Analyzing Crowd Rankings

Julia Stoyanovich
Drexel University
stoyanovich@drexel.edu

Marie Jacob
University of Pennsylvania
majacob@cis.upenn.edu

Xuemei Gong
Drexel University
xg45@drexel.edu

ABSTRACT

Ranked data is ubiquitous in real-world applications, arising naturally when users express preferences about products and services, when voters cast ballots in elections, and when funding proposals are evaluated based on their merits or university departments based on their reputation. This paper focuses on crowdsourcing and novel analysis of ranked data. We describe the design of a *data collection* task in which Amazon MT workers were asked to rank movies. We present results of data analysis, correlating our ranked dataset with IMDb, where movies are rated on a discrete scale rather than ranked. We develop an intuitive measure of *worker quality* appropriate for this task, where no gold standard answer exists. We propose a model of *local structure* in ranked datasets, reflecting that subsets of the workers agree in their ranking over subsets of the items, develop a data mining algorithm that identifies such structure, and evaluate in on our dataset. Our dataset is publicly available at `https://github.com/stoyanovich/CrowdRank`.

1. INTRODUCTION

Ranked data is ubiquitous in real-world applications, arising naturally when users express their preferences about products and services, when voters cast ballots in elections, and when funding proposals are evaluated based on their merits or university departments based on their reputation.

A ranking is a statement about the *relative* quality or relevance of the items being ranked. Rankings are intrinsically different from ratings, and have several advantages that make this kind of data valuable. A user who rates movies "Birdman", "Selma" and "Whiplash" with 5 out of 5 stars may still rank "Birdman" first, followed by "Selma" and then by "Whiplash", stating a finer-grained preference. At the same time, ranking is a very *natural and fun* way for consumers to provide feedback, i.e., it allows to specify *finer-grained preferences* without adding much to the cognitive load. Another argument in favor of rankings is that two rankings of the same set of items are *directly comparable*, even when issued by two different users. This may not be the case for ratings, in that some users are more conservative than others, and a normalization step may be required to make ratings comparable across users. In other words, when two users say that they prefer "Birdman" to "Selma", they unambiguously state the same preference. On the other hand, if two users give the same star rating to "Selma", they may not have the same preference, because one person's 5 stars is not necessarily the same as another person's 5 stars. Finally, certain domains such as preferential voting elections make use of only rankings rather than ratings. All these reasons argue for using rankings in data analysis, in addition to, or instead of, the more commonly used ratings data.

WebDB'15, May 31, 2015, Melbourne, VIC, Australia
Copyright 2015 ACM 978-1-4503-3627-7/15/05 ...$15.00
http://dx.doi.org/10.1145/2767109.2767110

This paper focuses on crowdsourcing and novel analysis of ranked data. We first describe the reasons why crowdsourcing this data is useful, and then discuss the challenges in the collection and analysis of ranked data.

Why crowdsource? Ranked preferences are interesting and can be used in many scenarios in which other types of user preferences (e.g., ratings) are useful. Rankings may be combined (or aggregated) to facilitate analysis. For example, aggregated opinions of Computer Science faculty may serve as a basis for a ranking of CS departments, while aggregated opinions of users about products or services may be used for recommendation, or in support of data exploration.

Modeling and analysis techniques are typically evaluated using synthetic and real datasets, including, e.g., APA [11], Meath [14] and Sushi [19]. These datasets are relatively small in terms of the number of items and rankings, and so cannot serve as basis for the development of truly scalable analysis techniques, which would be applicable to datasets with hundreds or even thousands of items. Certain other datasets, e.g., MovieLens (`http://www.movielens.org`) and IMDb (`http://www.imdb.com`), where movies are rated by users on a discrete scale, can be transformed into rankings, but this transformation may introduce a bias.

Local structure in ranked data. In [26], we hypothesized that rankings exhibit local structure, reflecting agreement of subsets of the judges with respect to subsets of the items. Further, we argued that aggregation of rankings is most meaningful in presence of structure (i.e., of agreement), and, conversely, that structure must be identified before meaningful aggregation can take place. Intuitively, an aggregated ranking is only meaningful if it is representative of the rankings it aggregates. Local structure refers to the following three properties:

(1) *Domain diversity*: Different sets of items may be ranked by different groups of users. For example, one group of users

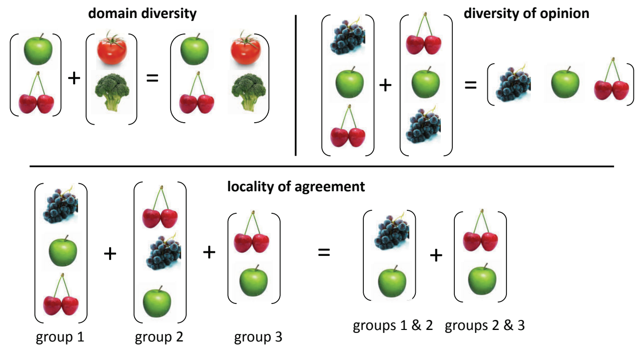

Figure 1: An illustration of local structure in ranked data, with fruits and vegetables representing items.

may rank only action movies, while another may rank only foreign dramas.

