Finding Outlying Items in Sets of Partial Rankings

Antti Ukkonen[1,3] and Heikki Mannila[1,2,3]

[1] Helsinki University of Technology
[2] University of Helsinki
[3] Helsinki Institute for Information Technology

Abstract. Partial rankings are totally ordered subsets of a set of items. For example, the sequence in which a user browses through different parts of a website is a partial ranking. We consider the following problem. Given a set D of partial rankings, find items that have strongly different status in different parts of D. To do this, we first compute a clustering of D and then look at items whose average rank in the cluster substantially deviates from its average rank in D. Such items can be seen as those that contribute the most to the differences between the clusters. To test the statistical significance of the found items, we propose a method that is based on a MCMC algorithm for sampling random sets of partial rankings with exactly the same statistics as D. We also demonstrate the method on movie rankings and gene expression data.

1 Introduction

Partial rankings are totally ordered subsets of a set of items. For example, the set of items might contain all products available at an Internet store, while a partial ranking contains only products viewed by one user, ranked in the order the user clicked on their descriptions. Partial rankings can be found for example in clickstream analysis, collaborative filtering and different scientific applications, such as analysis of microarray data.

Given a set of partial rankings we can construct a clustering so that similar rankings are assigned to the same cluster [6]. The rankings belonging to the same cluster can be aggregated to form a condensed representation of the cluster. This representation can be for example a total or partial order on the complete set of items. However, comparing the aggregate representations between clusters can sometimes be difficult. Especially if the number of items is very large, it can be hard to quickly identify features that separate the clusters from each other.

For example, consider microarray data where the expression levels of a number of genes have been measured in different tissues or under different conditions. Typically this kind of data is represented as a matrix, where the rows correspond to different genes and columns to different tissues/conditions. This data can be converted to partial rankings by sorting the tissues separately for each gene in decreasing order of the level of expression. These rankings are indeed partial, due to missing data. The partial rankings can be clustered to find out in what

J.N. Kok et al. (Eds.): PKDD 2007, LNAI 4702, pp. 265–276, 2007.

tissues the expression of genes belonging to a cluster is exceptionally strong or weak. This type of analysis is relevant in cases where one wants to identify the tissues or conditions in which a certain set of genes is more active than the rest.

As another example, consider movie ratings given by viewers. These ratings can be converted to partial rankings as well. In general people tend to prefer the same movies: if a movie is very good (or bad), then it is likely that the vast majority of all viewers considers it is good (or bad). But some titles divide the opinions of the viewers more than others. One of such films is for example Pulp Fiction by Quentin Tarantino. People either think it is a very good movie – maybe because of the distinct style of the director – or are appalled by the violence and use of language. What we might thus expect, is that when movie rating data is divided to, say, two clusters, the titles that end up as discriminative are movies that have a fairly strong fan base, but are frowned upon or otherwise disliked by others.

This leads to the idea of an alternative representation for a cluster. Instead of using a total or partial order as the aggregate representation, we can list those items that are ranked either distinctively high or low by the partial rankings in a cluster when compared to an aggregate representation of the entire data set. This provides a way of characterizing a cluster of partial rankings in terms of the items that separate it from the complete data. We call such items *outlying items*. A similar approach was proposed in [3], but it uses a different definition for outlyingness.

The second question concerns the statistical significance of the found items. For evaluating the "outlyingness" of an item we need a way to generate artificial sets of partial rankings with exactly the same statistics as the real input data. To this end we propose an MCMC algorithm for sampling sets of partial rankings from an equivalence class that is defined by the following statistics: number and length distribution of partial rankings, occurrence and co-occurrence frequencies of the items and the probabilities $Pr(u \prec v)$ that an item u precedes another item v in the partial rankings for all u and v. Especially important are the probabilities $Pr(u \prec v)$, as they play a major part in identifying the outlying items. We consider an item a significant outlier if it behaves differently in real data compared to random data sampled from the same equivalence class.

