An Unsupervised Distance-Based Model for Weighted Rank Aggregation with List Pruning

Leonidas Akritidis[a] (lakritidis@ihu.gr), Athanasios Fevgas[b] (fevgas@e-ce.uth.gr),
Panayiotis Bozanis[a] (pbozanis@ihu.gr), Yannis Manolopoulos[c] (yannis.manolopoulos@ouc.ac.cy)

[a] School of Science and Technology, International Hellenic University, Greece. Address: 14th km Thessaloniki - N. Moudania, Thermi, Thessaloniki, Greece.
[b] Department of Electrical and Computer Engineering, University of Thessaly, Greece. Address: Sekeri - Cheiden Str., Pedion Areos, ECE Building, 2nd floor, Volos, Greece.
[c] Faculty of Pure & Applied Sciences, Open University of Cyprus, Cyprus. Address: 33, Yiannou Kranidioti Avenue, 2220 Latsia, Nicosia, Cyprus.

Corresponding Author:
Leonidas Akritidis
School of Science and Technology, International Hellenic University, Greece.
Address: 14th km Thessaloniki - N. Moudania, Thermi, Thessaloniki, Greece.
Tel: (030) 6944 946071.
Email: lakritidis@ihu.gr

An Unsupervised Distance-Based Model for Weighted Rank Aggregation with List Pruning

Leonidas Akritidis[a,*], Athanasios Fevgas[b], Panayiotis Bozanis[a], Yannis Manolopoulos[c]

[a] *School of Science and Technology, International Hellenic University, Thessaloniki, Greece*
[b] *Department of Electrical and Computer Engineering, University of Thessaly, Volos, Greece*
[c] *Faculty of Pure & Applied Sciences, Open University of Cyprus, Nicosia, Cyprus*

Abstract

Combining multiple ranked lists of items, called voters, into a single consensus list is a popular problem with significant implications in numerous areas, including Bioinformatics, recommendation systems, metasearch engines, etc. Multiple recent solutions introduced supervised and unsupervised techniques that try to model the ordering of the list elements and identify common ranking patterns among the voters. Nevertheless, these works either require additional information (e.g. the element scores assigned by the voters, or training data), or they merge similar voters without the evidence that similar voters are important voters. Furthermore, these models are computationally expensive. To overcome these problems, this paper introduces an unsupervised method that identifies the expert voters, thus enhancing the aggregation performance. Specifically, we build upon the concept that collective knowledge is superior to the individual preferences. Therefore, the closer an individual list is to a consensus ranking, the stronger the respective voter is. By iteratively correcting these distances, we assign converging weights to each voter, leading to a final stable list. Moreover, to the best of our knowledge, this is the first work that employs these weights not only to assign scores to the individual elements, but also to determine their population. The proposed model has been extensively evaluated both with well-established TREC datasets and synthetic ones. The results demonstrate substantial precision improvements over three baseline and two recent state-of-the-art methods.

Keywords: information retrieval, metasearch, weighted rank aggregation, unsupervised data fusion, rank aggregation, ranking

1. Introduction

Rank aggregation is a well-studied problem with numerous applications in diverse fields of science, economy, and society. In general, rank aggregation methods collect individual lists of ranked items from various sources that may represent entities of any type including users, preferences, products, suggestions, events, etc. Then, they merge all the input lists into a single aggregate list by applying a data fusion technique, and they rearrange the elements of the aggregate list to generate an improved consensus ranking.

The problem of rank aggregation was firstly introduced over two centuries ago to satisfy the requirements of a fair election system, both single and multi-winner (Bartholdi et al., 1989; Kilgour, 2010). Since then, numerous state-of-the-art approaches have been proposed to confront problems related to Web metasearch (Akritidis et al., 2011; Wang et al., 2017), Bioinformatics (Chen et al., 2016; Li et al., 2019), Web spam detection (Dwork et al., 2001), Natural Language Processing (Rosti et al., 2007; Reyes Ayala et al., 2018), social choice platforms (Caragiannis et al., 2019), collaborative filtering (Chatterjee et al., 2018), and so on.