(2) *Diversity of opinion*: Even when the same set of items is ranked by two groups of users, these groups may express conflicting opinions over the items, e.g., ranking them in the opposite relative order.

(3) *Locality of agreement*: Users may rank some items in common, but not others, and they may rank some items similarly, giving rise to a consensus ranking, and others — dissimilarly, producing divergent rankings.

Figure 1 illustrates local structure in ranked data, and the related issues when ranked data is aggregated, with fruits and vegetables representing items.

What are the challenges? Several issues make crowdsourcing and analysis of ranked data challenging.

Size and sparseness. Typical applications deal with hundreds or even thousands of items. For example, there are over $10,000$ movies in the MovieLens 10M dataset. This results in a potentially intractable space of possible rankings, since there are $m!$ possible ways to order m items. Furthermore, because the space is so large, we expect the set of actual observations to be sparse. We design our data collection task to control for sparseness while still amassing user judgments for a reasonably large set of movies, albeit a smaller one than in the MovieLens datasets (Section 2).

Incompleteness. Rankings will often be incomplete, i.e., a ranking often includes a (small) subset of all available items. Rankings are incomplete for two reasons. First, a full set of preferences may not be feasible to collect — a person who has not seen the movie “Lincoln” cannot rank it relative to other movies. Second, not all items may be directly comparable — a person may prefer to give one ranking for action movies and another for romance, but cannot compare action movies to romance movies. We design our data collection task so as to allow users to specify incomplete rankings (Section 2). Our data mining methods are likewise designed to work with incomplete rankings (Section 4).

Subjectiveness. Many ranked datasets consist of inherently subjective opinions, i.e., it is usually the case that no ground truth ranking exists. This presents a challenge for estimating worker quality in a crowdsourcing scenario. We propose novel measures of worker quality that exploit the properties of the transitive closure of pairwise preferences for a particular worker (Section 3).

Contributions and roadmap. We describe the design of a *data collection* task in which Amazon MT workers were

dataset	# rankings	# items	items per ranking
CrowdRank	5000	471	up to 20
Sushi-a	5000	10	10
Sushi-b	5000	100	up to 10
APA	15449	5	up to 5
Meath	64081	14	up to 5
Idea	98	5	5
Car	279	4	4

Table 1: Properties of ranked datasets.

asked to rank movies, and present results of data analysis (Section 2). We develop an intuitive measure of *worker quality* appropriate for crowdsourcing ranked data in domains where no gold standard exists (Section 3). We propose a model of *local structure* in ranked datasets, reflecting that subsets of the workers agree in their ranking over subsets of the items. We develop a data mining algorithm that identifies such structure, and show that such structure indeed holds in the dataset we collected (Section 4). We discuss related work in Section 5, then discuss future work and conclude in Section 6.

2. DATA COLLECTION AND ANALYSIS

We collected a dataset consisting of 5,000 rankings of 471 distinct movies, given by 351 users. We will refer to the resulting dataset as CrowdRank in the remainder of the paper. While moderate in size, CrowdRank is, to our knowledge, the largest currently available ranked dataset. Other datasets include Sushi [19], Meath [14], APA [11], Idea [12], and Car [21]. Table 1 summarizes basic statistics about CrowdRank and other datasets.

2.1 Data collection methodology

CrowdRank was collected using Amazon MT during a consecutive 1-month period in 2013. We published 50 Human Intelligence Tasks (HITs), each consisting of 20 movies, and collected 100 responses per HIT. Workers were instructed to only rank the movies that they have watched and feel comfortable ranking. Workers were compensated based on the number of HITs they complete, not based on the number of movies they rank, at the rate of $0.50 per HIT. HITs with fewer than 3 ranked movies were rejected.

Our HITs were open to high-quality Amazon MT workers, those with an overall approval rate of at least 80% and with at least 1,000 Amazon MT HITs already completed. We collected the following demographic information about the workers: gender, age, highest completed education level, and frequency with which they watch movies.

