How to Find the Best Rated Items on a Likert Scale and How Many Ratings Are Enough

Qing Liu[1], Debabrota Basu[2], Shruti Goel[3],
Talel Abdessalem[4], Stéphane Bressan[5]

[1,2,5]School of Computing, National University of Singapore, Singapore, Singapore
[1]liuqing,[2]debabrota.basu@u.nus.edu, [5]steph@nus.edu.sg
[3]Jacobs School of Engineering, University of California San Diego, La Jolla, CA, USA, s2goel@eng.ucsd.edu
[4]LTCI/IPAL CNRS, Télécom ParisTech, Université Paris-Saclay, France
talel.abdessalem@telecom-paristech.fr

Abstract The collection and exploitation of ratings from users are modern pillars of collaborative filtering. Likert scale is a psychometric quantifier of ratings popular among the electronic commerce sites. In this paper, we consider the tasks of collecting Likert scale ratings of items and of finding the n-k best-rated items, i.e., the n items that are most likely to be the top-k in a ranking constructed from these ratings. We devise an algorithm, Pundit, that computes the n-k best-rated items. Pundit uses the probability-generating function constructed from the Likert scale responses to avoid the combinatorial exploration of the possible outcomes and to compute the result efficiently. Selection of the best-rated items meets, in practice, the major obstacle of the scarcity of ratings. We propose an approach that learns from the available data how many ratings are enough to meet a prescribed error. We empirically validate with real datasets the effectiveness of our method to recommend the collection of additional ratings.

1 Introduction

The collection and exploitation of ratings from users are modern pillars of collaborative filtering [6,8]. Likert scale is an ordinal rating scale popular among the electronic commerce sites and crowdsourced information systems such as TripAdvisor. Each value in the scale gauges the degree of satisfaction of the user towards a particular item, e.g., product or service.

Ranking the items based on Likert scale ratings is not always as obvious as it seems to be. For instance, ranking items using the expectations of the Likert scale responses can yield incorrect results [5]. Thus, we consider the task of finding the list of n items that are most likely to be the top-k in a ranking constructed from the ratings as our target recommendation task. For the sake of simplicity, we refer to this list as the *n-k best-rated items*. We define the problem of finding the n-k best-rated items as a probabilistic one. The uncertainty arises not from the unreliability of users but from the unavailability of ratings by all the users. We assume that the ratings from all users are correct and exact.

In this paper, we represent the scores of the items as discrete distributions on an L-valued Likert scale. We develop a polynomial-time algorithm, Pundit, that computes the n-k best-rated items. Pundit exploits the probability-generating functions of the discrete distributions of the items to avoid the combinatorial exploration of all possible outcomes and to compute the result efficiently. Our method is exact whereas the other methods like ranking by mean ratings or Monte Carlo sampling [15] are approximate algorithms.

In practice, the problem is not solved yet. Since selection of the n-k best-rated items is constrained by insufficient ratings. Corresponding discrete distributions are not 'true' representations of the score of the items. We devise a score distribution error model based on KL-divergence to answer the question "How many ratings are enough". This error model estimate the deviation of the discrete distribution formed with the available data from the 'true' universal distribution. As we empirically evaluate this model on the Amazon review dataset, we observe the KL-divergence based model follows inverse law. Following this we use the inverse law for KL-divergence based error to recommend how many additional ratings should be proactively sought to reach a certain error threshold.

2 Related Work

The problem of finding the n-k best-rated items is related to the probabilistic threshold top-k query [4] that returns the items having a probability of being in the top-k over a user specified threshold. [10] proposes a unified way to summarize a category of probabilistic top-k queries. Though the problem definitions seem similar, the category of probabilistic top-k queries is applicable to the scenario where the existence of an item is uncertain. Each item has a fixed known score representing its quality. However, the uncertainty modeled in our problem emerges from the unavailability of the ratings by the universal user-pool and is expressed as an evolving distribution over a Likert scale. [9] studies the problem of ranking continuous probabilistic data, where the score of each item is modelled as a continuous probability distribution. The authors focus on the probability of an item being ranked at a certain position. This result cannot be applied directly to our problem since the probability of an item being in top-k is not simply the sum over the probabilities of it being at different positions in the ranking. Another variant of these queries is UTop-Rank query [16]. This query searches for the item that has the highest probability of being ranked within a certain range of positions. They solve UTop-Rank query based on Monte Carlo sampling techniques which produce an approximate result. We construct a polynomial time exact algorithm for our problem which is more effective and efficient than the Monte Carlo method.

