An Unsupervised Learning Algorithm for Rank Aggregation

Alexandre Klementiev, Dan Roth, and Kevin Small

Department of Computer Science
University of Illinois at Urbana-Champaign
201 N. Goodwin Avenue, Urbana, IL 61801, USA
{klementi, danr, ksmall}@uiuc.edu

Abstract. Many applications in information retrieval, natural language processing, data mining, and related fields require a ranking of instances with respect to a specified criteria as opposed to a classification. Furthermore, for many such problems, multiple established ranking models have been well studied and it is desirable to combine their results into a joint ranking, a formalism denoted as *rank aggregation*. This work presents a novel *unsupervised* learning algorithm for rank aggregation (ULARA) which returns a linear combination of the individual ranking functions based on the principle of rewarding ordering agreement between the rankers. In addition to presenting ULARA, we demonstrate its effectiveness on a data fusion task across ad hoc retrieval systems.

1 Introduction

Ranking items is a fundamental computer and information sciences problem. Most closely associated with information retrieval [1], ranking has recently attracted significant attention from the machine learning [2–5] and natural language processing [6–8] communities. While classification is the standard task of inductive learning, many applications require the expressivity of ranking. A related, but less thoroughly studied problem is *rank aggregation*, where multiple existing rankings of an item set are combined into a joint ranking. In the information retrieval community, this is the *data fusion* problem [9, 10] and corresponds to deriving a document ranking based on the input of multiple retrieval systems. For domains where ranking algorithms exist which utilize different modalities or views over the data, rank aggregation is particularly appealing as these views are difficult to combine into a single model.

From a machine learning perspective, this work is most ostensibly related to [2] which extends ideas regarding expert ensembles [11] and boosting [12] to ranking. In addition to a different model representation, the fundamental difference is that our algorithm is an unsupervised learning algorithm. Another related vein is the study of deriving voting policies which satisfy specified axiomatic properties [13] (e.g. the *independence of irrelevant alternatives* [14]). Our algorithm is similar in that the input is a set of ranking functions and no supervised training is required. However, our work adaptively learns a parameterized linear combination to optimize the relative influence of individual rankers.

In the data fusion domain, one widely cited set of approaches is [15]. These solutions are deterministic functions which mix rankings heuristically, differing from our work in that we learn the mixing parameters. One data fusion approach which has learned parameters measuring the reliability of ranking functions is ProbFuse [9], where probabilities that specific rankers return relevant documents are combined to determine the rank ordering. A second supervised approach presented by [10] also resembles our approach in that they use a linear combination of ranking functions to determine the joint ranking. However, unlike both methods, we present an unsupervised learning algorithm.

This paper presents an *unsupervised* learning algorithm for rank aggregation (ULARA) based on a linear combination of ranking functions, guided by the simple but effective principle that the relative contribution of an individual ordering to the joint ranking should be determined by its tendency to agree with other members of the expert pool. To the best of our knowledge, ULARA prescribes the first method for learning a parameterized rank aggregation without supervision. The remainder of the paper proceeds as follows. In addition to deriving ULARA in section 2, we also demonstrate specific properties on synthetic data. Section 3 proceeds by presenting experimental results for a data fusion task. Finally, we conclude and describe future work in section 4.

2 Rank Aggregation Framework

Let $x \in \mathcal{X}$ denote an item (e.g. document) in the instance space and $\mathbf{x}$ represent a set of items (e.g. corpus) to be ranked relative to each other. Furthermore, let $q \in \mathcal{Q}$ represent a query and $r : \mathcal{Q} \times \mathcal{X} \rightarrow \mathbb{N}$ represent a ranking function (e.g. retrieval system) where the output represents the item position in the ranked list such that $r(q,x) < r(q,x')$ specifies that x is ranked higher than x' with respect to q. We use the notation $r_u \succ_{\mathcal{E}} r_v$ to signify that r_u is a better ranking function than r_v with respect to the application specific evaluation criteria $\mathcal{E}$. Examples of $\mathcal{E}$ include Spearman's rank correlation coefficient [16] for general ranking problems or F1 measure for information retrieval.