Each HIT belonged to one of 5 categories, with 10 HITs per category: top-200, director, genre, year of release (from 2004 through 2013) and random (consisting of a randomly selected set of movies). The type of a HIT was not announced to the user. Users were simply presented with a set of movies, and it was up to them to realize whether or not these movies shared a common theme.

Movies were assigned to the HITs in the following way. First, we selected 1000 movies from IMDb so as to cover the above categories. We then refined this initial set, by allowing movies to be reused across HITs. Our goal was to have sets of movies repeat, which would in effect solicit repeated rankings of certain movies from the same user. To accomplish this, we introduced the notion of a *block of movies* —

a set of movies that agree on values for one or several attributes, including genre, year of release, director and popularity (top-200 or not). For each HIT, we randomly selected which blocks to assign to it from among those that have the right attribute values, subject to HIT size being fixed at 20 movies. For example, a HIT in the genre category, with $genre = crime$ is comprised of 4 blocks of movies for which crime is one of the genres.

We manually generated 144 blocks, each consisting of at least 2 and at most 10 movies, and with an average of 4.5 movies per block. Blocks may overlap in terms of movies, and are reused across HITs. A HIT is made up of between 2 and 10 blocks. Importantly, the same block may be reused within a category of HITs and across categories. When selecting which blocks to reuse, we were careful to limit the number of HITs in which a particular movie is used, so as not to bias our data mining results (described in Section 4). A total of 471 distinct movies are used in our dataset, across all HITs. We randomly reordered movies in each HIT to make blocks less apparent. Given a HIT, all users were presented with movies in that HIT in the same order.

Our design of the HITs allowed us to address two issues. First, it helped control *data sparsity.* That is, although the same exact ranking of all movies is unlikely to be observed more than once, smaller subsets of movies, especially those repeating across HITs, are likely to repeat both for a particular user and across users. Second, our design allowed us to quantify *worker quality.* In a nutshell, a worker who does multiple HITs is likely to rank certain pairs relative to each other multiple times. This allows us to compute a measure of consistency for the particular worker, based on the number of conflicting rankings. We will discuss this point in detail in Section 3.

User engagement. Users spent an average of 5 minutes per HIT, earning an average of \$5.89 per hour. A worker completed 14.2 HITs and ranked 70.0 distinct movies on average. The average length (number of movies) of a ranking is 7.3. We observed no correlation between the number of HITs a user submitted and the number of movies a user ranked per HIT. This is encouraging, as it signals that worker commitment does not degrade over time in our data collection. We released our HITs in 4 batches over a 1-month period, and over half of the users came back to do several batches.

While remuneration is the primary incentive for Amazon MT users, we believe that the nature of the task provided an additional incentive. In fact, we received numerous unsolicited emails, saying how much users enjoyed the HITs, including: *Very fun HITs, wish I could've done more. Thanks so much! Loved the HITs! Thanks for the work!*

2.2 Correlations with IMDb

CrowdRank is of moderate size, but is the largest publicly available ranked dataset that we are aware of. One may argue that large datasets of ratings are available, and that these may be converted to rankings. We will argue against such an approach in this section. We considered the full CrowdRank dataset, and the portion of the IMDb dataset that includes 471 movies that appear in CrowdRank. For ease of exposition, we will refer to this portion of IMDb simply as "IMDb", but stress here that the analysis was not on the full IMDb dataset, but only on its relevant portion.

We looked at the relationship between movie quality and popularity, both within a dataset and between datasets. In IMDb, movie quality is captured by the attribute *average score*, a decimal number ranging between 1 and 10 (higher is better). Popularity is captured in IMDb by the attribute *votes*, which represents the number of votes the movie received. In CrowdRank, movie quality is captured by its rank, which, if averaged among all users, will produce *average rank*, a decimal number ranging between 1 and 20 (lower is better). Popularity of a movie in CrowdRank cannot be measured directly by its observed frequency, since the number of rankings in which a movie appears strongly depends on the number of HITs that included the movie (i.e., on the design of our data collection task, see Section 2). For this reason, we use *normalized frequency* to represent popularity, normalizing the number of times a movie was ranked by the number of HITs in which it appeared.

We used Pearson's correlation coefficient (r) and Spearman's rank correlation coefficient (r_s). Both measures vary from -1 to 1, with values close to 1 signifying strong positive correlation, values close to -1 signifying strong negative correlation, and values close to 0 signifying lack of correlation. We made the following observations.

