



An Empirical Comparison of Top-Lists Aggregation Methods

Ammar Eltigani and Tai Henrichs

The Problem



- Suppose there are n candidates and N voters.
- Define a **top-List** π_i to be the **strict** ordering of the candidates provided by voter i .
- Top-lists do not require a voter to rank all candidates. Instead
 - Each preference can rank up to n candidates $|\pi_i| \leq n$
- We assume unranked candidates are tied at the bottom of a top-list, i.e if u and v are not ranked by voter i , then $\pi_i(u) = \pi_i(v)$
- Define the **Generalized Kendall Tau Distance** as the traditional Kendall Tau Distance (number of pairwise disagreements between two ranking lists) that disregards ties. Formally,

$$K(\pi_i, \pi_j) = \sum_{u, v \in [n]} 1 \{ \pi_i(u) < \pi_i(v) \ \&\& \ \pi_j(v) < \pi_j(u) \}$$

- **Problem:** Given N top-lists, output a ranking of all candidates that minimizes average generalized Kendall Tau Distance to each top-list

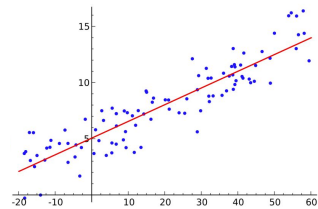
Ammar speaks

Note about how transformations from π_i to τ_i are made
How total distance is computed for the whole dataset
Could mention top-k special-case

Why Top-List Aggregation?



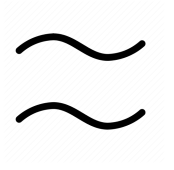
- Applications to
 - Likelihood approximation
 - Web content aggregation
 - Social choice / ranking
- Our own **motivations** for choosing it
 - More realistic to have top-lists than full rankings
 - Relatively new, no known implementations
 - Fun!



- Tai Speaks
- Also condorcet consistent and ranks those preferred by the majority at the top, if there is such a majority



Why Approximations?



- Complexity
 - Top-List Aggregation is NP-Hard for 4 or more candidates
- An approximation algorithm is an efficient (poly-time usually) approximate solutions to NP-Hard optimization problems with provable guarantees on the distance returned
 - ρ -Approximation for some problem guarantees that
$$\begin{cases} \text{OPT} \leq f(x) \leq \rho \text{OPT}, & \text{if } \rho > 1; \\ \rho \text{OPT} \leq f(x) \leq \text{OPT}, & \text{if } \rho < 1. \end{cases}$$
- Approximation algorithms we implemented*: **Borda+**, **Score-Then-Borda+**, **Score-Then-Adjust**, **FootRule+**, **RandomSort**

**these algorithms are mostly generalization from known full rank algorithms to the top-list problem (Mathieu & Simon 2020)*

Ammar Speaks

- Don't read off the algorithms

Example: Score-Then-Borda+*

Input: an instance (n, p) of TOP-AGG

Step 1, partition candidates into intervals:

$u \leftarrow$ uniformly random value on $[0, 1)$.

for all candidate $i \in [n]$ **do**

 Compute $Score_i \leftarrow p(\pi_i < \infty)$.

 Set $t \leftarrow \lfloor u - \ln(Score_i) \rfloor$ and put candidate i in interval E_t .

Step 2, solve the problem in each interval:

for all $t \in \mathbb{N} \cup \{\infty\}$ such that E_t is non-empty **do**

 Order E_t sorting candidates i by average rank $Rank_i \leftarrow \sum_{r=1}^n \frac{p(\pi_i=r)}{p(\pi_i < \infty)} \cdot r$

Concatenate the ranking of E_0 , ranking of E_1, \dots , and ranking of E_∞ .

Output resulting full-ranking.

Legend

- ★ E_i is the i^{th} partition
- ★ $p(\pi_i = r)$ is the probability that candidate i is placed in the r^{th} rank
- ★ $p(\pi_i < \infty)$ is the probability that candidate i is ranked

*randomized $(8e+4)$ approximation, where e is the euler number

Ammar Speaks

Define score and rank!

If score = 0, don't compute t. Automatically throw candidate into E_{inf}

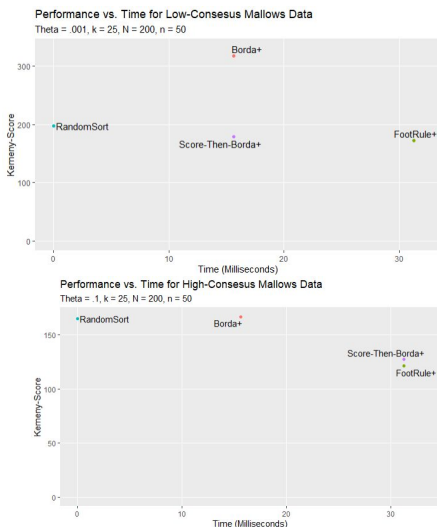
- Intuition for the approximation
 - Borda+ sorts candidates from best to worst rank and is a non-constant approximation $O(n)$
 - Score-Then-Borda+ is a constant approximation
 - Kemeny cares about swaps, but it doesn't care about swap distance. For Kemeny, it only matters that a is ranked before b, not that a is 5 vs. 4 places better than b
 - A candidate has a greater score the more often they appear in top-lists
 - if a candidate c is unranked by voter v, then no matter how c is ranked by an algorithm, v does not care. So, this roughly prioritizes candidates that are generally cared about more, in that voters have preferences about them
 - But, even if two candidates are ranked with similar frequency, one may be more highly ranked very often
 - So, we sort partitions by ranks

