Adaptive voting aggregation for partial ballots in crowdsourcing

Elizabeth Silver and Richard de Rozario

Abstract

We evaluate rating aggregation methods for an online crowdsourcing platform, where users are allowed to submit partial ballots and change their votes over time. Our goal is to identify good content as accurately as possible, using a rating aggregation method that is difficult for users to manipulate. We modify Range voting, Borda count and Copeland's method for partial ballots. We introduce a new method, which adapts to how users choose which items to rate. We evaluate all methods on synthetic data. If the synthetic agents submit top-truncated ballots, Borda count performs best, in the sense that it identifies the best content most often. If users submit randomly-censored partial ballots, then modified Range voting, Copeland's method and User Preferences Ranking perform about equally well, and dramatically outperform Borda count. Our new adaptive method performs well across both conditions.

1 Introduction

SWARM is collaborative software designed to improve reasoning, initially proposed by [7]. It is under current development as part of IARPA's CREATE research project [3].[1] Here we introduce and evaluate a novel voting aggregation method designed for SWARM and similar crowdsourcing platforms.

SWARM helps a team of 10-30 people to write a report (henceforth, "item") in response to a question. Individual team members submit items, rate each others' items, and provide feedback via comments. At the end of the work cycle, the highest-rated item is submitted as the team's collective work, so each team-member has an incentive to ensure the best item is highest-rated, and to improve the top item through feedback.

We assume that there is a true ranking of item quality, and each rater's assessment is a noisy measure of that ranking; some raters may be noisier than others. Unlike most social choice domains, SWARM aims not just to satisfy the preferences of voters/raters, but to learn from them – to use their ratings to estimate the true ranking. We are crowdsourcing the assessment of item quality.

Like other crowdsourcing platforms (e.g. Stack Overflow or Amazon reviews), SWARM is designed to put few demands or constraints on the user. SWARM allows users to rate items on a 0–100 scale, and users may rate as many or as few items as they wish; they usually rate only a small subset of items. Users are unable to rate their own items. SWARM allows users to rate at any time they wish during the workflow, and to alter their ratings as often as they wish. The item ranking is visible to users and is updated in real time as users add or change their ratings. Users may also edit their items or submit new items at any time.

[1] More information about the SWARM Project is available from the project website: `https://www.swarmproject.info/` . The system is currently invitation-only; the source code is scheduled to be released in December 2018.

This research is based upon work supported in part by the Office of the Director of National Intelligence (ODNI), Intelligence Advanced Research projects Activity (IARPA). The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of ODNI, IARPA, the U.S. Government, or the University of Melbourne. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright annotation therein.

Old ratings are retained when items are edited; users are encouraged but not required to update their ratings. The voting aggregation method for SWARM must work with partial ballots, votes changing over time, and late entry of new items.

In our initial design, SWARM aggregated the ratings by averaging the ratings given to each item. (When all users rate all items, averaging is equivalent to range voting.) No non-dictatorial voting system is strategy-proof [2, 6], but averaging is easy to manipulate, simply by giving extreme scores. In our initial testing, every team member independently discovered and exploited this strategy. The manipulation was well-intentioned: each user believed they were helping to correct the 'mistakes' of their team-members. However, it reduced the information conveyed about user preferences, coarsened the feedback to item authors, and undermined the democratic nature of SWARM; users who gave extreme scores had more leverage over the item rankings.

We sought an aggregation method that would resist manipulation, while identifying the best item as accurately as possible. We tested several different rating aggregation methods on synthetic data. The method that performed best in simulation was a novel adaptive ranking method, which we present here, along with the results of our simulation study.