*Corresponding author.
Email addresses: lakritidis@ihu.gr (Leonidas Akritidis), fevgas@e-ce.uth.gr (Athanasios Fevgas), pbozanis@ihu.gr (Panayiotis Bozanis), yannis.manolopoulos@ouc.ac.cy (Yannis Manolopoulos)

Hence, despite it is a fairly old problem, rank aggregation is still challenging for the research community due to its significant adoption by a multitudinous family of expert systems.

For this reason, the relevant literature includes a large number of supervised and unsupervised solutions to the problem. The supervised algorithms exhibit two significant drawbacks. First, they require the existence of training data, a requirement that is frequently expensive and hard to satisfy (Klementiev et al., 2008). The second issue is that the models learned by the supervised methods do not generalize well; in other words, they work satisfactorily only on the particular problem in which they have been trained. For instance, a supervised aggregation model that has been trained on the results of multiple Web search engines to support a metasearch engine will potentially be ineffective in other types of applications such as Bioinformatics, or collaborative filtering.

On the other hand, the unsupervised methods do not require training data and do not exhibit the aforementioned generalization problem, because they always operate on previously unseen data. However, the absence of training data removes ground truth, and limits the ability for optimization (e.g., minimize a cost function); this usually leads to suboptimal solutions. The robust unsupervised models overcome this limitation by employing exploratory analysis techniques, namely, by identifying hidden useful patterns, structure, and information in the underlying data (Díaz et al., 2008).

Attracted by the advantages of the unsupervised methods and motivated by i) the importance of the problem and ii) the challenge of discovering hidden information within the ranked lists themselves, we introduce an unsupervised method that automatically identifies the experts among a set of voters. Our research objectives are, firstly, to identify and, subsequently, exploit this valuable information with the aim of enhancing the quality of the aggregate list.

The majority of the state-of-the-art algorithms in the area are *unweighted*, that is, they consider that all voters are of equal importance and treat them equivalently (Aslam & Montague, 2001; Aledo et al., 2021). Nevertheless, in numerous cases (e.g., in recommender systems), some voters may have a greater knowledge, or a richer experience about a subject than others. Hence, their opinion (i.e. their rankings) should be assigned higher weights compared to the rankings of the normal, or irrelevant voters. The unweighted methods ignore the existence of the expert voters, failing to incorporate this useful information in the introduced models.

In the context of weighted rank aggregation, Pihur et al. (2007) treated the problem as an optimization problem and introduced a weighted list distance function. However, this function requires knowledge not only of the item rankings within the input lists, but also of the individual scores assigned by the voters to the items. Consequently, this method is inapplicable to scenarios where these scores are not available. Furthermore, it is computationally very expensive, even for short input lists (Onan & Korukoğlu, 2017).

On the other hand, Chatterjee et al. (2018) proposed a hierarchical algorithm that repeatedly merges the two closest rankings until a single output list is produced. The weight of each merged list is calculated from the weights of the two parent lists. A weakness of this approach is revealed when two or more low quality rankings are very similar. These rankings will be merged early, producing a weighted list that will propagate to the subsequent list merges. Moreover, the agglomerative methods introduce a cubic time complexity, and a quadratic space complexity. On the other hand, the majoritarian method of Desarkar et al. (2016) assigns weights by comparing the item rankings and by counting the number of winners. This work focuses on metasearch applications that usually involve few short input lists, whereas its computational cost is high.

To overcome these limitations, this paper introduces an unsupervised rank aggregation algorithm that quantifies the importance of the voters and exploits this information to generate an improved consensus list. The proposed method initially establishes a converging form for the voter weights. Then, it repeatedly updates the weights at each iteration by measuring the distances from a temporary aggregate list. The temporary list is updated at each iteration, leading to a final stable list when the voter weights converge. Furthermore, we enhance this algorithm by proposing a novel list pruning method that employs the learned voter weights to adjust the number of items that will participate in the aggregation process.

Our approach builds upon the idea that the important voters can be identified by the quality of their suggested elements. In the context of rank aggregation, the quality of an element can be determined by several factors, such as the number of the voters who included it in their submitted lists, its rankings in these lists, and so on. In other words, the quality of an element is reflected by its position in the final aggregate

list, because this position depends on the aforementioned factors. This means that the importance of a voter is connected to the number of the suggested elements that have been placed near the top of the aggregate list. Then, it is naturally derived that the weight of a voter depends on the distance between the submitted list and the aggregate list.