Given a set of ranking functions $\{r_i\}_{i=1}^N$, we desire an aggregate ranking function $R : \mathcal{Q} \times \mathcal{X} \rightarrow \mathbb{N}$ such that $R \succ_{\mathcal{E}} r_i$ for $i = 1, \ldots, N$. This work studies aggregation based on the functional form $R'(q,x) = \sum_{i=1}^N w_i \cdot r_i(q,x)$, where w_i are the linear combination parameters, noting that R' is the expected value of the rank if $\mathbf{w}$ is a probability distribution (i.e. $0 \leq w_i \leq 1$ for all i and $\sum_{i=1}^N w_i = 1$). This representation has been shown successful in supervised situations [10], which we look to extend to the unsupervised case. Our specific approach derives a surrogate supervision signal in the absence of labeled data, referred to as an *incidental supervision signal*. Given this seting, the technical goals are three-fold: (1) derive an incidental supervision signal, (2) develop a corresponding unsupervised learning algorithm to learn the parameters of R', and (3) demonstrate that this incidental supervision signal works well for multiple evaluation criteria $\mathcal{E}$.

2.1 Incidental Supervision based on Ranker Agreement

The fundamental intuition for our incidental supervision signal is that in non-adversarial situations, accurate ranking functions will rank items according to a similar conditional distribution while poor ranking functions will tend towards a more uniformly random ranking distribution. For our representation, rankers that tend to agree with the plurality of rankers will be given large relative weight, while rankers that tend to disagree will be given small relative weight. Let $\mu(q,x) = \frac{\sum_{i=1}^{N_x} r_i(q,x)}{N_x}$ signify the mean ranking where $N_x = \sum_{i=1}^{N} [\![r_i(q,x) \leq \kappa_i]\!]$[1] (i.e. the number of ranking functions which return a ranking higher than a threshold κ_i for x). For each item to be ranked, each ranker will return a value $r_i(q,x)$ which is used to measure agreement using the variance-like measure $\sigma_i(q,x) = [r_i(q,x) - \mu(q,x)]^2$. If this value is small for a given item ranking, the corresponding ranker agrees with other rankers for this item and should be given a large weight and vice versa. Therefore, if we use a scaled variance-like measure $\delta_i(q,x) = w_i \cdot \sigma_i(q,x)$ and sum this value across all $\{query, item, ranker\}$ triples, this statement leads to an optimization problem that minimizes the weighted variance of the aggregate ranking function over the component rankings,

$$\underset{\mathbf{w}}{\operatorname{argmin}} \sum_{q \in Q} \sum_{x \in \mathbf{x}} \sum_{i=1}^{N} \delta_i(q,x) \tag{1}$$

$$\text{s.t.} \quad \sum_{i=1}^{N} w_i = 1; \forall i, w_i \geq 0. \tag{2}$$

2.2 Unsupervised Learning Algorithm for Rank Aggregation

As opposed to optimizing this problem of section 2.1 directly, ULARA uses iterative gradient descent [17] to derive an effective online learning algorithm. Observing that $\delta(q,x)$ is linear in w, the gradient for equation 1 with respect to a single weight w_i is $\nabla = \sum_{q \in Q} \sum_{x \in \mathbf{x}} \sigma_i(q,x)$. To derive an update rule, we are interested in the gradient of the utility function components, $\delta_i(q,x)$, stated as

$$\nabla_i = \frac{\partial \delta_i(q,x)}{\partial w_i} = [r_i(q,x) - \mu(q,x)]^2 \tag{3}$$

and resulting in algorithm 1. ULARA takes as input a set of ranking functions $\{r_i\}_{i=1}^N$ along with the associated ranking function threshold values κ_i, a learning rate λ, and a significance threshold value θ, all discussed in greater detail below. Also, ULARA takes a set of queries Q of which we do not know the true ranking. In general, for each item x and query q, the expert ranking for each of the N rankers are determined using $r_i(q,x)$, the mean is calculated (line 8), the gradient is determined (line 11), and the weight update is made (line 14/15). Once all of these updates are completed, the weight vector is normalized (line 17) to generate a probability vector for evaluation in algorithm 2. The remaining discussion entails algorithmic details to accommodate practical situations:

[1] $[\![p]\!] = 1$ if the predicate p is true; else $[\![p]\!] = 0$.