The rest of this paper is organized as follows. The definition of an outlying item and significance testing is discussed in Section 2. The MCMC algorithm used for generating random data is described in Section 3. Experiments and results are presented in Section 4 while Section 5 provides a short conclusion.

2 Problem Statement

2.1 Basic Definitions

Let M be a set of items to be ranked. In the following when we discuss an item u, it is always assumed to belong to the set M. A *partial ranking* ϕ is a totally ordered subset of M. Note that this is not the same as a *partial order* that

concerns all of M but leaves the mutual ranking between some items unspecified. A set of partial rankings is denoted D.

Given the set D, we can compute a number of *statistics* that describe it. Some simple examples of such statistics are the size of D, length distribution of the partial rankings, occurrence-, and co-occurrence frequencies of the items. As the data contains rankings, the most important statistic is related to the mutual order of the items. For each pair (u, v), $u, v \in M$, we consider the probability that item u precedes the item v in D, denoted $Pr(u \prec v)$. We estimate $Pr(u \prec v)$ with the fraction of partial rankings in D that place u before v. Note that u and v do not need to be adjacent in the partial ranking. If u and v never occur together in a partial ranking, we set $Pr(u \prec v) = Pr(v \prec u) = 0.5$, and in general we always have $Pr(u \prec v) + Pr(v \prec u) = 1$. The probabilities are arranged in a $|M| \times |M|$ matrix C_D, so that $C_D(u, v) = Pr(u \prec v)$. We call C_D the *pair order matrix* associated with the set of partial rankings D.

All of the statistics discussed above can be used to define *equivalence classes* over the set of all possible sets of partial rankings. We denote with $\mathcal{C}(D)$ the class of sets of partial rankings that all have exactly the same statistics as the set D. Later, in Section 3, we will discuss an algorithm for sampling uniformly from $\mathcal{C}(D)$.

2.2 Finding Outlying Items

Let D' be a subset of D. Typically D' is obtained by computing a clustering of D and letting D' correspond to one cluster. Our aim in this paper is to discover items that behave differently in D' when compared to D. To do this we use the pair order matrices C_D and $C_{D'}$. More specifically, we are interested in the quantities

$$S_D(u) = \sum_v C_D(u, v) \quad \text{and} \quad S_{D'}(u) = \sum_v C_{D'}(u, v),$$

which are simply the row sums of the pair order matrices corresponding to D and D' for item u. These are indicators of the *global rank* of an item in a set of partial rankings . For example, if $S_D(u)$ is very large, then the item u should be placed before most of the other items in a global ranking based on D, as u tends to precede them in the partial rankings in D. Likewise, if $S_D(u)$ is small the item should be placed after most of the other items in the global ranking.

Given the subset D' of D, we consider an item *outlying* in D' if the difference

$$X(u) = S_{D'}(u) - S_D(u) \tag{1}$$

is significantly above (or below) zero. This would mean the rankings that belong to D' tend to place u more often before (or after) the other items than the rankings belonging to D in general. Hence, the subset D' is at least partially characterized by the way in which it ranks the item u. See [3] for a slightly different approach for finding outlying items.

Given the sets D and D' we can sort the items in decreasing order of their *outlyingness* $|X(u)|$ and pick the h topmost items as interesting representatives

for the set D'. These items may contain both items that are ranked unusually high or unusually low in D' when compared to D.

Thus, the definition of an outlying item is very simple. Computing $X(u)$ is almost trivial, as we only need to build the pair order matrices and compute their row sums, but doing this alone may lead to incorrect conclusions. Consider the case where all partial rankings in D are completely random, the probabilities $Pr(u \prec v)$ are all approximately 0.5 for all u and v. If we use some clustering algorithm to divide D to two non-overlapping clusters, the result will be arbitrary. However, in one of the clusters some of the items may have a slightly higher global rank than in the entire data D. These will be identified as outlying, even though they were found largely by coincidence. To prevent us from finding such items in real data sets, we propose methods for evaluating the significance of the outlyingness of an item.

