Social Choice for Human Computation

Andrew Mao
Harvard University
mao@seas.harvard.edu

Ariel D. Procaccia
Carnegie Mellon University
arielpro@cs.cmu.edu

Yiling Chen
Harvard University
yiling@seas.harvard.edu

Abstract

Designers of human computation systems often face the need to aggregate noisy information provided by multiple people. While voting is often used for this purpose, the specific voting methods that are employed are typically naïve. The theory of social choice provides powerful tools for exactly these settings. We conduct experiments on Amazon Mechanical Turk which demonstrate empirically that more intricate voting rules, which are known to provide theoretical guarantees under simple models of noisy votes, significantly outperform basic voting rules. Our work has the potential to spark a long-term interaction between human computation and (computational) social choice.

1 Introduction

Human computation is a fast-growing field that seeks to harness the relative strengths of humans to solve problems that are difficult for computers to solve alone. The field has recently been gaining traction within the AI community, as increasingly more deep connections between AI and human computation are uncovered (Dai, Mausam, and Weld 2010; Shahaf and Horvitz 2010).

Generally speaking, there are two streams of research in human computation. The first stream focuses on *games with a purpose* (von Ahn and Dabbish 2008), which are designed to extract useful information from people as they play an enjoyable game. While there is no need for monetary incentives in such systems, the game must inevitably be tailor-made for a very specific purpose. A prominent group of games with a purpose are known as *scientific discovery games* (Cooper et al. 2010). For example, in EteRNA (`http://eterna.cmu.edu`) players collaborate in folding RNA into its stable shape. Players submit different proposals for stable designs; a subset of designs are then synthesized in a laboratory to learn which design is truly the most stable (and to score the players). EteRNA's tagline, "played by humans, scored by nature", nicely summarizes the process.

The second strand of human computation systems is based on *online labor markets*. *Amazon Mechanical Turk* (`http://www.mturk.com`) — MTurk henceforth — is perhaps the paradigmatic online labor market. There are two sides to the market: requesters, who post human intelligence tasks (HITs), and workers, who perform HITs for monetary compensation. HITs often consist of repetitive tasks that are simple for humans but difficult for computers, such as labeling pictures. By combining different types of tasks into a process, requesters can achieve more complex objectives. MTurk's tagline, "artificial artificial intelligence", cleverly refers to the fact that requesters interact with a seemingly intelligent software system yet some of the actual computation is ultimately carried out by humans.

The input provided by humans via human computation systems is typically quite noisy, and in applications beyond very simple tasks there is often a need to aggregate information into a collective choice. A natural, common way of doing this is by crowdsourcing this stage as well, and specifically letting people *vote* over different proposals that were submitted by their peers. For example, in EteRNA thousands of designs are submitted each month, but only a small number k of them can be synthesized in the lab (as of late 2011, $k = 8$). To single out k designs to be synthesized, players vote by reporting their k favorite designs, each of which is awarded one point. The k designs with most points are then synthesized. This voting rule is known as *k-approval*.

Voting is also frequently used on MTurk. The popular TurKit toolkit for MTurk, recently created by Little et al. (2010b), is essentially a programming language for creating and managing tasks, and in particular provides an implementation of a voting function. This function receives two alternatives and a threshold as input, and posts HITs asking workers to single out their preferred alternative, until the number of votes received by one of the alternatives is greater than the given threshold; to implement the common best-3-out-of-5 vote, it is sufficient to elicit three votes, and elicit more only if the first three agents do not all favor the same alternative. The authors give an example where several suggestions for things to do in New York, themselves generated by workers, are sorted using such pairwise comparisons. Little et al. (2010a) also demonstrate how human computation processes can solve more complex problems using many iterations of voting. However, in general, ranking alternatives according to pairwise comparisons can lead to cycles; this phenomenon is known as the *Condorcet paradox*. So what is the best method to construct a ranking of the alternatives?

Enter social choice theory, a field that deals with the aggregation of individual preferences into a collective decision. Social choice theory has been studied for centuries by mathematicians and economists. In the last two decades, a field which marries computer science and social choice theory — *computational social choice* — has emerged. Most of the work in computational social choice focuses on applying computational paradigms to social choice theory, for example, by studying the computational complexity of winner determination (Hemaspaandra, Hemaspaandra, and Rothe 1997; Conitzer 2006; Brandt et al. 2008) and manipulation (Faliszewski and Procaccia 2010; Conitzer, Sandholm, and Lang 2007; Procaccia and Rosenschein 2007) in elections. There is very little work that applies social choice theory to computer science (see (Dwork et al. 2001) for one exception).