Algorithm 1 ULARA - Training

```
1: Input: $Q, \{r_i, \kappa_i\}_{i=1}^N, \lambda, \theta$
2: $\mathbf{w} \leftarrow \mathbf{0}$ {additive}
3: $\mathbf{w} \leftarrow \frac{\mathbf{1}}{\mathbf{N}}$ {exponentiated}
4: $t \leftarrow 1$
5: for all $q \in Q$ do
6:     for all $x \in \mathbf{x}$ do
7:         if $N_x \geq \theta$ then
8:             $\mu(q_q, x) = \frac{\sum_{i=1}^{N_x} r_i(q,x)}{N_x}$
9:             for $i \leftarrow 1, \ldots, N$ do
10:                if $r_i(x) \leq \kappa_i$ then
11:                    $\nabla_i \leftarrow [r_i(q,x) - \mu(q,x)]^2$
12:                else
13:                    $\nabla_i \leftarrow [\kappa_i + 1 - \mu(q,x)]^2$
14:                $w_i^t \leftarrow w_i^{t-1} + \lambda \cdot \nabla_i$ {additive}
15:                $w_i^t \leftarrow \frac{w_i^{t-1} e^{-\lambda \nabla_i}}{\sum_{j=1}^N w_j^{t-1} e^{-\lambda \nabla_i}}$ {exp.}
16:            $t \leftarrow t + 1$
17: NORMALIZE($w$) {additive}
18: Output: $\mathbf{w} \in [0,1]^N$
```

Algorithm 2 ULARA - Evaluation

```
1: Input: $q, \mathbf{w}, \{r_i, \kappa_i\}_{i=1}^N$
2: for all $x \in \mathbf{x}$ do
3:     $R_x \leftarrow 0$
4:     for $i \leftarrow 1, \ldots, N$ do
5:         if $r_i(q,x) \leq \kappa_i$ then
6:             $R_x \leftarrow R_x + w_i \cdot r_i(q,x)$
7:         else
8:             $R_x \leftarrow R_x + w_i \cdot (\kappa_i + 1)$
9: ARGSORTASCENDING($R_x$)
10: $R(q,x) \leftarrow$ RANKING($R_x$)
11: Output: $R : \mathcal{Q} \times \mathcal{X} \rightarrow \mathbb{N}$
```

Fig. 1: An Unsupervised Learning Algorithm for Rank Aggregation (ULARA). Note that lines 2, 14, and 17 are only used in the case of additive updates and lines 3 and 15 are only used in the case of exponentiated updates.

– Additive vs. Exponentiated Updates - Two methods for gradient descent are additive and exponentiated updates [17], each with specific properties that will be explored. If using additive updates, lines 3 and 15 should be removed. If using exponentiated updates, lines 2, 14, and 17 should be removed noting that normalization is not necessary.
– Missing Rankings (κ_i) - For most settings, there are more items in the instance space than the individual ranking functions will return. For data fusion, systems return rankings for a fixed set of documents and most corpus documents remain unranked. We denote this threshold value as κ_i, noting that ranking functions may have different thresholds. If a ranking is not returned by a specific ranker, we substitute $\kappa_i + 1$ for update calculations (line 13), assuming unranked items are ranked just below the last ranked item.
– Learning Rate (λ) - For ULARA$_{\text{add}}$, $\lambda = 1$ was used for all experiments. For ULARA$_{\text{exp}}$, λ was set proportional to κ^{-2}.
– Variable Number of Rankers (θ) - For some items, only a small fraction of the ranking functions return a valid rank. If less than θ rankers, as defined by the user, return a valid rank for an item, the information is deemed untrustworthy and no updates are made for this item (line 7).