2.3 Testing the Significance of Outlying Items

There are two possible pitfalls when using $|X(u)|$ to find outlying items for a subset D'. First, we must address the reliability of $|X(u)|$ as the indicator of outlyingness for a fixed item u. Especially we want to determine if the deviation of $X(u)$ from zero for a specific u is only caused by the values in the pair order matrix C_D. If this were the case then we should observe high deviations from zero for $X(u)$ also with sets of partial rankings that differ from D but have the same pair order matrix.

The second issue is related to the validity of the set of outlying items in general. Suppose that we generate a random set $\hat{D}$ of partial rankings so that $C_D = C_{\hat{D}}$ and compute a clustering of $\hat{D}$ to k clusters. The meaningfulness of the set of outlying items found in the real data can be questioned if roughly the same number of (possibly different) items u have equally high deviations of $X(u)$ from zero in a cluster of the random data.

The basic approach for using random data is the following:

1. Compute a clustering of the real data D and find the sets of outlying items for each cluster using $|X(u)|$.
2. Pick a set $\hat{D}$ of partial rankings from $\mathcal{C}(D)$ (the equivalence class of sets of partial rankings with same statistics as D) uniformly at random.
3. Compute a clustering of $\hat{D}$ and record the $|X(u)|$ values for each item in each cluster. If enough samples of $|X(u)|$ have been obtained, go to next step, otherwise go to step 2.
4. Estimate $E[|X(u)|]$ and $Var[|X(u)|]$ for all $u \in M$ from the samples.
5. Compute a significance measure for $|X(u)|$ based on $E[|X(u)|]$ and $Var[|X(u)|]$.

Hence, we assume the $X(u)$s are normally distributed. The significance is measured by the distance of $|X(u)|$ from $E[|X(u)|]$ in standard deviations. That is, we let

$$Z(u) = \frac{\big||X(u)| - E[|X(u)|]\big|}{\sqrt{Var[|X(u)|]}}.$$

For example, if $Z(u) > 3$, it is fairly safe to assume that u indeed is a significantly outlying item.

To address the significance of the entire set of outlying items, we compute the quantity $Y(D') = \sum_u X(u)^2$, where the $X(u)$s are deviations in subset D'. This is done both for the original set of partial rankings and each random data. The significance of the deviations in the original data is again expressed as the distance of $Y(D')$ from $E[\sum_u X(u)^2]$ computed over all the clusters in random data sets. Denote this by Q. Large values of Q indicate that the set of outlying items in D' is more significant, as it means that in random data the $X(u)$s deviate on the average less from zero.

3 Sampling from $\mathcal{C}(D)$

In order to test for the outlyingness $X(u)$ of an item u we must have a way of generating random sets of partial rankings with exactly the same features as the original data D. Features we want to preserve are the size of D, length distribution of the partial rankings, occurrence and co-occurrence frequencies of all items, and most importantly the pair order matrix C_D. Recall, that data sets with the same statistics as a given data set D belong to the equivalence class $\mathcal{C}(D)$. We present a simple MCMC algorithm for sampling sets of partial rankings uniformly from $\mathcal{C}(D)$.

3.1 The SWAP-PAIRS Algorithm

The basic idea of the algorithm is to perform swaps of adjacent items in the partial rankings of D. Suppose that u and v are adjacent in the partial ranking ϕ with u before v. Swapping u and v in ϕ has no effect on the length of ϕ or the frequencies of u or v in general, but only on $C_D(u, v)$ which is decremented and $C_D(v, u)$ which is incremented, both by the same amount. To preserve the values of $C_D(u, v)$ and $C_D(v, u)$ we must perform another swap with the opposite effect, i.e., we must find a partial ranking ψ where u and v are adjacent, but v precedes u, and swap those. Combining these two swaps results in a new set of partial rankings $\hat{D}$ with ϕ and ψ changed, but having the pair order matrix $C_{\hat{D}}$ equal to the original C_D. The algorithm is called SWAP-PAIRS and it simply starts from D and performs a sequence of such swaps at random.