Crowdsourcing-based approaches have been proposed for the ranking and top-k problems in recent years. For example, [3,2] study the problem of finding the 'max' item or ranking the items by asking the crowd to compare pairs of items. Then, heuristic algorithms or learning approaches are proposed to aggregate the opinions collected from the crowd and to find the item with the

maximal score. Beside these, [18] provides a thorough experimental study of the crowdsourced top-k queries. Most of the works in crowdsourcing use the *preference judgement* scheme which is based on the pairwise comparisons results from the crowd for inferring the global ranking. Hybrid approaches, such as [17,14], combine preference judgement and *absolute judgement*, like ratings, to infer the ranking. These approaches either transform the absolute judgement into the preference judgement [14] or use the parametric analysis [17] which may not be suitable for the ordinal data[5]. In this paper, we adopt the absolute judgement in form of the correct and exact ratings to infer the ranking of the items. The score of the item is modeled as a discrete distribution over a L-valued Likert scale.

3 How to Find the n-k Best-Rated Items?

3.1 Problem Definition

We use similar notations as in [11]. Consider a set of N items, $\mathcal{O} = \{o_1, \cdots, o_N\}$. A *scoring function* s maps the set of items $\mathcal{O}$ to a totally ordered domain $\mathcal{D}$, i.e. $s : \mathcal{O} \to \mathcal{D}$. $(\mathcal{D}, \geq)$ denotes a total order and $(\mathcal{D}, >)$ is the corresponding strict total order of $\mathcal{O}$ induced by s. We call the image $s(o)$ of an item $o \in \mathcal{O}$ by the function s the score of the item. A *ranking* $r : \mathcal{O} \to S_N$ is an indexing function induced on $\mathcal{O}$ by the total order $(\mathcal{D}, \geq)$. Here, S_N denotes the permutation group on $\{1, \ldots, N\}$. It is the set of all possible rankings of N items. For any two items o_i and $o_j \in \mathcal{O}$, if score of o_i is greater than or equal to that of o_j, i.e., $s(o_i) \geq s(o_j)$, we say that o_i is ranked equally with or above o_j, i.e., $r(o_i) \leq r(o_j)$.

In our problem, the score of each item is constructed from a collection of ratings. This epistemic uncertainty introduced by insufficiency of ratings prohibits existence of a deterministic score. Thus, we model the score $s(o_i)$ of an item $o_i \in \mathcal{O}$ as a random variable X_i with a probability mass function f_i. We define the score function as $s : \mathcal{O} \to \{f : \mathcal{L} \to [0, 1]\}$. Here, $\mathcal{L} \triangleq \{1, \ldots, L\}$ is the L-valued Likert scale and f is a probability mass function defined over the support $\mathcal{L}$. For example, $\mathcal{L}$ is $\{1, \ldots, 5\}$ for a 5-valued Likert scale. We call f a *score distribution*.

If $x_1, \cdots, x_N \in \mathcal{L}$ are the observed ratings for the N items correspondingly, the probability of an item o_i to be ranked in top-k is expressed in Equation 1.

$$\mathbb{P}(r(o_i) \leq k) = \sum_{\{x_1, \cdots, x_N\} \in S_i^k} f_1(x_1) \cdots f_N(x_N). \tag{1}$$

Here, S_i^k is the set of all N-tuples $\{x_1, \cdots, x_N\}$, such that for each $\{x_1, \cdots, x_N\}$ there exist at least $(N - k + 1)$ number of x's which are less than or equal to x_i. We call $\mathbb{P}(r(o_i) \leq k)$ the *positional probability* of o_i.