2 Ballot structure

Each problem on SWARM prompts users to submit items and rate each others' items. If there are n users and m items, the ratings can be turned into an n-by-m matrix $\mathbf{A}$, where each row is a user's ballot, and each column is a item:

$$\mathbf{A} = \begin{array}{c} \\ u_1 \\ u_2 \\ \vdots \\ u_n \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & \cdots & m \end{array} \\ \begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,m} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n,1} & a_{n,2} & \cdots & a_{n,m} \end{bmatrix} \end{array}$$

Each entry a_{ij} may be null, if user i has not rated item j, or else an integer between 0 and 100 (inclusive).[2] Users are not required to submit an item and may submit multiple items. Consequently, m may be larger or smaller than n, and item i might not have been written by user i. Users are unable to rate their own items so at least one cell per column must be null. Users are not required to rate all other items so the matrix may be sparse.

Each ranking method must take $\mathbf{A}$ as input and produce a score vector $\mathbf{s}$, where the entry s_j is the score of item j. The rank order of items in the score vector gives us the rank order of the items.

3 Ranking Methods

We implemented and tested the following ranking methods:

3.1 Range voting modified for partial ballots

We considered two extensions of range voting to partial ballots:

[2] Users can submit either a 'simple rating' that is a single score out of 100, or they may submit a 'detailed rating' where they rate items on seven sub-scales related to quality of reasoning and clarity of communication. These sub-scales provide detailed feedback for the item author, but for ranking purposes they are aggregated into an 'overall score' out of 100, so in this paper we only consider aggregating these overall scores between raters.

3.1.1 Averaging

Average all the ratings given to each item. The score for item j is:

$$s_j = \begin{cases} \frac{1}{n_j} \sum_{\{i:\ a_{ij} \text{ is not null}\}} a_{ij} & \text{if } n_j \neq 0 \\ 0 & \text{otherwise} \end{cases}$$

where n_j is the number of users who have rated item j, and a_{ij} is the score user i gives to item j, or null if user i has not rated item j.

Note that if some users are 'easy graders', who give unusually high ratings, then the items they choose to rate will have an advantage, and items rated by 'harsh graders' will be disadvantaged. When voters can submit partial ballots, easy graders can bias the results of range voting/averaging.

Note: There exists a version of range voting, Normalized Range Voting [5], which is neither affected by easy graders, nor vulnerable to manipulation by giving extreme scores. Each user's ballot is standardized so that the lowest- and highest scoring items are given the minimum and maximum score, respectively. However, we did not consider this form of normalization because we did not see a way to extend it to partial ballots. If a user rates only the items she considers the best, then her lowest-rated item should still receive a high score. Without knowing how the user decided which items to rate, it is unclear what the lower bound on the normalized range should be.

3.1.2 Averaging except for single-rated answers

Under Averaging, a item with one rating of 95 will outrank an item with 10 ratings and an average score of 94. In our initial testing, sometimes a item with one rating was top-ranked, but we felt that a single rating provides very little information about item quality. We tried a heuristic to avoid this situation in future: rank every item with a single rating below the items with two or more ratings (and above the unrated items). Among the items with a single rating, rank by the score of that rating; and among the items with two or more ratings, rank by average score. In other words:

$$s_j = \begin{cases} \sum_{\{i:\ a_{ij} \text{ is not null}\}} a_{ij} & \text{if } n_j = 1 \\ b + \frac{1}{n_j} \sum_{\{i:\ a_{ij} \text{ is not null}\}} a_{ij} & \text{if } n_j > 1 \\ 0 & \text{otherwise} \end{cases}$$

where b is the maximum score among the items with only one rating, and n_j is the number of users who have rated item j.

Note: This approach is vulnerable to some of the same problems as averaging: manipulation by giving extreme ratings, and easy grader bias. It is also ad hoc; there are more principled ways of handling the uncertainty that we did not explore, such as using the lower bound of a confidence interval around the average rating. We present this simple method because it performed well on synthetic data (where no strategic voting took place).

3.1.3 Median score

We also tried using just the *median* rating given to each item. The median is robust to outliers, making it difficult to manipulate by giving extreme scores.