Studying ULARA via Controlled Experiments We first explore ULARA on synthetic data to: (1) examine robustness (2) compare additive and exponentiated updates. We begin by generating $M = 1000$ items with rank designated as $r_\star(x) = \{1, 2, \ldots, 1000\}$. We then specify $N = 14$ ranking functions $r_i(x)$ with varying degrees of perturbation from the true ranking by perturbing $r_\star(x)$ by a random sample from the discrete uniform distribution with window size ω. Formally, $r_i(x) \sim \mathcal{U}\left[\max\left(0, r_\star(x) - \frac{\omega}{2} - \epsilon\right), \min\left(r_\star(x) + \frac{\omega}{2} + \epsilon, M\right)\right]$ where $\epsilon \geq 0$ is the amount necessary to maintain window size ω. Ties are broken with the policy that all tied items receive an the average rank of the item positions as per [16]. $|Q| = 50$ queries were generated for $N = 14$ ranking function with two $r_i(x)$ where $\omega = 200$, two $r_i(x)$ where $\omega = 600$, and 10 $r_i(x)$ where $\omega = 1000$.

Figure 2 displays performance of both ULARA$_{\text{add}}$ and ULARA$_{\text{exp}}$ on the synthetic data. As a baseline, we modify the $CombMNZ$ [15] to use rank information as opposed to the underlying real-valued score since we do not have this information in our task; $CombMNZ_{rank} \leftarrow \sum_{i=1}^{N} r_i(q, x) \cdot N_x$. We use a modified $CombMNZ$ as it is unsupervised (albeit without learning) and widely used for data fusion. For evaluation, we use Spearman's rank correlation coefficient, $\rho \in [-1, 1]$ [16]. ULARA$_{\text{exp}}$ achieves an increase of 0.28 (39% relative increase) in ρ relative to $CombMNZ_{rank}$ after 40 queries and ULARA$_{\text{add}}$ achieves an increase of 0.24 (35% relative increase) in ρ after only 5 queries, demonstrating that ULARA can effectively weigh rankers without explicit supervision.

The second result is presented in figure 3, showing the average weights associated with the rankers in the three groups (ω = 200, 600, and 1000) during training. ULARA assigns the most weight to the best ranking functions, some weight to the reasonable ranking functions, and almost no weight to the worst ranking functions. In accordance with theory [17], ULARA$_{\text{exp}}$ tends to select the best ranking functions by assigning all of the weight to these rankers. Conversely, ULARA$_{\text{add}}$ tends to mix the ranking functions relative to their performance, which proves beneficial in cases where many reasonable rankers exist and can complement each other in different areas of the item distribution.

3 Data Fusion

The real-world task we study empirically is data fusion [9], utilizing data from the ad hoc retrieval shared task of the TREC-3 conference. For this task, each group of $N = 40$ shared task participants was provided with a set of documents and $|Q| = 50$ of queries, returning the $\kappa = 1000$ documents for which the expected relevance is greatest for each query. ULARA was used to combine the rankings of the individual research groups into an aggregate ranking function R. Performance is quantified by the precision/recall curves and mean average precision metric as provided by the software (`trec_eval`) from the TREC conference series [18].

Figure 4 shows the results of the top individual submissions, $CombMNZ_{rank}$, and ULARA for the data fusion task. We observe that ULARA outperforms all component ranking functions as well as $CombMNZ_{rank}$. More significantly,

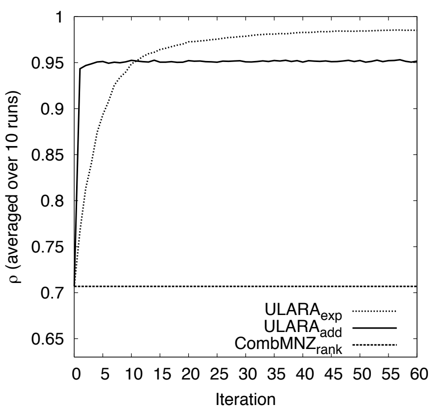

Fig. 2: Experimental results on synthetic data comparing ULARA to $CombMNZ_{rank}$. Exponentiated updates achieve a 39% relative increase compared to $CombMNZ_{rank}$ in the Spearman's ranking coefficient with only 40 queries, while additive updates achieve a 35% relative increase in only 5 queries.