To do the swaps efficiently we preprocess the data and compute the set A of *swappable pairs*. More formally we let

$$A = \{\{u, v\} | uv \in \phi \wedge vu \in \psi \wedge \phi, \psi \in D\}, \tag{2}$$

where $uv \in \phi$ denotes that items u and v are adjacent in ϕ with u before v. Note that if the pair $\{u, v\}$ is swappable and u and v are swapped, then $\{u, v\}$ remains swappable. However, as a consequence of a swap some other pairs may become swappable and some other unswappable. For example, consider the following set of partial rankings:

SWAP-PAIRS:

1. Pick the swap (u, v, ϕ, ψ) uniformly at random from all possible swaps in the current state.
2. Swap the positions of u and v both in ϕ and ψ with probability $\min(1, \frac{N(current)}{N(swapped)})$. Update the set of possible swaps accordingly.
3. If we have done enough swaps, output $\hat{D}$, otherwise go to step 1.

Fig. 1. A high level description of the SWAP-PAIRS algorithm

$$\begin{array}{l}\phi_1\text{: } 1\ 2\ 3\ 4\ 5\\ \phi_2\text{: } 7\ 8\ 4\ 3\ 6\\ \phi_3\text{: } 3\ 2\ 6\ 4\ 1\end{array}$$

It is quickly seen that 3 and 4 are a swappable pair as they are adjacent in both ϕ_1 and ϕ_2 but in different order. Also note that 2 and 3 form a swappable pair with partial rankings ϕ_1 and ϕ_3. In this case we have thus $A = \{\{2, 3\}, \{3, 4\}\}$. Lets say we decide to swap 3 and 4 and obtain:

$$\begin{array}{l}\phi'_1\text{: } 1\ 2\ 4\ 3\ 5\\ \phi'_2\text{: } 7\ 8\ 3\ 4\ 6\\ \phi_3\text{: } 3\ 2\ 6\ 4\ 1\end{array}$$

Now $\{2, 3\}$ is no longer swappable, as 2 and 3 are no longer adjacent in ϕ'_1 and we must remove this pair from A. However, now 4 and 6 are adjacent in both ϕ'_2 and ϕ_3 and their order is different, so we can add $\{4, 6\}$ to the set of swappable pairs and are left with $A = \{\{3, 4\}, \{4, 6\}\}$.

In addition to the list of swappable pairs we use a data structure, denoted S, that quickly returns the set of relevant partial rankings when given a swappable pair. We let $S(u, v) = \{\phi \in D | uv \in \phi\}$. The structure S is also computed during preprocessing and updated during the execution of the algorithm.

Finally, to sample uniformly from $\mathcal{C}(D)$ we must address one additional issue. We discuss some notation first. A *swap* is the tuple (u, v, ϕ, ψ), where u and v are items and ϕ and ψ are partial rankings in D. The swap (u, v, ϕ, ψ) means that items u and v are swapped in rankings ϕ and ψ. Denote the number of different possible swaps at the current state of the Markov chain with $N(current)$, and with $N(swapped)$ the same number for a state reachable by one swap from the current state. It is easy to see that $N(current) = \sum_{\{u,v\} \in A} |S(u, v)||S(v, u)|$. As it is possible that $N(current) \neq N(swapped)$, a simple algorithm that just picks possible swaps at random doesn't converge to the uniform distribution over $\mathcal{C}(D)$. To remedy this we use the Metropolis-Hastings step when performing a swap. That is, first the swap (u, v, ϕ, ψ) is picked uniformly at random from the set of all possible swaps at the current state, and we accept (u, v, ϕ, ψ) with probability $\min(1, \frac{N(current)}{N(swapped)})$. Intuitively the chain always moves to states with fewer possible swaps than the current state, and if the next state has a larger number of possible swaps, the transition is accepted with a probability less than 1. Pseudocode for the SWAP-PAIRS algorithm is given in Figure 1.