Example 1. Suppose there are three items, o_1, o_2 and o_3. If $k = 1, i = 2$, $\{x_1 = 5, x_2 = 1, x_3 = 5\}$ is not in S_2^1. Because it consists no rating lower than or equal to x_2. But $\{x_1 = 1, x_2 = 1, x_3 = 1\}$ is in S_2^1. Because ratings of o_1 and o_3 are equal to the rating of o_2.

We are looking for the list of n items $\Omega = [o_1, \ldots, o_n]$ that are most likely to be the top-k. That is, $\mathbb{P}(r(o_1) \leq k) \geq \cdots \geq \mathbb{P}(r(o_n) \leq k)$, and $\mathbb{P}(r(o_n) \leq k) \geq \mathbb{P}(r(o_{i'}) \leq k)$ for all $o_{i'} \notin \Omega$. This means that the items in Ω are ranked according to their positional probability and probability of other $N-n$ items to be in top-k is less than that of any item in Ω.

3.2 An Exact Algorithm for Finding the n-k Best-Rated Items

Approaches like ranking by the mean scores and Monte Carlo approaches give approximate results. Here, we develop an exact algorithm, `Pundit`, that finds the n-k best-rated items in polynomial time. The idea is to construct a degree N polynomial such that its coefficients are dependent on the positional probability. In the following, we will explain how to construct such a polynomial and then how to compute the coefficients. Once we can compute the positional probabilities efficiently, the n items with the highest positional probabilities are the result.

Construction of the Polynomial. We observe that by construction "rank of o_i is higher than or equal to k, i.e., $r(o_i) \leq k$" is equivalent to the fact that "at least $N-k$ items other than o_i have scores lower than or equal to score of item o_i, i.e., $s(o_i)$". This fact includes k mutually exclusive cases. Case $j \in \{1, \cdots, k\}$ occurs if exactly $N-k+j-1$ items other than o_i have scores lower than or equal to $s(o_i)$ and other $k-j$ scores are higher than $s(o_i)$. Thus, if we can calculate the probability for each of the k cases, the positional probability is the sum of the probabilities of these k cases.

In order to calculate the probability of each of the k cases, we construct a probability-generating function as shown in Equation 2. This construction connects the probability of each of the k cases to the coefficients of the polynomial.

$$\mathcal{F}_i(x, l) \triangleq \prod_{j \neq i} (\mathbb{P}(s(o_j) \leq l) + \mathbb{P}(s(o_j) > l)x) \tag{2}$$

In Equation 2, $\mathbb{P}(s(o_j) \leq l)$ denotes the probability that score of o_j is lower than or equal to l. For a given l, $\mathcal{F}_i(x, l)$ is a polynomial of x. In particular, the coefficient of the term x^k equals to $\mathbb{P}(\sum_{j \neq i} \mathbb{I}(s(o_j) > l) = k)$ [10]. Here, $\mathbb{I}(s(o_j) > l)$ is the indicator function that returns 1 or 0 depending on whether $s(o_j) > l$ is true or not. This implies that the coefficient of the term x^k is the probability that there are exactly k items having scores higher than l. If $s(o_i) = l$, the coefficient of the term x^{k-j} is the probability that there are exactly $k-j$ items having scores higher than $s(o_i)$. Thus, the coefficient of x^{k-j} exactly quantifies the j^{th} ($j \in \{1, \cdots, k\}$) case.

Now, we just need to think about how to compute the coefficients in Equation 2 efficiently.

Coefficients Calculation. We reconstruct the generating function of Equation 2 into the polynomial expression of x.

$$\mathcal{F}_i(x, l) = c_0(l)x^0 + \cdots + c_{N-1}(l)x^{N-1} \tag{3}$$

where $c_q(l)$ represents the q^{th} coefficient. The coefficients $c_0(l), \cdots, c_{N-1}(l)$ can be computed in $O(N^2)$ time by expanding Equation 2 into Equation 3.

We propose an efficient divide-and-conquer algorithm which applies Fast Fourier transform (FFT) to compute the coefficients $c_0(l), \cdots, c_{N-1}(l)$ more efficiently. Time complexity of this divide-and-conquer algorithm is $O(Nlog^2N)$. Due to the limitation of space, we refer the readers to our technical report [1] for more details of the efficient computation of the coefficients.