Our main conceptual contribution lies in the suggestion that the huge body of literature on social choice theory can be leveraged to design better human computation systems. Specifically, we suggest replacing the somewhat naïve voting procedures used so far with more sophisticated voting rules that are based on a well-studied, principled approach to collective decision making.

Our approach and results. We are interested in settings where there is a true ranking of several alternatives according to quality. Each voter provides us with his own ranking of the alternatives, and our goal is to identify either the entire true ranking or its top alternative. The question is which voting rule to use for vote aggregation. This is a slightly unusual setting because there are no "preferences" over alternatives, only rankings that reflect subjective estimates of the quality of different alternatives. Nevertheless, this setting is a perfect fit with the view of voting rules as maximum likelihood estimators. This view was proposed by the Marquis de Condorcet as early as the 18th Century; it was picked up by Young (1988) two centuries later, and more recently studied by AI researchers (Conitzer and Sandholm 2005; Conitzer, Rognlie, and Xia 2009). The premise is that the ranking provided by each voter is a noisy estimate of the true ranking, which is generated using a known noise model, and a voting rule aggregates such noisy information. An ideal voting rule then outputs the ranking (resp., alternative) that is most likely to be the true ranking (resp., most likely to be the true top alternative).

But do voting rules that are provably maximum likelihood estimators (for certain theoretical noise models) perform better than naïve voting rules in practical human computation settings? To answer this question, we designed a voting experiment and gathered data extensively from real subjects on MTurk. Our core design insight allows for adjustment of the amount of implicit noise with which voters perceive an underlying ground truth that is known to the experimenter. However, as the noise itself is still generated by the voters, this allows us to compare the performance of several voting rules in realistic conditions. Previous empirical comparisons of voting rules, such as the one by Forsythe et al. (1996), do not measure the accuracy of different voting rules in teasing out an underlying truth, nor do they involve large-scale experiments in human computation settings.

In our voting experiments, the alternatives were 8-puzzles, each taking some minimum number of sliding moves to reach a goal state. Workers were asked to rank a set of four 8-puzzles according to how close they are to the goal state. By varying how close the puzzles were (in number of moves from the goal) to each other and how far the entire set was from the goal state, we were able to reliably control the noise with which voters evaluated the rankings.

After collecting almost 3500 rankings at varying levels of noise from almost 800 unique voters, we found that the Kemeny voting rule clearly and significantly outperforms the simpler plurality rule that is often used (itself or close variants thereof) in human computation. We examined the power of a voting rule to both correctly elect a single winner and output the correct ranking of all alternatives. For both criteria, Kemeny consistently made less mistakes, in the statistically significant sense, than plurality, Borda, and maximin. Moreover, using the collected votes, we tested a trained positional scoring rule using cross-validation and also computed an ex-post optimal positional scoring rule. To our surprise, Kemeny consistently outperformed both the trained and the ex-post optimal positional scoring rules.

2 Voting Rules

A typical social choice setting has a set of n *voters*, $\mathcal{N} = \{1, 2, \ldots, n\}$ and a set of m *alternatives* (or candidates), $\mathcal{A} = \{a_1, a_2, \ldots, a_m\}$. Each voter i has a preference σ_i, which is a total order over $\mathcal{A}$. In other words, each voter ranks the alternatives. Let $\mathcal{L}$ denote the set of all total orders over $\mathcal{A}$. Then, $\sigma_i \in \mathcal{L}$, $\forall i \in \mathcal{N}$. A *preference profile* $\vec{\sigma}$ is a collection of the preferences of the n agents, $\vec{\sigma} \in \mathcal{L}^n$.

Social choice theorists have developed a large number of voting rules for aggregating individual preferences. Depending on whether the output is a single winning alternative or a preference ranking of all alternatives, a voting rule can correspond to a *social choice function* or a *social welfare function*.[1] A social choice function is a function $C : \mathcal{L}^n \to \mathcal{A}$, while a social welfare function is a function $W : \mathcal{L}^n \to \mathcal{L}$. Note that both functions receive a preference profile as input. Clearly, any social welfare function induces a social choice function by selecting the alternative at the first position in the social preference ranking.

In this paper, we consider the following four popular voting rules:

- **Plurality**: Each voter casts a single vote for his most preferred alternative. The alternative that receives the most votes wins. If a ranking is desired, alternatives can be ranked by the number of votes received.
- **Borda**: For each voter who places an alternative at position k in his ranking σ_i, the alternative receives a score of $m-k$. The alternative that receives the highest total score wins. Alternatives are ranked by their total scores.