The introduced algorithm cures the weaknesses of the aforementioned solutions. In summary, the contributions of this work are:

- We introduce an unsupervised rank aggregation algorithm that automatically quantifies the degree of expertise of each voter. The proposed method builds upon the concept that the importance of the voters is reflected by the distance of their submitted lists from the aggregate list. According to the data analysis that we conducted, this concept is statistically significant.
- In contrast to the score-based methods, the suggested algorithm takes into consideration only the individual rankings of the elements within the input lists, and does not rely on the element scores, in opposition to the approach of Pihur et al. (2007). Consequently, its usefulness is broader, since it is applicable to problems where this additional information (i.e., the individual item scores) is missing.
- Instead of performing pairwise list comparisons, our method updates the voter weights at each iteration, by measuring the distances between each input list and a temporary aggregate list. This approach has two important benefits. Firstly, the utilization of a temporary aggregate list allows us to avoid expensive optimization problems (Pihur et al., 2007; Coppersmith et al., 2010). And, secondly, it is less prone to local problems that occur when two or more low quality rankings happen to be very similar.
- To the best of our knowledge, this is the first work to study how the learned weights can be exploited not only to assign scores to the individual elements, but also to adjust the lengths of the input lists. This goal is achieved by setting special cut-off points that determine the number of items from each voter that will be taken into account during aggregation. The rationale is that the low ranked elements of the less important voters are apparently of small value and should be discarded; otherwise, they would append undesired noise.
- We also introduce a novel distance function named *CODRA (Cosine Distance for Rank Aggregation)* that originates from the well-established cosine similarity metric. We experimentally demonstrate that, compared to the traditional list correlation measures such as Spearman's footrule distance, CODRA yields better results.
- We evaluate the performance of the introduced algorithm by using both synthetic and real datasets with different attributes (number of voters, list lengths, etc.). The statistical analysis of the data indicates a significant correlation between the importance of the voters and the distances of their lists from the aggregate list. Moreover, we demonstrate the superiority over the baseline and competitive methods.

The rest of the paper is organized as follows: The most important relevant works are presented in Section 2. Section 3 describes the necessary background elements of the problem and establishes the article's notation and research objectives, whereas Section 4 analyzes the details of the proposed model for learning the voter weights. Moreover, in Section 5, two methods for exploiting the weights are proposed, and a list pruning method based on them is introduced. The results of the experimental evaluation of the model are presented in Section 6, and discussed in Section 7. Finally, Section 8 summarizes the conclusions of the paper and provides some insights of our future work.

2. Related Work

The current relevant literature includes a significant number of unsupervised methods that introduce state-of-the-art solutions to the problem of rank aggregation. These solutions can be classified into several categories according to the method they apply to perform aggregation.

More specifically, the score-based methods take into consideration the scores assigned by the voters themselves to the individual elements of their submitted lists (Amin & Emrouznejad, 2011). Since, in general, these approaches introduce aggregate scoring functions, their most significant drawback is that they cannot be applied when the item scores are not provided by the data sources (e.g., Web metasearch, recommender systems, etc.), or they are assigned in a non uniform fashion.

On the other hand, the positional methods assign scores based solely on the individual rankings of the examined elements. The most popular positional method is Borda Count (de Borda, 1781), introduced nearly two centuries ago with the aim of providing an unbiased system for fair elections. It has been used in numerous applications, ranging from voting platforms to social choice systems and metasearch engines. In most cases, it has been proved both fast and effective. Hence, it attracted many researchers to further study its properties (Emerson, 2013). Another group of methods identify Condorcet winners in accordance to the Condorcet criterion; that is, elements that win (ranked higher than) every other element in the input lists (De Condorcet, 1785; Montague & Aslam, 2002). The execution of pairwise comparisons between every element in the dataset leads to a quadratic time complexity, rendering the majoritarian methods computationally expensive, or even infeasible.