1. There is a positive correlation between popularity in CrowdRank and in IMDb: $r = 0.760$ and $r_s = 0.845$. Thus popularity of movies among our users was consistent with that of IMDb, which represents a far larger sample of the population.
2. There is a positive correlation between quality and popularity in IMDb: $r = 0.620$ and $r_s = 0.759$. This is expected: more popular a movie, i.e., the more frequently it is voted on, the higher its average rating.
3. There is a negative correlation between average rank (complement of quality) and popularity in CrowdRank: $r = -0.623$ and $r_x = -0.706$. This is expected, and is in-line with the finding for IMDb. The more popular a movie, i.e., the more frequently it is ranked (subject to normalization), the lower its average rank.
4. There is a moderate negative correlation between CrowdRank average rank and IMDb average score: $r = -0.395$ and $r_s = -0.403$, i.e., the higher a movie's IMDb score the lower its CrowdRank rank.

The most interesting statistic we observed was the moderate correlation between average score and average rank. This result supports our conjecture that rankings and ratings capture related but different preferences. Further, it argues for the need to collect ranked datasets directly, in support of modeling and analysis of this type of data. An alternative of converting ratings data to rankings is not ideal, since the resulting data will not have the same statistical properties as if rankings were collected directly.

In summary, our analysis of CrowdRank data shows that this dataset exhibits some interesting properties. The relationship between movie popularity and quality in CrowdRank is in-line with that observed in IMDb. Popularity of a movie in CrowdRank agrees with that in IMDb, while average rank of a movie in CrowdRank shows only a moderate correlation with average score in IMDb.

3. MEASURING WORKER QUALITY

In this section we address a question that is central to crowdsourcing, that of estimating worker quality. This task is difficult in our scenario in that, because we are collecting user opinions, which are inherently subjective, there is no gold standard. Nonetheless, worker quality is relevant here, and by it we understand not whether the user is competent but rather whether he is performing due diligence.

We describe several measures of worker quality in the remainder of this section that are based on *inconsistencies of pairwise preferences*. Recall from Section 2 that, because of the way our data collection was designed, movies may repeat across HITs. Consider movies a, b and c, and suppose first that the pair a, b appears in two or more HITs. Intuitively, a high-quality worker who thinks carefully about his preferences will rank these movies in the same relative order whenever they occur together in a HIT. If this is not the case, i.e., if the worker ranks $a \mid b$ in one HIT and $b \mid a$ in another, we observe an inconsistency. Another kind of an inconsistency that can arise is when a worker ranks $a \mid b$, $b \mid c$ and $c \mid a$. (These pairwise preferences would be extracted from two or more HITs submitted by the same worker.) Here, while no pairs of preferences are in direct conflict, an inconsistency exists over the transitive closure of the pairwise preferences.

We will discuss our proposed ways of quantifying worker quality shortly. But before diving into the details of quality measures, we observe that a certain amount of inconsistencies is expected to occur in ranked data, and does not immediately disqualify the worker. Inconsistencies may arise due to two reasons. First, our HITs ask workers to give a total (as opposed to partial) ordering among the movies. An inconsistency may signal a tie — a case where a user does not have a clear preference between two movies. Second, and perhaps even more interestingly, inconsistencies may arise because preferences are context-sensitive. Specifically, a user may prefer one item to another based on one feature (e.g., action movie), and have the opposite preference based on another feature (e.g., a movie by Steven Spielberg). Our data collection method is conducive to context playing a role in how users rank movies. This is because HITs are set up based on a specific feature of the movies (genre, director, popularity). While this is not explicitly stated, it will often become apparent to the user. These points are discussed further in Section 6, in the context of ongoing and future work.

We now formally define our measures of worker quality for worker w. Consider a dataset in which items from the set I of cardinality m are being ranked. We use permutations to represent rankings.

Definition 3.1 (Permutation). *A* permutation π *is a bijective function* $\pi : I \to \{1, \ldots, m\}$*, associating with each item a rank* $\pi(i)$*.* $\pi(i)$ *denotes the rank of item* i *and* $\pi^{-1}(i)$ *denotes the item at rank* i*.*

Definition 3.2 (Preference graph). *Consider a worker* w*, with a corresponding set of permutations* R*. A* preference graph $G = (V, E)$ *is a directed graph, such that: (1)* $\forall i \in V$*,* i *appears in some* $\pi \in R$*; and (2)* $\forall e = (i, j) \in E$*,* i *and* j *appear in some* π *and* $\pi(j) = \pi(i) + 1$*.*

An example of a set of rankings R and of the corresponding preference graph G is given in Figure 2. There is a directed edge from i to j in G if i appears immediately before j in some ranking. More generally, there is a path from i to j in G if i is preferred to j. Such a preference may be directly observed, i.e., it may be derived from a single ranking in which both i and j appear, as is the case with the path from a to c. Alternatively, a path may be derived

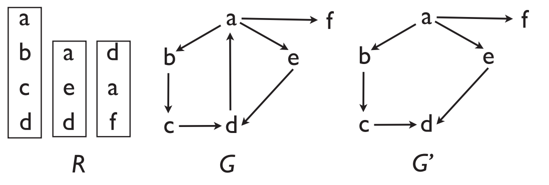

Figure 2: Example of a preference graph.

transitively from multiple rankings, as is the case with the path from e to f.