Pundit: The Algorithm. Once $c_0(l), \cdots, c_{N-1}(l)$ are computed, the positional probability for a L-valued Likert scale is calculated using $\mathbb{P}(r(o_i) \leq k) = \sum_{l=1}^{L}(c_0(l) + \cdots + c_{k-1}(l))\mathbb{P}(s(o_i) = l)$. Once the positional probabilities for all the N items are computed, the n items that have the highest positional probabilities are the n-k best-rated items. Calculating the positional probability $\mathbb{P}(r(o_i) \leq k)$ for each item takes $O(Nlog^2N)$ time, it would take $O(N^2log^2N)$ time for all the items. Here, we propose two techniques to accelerate the computation. The first technique is to pre-compute the coefficient C^l of $\mathcal{F}' = \prod_{o_j \in \mathcal{O}}(\mathbb{P}(s(o_j) \leq l) + \mathbb{P}(s(o_j) > l)x)$ for all $1 \leq l \leq L$. Using the shorthand notation, we get $\mathcal{F}_i(x, l) = \mathcal{F}'\left[p_i^l + (1 - p_i^l)x\right]^{-1}$, where $p_i^l = \mathbb{P}(s(o_i) \leq l)$. Thus, we need to compute the set of coefficients once for each l and all the coefficients can be deduced correspondingly. Secondly, we observe that explicit calculation of all the coefficients is not needed, we calculate only the first k coefficients $c_0(l), \cdots, c_{k-1}(l)$. Time complexity of `Pundit` reduces to $O(Nlog^2N + Nk)$.

4 How Many Ratings Are Enough?

Though we have formulated an exact algorithm, `Pundit`, for finding n-k best-rated items, the problem is not solved yet. For real applications, ratings of some items are either missing or insufficient. For example, more than 30000 books in our Amazon book dataset have only one rating while the entire population of our datasets is 8726569. If we try to find the 10 best-rated books from this dataset, we would get 10 books which are rated as 5-star by only one user. This result is statistically insignificant and probably biased. Thus, the question that naturally appears is– "how many ratings are enough to construct the score of an item?" We investigate error of the score distribution of an item if we have a finite number of ratings. This model allows us to set a threshold in the required number of ratings for ranking the items without introducing remarkable error.

4.1 Score Distribution Error Model

We represent the 'true' score of an item by the *oracle score distribution* f^* constructed with all the ratings from the universal user pool while the *observed score distribution* f is constructed with a limited number of ratings. For brevity, we call f^* and f the *oracle distribution* and the *observed distribution* respectively.

The Optimization Problem. Consider the scenario when m ratings of an item are collected in the form of L-valued Likert scale. Suppose $z_1, \cdots, z_L$ are the number of ratings for each of the L values, such that $\sum_{i=1}^{L} z_i = m$. Such a rating pool can be represented by a multinomial distribution, $\mathbb{P}(z_1, \cdots, z_L) =$

$\frac{m!}{z_1!\cdots z_L!} {p_1^*}^{z_1} \cdots {p_L^*}^{z_L}$. Here, $\{p_1^*, \cdots, p_L^*\}$ is the oracle distribution f^* of this item. The observed distribution f^m based on m ratings is $\{\frac{z_1}{m}, \cdots, \frac{z_L}{m}\}$.

In order to model the information gap between the observed distribution f^m and the oracle distribution f^*, we define the *expected score distribution error* as $E_m^{\mathrm{KL}} \triangleq \sum \mathbb{P}(z_1, \cdots, z_L)\ Dist(f^m, f^*)$ with a distance function $Dist(f^m, f^*)$. The sum is calculated over all $\{z_1, \cdots, z_L\}$ in the set of all possible L-partitions of m, $P(m, L)$. Thus, the expected error depends on three factors– the number of ratings m, the oracle distribution f^* and the distance function $Dist$. As m is given at an instance and the oracle distribution f^* is constructed with the universal review pool, modeling the expected error reduces to choice of the distance function. Since KL-divergence [7] quantifies the expected information per sample to discriminate between the uncertainty encompassed by one distribution against the other, we choose KL divergence as the eligible choice of distance function between the oracle and the observed score distributions. Thus, the expected error can be written as in Equation 4.