[1] In the computational social choice literature, the term "voting rule" sometimes coincides with social choice function, whereas "rank aggregation rule" is equivalent to social welfare function. We do not make this distinction here.

- **Maximin**: Let $N(a_i, a_j)$ be the number of voters who rank alternative a_i higher than alternative a_j. An alternative i's maximin score is its worst score in a pairwise election, that is $\min_{j:j\neq i} N(a_i, a_j)$. Alternatives are ranked by their maximin scores.
- **Kemeny**: The Kendall-tau distance between two preferences σ and σ' is given by

$$K(\sigma, \sigma') = \frac{1}{2} \sum_{(a,a')\in\mathcal{A}^2:a\neq a'} K_{a,a'}(\sigma, \sigma'), \qquad (1)$$

  where $K_{a,a'}(\sigma, \sigma')$ equals 0 if alternatives a and a' are in the same order in σ and σ' and 1 if they are in the opposite order. The ranking that has the smallest total Kendall-tau distance summed over all individual preferences is selected. That is, $W(\vec{\sigma}) = \min_{\pi\in\mathcal{L}} \sum_{i\in\mathcal{N}} K(\pi, \sigma_i)$. Although the Kemeny ranking is NP-hard to compute, heuristics are available (Conitzer, Davenport, and Kalagnanam 2006), and in cases like ours with few alternatives it is a simple matter (we just enumerate all 24 rankings of four alternatives).

The plurality and Borda voting rules belong to a larger family of voting rules called *positional scoring rules*. A positional scoring rule is given by a scoring vector $\vec{s} = \langle s_1, ..., s_m\rangle$ with $s_1 \geq s_2 \geq \cdots \geq s_m$ and $s_1 > s_m$. An alternative receives a score of s_k for each voter who places it at position k. Alternatives are then ranked by score. Plurality can be viewed as a positional scoring rule with $\vec{s} = \langle 1, 0, \ldots, 0\rangle$, and Borda is a positional scoring rule with $\vec{s} = \langle m-1, m-2, \ldots, 0\rangle$.

Voting Rules as Maximum Likelihood Estimators. An interesting view of the goal of voting rules is to reconstruct an underlying true ranking of alternatives given noisy information. Taking this perspective, there is a body of research seeking to single out voting rules that are maximum likelihood estimators (MLE) under a model of noisy votes.

More than two centuries ago, Condorcet suggested a natural noise model with an intuitive interpretation: given a true ranking, a voter ranks each pair of alternatives correctly with probability $p > 1/2$.[2] Condorcet solved the case of two alternatives, proving that plurality (known in this case as *majority*) is the maximum likelihood estimator. Moreover, as the number of voters n grows, the probability that plurality will elect the correct alternative increases; it approaches 1 as n approaches infinity (this seems obvious today, but probability theory was in its infancy in the 18th Century).

Young (1988) extended Condorcet's solution to the case of more than two alternatives. He showed that, under Condorcet's natural noise model, the voting rule that is most likely to output the correct *ranking* coincides with the Kemeny rule. Young also observed that if p is very close to 1/2 (i.e., when there is a lot of noise), the voting rule that is most likely to select the correct *winner* is Borda. More formally, for every number of voters n and number of alternatives m there is p sufficiently close to $1/2$ such that Borda is an MLE for the top alternative. Finally, Young observed that when p is very close to 1, there are examples where Maximin is the MLE for the top alternative.

Conitzer and Sandholm (2005) showed that Borda is an MLE under a different noise model. If, for any voter, the probability of ranking the correct winner at position k is $2^{(m-k)}/(2^m - 1)$, i.e., proportional to 2^{s_k} where s_k is the Borda score at position k, then Borda elects the most likely winner. More generally, Conitzer and Sandholm showed that every positional scoring rule is an MLE for the true ranking, or true top alternative, under some noise model with votes that are independent and identically distributed.

3 Sliding puzzles

In order to empirically test the efficacy of voting rules, we enlist the help of another usual suspect in AI. The 8-puzzle (and its larger cousin, the 15-puzzle) has a rich history in AI research, and is often used as a test environment for various search algorithms. However, the notion of sliding puzzles itself is much older—it was reportedly popularized in as early as 1880 in North America[3], and has been a well-known game for many generations ever since. Hence, this kind of puzzle game is natural and familiar to many people.