Definition 3.3 (Inconsistent pair). *Consider a preference graph* $G = (V, E)$*. We say that a pair of items* i, j *is* inconsistent *if there exists a path from* i *to* j *and a path from* j *to* i *in* G*. We denote by* Υ *the set of inconsistent pairs of* G*.*

We observe several inconsistencies in the graph in Figure 2. The cycle $a \to b \to c \to d \to a$ contributes $\binom{4}{2} = 6$ inconsistent pairs, and the cycle $a \to e \to d \to a$ contributes $\binom{3}{2} = 3$, for a total of 8 distinct inconsistent pairs. We use the number of inconsistent pairs to define our first worker quality measure.

Definition 3.4 (Inconsistency). *Consider a preference graph* $G = (V, E)$*, with the corresponding set of inconsistent pairs* Υ*. The* normalized inconsistency *of* G *is* $q_i = |\Upsilon| / \binom{|V|}{2}$*.*

To make the inconsistency score comparable across workers, we normalize it by the total number of possible item pairs. Normalized inconsistency of G in Figure 2 is $q_i = \frac{8}{15}$.

Among 351 workers in the CrowdRank dataset, 175 (50%) had $q_i = 0$, 150 (43%) had $0 < q_i \leq 0.2$, and 26 (7%) of the workers had $q_i > 0.2$ (with $q_i = 0.66$ as the maximum). Based on this distribution, one alternative is to take 0.2 as a threshold and remove contributions of workers whose normalized inconsistency score is above this threshold from the dataset. However, this still leaves a large fraction of the data with some inconsistencies, which make analysis of ranked data challenging. To address this issue, we developed a data cleaning procedure that removes cycles from preference graphs. This procedure also gave rise to another natural worker quality measure, based on the idea of *repairs*.

Definition 3.5 (Repair, Normalized repair size). *Consider a preference graph* $G = (V, E)$*, with the corresponding* $\Upsilon \neq \emptyset$*. A* repair *of* G *is a directed acyclic graph* $G' = (V', E')$*, with* $V' = V$*,* $E' \subset E$ *and* $\Upsilon' = \emptyset$*. For the given* G*,* G'*, the* normalized repair size *is* $q_r = (|E| - |E'|)/|E|$*.*

A repair removes some subset of the edges of G so as to make G cycle-free. Clearly, removing any edge that participates in a cycle will break that cycle. For example, we could remove edges $b \to c$ and $e \to d$, making the graph in Figure 2 cycle-free, which would result in normalized repair size $q_r = \frac{2}{7}$. Other repairs of G are possible, including a particular repair that has a lower size: removing the edge $d \to a$ will break both cycles in G, resulting in a normalized repair size of $q_r = \frac{1}{7}$. The latter is known as a minimal repair [5, 8]. It is easy to show that, while a minimal repair may not be unique, all minimal repairs have the same size. Finding a minimal repair is NP-hard (it reduces to the problem of finding the feedback arc set of a directed graph). We use a greedy heuristic to find a repair that has cost close to that of a minimal repair.

To construct a repair, we remove edges in decreasing order of the number of cycles in which they participate, and stop when the graph becomes cycle-free. We ran this procedure on our dataset and found that removing 1 edge repaired the preference graph for 25 (7%) workers, removing between 2 and 9 edges repaired the graph for 100 (28%) workers, while 51 (15%) workers required the removal of 10 or more edges (115 was the maximum number of edges to be removed). We found that normalized repair size and normalized inconsistency score are strongly correlated, Pearson's $r = 0.830$.

In summary, while no gold standard exists, we can still quantify worker quality in our setting, which we do by using the properties of the transitive closure of preferences. We described a data cleaning procedure, which is used before the data is mined. We describe our data mining task and its results in the next section.

4. IDENTIFYING LOCAL STRUCTURE

We argued that ranked data exhibits *local structure*, reflecting agreement of different, possibly overlapping, subsets of workers over different, possibly overlapping, subsets of items. We now describe more precisely what we mean by local structure, and show how it can be mined.