3.2 Borda count modified for partial ballots

We considered two extensions of Borda count to partial ballots:

3.2.1 Pessimistic Borda

This is the modification that [1] calls "pessimistic Borda", which is used in Slovenian elections. Each unrated item in a ballot receives zero points. Each rated item receives a point for being rated, and a point for every rated item that they outrank. The Pessimistic Borda score of item j is:

$$s_j = \sum_i \mathbb{1}(a_{ij} \text{ not null}) \; + \sum_{k \neq j} \mathbb{1}(a_{ij} > a_{ik})$$

where $\mathbb{1}(a_{ij} > a_{ik}) = 0$ if either a_{ij} or a_{ik} is null.

Pessimistic Borda creates a strong incentive for a user to rate more items: with each extra rating, the number of points given to the top-rated item increases by 1. A user who ranks all items contributes $\frac{m(m+1)}{2}$ points to the overall tally, whereas a user who rates only one item contributes only 1 point. We would like users to rate many items, but this incentive is perhaps too strong – it gives some users a lot of power over the overall rankings. We also tried a modification of Borda that weakens this incentive:

3.2.2 Normalized Pessimistic Borda

Similar to Pessimistic Borda, except that the scores are normalized so that the maximum number of points each user awards to a item is 1:

$$s_j = \sum_i \frac{1}{n_i} \left[\mathbb{1}(a_{ij} \text{ not null}) + \sum_{k \neq j} \mathbb{1}(a_{ij} > a_{ik}) \right]$$

where $\mathbb{1}(a_{ij} > a_{ik}) = 0$ if either a_{ij} or a_{ik} is null, and where n_i is the number of items rated by user i.

This approach is similar to the "optimistic Borda" extension discussed by [1], in the sense that each user's top-ranked item gets the same number of points from that user's vote, regardless of how many other items the user has rated. However, unlike optimistic Borda, Normalized Pessimistic Borda does not create a large gap between the lowest-ranked items and the unranked items. Users who rate more items still contribute more points to the score vector, up to a total of $\frac{m+1}{2}$ points, so there is still an incentive to rate more items, it is just a weaker incentive than in Pessimistic Borda.

3.3 User Preferences Ranking

[4, Chapter 10] describe User Preferences Ranking, which was designed for settings like Amazon Reviews, where users rate items on a five-star scale, but each user rates only a small subset of all items. User Preferences Ranking is a pairwise comparison ranking system: it looks at the differences in scores between pairs of items that were both rated by the same user. For example, if user u gives item i a rating of 5 and item j a rating of 3, then item i gets 2 points from that user (and item j gets -2 points).

[4, page 118] start by defining an m-by-m skew-symmetric matrix, $\mathbf{K}$. Each cell k_{ij} holds the average score difference between items i and j, among all the users who rated both items:

$$\mathbf{K} = \begin{pmatrix} 0 & k_{1,2} & \cdots & k_{1,m} \\ k_{2,1} & 0 & \cdots & k_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ k_{m,1} & k_{m,2} & \cdots & 0 \end{pmatrix}$$

Where the k_{ij} are given by:

$$k_{ij} = -k_{ji} = \begin{cases} \frac{1}{n_{ij}} \sum_{U_i \cap U_j} a_{ui} - a_{uj} & \text{if } n_{ij} \neq 0 \\ 0 & \text{if } n_{ij} = 0 \end{cases}$$

where $U_i \cap U_j$ is the set of users who have rated both items i and j, and n_{ij} is the number of users in $U_i \cap U_j$. The pairwise differences between item i and all other items is then averaged to get an overall score for item i. In other words, the overall score vector $\mathbf{s}$ is just the row means of $\mathbf{K}$:

$$\mathbf{s} = \frac{\mathbf{Ke}}{m}$$

where $\mathbf{e}$ is a vector of ones.

However, if item j received no ratings at all, it will have a score of 0, whereas if item i was consistently rated lower than other items, its score will be less than 0. We believe that items with no ratings should have the lowest scores, so we modified the score vector slightly:

$$s_j = \begin{cases} b + \frac{1}{m} \sum_i k_{ij} & \text{if } n_j \neq 0 \\ 0 & \text{if } n_j = 0 \end{cases}$$

where $b = 1 - \min_j \frac{1}{m} \sum_i k_{ij}$, and n_j is the number of users who have rated item j.