The number of possible swaps at a given state is of order $O(|M|^2|D|^2)$ in the worst case, which can get prohibitively large. In practice our implementation never stores the set of possible swaps explicitly, but only uses the A and S structures. The swap (u, v, ϕ, ψ) is computed (step 1 of algorithm) by first picking the pair $\{u, v\}$ from A with probability $\frac{|S(u,v)||S(v,u)|}{N(current)}$ and then picking ϕ and ψ uniformly at random from $S(u, v)$ and $S(v, u)$, respectively. This can be done in time $O(|M|^2)$ when elements of S are accessible in constant time. Complexity of the swap (step 2 of the algorithm) depends on the type of data structure used for A and $S(u, v)$ as they must be modified as a result of the swap. Our simple implementation uses sorted random access lists that can be updated in time $O(\log |M|^2)$ and $O(\log |D|)$ in case of A and $S(u, v)$, respectively. Hashing would provide a constant time solution, but might make step 1 more complicated as we would need to sample uniformly from the values of a hash table. In practice, however, the biggest bottleneck of the algorithm is sampling the swappable pair $\{u, v\}$ from A, because the probabilities need to be updated on every iteration for each pair.

3.2 Theoretical Questions and Convergence Diagnostics

The problem of creating sets of partial rankings with the same features as a given initial set by performing swaps of adjacent items is interesting in its own right. The problem is very similar to the one discussed in [4] and more recently in [2] and [1] in the context of 0-1 matrices. There the problem is to generate 0-1 matrices with exactly the same row and column sums as a given initial matrix.

It is easy to see that SWAP-PAIRS preserves the statistics used to define the class $\mathcal{C}(D)$. However, it is not obvious that $\mathcal{C}(D)$ is connected with respect to the swap operation. In [4] it is shown that the set of 0-1 matrices with same row and column sums is connected with respect to a certain local transformation of the matrix values. Whether this holds also with $\mathcal{C}(D)$ is an interesting open question. Moreover, estimating the size of $\mathcal{C}(D)$ is another task for future work.

A more practical problem concerns the convergence of the Markov chain defined using the swap operation. To use the algorithm for sampling sets of partial rankings we must know how many swaps to make to be sure that the resulting rankings are uncorrelated with the initial state D. In general analyzing the mixing times of Markov chains formally is nontrivial. We can, however, empirically evaluate the sequence of sampled sets $\hat{D}$ of partial rankings in terms of their distance from D.

To do this, we define the function δ as measure of the distance between two sets of partial rankings generated by the swap randomization algorithm. As SWAP-PAIRS only swaps items within a partial ranking, the items belonging to each ranking stay the same. Hence, the i:th partial ranking in the swapped data set, denoted $\hat{D}(i)$, is in fact a permutation of the i:th partial ranking of the original data set, denoted $D(i)$. We define $\delta(D, \hat{D}) = |D|^{-1} \sum_i d(D(i), \hat{D}(i))$, where d is some distance function between permutations. We use Kendall's tau, which is the number of pairs of items that are ordered differently by D and $\hat{D}$. The

measure δ is thus the average permutation distance between the partial rankings in D and $\hat{D}$.

To see when the chain has converged we must see how δ behaves as the swapping progresses. Denote by $\hat{D}_j$ the set of partial rankings obtained from D by performing j swaps. We say the chain has converged after r swaps when $\delta(D, \hat{D}_r) \approx \delta(D, \hat{D}_\infty)$. In practice we can determine r by starting the chain from D and stopping when $\delta(D, \hat{D}_j)$ no longer increases. This way of measuring convergence can be questioned as it does not directly use any of the estimated parameters ($X(u)$ in this case), but it is sufficient for our immediate concern of keeping consecutive samples uncorrelated.

4 Experiments

4.1 Data Sets

In the following we briefly discuss the data sets used in the experiments. Table 2 summarizes some of their statistics.

Movielens data. The MovieLens data[1] was originally collected by the GroupLens research group at University of Minnesota. It contains 10^6 ratings for about 3900 movies from over 6000 users. The ratings are given on a scale of 1-5.