Figure 1 shows an example of some 8-puzzle states. The goal is to take any board state, and obtain a board state where the numbers are correctly ordered (from top to bottom and left to right) by sliding tiles into the empty space. Each movement of a single tile counts as one "move", and the general goal of the game is to solve the puzzle in as few moves as possible. An optimal solution to the 8-puzzle game, using the fewest number of moves, can be found by using A^* search or another search algorithm. However, when humans play this game, they will rarely be able to find a solution in the fewest number of moves without significant effort.

Why 8-puzzle? 8-puzzles have two main advantages from our point of view. First, because of the game's popularity and simplicity, many people intuitively understand what needs to be done without requiring elaborate instructions. Second, we can easily configure the game's difficulty by varying the distance between the starting state and the goal state; this feature will allow us to smoothly control the level of noise in workers' estimates. We describe in the next section how this feature allows a collection of 8-puzzles to facilitate an experiment that presents voters with a noisy perception of an underlying truth.

4 Methodology

Our goal was to test how different voting rules fare when applied to noisy collective estimates of a ground truth, and to evaluate how their performance changes with different amounts of noise. The design of our experiment required a balancing act between two extremes: if the users' perception is too noisy, all of the resulting rankings will be almost random, with the effect that no voting rule will outperform another except by random chance. However, if it is too clear to users what the true ranking is, then almost everyone will submit the true ranking and all voting rules will obtain the

[2] If cyclical preferences are generated, the process is restarted.

[3] See http://en.wikipedia.org/wiki/8-puzzle.

same result. We aimed to find a middle ground where the noise is neither too much nor too little.

We used four 8-puzzles as the alternatives in a voting experiment, where we asked workers to submit rankings of how far the puzzles are from the solution by the minimum number of moves. The ground truth of each puzzle is the optimal number of moves to the solution as computed by A^* search. While it may be hard for humans to directly come up with these numbers, asking users to compare items to each other naturally produces rankings, and collects preferences more effectively than requesting absolute judgments (Carterette et al. 2008). By making these comparisons, users produce perceived rankings, and by varying the distance to the solution we can adjust the difficulty of the task and tune the level of noise.

To collect votes at a certain level of noise, we chose a sequence of numbers, such as $(7, 10, 13, 16)$, and generated a set of four random puzzles that correspond to each number by minimum moves to solution. For example, for the above sequence, we would generate one puzzle that is 7 moves away from the goal (selected approximately uniformly over all such puzzles), one puzzle that is 10 moves away, etc. We then shuffled the puzzles randomly and presented them to voters in a convenient interface that allowed for comparisons with minimal effort.

We used MTurk to collect rankings over the set of puzzles in increasing distance to the goal state. Hence, the "winning" alternative is the puzzle that is closest to the goal. This is a design choice we made as we believe it may be easier for voters to identify the closest puzzle to the goal state than the furthest. However, the same methodology can be used to rank puzzles in decreasing distance to the goal state and elect the one that is furthest.

Experiment Setup. After initial tests to identify puzzles with the ideal level of noise, we focused on four sequences of four numbers $(d, d+3, d+6, d+9)$, for $d = 5, 7, 9, 11$. For each of these four sequences we generated 40 sets of four puzzles at the corresponding distances from the goal. We then collected approximately 20 preference rankings on each set. In other words, we collected 40 *preference profiles* per sequence with 20 voters each; overall our analysis is based on 3,200 rankings. All sets of data were gathered at a constant rate over 24-hour periods to minimize the effect of different types of workers throughout the day.

We paid \$0.10 for each HIT, which consisted of ranking one set of puzzles. After passing a short quiz, users were asked to sort the set of four puzzles using a simple drag-and-drop interface. For example, Figure 1 shows four puzzles which are 7, 10, 13, and 16 moves away from the goal. The puzzles were presented in randomized order in a square grid, and users were asked to order them in a line on a different part of the screen. Puzzles could be picked up and inserted at any point in the line to rearrange the order.

The task interface was designed to minimize the effect of laziness or difficulty of communication on the results. By randomizing the initial set of puzzles and arranging them in a square, we prevented lazy workers from dragging puzzles straight to their closest possible positions in the ranking line. The drag-and-drop interface also allowed voters to express their preferences with very little effort.