Users influence pairwise comparisons whenever they rate both items in the pair. If a user has rated k items, she has influenced $\frac{k^2-k}{2}$ pairwise comparisons. This creates a strong incentive to rate more items.

Unlike Averaging, User Preferences Ranking is not biased by 'easy graders'. The absolute values of users' ratings don't matter, only the score differences between pairs of items they have rated.

However, User Preferences Ranking is still vulnerable to manipulation by giving extreme scores, and the strategy is fairly simple. A user who wished to give the maximum possible boost to j's score would give j a rating of 100, and all other items ratings of 0. We therefore explored a combination of User Preference Ranking and Copeland's method, extending Copeland's method to handle partial ballots.

3.4 Combination of Copeland's Method and User Preferences Ranking for partial ballots

Copeland's method tallies the 'wins' and 'losses' for each item, rather than the point differences. Because some items were rated by more users than others, we normalized by the number of users who rated both items. The resulting method is a combination of Copeland's method and User Preferences Ranking. We start by defining a matrix $\mathbf{K}$ of pairwise differences, as above, but this time the entries k_{ij} are given by:

$$k_{ij} = -k_{ji} = \begin{cases} \frac{1}{n_{ij}} \sum_{u \in U_i \cap U_j} \mathbb{1}(a_{ui} > a_{uj}) - \mathbb{1}(a_{ui} < a_{uj}) & \text{if } n_{ij} \neq 0 \\ 0 & \text{if } n_{ij} = 0 \end{cases}$$

Note that if user u prefers item i to item j, so $a_{ui} > a_{uj}$, then she contributes $\frac{1}{n_{ij}}$ to the entry k_{ij}. If she rates j higher, she contributes $\frac{-1}{n_{ij}}$. If she's indifferent, she contributes 0 points, but the fact that she has rated both items means the denominator n_{ij} will be larger by 1 than if she had rated neither of them; so ties matter.

Similarly to User Preferences Ranking, when we calculate the final score vector, we add a bonus to all items that received ratings so they score higher than unrated items:

$$s_j = \begin{cases} b + \frac{1}{m} \sum_i k_{ij} & \text{if } n_j \neq 0 \\ 0 & \text{if } n_j = 0 \end{cases}$$

where $b = 1 - \min_j \frac{1}{m} \sum_i k_{ij}$, and n_j is the number of users who have rated item j.

Modified Copeland is not vulnerable to manipulation by giving extreme scores, because only the relative rank of the items influences the score, not the raw numbers.

3.5 Adaptive Modified Copeland

In our preliminary testing on synthetic data, we included a parameter for how users decided which items to rate: either they rated the items they considered the best (called 'top-truncated ballots' by [1]), or they rated a random selection of items. Both scenarios are plausible. We don't know how our users will choose which items to rate, and their behavior may change depending on the problem they are working on, or even the time in the workflow (e.g. users may have top-truncated ballots until some new items are posted).

We noticed that both modifications of Borda count performed best with top-truncated ballots, but all other methods outperformed Borda count under random selection. This makes sense: with top-truncated ballots, the number of ratings received provides extra information about answer quality. Under Borda count, more ratings always means more points – an answer cannot be penalized for being rated more often – but this is not the case for our other aggregation methods.

This inspired an adaptive method. We attempt to *learn* whether users are posting top-truncated ballots or rating a random selection of items, and alter the score vector accordingly; if users appear to be submitting top-truncated ballots, we add a bonus for number of ratings received. The test is simple. Under random selection, the number of ratings received should be uniformly distributed and uncorrelated with the Modified Copeland score. By contrast, if users choose to rate their top k items, then the number of ratings received will correlate with item quality, and therefore will also correlate with the Modified Copeland score vector.