Before turning the ratings into partial rankings we preprocess the data as follows. First we discard movies that have been ranked by less than 1000 users. This is done to reduce the number of different movies to 207. As many movies have been seen by only very few users the data does not contain enough information about their relation to the other movies. Next we prune users who have not used the entire scale of five stars in their ratings. This way the resulting partial rankings are more useful as they all reflect the entire preference spectrum from "very bad" to "excellent". This leaves us with 2191 users. For each user we create a partial ranking by picking uniformly at random at most three movies with the same number of stars and ordering them according to the number of stars so that better movies are ranked before the worse ones. The mutual order between two movies with the same number of stars is arbitrary. We call the resulting data set MOVIELENS.

Microarray data. We use a publicly available[2] microarray data from [5]. The data contains expression levels of 1375 genes in 60 cell lines associated to different types of cancer. This data is converted to partial rankings by first sorting the cell lines in decreasing order of expression, separately for each gene. If the expression of a gene for some cell line is unavailable, then this cell line is omitted from the ranking for that gene. Finally we select a random sample of 20 cell lines, again separately for each gene, to have one partial ranking of 20 items for each gene. We call this data set NCI60.

[1] `http://www.movielens.org/` (24.4.2007)

[2] `http://discover.nci.nih.gov/datasetsNature2000.jsp` (24.4.2007)

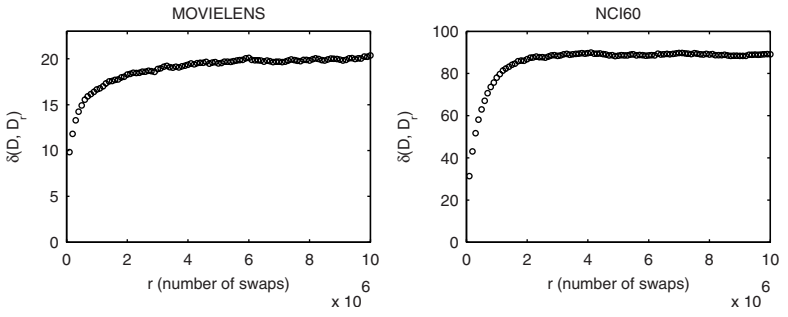

Fig. 2. Convergence of the SWAP-PAIRS algorithm with a movie ranking data (left) and a gene expression data (right). The measure $\delta(D, D_r)$ is the average Kendall distance between rankings in the original data D and the permuted data D_r after r swaps.

4.2 Convergence of SWAP-PAIRS

To make sure that the samples we obtain from $\mathcal{C}(D)$ are not correlated, we use the measure δ discussed above. Before sampling random sets of partial rankings to be used in the actual experiments, we ran the SWAP-PAIRS algorithm on both NCI60 and MOVIELENS data sets for ten million swaps and measured the distance $\delta(D, \hat{D}_r)$ every 10^5 steps. Running time of this test was a little over 14 minutes with the MOVIELENS data and about five minutes with the NCI60 data using a simple Java implementation of SWAP-PAIRS with JRE 1.5 on a 1.8GHz CPU. This difference is due to the different number of items in the data sets. The results are shown in Figure 2.

It is immediate that with the NCI60 data the algorithm seems to have converged after roughly four million swaps. The subsequent samples are all at approximately equal distance from the original set of partial rankings. With MOVIELENS the convergence is not as rapid. Even after ten million swaps it appears that $\delta(D, \hat{D}_r)$ is still slightly increasing, albeit extremely slowly.

For our purposes we considered it is enough to make 5 million swaps between samples with both data sets. In case of NCI60 this should yield very uncorrelated samples and also with MOVIELENS the samples should be usable.

4.3 Results

With both data sets we first computed a clustering to three clusters, and then determined the list of outlying items for each cluster. The validity of the found items was tested using the method discussed. We used 100 random data sets sampled with the SWAP-PAIRS algorithm.