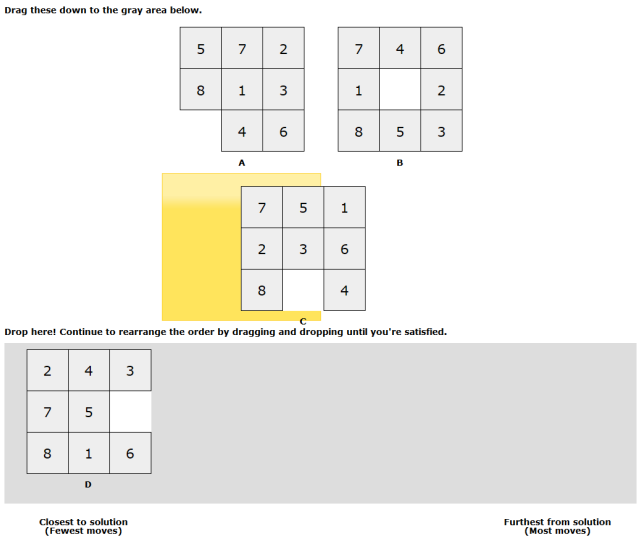


Figure 1: The experiment interface. Voters see a randomly shuffled set of four puzzles in a square, which they must arrange into a sequence from "closest to solution" to "furthest from solution". We use a drag and drop interface to minimize the effort for elicitation.

Each set of data collected over a 24-hour period enforced a limit of 5 HITs per user, and ensured that no user saw the same set of puzzles twice. We used the maximum of 5 HITs in order to increase the rate of data collection, but to avoid situations where workers become lazy from doing many HITs repeatedly.

5 Results

We tested the accuracy of the common voting rules described above in teasing out the correct top alternative and ranking. In addition, we tested optimized positional scoring rules using mixed integer linear programs that compute a positional score vector given a set of preference profiles. For first-place mistakes, the optimized rule first minimizes the number of flips between the correct first-place alternatives and all other alternatives, then maximizes the sum of minimum differences between the total score for the first-place alternative and all other total scores (for non-flipped pairs) across all preference profiles. For total mistakes, the optimized rule first minimizes the total number of flips to the correct ranking, then maximizes the sum of minimum differences between all correctly ranked (relative to each other) pairs across all preference profiles. The secondary objectives here are implemented using appropriate normalization and used to reduce generalization error.

The optimized positional scoring rules are trained in two ways: by using leave-one-out cross-validation (and averaged), and also by using the entire data set, which produces an ex-post optimal positional scoring rule. The rules trained using cross-validation are practical, and should be viewed as competing on the same ground as the other voting rules. In our results below, there appears to be some room to improve

the trained positional scoring rules to be closer to the ex-post optimum.

In the case of the ex-post optimal positional scoring rule the generalization error is irrelevant, but its performance is still subject to limits. Clearly a voting rule that returns the right ranking for each of the forty preference profiles would make no mistakes. However, due to the constraint imposed by the specific way positional scoring rules aggregate votes, even omniscience about the true rankings does not lead to a positional scoring rule that makes no mistakes. While the ex-post optimal positional scoring rule is not practical, we think of it as providing a very competitive benchmark.

Our results were computed by averaging the number of mistakes (first-place or total) made over 100 random subsets of 10 votes on each of 40 preference profiles for each sequence. In other words, the voting rules are applied to preference profiles consisting of 10 voters, each ranking the four alternatives. The same random subsets were used to evaluate each voting rule. This approach, using subset bootstrap sampling, not only simulates the effect of having fewer voters (and therefore more noise) in an election setting, but reduces the amount of statistical variance by computing an average number of mistakes on each set of puzzles. In addition, averaging brings the differences in number of mistakes closer to a normal distribution, which justifies using a paired t-test for different voting rules on the same preference profiles to test for statistical significance. The results computed using entire preference profiles of 20 voters (instead of subsets of size 10) are qualitatively similar. However, they make it slightly harder to establish statistical significance as strongly due to the increased variance.

First-place mistakes. To evaluate the voting rules' performance as a social choice function, which elects a single winner, we use the metric of first-place mistakes, i.e., the number of preference profiles where a voting rule fails to rank the puzzle that is closest to the goal state at the first position. For voting rules that compute score for alternatives, we use the expected number of mistakes in random tie-breaking: 1/2 a mistake if two alternatives are tied for first place, and 2/3 a mistake if three alternatives are tied for first place, etc.[4]

Figure 2a plots the first-place mistakes computed over random subsets, and averaged over 40 preference profiles for 4 common voting rules. It also includes the average error for the global (ex-post) optimal positional scoring rule, which minimizes the number of first-place mistakes given knowledge of the true winner, and the average cross-validation error for our trained positional scoring rule.

The mean first-place mistakes for all rules increases as the minimal distance from the goal state of a sequence increases. This confirms our premise that varying the distance to the goal state changes the noisiness of the votes collected. According to Figure 2a, Borda consistently has among the highest number of first-place mistakes, while Kemeny consistently has the lowest number of first-place mistakes and beats other rules by a relatively large margin. Strikingly, Kemeny even outperforms the (omniscient) global optimal scoring rule.