Our adaptive method calculates the Spearman rank correlation $\rho_{\mathbf{c},\mathbf{n}}$ between the Modified Copeland score vector $\mathbf{c}$ and the number of ratings received $\mathbf{n}$. It then tests whether $\rho_{\mathbf{c},\mathbf{n}}$ is significantly different from zero. If the test fails ($p \geq 0.05$) or the correlation is negative ($\rho_{\mathbf{c},\mathbf{n}} < 0$), the method returns $\mathbf{c}$ as the score vector. If the test passes and the correlation is positive ($\rho_{\mathbf{c},\mathbf{n}} > 0$), then return a weighted sum of $\mathbf{c}$ and $\mathbf{n}$, where the weight on $\mathbf{n}$ depends on the observed correlation $\rho_{\mathbf{c},\mathbf{n}}$. In other words:

$$s_j = \begin{cases} c_j & \text{if } \rho_{\mathbf{c},\mathbf{n}} < 0 \text{ or } p \geq 0.05 \\ c'_j + \rho_{\mathbf{c},\mathbf{n}} \cdot n'_j & \text{if } \rho_{\mathbf{c},\mathbf{n}} > 0 \text{ and } p < 0.05 \end{cases}$$

where $\mathbf{c}'$ and $\mathbf{n}'$ were normalized to have the same range, i.e.

$$c'_j = \begin{cases} \frac{c_j - \min(\mathbf{c})}{\max(\mathbf{c}) - \min(\mathbf{c})} & \text{if } \max(\mathbf{c}) - \min(\mathbf{c}) \neq 0 \\ 0 & \text{if } \max(\mathbf{c}) - \min(\mathbf{c}) = 0 \end{cases}$$

and likewise for n'_j.

Adaptive Modified Copeland is not vulnerable to manipulation by giving extreme scores, becauseit only considers the number of ratings received and the relative rank of each item on a user's ballot. It ignores the exact value of ratings. It is not clear how robust Adaptive Modified Copeland is to other kinds of manipulation. With partial ballots, votes changing over time, and late entry of new item, the space of potential strategies is large and difficult to study.

We evaluated performance on synthetic data, without modeling strategic voting.

4 Performance on synthetic data

We developed a sampling scheme for synthetic data that represents some aspects of how ballots are cast on SWARM, and allows us to evaluate how well the rating aggregation methods identify good content. We first generate an underlying quality vector, $\mathbf{q}$, which represents the true quality of the items. Each user's ratings are noisy estimates of $\mathbf{q}$, and some raters are more accurate than others. We then redact some ratings; either from the items each user considers worst (generating top-truncated ballots) or a random selection of items. The final ballot is sparse and noisy. To evaluate each rating aggregation method, we measure how well it recovers the rank order of $\mathbf{q}$.

We simplify the sampling process in several ways. We assume that each user submits at most one answer, even though on SWARM they are able to submit multiple answers. We assume that all users are rating honestly, rather than strategically. We ignore the passage of time, drawing all data simultaneously. As such we don't model (a) users' reactions to the current rank order at any point in time, (b) the changing visibility of items as the rank order changes, or (c) new items being posted.

4.1 Sampling scheme for synthetic data

We assume that each user has some level of skill, which influences both their answer quality, and their discernment – how well they can rate the quality of other users' answers. Our sampling procedure is as follows:

1. For each of 30 users, draw the user's discernment, $d_i \sim \mathcal{U}(0,1)$
2. Draw the quality of each user's answer, q_i, from a truncated Normal distribution $\mathcal{N}(d_i, 0.05)$ where $q_i \in [0,1]$
3. For each user u_i and each item j, draw that user's assessment of the answer from a truncated Gaussian, where the variance is inversely related to the user's discernment: $a_{ij} \sim \mathcal{N}(q_j, (1-d_i)\cdot\sigma_a)$ restricted to $a_{ij} \in [0,1]$ and where $\sigma_a \in \{.4, .6\}$
4. Draw each user's easiness bias, $e_i \sim \mathcal{N}(0, \sigma_e)$ where $\sigma_e \in \{.1, .5\}$. Transform each users ballot to push the scores up or down: $\mathbf{a}_{i\cdot} = \text{expit}(\text{logit}(\mathbf{a}_{i\cdot}) + e_i)$
5. For each user u_i, draw the number of answers they left unrated from a binomial distribution: $k_i \sim Binom(30, p_k)$, where $p_k \in \{.3, .7\}$.
6. For each user u_i, redact (set to null) k_i ratings. If `toprated = True`, redact the lowest k_i numbers in $\mathbf{a}_{i\cdot}$. Otherwise if `toprated = False`, redact a random set of k_i ratings from $\mathbf{a}_{i\cdot}$
7. Redact all ratings on the diagonal of $\mathbf{A}$, as users cannot rate their own items.
8. For each item j, flip a coin to decide whether to remove its entire column from $\mathbf{A}$, since many users do not submit answers. Sample $l_j \sim Bernoulli(0.5)$, and if $l_j = 1$, remove $\mathbf{a}_{\cdot j}$
9. For each user u_i, flip a biased coin to decide whether to remove their entire row from $\mathbf{A}$, since some users do not submit any ratings. Sample $g_i \sim Bernoulli(p_g)$, where $p_g \in \{.3, .6\}$ If $g_i = 1$, remove $\mathbf{a}_{i\cdot}$

The result of this sampling procedure is a set of rating matrices $\mathbf{A}$ that are sparse and of varying sizes. The tuning parameters are:

- $\sigma_a \in \{.4, .6\}$, which controls how 'noisy' the ratings are overall
- $\sigma_e \in \{.1, .5\}$, which controls the range of how 'easy' or 'harsh' raters are
- $p_k \in \{.3, .7\}$, which controls how likely a user is to leave an item unrated
- `toprated` $\in \{True, False\}$, which controls whether users' partial ballots are top-truncated or randomly-selected

- $p_g \in \{.3, .6\}$, which controls how likely a user is to submit no ratings at all

For each combination of settings of the tuning parameters, we generated 100 such matrices. This produced a sample of 3200 rating matrices.

4.2 Performance

4.2.1 Metrics

We used two metrics to assess performance:

1. Average Spearman's ρ between the score vector $\mathbf{s}$ and the quality vector $\mathbf{q}$
2. The proportion of samples in which the rating aggregation method correctly selected the best item, i.e. $\arg\max(\mathbf{q}) = \arg\max(\mathbf{s})$

4.3 Results on synthetic data

We found that the variance of raters' easiness bias, σ_e, had little effect on the results and did not appear to interact with any of the other conditions, so all results are averaged over both values of σ_e.

Figure 1 shows performance under all combinations of the other four tuning parameters. The methods are grouped, with the three methods based on Range voting shown first, then the two Borda methods, then the three methods based on pairwise comparisons.