$$E_m^{\mathrm{KL}} = \sum_{\{z_1, \cdots, z_L\} \in P(m.L)} \left(\frac{m!\ {p_1^*}^{z_1} \cdots {p_L^*}^{z_L}}{z_1! \cdots z_L!} \sum_{i=1}^{L} \left(\frac{z_i}{m} \log \frac{z_i}{m p_i^*} \right) \right) \tag{4}$$

We want to find the minimal number of ratings m^* such that the expected error between the oracle and the observed distribution is less than a predefined threshold ϵ. Our objective is mathematically expressed in Equation 5.

$$m^* = \arg\min_m \ m \quad \text{such that, } E_m^{\mathrm{KL}} \leq \epsilon. \tag{5}$$

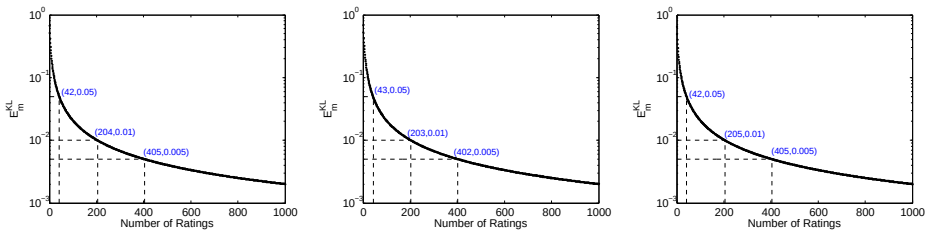
Efficient Solution. In order to compute m^* in Equation 5, we need to compute E_m^{KL}. E_m^{KL} depends on the oracle distribution, which is a choice, and the observed distribution f^m, which is observable. As we focus on the method for efficient calculation of the error, let us assume f^* is either given as a model parameter or constructed from the user-pool of a given dataset. But even when the oracle distribution is given, the expected error is not easy to compute. Because we sum over the set $P(m, L)$ that contains $\binom{m+L-1}{L-1}$ elements. It makes exact calculation of E_m^{KL} combinatorially expensive.

Thus, we propose a sampling approach to estimate the expected error based on the Ergodic Theorem [15]. Due to the limitation of space, we refer readers to our technical report [1] for more details of the computation of the expected error. Once we are able to calculate a sufficient approximation of the expected error E_m^{KL} efficiently, we can formulate the relation between E_m^{KL} and m. This allows us to find the minimal number of ratings required (m^*) to reach a prescribed error.

4.2 Experimental Investigation of Error Models

Dataset and Set-up. We use Amazon review dataset[1][13,12] with six categories of products– 'Apps for Android', 'Beauty', 'Books', 'Cell phones and

[1] `http://jmcauley.ucsd.edu/data/amazon/`



(a) Apps for Android ($N = 906$) (b) Beauty ($N = 174$) (c) Books ($N = 2816$)

Figure 1. Expected Error E_m^{KL} with Different Number of Ratings

Accessories', 'Electronics' and 'Movies and TVs'. Each review contains a rating for an item collected using a 5-valued Likert scale. This dataset is collected from May 1996 to July 2014. We consider only the items with more than 500 reviews. The remaining number of items is summarized in Figure 1. We only show the results on three datasets in Figure 1 due to the limitation of space, the results on other datasets are similar to those in Figure 1 [1]. We aggregate the ratings for each item to construct the oracle distributions of the items. Once we obtain the oracle distributions, we focus on uncovering the relation between the score distribution error and the number of ratings m. In the experiments, we increase the number of ratings accumulated for the items and then observe evolution of the error.

Score Distribution Error and Number of Ratings. In Figure 1, we present a smooth curve that fits the evolution of the error. We observe that the score distribution error decreases with increase in the number of ratings. This observation proves that this error model is consistent. Because the observed score distribution would converge to the oracle distribution with accumulation of more ratings, i.e., information about the item.