We performed two-sided paired t-tests on the average first-place mistakes over 40 preference profiles by pairing the average number of mistakes by different voting rules on the same profiles. The difference in first-place mistakes between Kemeny and any other rule is highly statistically significant ($p < 0.003$) for all four sequences. Table 1 contains the pairwise p-values for each voting rule we tested, corresponding to the different values along the x-axis in Figure 2a.

p-values	Plurality	Borda	Kemeny	Maximin	Glob.Opt	Train
Plurality		0.9266	<0.0001	0.3700	0.0149	0.9551
Borda	0.1028		<0.0001	0.2234	<0.0001	0.7643
Kemeny	0.0021	<0.0001		<0.0001	0.0001	<0.0001
Maximin	0.8176	0.0246	<0.0001		0.0066	0.1670
Global Opt.	0.1175	0.0005	0.0007	0.0981		<0.0001
Trained	0.5321	0.0256	0.0001	0.9102	<0.0001	

(a) $d = 5$ (lower triangle) and $d = 7$ (upper triangle)

p-values	Plurality	Borda	Kemeny	Maximin	Glob.Opt	Train
Plurality		0.7212	0.0002	0.6856	0.0177	0.8858
Borda	0.5810		<0.0001	0.9656	<0.0001	0.0134
Kemeny	<0.0001	<0.0001		<0.0001	0.0002	<0.0001
Maximin	0.7064	0.5992	<0.0001		0.0001	0.3312
Global Opt.	0.0115	0.0003	0.0002	0.0005		<0.0001
Trained	0.4858	0.7681	<0.0001	0.6854	<0.0001	

(b) $d = 9$ (lower triangle) and $d = 11$ (upper triangle)

Table 1: Table of p-values for paired, two-sided t-tests on first-place mistakes averaged over 100 samples, corresponding to x-values in Figure 2a.

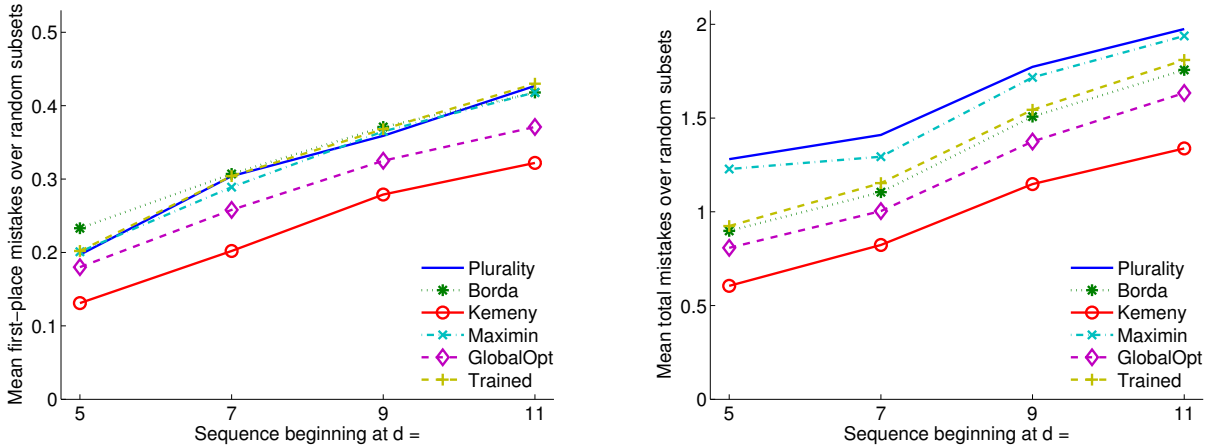
All mistakes/Kendall-tau distance. To evaluate how well voting rules perform as social welfare functions, we are interested in how the aggregate ranking stacks up against the ground truth. The standard, most natural way to measure the difference between two rankings is using the Kendall-tau distance, given in Equation (1). The Kendall-tau distance between two rankings is simply the total number of pairs on which the rankings disagree, or when compared to the ground truth, the number of pairwise mistakes. Once again, count the number of mistakes for ties in rankings as 1/2, equal to the expected value for random tie-breaking.

Figure 2b shows the mean Kendall-tau distance between the voting rules we tested and the ground truth. It also shows the globally optimal positional scoring rule that ex-post minimizes the total number of mistakes given all preference profiles, as well as the average number of mistakes when trained using cross-validation. Once again, we see that the total number of mistakes for all voting rules rise as the sequence moves further from the goal, confirming that we can control the level of noise in this way.