Our starting point is a set $\mathcal{G} = \{G_1, \ldots, G_n\}$ of preference graphs (see Definition 3.2) that have been repaired, and so are cycle-free. Each graph represents the transitive closure of pairwise preferences of a particular worker. Consider items i, j, k. If a worker has a preference regarding how these items should be ranked relative to each other, then there will exist exactly one path in G with i, j, k occurring along that path, but not necessarily consecutively. Further, i, j, k may occur in different orders in different graphs in $\mathcal{G}$. We capture these concepts in the following two definitions.

DEFINITION 4.1 (ITEMSET SUPPORT). *Consider a set of preference graphs $\mathcal{G}$ and an itemset A.* Support set *of A in $\mathcal{G}$, denoted $suppSet(A, \mathcal{G})$ is the set of graphs in which all items in A occur along some path. We refer to the size of $suppSet(A, \mathcal{G})$ as* support *of A.*

For example, G' in Figure 2 is in the support set of $\{a, c, d\}$ but it is not in the support set of $\{a, e, f\}$. Next we define the support of a permutation.

DEFINITION 4.2 (PERMUTATION SUPPORT). *Consider a set of preference graphs $\mathcal{G}$ and a permutation π.* Support set *of π in $\mathcal{G}$, denoted $suppSet(\pi, \mathcal{G})$, is the set of graphs G such that, $\forall i, j \in items(\pi)$, if $\pi(i) < \pi(j)$ then there is a path from i to j in G. We refer to the size of $suppSet(\pi, \mathcal{G})$ as* support *of π.*

For example, graph G' in Figure 2 is in the support set of $a|c|d$ but not in the support set of $a|d|c$. Typically, we would compute the support of a permutation π w.r.t. $suppSet(items(\pi), \mathcal{G})$.

Let us quickly revisit the way in which preference graphs represent local structure. The fact that different sets of items are ranked by different workers (and so appear as nodes in the respective preference graphs) corresponds to the *domain diversity* property of local structure. The fact that certain items, say, a, b, c, occur in different relative order corresponds to *diversity of opinion.* Finally, a situation where G_i and G_j both support $a|b|c$, while G_i and G_k both support $b|d|e$ corresponds to *locality of agreement.*

4.1 Identifying local structure

We use a bottom-up procedure in the style of Apriori [3] to find permutations over which there is agreement. We use a frequency threshold to specify itemset support (per Definition 4.1). If an itemset has sufficient support, we go on to evaluate the support of its permutations. While specifying a frequency threshold for permutation support would have been reasonable, we opted instead for a measure based on entropy, because this allows to set the threshold as a function of permutation size.

Recall that entropy of the random variable X is calculated as $H(X) = -\sum_i p_i log(p_i)$ where p_i is the probability that $X = i$. We assume a non-parametric probability distribution over the permutations, use $supp(\pi, suppSet(items(\pi), \mathcal{G})$ to estimate this probability, and then compute the entropy of this estimated probability distribution. We refer to this measure as *permutation entropy.* For example, suppose that $supp(\{a, b, c\}, \mathcal{G}) = 100$. We denote $suppSet(\{a, b, c\}, \mathcal{G}) = \mathcal{G}'$ for convenience. Suppose now that the following support was observed for the different permutations of $\{a, b, c\}$ in $\mathcal{G}'$: $supp(a|b|c, \mathcal{G}') = 50$, $supp(a|c|b, \mathcal{G}') = \ldots = supp(c|b|a, \mathcal{G}') = 10$. We use permutation support to estimate the probability distribution of the $3! = 6$ permutations of $\{a, b, c\}$, and compute its entropy.

We set the threshold for our entropy-based measure as a percentage of the maximum possible entropy value for the given number of outcomes of the random variable X. Specifically, there are $m!$ possible outcomes and the threshold is computed as $t = \alpha * log(m!)$. Since entropy is measured in the number of bits taken to encode the distribution, we look for samples with a probability distribution that could be encoded in as a few bits as possible.

The entropy-based measure above can be used in scope of a bottom-up mining procedure because it exhibits downward closure, which ensures that, as more items are added to an itemset, the entropy of the probability distribution over their permutations does not decrease. Precise statement of this claim and its proof are omitted due to lack of space.

4.2 Identifying user populations

The mining process described above will produce a set of interesting permutations over which there is agreement among a set of users. As the final step, we post-process this data, with the goal of grouping several interesting permutations by population, i.e., of identifying sets of users who agree on how they rank sets of items, together with these items. Note that it is important to allow the sets of users and the sets of items to overlap. For example, we may output that users Ann, Bob, Cat, and Dan rank "Fargo" | "The Big Lebowski" | "No Country for Old Men" (directed by the Coen brothers) and "Fargo " | "The Departed" | "The Big Lebowski" | "Goodfellas" (crime), while Ann, Mary and Sue rank "Life of Pi" | "Django Unchained" (adventure).