When MOVIELENS is clustered to three clusters, we obtain one group with 195, one with 985 and a third one with 1011 partial rankings. The topmost part of Table 3 shows the five movies with largest positive and negative deviations of $X(u)$ from zero in cluster 1. As it contains so few items, the deviations are not very significant in terms of the $Z(u)$ measure. We report the items nonetheless,

Table 1. Outlying items found in clusters 2 and 3 computed from the NCI60 data

Cell line ID	$X(u)$	$E[\|X(u)\|]$	$Std[\|X(u)\|]$	$Z(u)$
Cluster 2: (412 rankings), $Q = 18.92$				
CO:COLO205	14.34	3.9	2.76	3.78
CO:HT29	13.74	3.93	2.72	3.61
CO:SW-620	13.45	3.79	2.57	3.76
CO:KM12	12.14	4.92	3.18	2.27
CO:HCC-2998	11.15	3.47	2.3	3.34
LE:HL-60	10.87	3.66	2.66	2.71
CO:HCT-15	10.71	3.89	2.98	2.29
Cluster 3: (434 rankings), $Q = 12.33$				
BR:MDA-N	19.31	3.8	2.57	6.04
BR:MDA-MB-435	19.05	3.59	2.64	5.86
ME:SK-MEL-5	17	4.71	3.24	3.79
ME:SK-MEL-28	13.83	4.71	3.11	2.93
ME:M14	13.48	3.25	2.27	4.5
ME:UACC-62	13.48	3.96	2.76	3.45
ME:SK-MEL-2	12.22	4.5	3.07	2.52
ME:MALME-3M	11.74	4.15	2.98	2.55
ME:UACC-257	11.17	3.65	2.48	3.03

Table 2. Statistics of the data sets used in the experiments

	MOVIELENS	NCI60
number of partial rankings	2191	1375
total number of items to rank	207	60
average length of partial ranking	13.25	20
length of shortest partial ranking	6	20
length of longest partial ranking	15	20

as they have a nice interpretation. Items that have a high value of $X(u)$ in cluster 1 are successful mainstream action titles, while those that have a low value (are disliked by the viewers) are older, maybe movies for a more mature audience. Clusters 2 and 3 are even more interesting. Table 3 shows the five most preferred and disliked movies for both clusters that are also significant in terms of $Z(u)$. One immediately notices that three movies (Being John Malkovich, Fargo and Reservoir Dogs) that are preferred by cluster 2 have a large negative value of $X(u)$ in cluster 3. In fact, both Pulp Fiction and Election almost made the list as well, as $X = -31.73$ (with $Z = 2.94$) for Pulp Fiction and $X = -35.09$ (with $Z = 4.43$) for Election in cluster 3. These movies are titles that viewers typically either love or hate. This result suggests that the outlying items can be used to identify "causes of controversy" in partial rankings. Also, using the randomization for identifying significant items proved useful, as for instance in cluster 3 the movie having the highest $X(u)$ was The Blair Witch Project with $X = 70.47$ but as its Z was only 0.92, we omitted it from Table 3. There were

Table 3. Outlying items found in three clusters computed from the MOVIELENS data

Movie title	$X(u)$	$E[\lvert X(u)\rvert]$	$Std[\lvert X(u)\rvert]$	$Z(u)$
Cluster 1: (195 rankings), $Q = 1.10$				
Twister	40.55	18.35	14.78	1.5
The Lost World: Jurassic Park	34.68	13.87	11.29	1.84
Robocop	33.51	16.03	12.2	1.43
The Fifth Element	31.82	14.39	12	1.45
Mad Max 2	31.25	17.64	13.36	1.02
Glory	-37.88	17.52	14.94	1.36
The Godfather	-37.92	14.74	11.31	2.05
Amadeus	-39.47	17.79	13.76	1.58
The Sting	-44.54	19.98	15.92	1.54
North by Northwest	-50.41	25.64	20.24	1.22
Cluster 2: (985 rankings), $Q = 0.85$				
Being John Malkovich	50.27	10.66	9.8	4.04
Fargo	46	10.27	10.37	3.45
Pulp Fiction	32.63	8.9	7.76	3.06
Election	31.91	7.65	6.19	3.92
Reservoir Dogs	30.82	8.02	7.13	3.2
Star Trek: First Contact	-21.67	6.96	5.23	2.81
Mary Poppins	-21.84	6.17	5.6	2.8
Apollo 13 (1995)	-24.24	7.03	5.63	3.06
Forrest Gump (1994)	-29.41	7.7	7.04	3.08
Star Wars: Episode V	-45.31	11.37	9.94	3.41
Cluster 3: (1011 rankings), $Q = 1.46$				
Independence Day	47.15	10.95	9.47	3.82
Ghost	34.69	9.16	7.73	3.3
The Rock	30.91	9.47	7.1	3.02
Men in Black	26.62	8.01	7.37	2.52
Big	24.26	5.5	4.84	3.87
Young Frankenstein	-38.71	7.92	5.94	5.18
Reservoir Dogs	-45.71	8.02	7.13	5.28
Taxi Driver	-46.28	10.62	7.86	4.53
Being John Malkovich	-53.64	10.66	9.8	4.39
Fargo	-60.22	10.27	10.37	4.82