Similarly to the analysis of first-place mistakes, we see that Kemeny definitively beats all other voting rules, and even the global optimal positional scoring rule. Moreover, the stratification of the voting rules stays the same even as the noise levels change, so that the voting rules perform the same relative to each other at different noise levels.

We omit the table of statistical significance for this case,

[4]Ties between rankings under Kemeny, which are rare, are broken arbitrarily.



(a) First-place mistakes.

(b) Total mistakes (Kendall-tau distance to ground truth.)

Figure 2: First-place and total mistakes at different noise levels using random subset averaging.

as it is similar to the previous one (in fact the p-values are in general even smaller, and distinguish between any rules that are visibly separate in Figure 2b).

Observations and user heuristics. Our entire dataset, including the initial exploratory data, consists of 3,435 individual rankings from 789 unique voters.[5] At 10 cents per ranking, this was an extremely economical way to collect a large amount of data from a very diverse population. This large amount of data allowed us to obtain very precise results when combined with bootstrap sampling and averaging.

We collected data from users about how they approached comparing puzzles. While the individual heuristics users used do not affect how the voting rules perform, we thought it was rather interesting how approaches were different. Some results were close to spam. As expected, the majority of users compared puzzles mentally and did the task quickly, but many also tried to solve the puzzle on pencil-and-paper, or went even further by cutting out the 8-puzzle on paper and sliding the pieces around. Some computed what was essentially the Manhattan distance heuristic for each puzzle (we point out that this is admissible and often used for A^* search). A few workers actually went all the way and wrote programs to solve each puzzle and entered in the minimum number of moves in their comments, even though this was far beyond what we requested. These varying user effort levels are only natural when it comes to human computation settings.

6 Discussion

Our results indicate that in realistic human computation settings, sophisticated voting rules—especially Kemeny—are considerably more accurate in aggregating people's noisy estimates compared to naïve voting rules such as plurality. Note that our results focus on the relative performance of different voting rules, not on whether we were able to solve 8-puzzles well in absolute terms. Although our experimental platform is specific to 8-puzzles (for reasons discussed above), we believe that the implications of our results are quite general for voting in any noisy setting. At the very least, we argue that designers of human computation systems that involve voting should be aware of the different options, and consider ranked voting and more advanced voting rules.

As an alternative to our approach, one can consider employing machine learning techniques (Adali, Magdon-Ismail, and Marshall 2007; Sculley 2007; Welinder et al. 2010). For example, given enough data, it may be possible to learn the parameters of the noise model (Lu and Boutilier 2011), and analytically pinpoint the alternative or ranking that is most likely to be correct. This approach requires a significant amount of data, work, knowledge, and computational resources. In contrast, given that a human computation system already employs voting, switching between voting rules is almost effortless.

Long-term implications. Despite its great appeal, many economists seem to believe that social choice theory has not achieved its full potential. One of the main reasons is that it is extremely difficult to change a political system, even to an alternative system that is clearly superior. A recent example is the referendum held in the UK in 2011 on whether to change the voting system (from plurality to a preferential system called IRV—instant-runoff voting). Although most academics agree that IRV is superior to plurality voting, the successful No campaign played on the unpopularity of Nick Clegg—the leader of the Liberal Democrats—who would have benefited politically from changing the system.

Therefore, human computation systems provide an exciting testbed for the principles of social choice. Indeed, the designer of a human computation system is free to employ any method of collective decision making. By informing the construction of real-world systems, the scope and impact of

[5] We are happy to share this collected preference data for others who would like to test social choice and rank aggregation methods.

social choice theory may greatly increase.

We also believe that the study of human computation can inform social choice theory, by motivating fundamentally new theoretical models. For example, voting in EteRNA is special in that the alternatives are situated in a metric space, and the quality of close alternatives is correlated. Hence, when a set of alternatives is selected, it is often best to avoid selecting two close alternatives. Even if this consideration is reflected in each individual vote, a naïve aggregation of the votes (using k-approval, as is the case today) can easily select close alternatives.

To summarize, we envision a future where there is a mutually beneficial interaction between (computational) social choice and human computation. We hope that our work, in addition to its more immediate practical implications, will spark this long-term interaction between the two fields.

Acknowledgements.

This material is based upon work supported by NSF Grant No. CCF-0915016 and Xerox Foundation. We would like to thank Haoqi Zhang for his implementation of the 8-puzzle in Javascript, which we modified for our experiments.