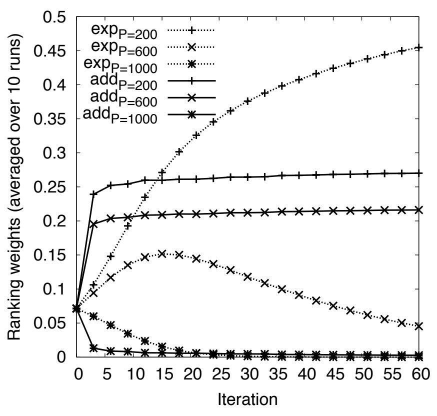

Fig. 3: ULARA ranker weights throughout training on synthetic data, noting the best rankers are weighted the highest, while the poorest rankers have negligible weight. Furthermore, exponentiated updates tends to select the best rankers while additive updates mix rankers relative to their performance.

while $CombMNZ_{rank}$ performs slightly better than the top system, ULARA$_{\text{add}}$ achieves a relative increase in average precision of 4.0% at the top ranking, 6.4% at 0.1 recall, and 6.0% at 0.2 recall over $CombMNZ_{rank}$. ULARA$_{\text{exp}}$ achieves a relative increase in average precision of 4.3% at the top ranking, 7.7% at 0.1 recall, and 8.1% at 0.2 recall over $CombMNZ_{rank}$. These results are significant in that they demonstrate that ULARA can not only distinguish good ranking functions from bad as in the synthetic data task, but in practice can generate a joint ranking function superior to its best components on the data fusion task.

The second experiment demonstrates the robustness properties of ULARA by looking not at the situation where we are working with world-class retrieval systems, but at the hypothetical situation where many of the ranking systems were poor. Specifically, we replaced a specified number of the $N = 40$ systems with a rankings drawn uniformly from all documents returned by all systems for a given query, denoted as *random rankings*. As figure 5 shows, the mean average precision of ULARA versus $CombMNZ_{rank}$ is consistently superior, becoming more pronounced as the number of random rankings is increased. To further explore this effect, we varied θ and observe that as more noise is added, θ must be lowered to accommodate the lack of agreement between rankers. However, even under relatively extreme cicumstances, ULARA returns a joint ranking function competitive with a noise free system. This experiment demonstrates that ULARA is capable of discounting bad rankers in a real-world setting such that they are not detrimental to the aggregate ranking function R.

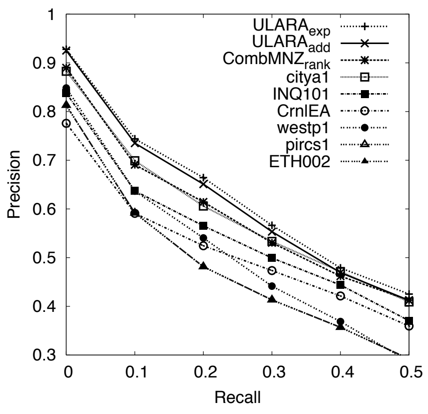

Fig. 4: Experimental results for data fusion of the retrieval systems submitted to the TREC-3 shared task. While $CombMNZ_{rank}$ only negligibly outperforms the top system, ULARA performs significantly better than any component system at multiple recall levels.

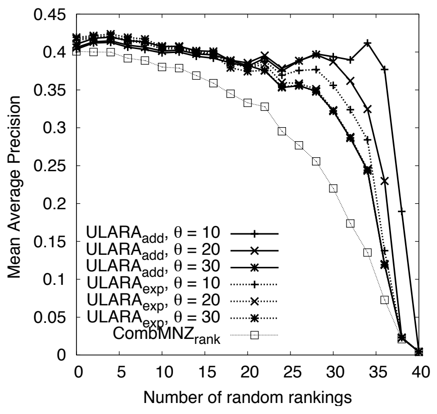

Fig. 5: Experimental results where a number of TREC-3 systems are replaced with random rankings, demonstrating robustness of ULARA. While the performance of the $CombMNZ_{rank}$ algorithm deteriorates rapidly, ULARA performs well even when more than half of the systems are replaced.