several other examples as well, such as American Beauty with $X = 50.87$ and $Z = 0.94$ in cluster 2. Maybe somewhat unfortunately, in MOVIELENS the $X(u)$s tend to deviate by the same amount also in random data, as shown by the small Q values.

Results for NCI60 are not interpreted as easily. Table 1 shows the cell lines with a large positive deviation of $X(u)$ in clusters 2 and 3 as well as their $Z(u)$ values. In both clusters the cell lines with high values of $X(u)$ come from the same type of cancer. In cluster 2 all but one cell line is a sample from a colon cell, while in cluster 3 all but two are samples associated to melanoma. Interestingly,

in this case all the Q values are very high indicating that in random data the deviations of $X(u)$ from zero are in general far smaller.

5 Conclusions and Future Work

We have presented a method for finding items from sets of partial rankings that have different status in different parts of the input. We called such items outlying items. The method is based on clustering of the partial rankings and subsequently computing a simple statistic $X(u)$ for each item. Those items u with $X(u)$ deviating from zero are considered outlying. We also discussed how to generate random data to evaluate the $X(u)$ statistic. The results indicate that the methods can be used to discover items that "divide the opinions" of the partial rankings.

Some interesting questions for future work include properties of the equivalence class $\mathcal{C}(D)$ defined by a set of partial rankings. Another direction for is sampling from $\mathcal{C}(D)$ in a more efficient way than what is possible with the SWAP-PAIRS algorithm. For example, replacing the swap with another kind of transformation that result in shorter mixing times of the Markov chain would be very useful. Another strategy would be to construct a model that can be used to sample the sets of partial rankings more easily. One could, for example, find a partial order P, such that the pair order matrix $C_{\hat{D}}$ obtained, when $\hat{D}$ is generated from uniformly sampled linear extensions of P, would approximate C_D as good as possible.

References

1. Cobb, G., Chen, Y.: An application of markov chain monte carlo to community ecology. American Mathematical Monthly 110, 264–288 (2003)
2. Gionis, A., Mannila, H., Mielikäinen, T., Tsaparas, P.: Assessing data mining results via swap randomization. In: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 167–176. ACM Press, New York (2006)
3. Kamishima, T., Akaho, S.: Efficient clustering for orders. In: Proceedings of the Sixth IEEE International Conference on Data Mining - Workshops (ICDMW'06), pp. 274–278. IEEE Computer Society Press, Los Alamitos (2006)
4. Ryser, H.J.: Combinatorial properties of matrices of zeros and ones. Canad. J. Math. 9, 371–377 (1957)
5. Scherf, U., Ross, D.T., Waltham, M., Smith, L.H., Lee, J.K., Tanabe, L., Kohn, K.W., Reinhold, W.C., Myers, T.G., TAndrews, D., Scudiero, D.A., Eisen, M.B., Sausville, E.A., Pommier, Y., Botstein, D., Brown, P.O., Weinstein, J.N.: A gene expression database for the molecular pharmacology of cancer. Nature Genetics 24(3), 236–244 (2000)
6. Ukkonen, A., Mannila, H.: Finding representative sets of bucket orders from partial rankings (submitted for review)