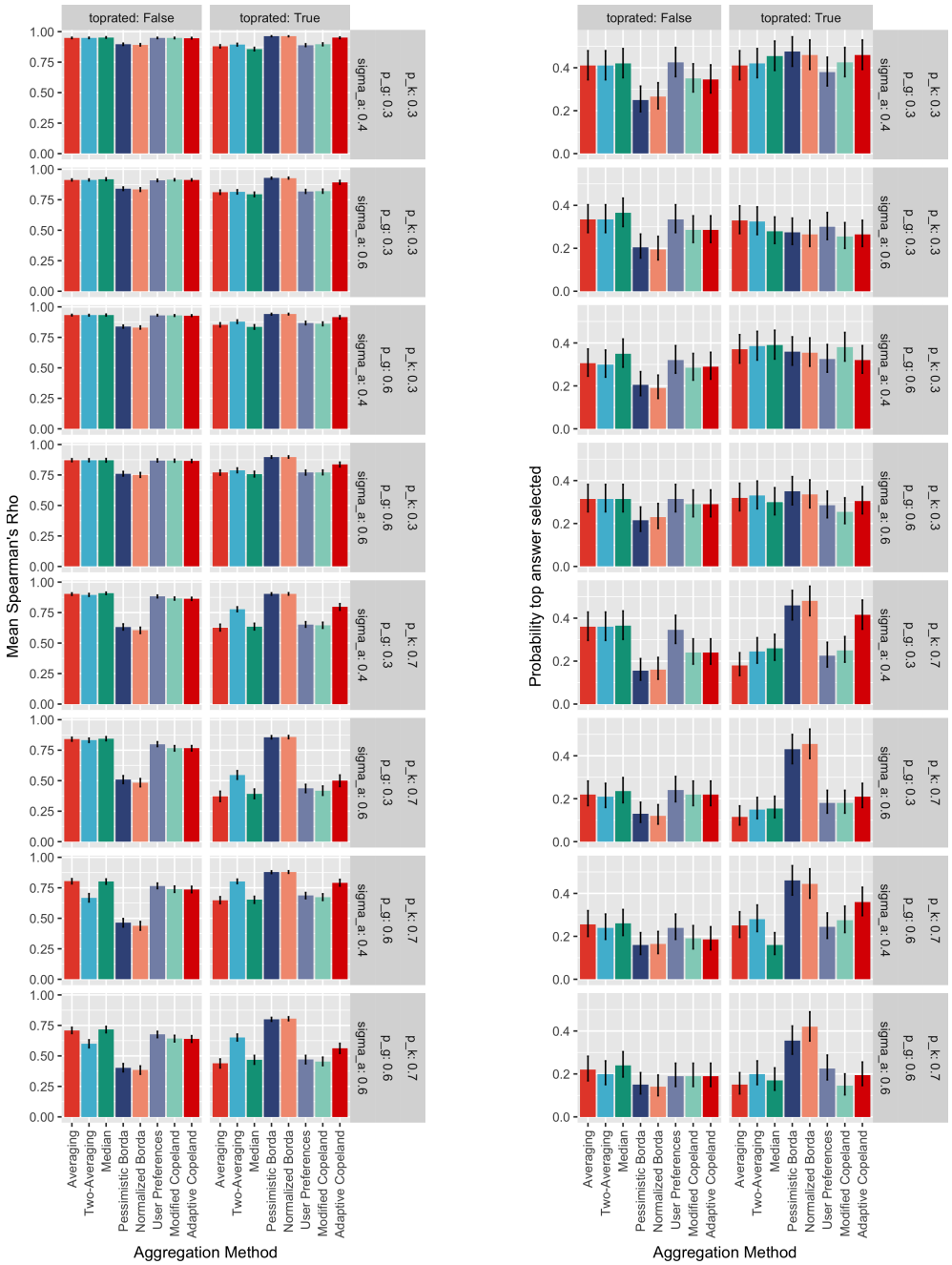
The facet rows in Figure 1 show the differences in performance as raters are more likely to leave an item unrated ($p_k = 0.7$ vs 0.3), to rate no items at all ($p_g = 0.6$ vs 0.3), or to rate less accurately ($\sigma_a = 0.6$ vs 0.4). Lower rows show the cases with sparser and noisier ratings. The facet columns show the difference in performance when raters rate a random selection of items (`toprated = False`) vs. the items they consider best (`toprated = True`).

The striking result is that `toprated` controls which methods perform best. If users rate a random selection of items, then Pessimistic Borda and Normalized Borda perform worse than Averaging, Two-Averaging, Median, User Preferences, Modified Copeland and Adaptive Copeland, and the difference grows more pronounced when the data is sparse and noisy. However, when users rate the items they consider best, the two Borda methods outperform the other methods, and perform particularly well when the data is sparse.

We are most interested in Median, the two Borda methods, Modified Copeland and Adaptive Copeland, because these five methods are resistant to manipulation by giving extreme scores. We do not know how users will choose which items to rate, or how sparse or noisy their ratings will be, so we want to choose a method that performs adequately in all conditions.

Adaptive Copeland appears to be the minimax method, i.e. it has the best performance in worst-case scenarios. It does not perform quite as well as Borda count when `toprated = True`, but it does better than the Median and Modified Copeland; and when `toprated = False`, it far outperforms Borda count.