To identify user populations we build an undirected graph in which nodes represent users and edges represent agreement on at least one ranking. Maximal cliques in this graph are candidate user population, and become actual user populations if users in the clique have at least one permutation in common. More sophisticated ways of analyzing the graph of users and their agreement are possible, e.g., finding overlapping clusters in the bipartite graph with one set of nodes representing users and the other set of nodes representing rankings. Generally, any reasonable community detection approach is applicable here, as long as it allows for communities to overlap.

4.3 Results

We mined the cleaned (cycle-free) CrowdRank dataset using the procedure described above. Interesting permutations, together with the sets of workers who agree on these permutations, were identified with itemset support threshold set to 5 (1.7% of the workers) and parameter of the entropy threshold α set to 0.7. We tuned these values manually so as to get enough interesting permutations, but not an overwhelming number. In the future we hope to move away from explicit thresholds and work with more robust statistical measures once the properties of the probabilistic process underlying this data are better understood.

Using the thresholds above, we identified 6197 itemsets for which the entropy of the estimated probability distribution was sufficiently low. Of these, 5501 itemsets were of size 2, 693 were of size 3 and 3 were of size 4.

Interestingly, we noticed an output sample of permutations showing a relatively diverse set of opinions among users who ranked the particular itemsets. For movies “Jaws”, “Inception”, “Underworld: Rise of the Lycans”, there were 4 different permutations that we encountered, with all three items appearing in the top-rank of at least one permutation.

We found 61 populations, with 2 to 7 users (3.9 on average). Users agreed on 1 to 19 rankings (6.4 on average). Among the largest populations, a group of 6 users ranked the following movies in the same way: *Jurassic Park, Xmen Last Stand, Stealth* ; *Serenity, Star Wars III, Stealth* ; *The Matrix, Star Wars III, Stealth* ; *The Dark Knight, X Men Last Stand, Stealth.* We observe that the movie “Stealth” is positioned at the last rank in each of the permutations — this correlates positively with its average IMDb rating of 2.5, suggesting it as a less favorable movie, in general. Note that most of the above movies are extremely popular, and when using a ratings-based system, most users would have rated them highly, but one could not extract such fine-grained preferences as the above.

In summary, we presented an approach for mining local structure in ranked datasets and showed its results in CrowdRank. To the best of our knowledge, this kind of structure has not been mined before, and so both our objective and the approach we take are novel. Our results support the hypothesis that local structure holds in ranked data.

5. RELATED WORK

There has been considerable recent work on crowdsourcing [4, 6, 10, 13, 22, 23], and significant progress has been made towards understanding how to harness the wisdom of the crowds. Numerous successful crowd sourcing platforms are in use today, including Amazon MT [1] and CrowdFlower [2]. Our data was collected using Amazon MT.

There has been recent work on crowdsourcing preference data. In [7] and [20] the authors develop methods for deriving a gold-standard ranking from crowdsourced pairwise preferences. In [15] the authors study the problem of aggregating crowdsourced preferences to find the highest ranking object in a datasets, while in [9] the goal is to compute answers to top-k and group-by queries using the crowd. In contrast to these lines of work, we do not assume the existence of a gold-standard maximum, top-k list or ranking. Instead we aim to understand the diversity of opinions that hold in a population.

Quantifying worker quality is of central importance in crowdsourcing, with recent approaches discussed in [16, 24, 27, 28, 29]. We also tackle this question, and, unlike in prior work, propose measures of worker quality that are appropriate specifically for ranked subjective preferences.

Our data cleaning approach is inspired by the work on minimal repairs in databases [5, 8], where the goal is to delete a minimal set of tuples to restore consistency. Our work deals in particular with repairing the set of pairwise preferences.

Our mining of local structure builds on our prior work on rank-aware clustering [25], where we identify correlations that hold between item attributes and ranking. Scalability was achieved using bottom-up search to identify local correlations in subspace projections of high-dimensional datasets. We leveraged this insight here, and developed a bottom-up procedure that exploits locality.

Our work is also related to mining subgraphs from directed graphs [18]. These algorithms work in a similar bottom-up way, building potential candidates and expanding the set of nodes in each iteration. Our work differs from prior work in that we are estimating a probability distribution over all observed orders of items in a particular itemset that occur along a path, but not necessarily consecutively.