Example Input

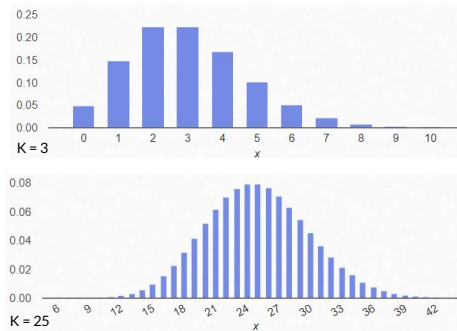
- Suppose the candidates are (1,2,3,4,5)
 - Three voters submit the following top-lists: $\pi_1 = (1,2,3), \pi_2 = (1,4,2), \pi_3 = (2,1)$
1. Suppose $u \in [0,1]$ equals 0.5
 2. Scores: $\text{Score}_1 = \text{Score}_2 = 3/3 = 1, \text{Score}_3 = \text{Score}_4 = 1/3, \text{Score}_5 = 0$
 3. $\forall i \in \{1,2,\dots,5\} t_i = Lu - \ln(\text{score}_i)J$ if $\text{score}_i > 0$ else $t_i = \infty$
 - a. $t_1 = t_2 = Lu - \ln(\text{score}_1)J = Lu - \ln(\text{score}_2)J = Lu - \ln(1)J = L.5 - 0J = 0$
 - b. $t_3 = t_4 = Lu - \ln(\text{score}_3)J = Lu - \ln(\text{score}_4)J = Lu - \ln(1/3)J = L.5 - (-1.09)J = L.5 + 1.09J = 1$
 - c. $t_5 = \infty$
 4. $E_t = \{i : t_i = t\} \rightarrow E_0 = \{1,2\}, E_1 = \{3,4\}, E_\infty = \{5\}$
 5. Average Ranks: $\text{Rank}_1 = 1 * 2/3 + 2 * 1/3 = 4/3, \text{Rank}_2 = 1/3 (2 + 3 + 1) = 2, \text{Rank}_3 = 1/1 * 3 = 3, \text{Rank}_4 = 1/1, \text{Rank}_5 = N/A$
 6. Sort each E_t by increasing ranks: $E_0 = \{1,2\}, E_1 = \{4,3\}, E_\infty = \{5\}$
 7. Output full-rank: $\sigma = (1,2,4,3,5)$

- Tai speaks

Preliminary Results



- Inspired by Ali and Meilă to consider data generated by the **Mallows Model** for varying levels of consensus
- Consensus increases with theta in the Mallows Model
- Model the varying lengths of submitted top-lists with a Poisson distribution



Poisson Visual Credits: Dr. Matt Bognar, University of Iowa

Tai

- Mallows is a statistical model for ranks, has a ground truth ranking, which is the most common ranking. As theta grows, ranks that are very different from the ground truth ranking become rarer
- Ali and Meila found that performance and time varied with consensus level - intuitively, high-consensus regimes have fewer swaps between the provided lists, hence the better scores above when there is more consensus.
- Poisson distribution chosen because it is left-skewed for small k, modelling situations where voters tend to rank a small number of candidates. Voters may not submit exactly the same length lists, so lengths are sampled from a left-skew Poisson distribution rather than making all voters submit top-lists of identical length

Experience thus far, hurdles, and what's next...

- Finding a library for generating synthetic data from a **Mallows Model*** for top-lists
- Learning **Integer Programming** for exact solutions because brute force was very slow
- Gave up on one algorithm that was a **PTAS** that would call multiple other PTASes as black boxes
 - Would have been very slow for reasonably small epsilon
 - Very hard to implement (even the PTAS co-designer advised against it!)
- Examine real-world data-sets from PrefLib
- Implement algorithms not considered in Simon and Mathieu
- Model **voting blocks** in synthetic data



*Fabien, Collas, and Irurozki Ekhine. Concentric mixtures of Mallows models for top-k rankings: sampling and identifiability.

We both speak, tai first

Define words in bold while presenting:

Voting Block consists of with preferences similar to each other, allowing us to consider cases where groups internally agree but disagree with other groups. Not considered in full-list aggregation literature that we could see.

An **integer programming** problem is a **mathematical optimization** or **feasibility** program in which some or all of the variables are restricted to be **integers**. In many settings the term refers to **integer linear programming** (ILP), in which the objective function and the constraints (other than the integer constraints) are **linear**.

Integer programming is **NP-complete**. In particular, the special case of 0-1 integer linear programming

A **PTAS** is an algorithm which takes an instance of an optimization problem and a parameter $\epsilon > 0$ and, in polynomial time, produces a solution that is within a factor $1 + \epsilon$ of being optimal



Resources and Papers

- Ailon, Nir. "Aggregation of partial rankings, p-ratings and top-m lists." *Algorithmica* 57.2 (2010): 284-300. **[Paper]**
- Ali, Alnur, and Marina Meilă. "Experiments with Kemeny ranking: What works when?" *Mathematical Social Sciences* 64.1 (2012): 28-40. **[Paper]**
- Fabien, Collas, and Irurozki Ekhine. "Concentric mixtures of Mallows models for top-k rankings: sampling and identifiability." *arXiv preprint arXiv:2010.14260* (2020). **[Code]**
- Mathieu, Claire, and Simon Maura. "How to aggregate Top-lists: Approximation algorithms via scores and average ranks." *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*. Society for Industrial and Applied Mathematics, 2020. **[Paper]**
- Nicholas Mattei and Toby Walsh. "PrefLib": A Library of Preference Data, Springer. **[Dataset]**
- Schalekamp, Frans, and Anke van Zuylen. "Rank aggregation: Together we're strong." *2009 Proceedings of the Eleventh Workshop on Algorithm Engineering and Experiments (ALENEX)*. Society for Industrial and Applied Mathematics, 2009. **[Paper]**
- **Python Libraries:** numpy, scipy, and pulp.
- **R Libraries:** ggplot2