When `toprated = True`, Adaptive Copeland does well provided the data is not very noisy ($\sigma_a = 0.4$). Its advantage is reduced when the data is noisier ($\sigma_a = 0.4$), especially by the second metric, the probability that it selects the top answer. This may reflect the fact that Adaptive Copeland relies on a statistical significance test in order to adapt to user behavior, and the significance test becomes less reliable when the ratings are noisier.



(a) Average rank correlation between true quality and aggregated ratings (±95% CIs)

(b) Proportion of trials in which the best answer was selected (±95% Wilson CIs)

Figure 1: Performance on two metrics. Row facets show how performance degrades as the ratings become sparser (when p_k and p_g increase) and noisier (when σ_a increases). Column facets give different values of `toprated`. Borda performs worst when `toprated = False`, but best when `toprated = True` and the data are sparse. Among methods that can't be manipulated by extreme scores, Adaptive Copeland has the best worst-case performance.

5 Conclusion

SWARM seeks to aggregate the opinions of a team of 10–30 people, while allowing them to submit partial ballots, change their votes over time, and create new items at any time. Users rate items on a 0–100 scale.

Initially we aggregated ratings by averaging these numbers, but our initial testing revealed that every user was manipulating the voting process by submitting extreme scores. We therefore evaluated a range of methods to find one that is relatively accurate while resisting this form of manipulation. We extended Borda count and Copeland's method to partial ballots.

Our tests on synthetic data showed that the ideal method depended entirely on how users choose which items to rate. If users submit top-truncated ballots, then Borda Count – which awards points simply for being rated – outperformd other methods; if users rated a random selection of items, then Borda Count dramatically underperformed the other methods. Unfortunately, we do not know how each user selects items to rate; it may change from problem to problem, or when new items are entered.

We developed a new method, Adaptive Copeland, which attempts to learn whether users are submitting top-truncated ballots or rating random items, based on the correlation between 'number of ratings received' and the Copeland score; if users seem to be submitting top-truncated ballots, Adaptive Copeland awards a bonus for each rating received.

In our simulation study, Adaptive Copeland performs well regardless of how users choose which items to rate. Based on these results, we have implemented Adaptive Copeland for the SWARM platform and are currently planning a study with real users.

Because of our application in crowdsourcing, we have focused on finding a vote aggregation method that is *accurate*, rather than one that satisfies the preferences of users. We hope that future research will examine the properties of Adaptive Copeland as a social choice mechanism.

References

[1] Dorothea Baumeister, Piotr Faliszewski, Jérôme Lang, and Jörg Rothe. Campaigns for lazy voters: Truncated ballots. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*, pages 577–584. International Foundation for Autonomous Agents and Multiagent Systems, 2012.

[2] Allan Gibbard. Manipulation of voting schemes: a general result. *Econometrica: journal of the Econometric Society*, pages 587–601, 1973.

[3] Intelligence Advanced Research Projects Activity (IARPA). Crowdsourcing Evidence, Argumentation, Thinking and Evaluation. IARPA-BAA-15-11, 2016. URL `https://www.iarpa.gov/index.php/research-programs/create/create-baa`.

[4] Amy N Langville and Carl D Meyer. *Who's #1?: the science of rating and ranking*. Princeton University Press, 2012.

[5] Curtis Menton. Normalized range voting broadly resists control. *Theory of Computing Systems*, 53(4):507–531, 2013.

[6] Mark Allen Satterthwaite. Strategy-proofness and Arrow's conditions: Existence and correspondence theorems for voting procedures and social welfare functions. *Journal of Economic Theory*, 10(2):187–217, 1975.

[7] Timothy van Gelder and Richard de Rozario. Pursuing fundamental advances in human reasoning. In *International Conference on Artificial General Intelligence*, pages 259–262. Springer, 2017.

Elizabeth Silver
The University of Melbourne
Melbourne, Australia
Email: `elizabeth.silver@unimelb.edu.au`

Richard de Rozario
The University of Melbourne
Melbourne, Australia
Email: `richard.derozario@unimelb.edu.au`