6. CONCLUSIONS AND FUTURE WORK

We described the collection and novel analysis of a dataset of crowd rankings. This work is part of our ongoing projects on understanding the structure of ranked data [26] and on building a system for the management and analysis of preferences [17]. Our dataset is publicly available at `https://github.com/stoyanovich/CrowdRank`.

This work has led to many interesting challenges that warrant future research. From the perspective of database research, rankings are a novel type of data, and as such can benefit from declarative models and query languages that would support efficient computation of inferences such as transitive closure. The presence of inconsistencies also raises interesting questions regarding the semantics of answers to general preference queries over a user’s data. One aspect of this is that it is important to differentiate between different kinds of inconsistencies, e.g., those due to ties (lack of preference) vs. those due to context sensitivity. In our ongoing work we aim to appropriately model different kinds of inconsistencies, and to develop data cleaning and query answering procedures that incorporate this information.

From the perspective of crowdsourcing, there are also many future directions that one can take. It is important to consider additional ways to design ranking tasks that tackle the data sparsity problem, but do not impose significant cognitive load on the workers. Additionally, evaluating worker quality based on ranked data is still an open area of investigation. We would like to extend our worker quality measures that are based on pairwise inconsistencies and minimal repairs, e.g., integrating temporal aspects and evaluating inconsistencies or contradictions that arise over time.

7. REFERENCES

[1] Amazon Mechanical Turk. `https://www.mturk.com`.

[2] CrowdFlower. `http://crowdflower.com/`.

[3] R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. In *VLDB*, 1994.

[4] S. Amer-Yahia et al. Crowds, clouds, and algorithms: exploring the human side of "big data" applications. In *SIGMOD*, 2010.

[5] M. Arenas, L. E. Bertossi, and J. Chomicki. Consistent query answers in inconsistent databases. In *PODS*, 1999.

[6] A. Bozzon, M. Brambilla, and S. Ceri. Answering search queries with CrowdSearcher. In *WWW*, 2012.

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

[8] J. Chomicki and J. Marcinkowski. Minimal-change integrity maintenance using tuple deletions. *Inf. Comput.*, 197(1-2):90–121, 2005.

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

[10] D. Deutch and T. Milo. Mob data sourcing. In *SIGMOD*, 2012.

[11] P. Diaconis. A generalization of spectral analysis with applications to ranked data. *Annals of Statistics*, 17(3):949–979, 1989.

[12] M. A. Fligner and J. S. Verducci. Distance-based ranking models. *Journal of the Royal Statistical Society. Series B*, 48(3):359–369, 1986.

[13] M. J. Franklin et al. CrowdDB: answering queries with crowdsourcing. In *SIGMOD*, 2011.

[14] I. C. Gormley and T. B. Murphy. A latent space model for rank data. In *ICML*, 2006.

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

[16] P. G. Ipeirotis and P. K. Paritosh. Managing crowdsourced human computation: a tutorial. In *WWW (Companion Volume)*, pages 287–288, 2011.

[17] M. Jacob, B. Kimelfeld, and J. Stoyanovich. A system for management and analysis of preference data. *PVLDB*, 7(12), 2014.

[18] C. Jiang, F. Coenen, and M. Zito. A survey of frequent subgraph mining algorithms. *The Knowledge Engineering Review*, 1(1):1–31, 2013.

[19] T. Kamishima and S. Akaho. Supervised ordering by regression combined with thurstone's model. *Artif. Intell. Rev.*, 25(3):231–246, 2006.

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

[21] A. Maydeu. Thurstonian modeling of ranking data via mean and covariance structure analysis. *Psychometrika*, 64(3):325–340, 1999.

[22] A. G. Parameswaran et al. CrowdScreen: algorithms for filtering data with humans. In *SIGMOD*, 2012.

[23] C. V. Pelt and A. Sorokin. Designing a scalable crowdsourcing platform. In *SIGMOD*, 2012.

[24] V. S. Sheng, F. J. Provost, and P. G. Ipeirotis. Get another label? improving data quality and data mining using multiple, noisy labelers. In *KDD*, 2008.

[25] J. Stoyanovich, S. Amer-Yahia, and T. Milo. Making interval-based clustering rank-aware. In *EDBT*, 2011.

[26] J. Stoyanovich et al. Understanding local structure in ranked datasets. In *CIDR*, 2013.

[27] J. Wang, A. Ghose, and P. Ipeirotis. Bonus, disclosure, and choice: What motivates the creation of high-quality paid reviews? In *ICIS*, 2012.

[28] P. Welinder et al. The multidimensional wisdom of crowds. In *NIPS*, 2010.

[29] J. Whitehill et al. Whose vote should count more: Optimal integration of labels from labelers of unknown expertise. In *NIPS*, 2009.