4 Conclusions and Future Work

We have presented a novel approach to the rank aggregation problem by specifying an optimization problem to learn a linear combination of ranking functions which maximizes agreement. Secondly, we introduce an unsupervised learning algorithm, ULARA, to solve this problem. This criteria is driven by the assumption that correctly ranked instances will possess a similar position in multiple ranking functions, allowing us to assign a high weight to rankers that tend to agree with the expert pool and reduce the influence of those rankers which tend to disagree. We have successfully demonstrated the effectiveness of our algorithm in two diverse experimental settings which each use a different evaluation function: on synthetic data which quantifies performance with Spearman's rank correlation coefficient and an information retrieval data fusion task which quantifies performance using precision/recall. For future work, we have already generated preliminary results extending ULARA to generalize a reranking approach [6] to named entity discovery [8], which we expect to pursue further.

Acknowledgments

This work is supported by NSF grant NSF ITR IIS-0428472 and by MIAS, a DHS Institute of Discrete Science Center for Multimodal Information Access and Synthesis.

References

1. Baeza-Yates, R.A., Ribeiro-Neto, B.A.: Modern Information Retrieval. ACM Press / Addison-Wesley (1999)
2. Cohen, W.W., Schapire, R.E., Singer, Y.: Learning to order things. Journal of Artificial Intelligence Research **10** (1999) 243–270
3. Agarwal, S., Roth, D.: Learnability of bipartite ranking functions. In: Proc. of the Annual ACM Workshop on Computational Learning Theory (COLT). (2005) 16–31
4. Clémençon, S., Lugosi, G., Vayatis, N.: Ranking and scoring using empirical risk minimization. In: Proc. of the Annual ACM Workshop on Computational Learning Theory (COLT). (2005) 1–15
5. Rudin, C., Cortes, C., Mohri, M., Schapire, R.E.: Margin-based ranking and boosting meet in the middle. In: Proc. of the Annual ACM Workshop on Computational Learning Theory (COLT). (2005) 63–78
6. Collins, M., Koo, T.: Discriminative reranking for natural language parsing. In: Proc. of the International Conference on Machine Learning (ICML). (2000) 175–182
7. Li, H., Rudin, C., Grishman, R.: Re-ranking algorithms for name tagging. In: HLT/NAACL Workshop on Joint Inference. (2006)
8. Klementiev, A., Roth, D.: Weakly supervised named entity transliteration and discovery from multilingual comparable corpora. In: Proc. of the Annual Meeting of the Association for Computational Linguistics (ACL). (July 2006) 817–824
9. Lillis, D., Toolan, F., Collier, R., Dunnion, J.: Probfuse: A probabilistic approach to data fusion. In: Proc. of the International ACM SIGIR Conference on Research and Development in Information Retrieval. (2006) 139–146
10. Vogt, C.C., Cottrell, G.W.: Fusion via a linear combination of scores. Information Retrieval **1**(3) (1999) 151–173
11. Cesa-Bianchi, N., Freund, Y., Helmbold, D.P., Haussler, D., Schapire, R.E., Warmuth, M.K.: How to use expert advice. Journal of the Association for Computing Machinery **44**(3) (1997) 427–485
12. Freund, Y., Schapire, R.E.: A decision-theoretic generalization of on-line learning and an application to boosting. Journal of Computer and System Sciences **55**(1) (1997) 119–139
13. Conitzer, V.: Computational Aspects of Preference Aggregation. PhD thesis, Carnegie Mellon University (2006)
14. Arrow, K.: Social Choice and Individual Values. 2nd edn. Cowles Foundation, New Haven (1963)
15. Shaw, J.A., Fox, E.A.: Combination of multiple searches. In: Text REtrieval Conference (TREC). (1994) 243–252
16. Kendall, M., Gibbons, J.D.: Rank Correlation Methods. 5th edn. Edward Arnold, London (1990)
17. Kivinen, J., Warmuth, M.K.: Additive versus exponentiated gradient updates for linear prediction. In: Proc, of the Annual ACM Symposium on Theory of Computing. (1995) 209–218
18. Voorhees, E.M., Buckland, L.P., eds.: The Fourteenth Text REtrieval Conference Proceedings (TREC). (2005)