References

Adali, S.; Magdon-Ismail, M.; and Marshall, B. 2007. A classification algorithm for finding the optimal rank aggregation method. In *Proceedings of the 22nd International Symposium on Computer and Information Sciences (ISCIS)*, 1–6.

Brandt, F.; Fischer, F.; Harrenstein, P.; and Mair, M. 2008. A computational analysis of the Tournament Equilibrium Set. In *Proceedings of the 23rd AAAI Conference on Artificial Intelligence (AAAI)*, 38–43.

Carterette, B.; Bennett, P. N.; Chickering, D. M.; and Dumais, S. T. 2008. Here or there: Preference judgments for relevance. In *Proceedings of the 30th European Conference on Information Retrieval (ECIR)*, 16–27.

Conitzer, V., and Sandholm, T. 2005. Common voting rules as maximum likelihood estimators. In *Proceedings of the 21st Annual Conference on Uncertainty in Artificial Intelligence (UAI)*, 145–152.

Conitzer, V.; Davenport, A.; and Kalagnanam, H. 2006. Improved bounds for computing Kemeny rankings. In *Proceedings of the 21st AAAI Conference on Artificial Intelligence (AAAI)*, 620–626.

Conitzer, V.; Rognlie, M.; and Xia, L. 2009. Preference functions that score rankings and maximum likelihood estimation. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI)*, 109–115.

Conitzer, V.; Sandholm, T.; and Lang, J. 2007. When are elections with few candidates hard to manipulate? *Journal of the ACM* 54(3):1–33.

Conitzer, V. 2006. Computing Slater rankings using similarities among candidates. In *Proceedings of the 21st AAAI Conference on Artificial Intelligence (AAAI)*, 613–619.

Cooper, S.; Khatib, F.; Treuille, A.; Barbero, J.; Lee, J.; Beenen, M.; Leaver-Fay, A.; Baker, D.; and Popović, Z. 2010. Predicting protein structures with a multiplayer online game. *Nature* 466:756–760.

Dai, P.; Mausam; and Weld, D. S. 2010. Decision-theoretic control of crowd-sourced workflows. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence (AAAI)*, 1168–1174.

Dwork, C.; Kumar, R.; Naor, M.; and Sivakumar, D. 2001. Rank aggregation methods for the web. In *Proceedings of the 10th International World Wide Web Conference (WWW)*, 613–622.

Faliszewski, P., and Procaccia, A. D. 2010. AI's war on manipulation: Are we winning? *AI Magazine* 31(4):53–64.

Forsythe, R.; Rietz, T.; Myerson, R.; and Weber, R. 1996. An experimental study of voting rules and polls in three-candidate elections. *International Journal of Game Theory* 25(3):355–383.

Hemaspaandra, E.; Hemaspaandra, L. A.; and Rothe, J. 1997. Exact analysis of Dodgson elections: Lewis Carroll's 1876 voting system is complete for parallel access to NP. *Journal of the ACM* 44(6):806–825.

Little, G.; Chilton, L. B.; Goldman, M.; and Miller, R. C. 2010a. Exploring iterative and parallel human computation processes. In *Proceedings of the 2nd Human Computation Workshop (HCOMP)*.

Little, G.; Chilton, L. B.; Goldman, M.; and Miller, R. C. 2010b. Turkit: Human computation algorithms on Mechanical Turk. In *Proceedings of the 23rd Annual ACM symposium on User interface software and technology (UIST)*, 57–66.

Lu, T., and Boutilier, C. 2011. Learning Mallows models with pairwise preferences. In *Proceedings of the 28th International Conference on Machine Learning (ICML)*, 145–152.

Procaccia, A. D., and Rosenschein, J. S. 2007. Junta distributions and the average-case complexity of manipulating elections. *Journal of Artificial Intelligence Research* 28:157–181.

Sculley, D. 2007. Rank aggregation for similar items. In *Proceedings of the 7th SIAM International Conference on Data Mining (SDM)*, 587–592.

Shahaf, D., and Horvitz, E. 2010. Generalized task markets for human and machine computation. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence (AAAI)*, 986–993.

von Ahn, L., and Dabbish, L. 2008. Designing games with a purpose. *Communications of the ACM* 51(8):58–67.

Welinder, P.; Branson, S.; Belongie, S.; and Perona, P. 2010. The multidimensional wisdom of crowds. In *Proceedings of the 24th Annual Conference on Neural Information Processing Systems (NIPS)*, 2424–2432.

Young, H. P. 1988. Condorcet's theory of voting. *The American Political Science Review* 82(4):1231–1244.