We also observe that decay of the expected score distribution error follows the inverse law, it fits with the hyperbolic equation $E_m^{\mathrm{KL}} = \frac{c}{m}$. For the six categories, c is a constant between 2.01 and 2.024. Also, evolution of the error is almost category independent as it quantifies the evolution of observed distribution with accumulation of ratings. Thus, the score distribution error depends on the accumulation of ratings but not on the exact object names or categories. Now, we are able to answer the question "How many ratings are enough". For example, in order to restrict the score distribution error to a prescribed value 0.005, we need around 405 ratings for each item.

5 Conclusion

In this paper, we study the problem of finding the n-k best-rated items by exploiting the ratings from the users. We devise an exact algorithm, Pundit, that

solves this problem efficiently. We develop the score distribution error model to quantify the effect of the accumulation of ratings and to answer "how many ratings are enough". Then, we uncover the fact that the score distribution error follows the inverse law, which enable us to predict minimal number of ratings that should be sought to meet a prescribed error.

Acknowledgement. This work is supported by the National University of Singapore under a grant from Singapore Ministry of Education for research project number T1 251RES1607 and is partially funded by the Big Data and Market Insights Chair of Télécom ParisTech.

References

1. http://www.comp.nus.edu.sg/~liuqing/tech-reports/TRB6-17-likert.pdf
2. Chen, X., Bennett, P.N., Collins-Thompson, K., Horvitz, E.: Pairwise ranking aggregation in a crowdsourced setting. In: WSDM. pp. 193–202 (2013)
3. Guo, S., Parameswaran, A., Garcia-Molina, H.: So who won?: dynamic max discovery with the crowd. In: SIGMOD. pp. 385–396 (2012)
4. Hua, M., Pei, J., Zhang, W., Lin, X.: Ranking queries on uncertain data: a probabilistic threshold approach. In: SIGMOD. pp. 673–686 (2008)
5. Jamieson, S., et al.: Likert scales: how to (ab)use them. Medical education 38(12), 1217–1218 (2004)
6. Jin, R., Si, L., Zhai, C., Callan, J.: Collaborative filtering with decoupled models for preferences and ratings. In: CIKM. pp. 309–316. ACM (2003)
7. Kullback, S., Leibler, R.A.: On information and sufficiency. The annals of mathematical statistics 22(1), 79–86 (1951)
8. Lee, J., Lee, D., Lee, Y.C., Hwang, W.S., Kim, S.W.: Improving the accuracy of top-n recommendation using a preference model. Information Sciences 348, 290–304 (2016)
9. Li, J., Deshpande, A.: Ranking continuous probabilistic datasets. VLDB 3(1-2), 638–649 (2010)
10. Li, J., Saha, B., Deshpande, A.: A unified approach to ranking in probabilistic databases. VLDB 2(1), 502–513 (2009)
11. Liu, Q., Basu, D., Abdessalem, T., Bressan, S.: Top-k queries over uncertain scores. In: COOPIS. pp. 245–262 (2016)
12. McAuley, J., Pandey, R., Leskovec, J.: Inferring networks of substitutable and complementary products. In: SIGKDD. pp. 785–794 (2015)
13. McAuley, J., Targett, C., Shi, Q., van den Hengel, A.: Image-based recommendations on styles and substitutes. In: SIGIR. pp. 43–52 (2015)
14. Niu, S., Lan, Y., Guo, J., Cheng, X., Yu, L., Long, G.: Listwise approach for rank aggregation in crowdsourcing. In: WSDM. pp. 253–262 (2015)
15. Robert, C.P., Casella, G.: Monte Carlo Statistical Methods. Springer (2004)
16. Soliman, M.A., Ilyas, I.F., Ben-David, S.: Supporting ranking queries on uncertain and incomplete data. VLDB 19(4), 477–501 (2010)
17. Ye, P., Doermann, D.: Combining preference and absolute judgements in a crowdsourced setting. In: ICML. pp. 1–7 (2013)
18. Zhang, X., Li, G., Feng, J.: Crowdsourced top-k algorithms: An experimental evaluation. VLDB 9(8), 612–623 (2016)