The works of Wang et al. (2017) and Akritidis et al. (2008) studied the problem of rank aggregation from the perspective of personalized Web metasearch. They both developed effective score functions that incorporate user-oriented data such as geographic location, language, user interest distribution, and so on. In addition, Akritidis et al. (2011) introduced QuadRank, a personalized method that takes into account additional IR features regarding the individual elements themselves. Another work which exploits the meta-information of the objects to be ranked was presented by Bhowmik & Ghosh (2017). This method augments the standard approaches and aids the recovery of the "true" rank order. Interestingly, Dwork et al. (2001) considered the elements of the input lists as states, and employed Markov chains to address the problem.

Additionally, a portion of the relevant research adopted the Kemeny optimal aggregation criterion and introduced approaches that optimize the average distance between the aggregate list and the input lists (Ailon et al., 2008; Coppersmith et al., 2010). Notice that Kemeny optimal aggregation is a NP-hard problem, even when the input lists are few, e.g., 4 (Dwork et al., 2001).

Regarding the applications of rank aggregation in multi-criteria decision making, the method of Samanlioglu & Ayağ (2021) employed hesitant fuzzy processes to calculate the criteria weights by using dependence and feedback interactions. Omidi et al. (2021) applied the technique for order preference by similarity to an ideal solution (termed TOPSIS) to rank the impact of various climate-related factors on safety performance. On the other hand, Tseng et al. (2021) proposed a fuzzy Delphi method in combination with a bibliometric approach that identifies and aggregates several indicators in order to solve supply chain problems.

Similarly to our work, Pihur et al. (2007) took into consideration the weights of the input lists during rank aggregation. Their optimization problem is based on a weighted variant of Spearman's footrule distance that is used to measure the list distances. Nevertheless, this variant requires knowledge not only of the item rankings within the input lists, but also of the individual scores that the items received by their voters. Hence, similarly to all score-based methods, its functionality is limited when these scores are not provided. Moreover, the combinatorial nature of the model renders it prohibitively expensive even for small-sized datasets (Onan & Korukoğlu, 2017). In contrast to these approaches, the proposed algorithm does not minimize the distance of the aggregate list from all individual input lists, but only from those that have been submitted by experts. Moreover, we avoid expensive distance optimizations and propose a converging form for the voter weights.

The usage of probabilistic models, such as the Mallows models (Mallows, 1957; Lebanon & Lafferty, 2002), on permutations is a common technique encountered in a handful of works. These approaches compute a distance-based probability for each permutation (Klementiev et al., 2008, 2009), and select the one that maximizes this probability. Their greatest drawback is their high computational complexity which is $O(n!)$. The approach of Clémençon & Jakubowicz (2010) employs the Luce model, and uses generalized Kantorovich distances between rankings. Its $O(n^6)$ complexity is also prohibitively high.

The method of Caragiannis et al. (2019) assumes that partial access to an underlying true ranking is available, and computes a scoring rule with respect to it. By applying this rule to the profile of individual rankings, an outcome that is as close as possible to the aforementioned true ranking is obtained. Nevertheless,

since the existence of a ground-truth ranking is necessary, this approach is rather considered as supervised. Reyes Ayala et al. (2018), used rank aggregation methods to combine the outputs of multiple search engines with the aim of comparing the performance of three machine translation techniques.

Another state-of-the-art paper employed the popular Bradley-Terry-Luce model, assuming that the pairs can be selected in a random and non-adaptive fashion (Klementiev et al., 2008). The method of Cohen et al. (1998) learns a query-independent vector that represents the voter weights, and then, it uses this vector to perform rank aggregation. Moreover, Farah & Vanderpooten (2007) introduced an outranking approach, whose operation also depends on pairwise comparisons, accompanied by four user-defined thresholds that determine the degree of agreement or disagreement between the submitted input lists.

A weighted aggregation algorithm that is related to the proposed one was introduced by Desarkar et al. (2016). The authors initially consider each input ranking as a preference graph, and derive the aggregate list by aggregating these preference graphs. Simultaneously, the voter weights are determined by checking the truth of several rules, based on pairwise comparisons between the list elements. As a majoritarian algorithm, this method has a quadratic time complexity. In addition, it does not examine how the computed weights can be further exploited. In contrast, we study the possibility of introducing a list pruning mechanism that increases the performance substantially in many cases.

Inspired by the theory of clustering algorithms, the study of Chatterjee et al. (2018) introduced a weighted aggregation algorithm that works similarly to agglomerative clustering. Initially, the two closest input lists are merged. The resultant list is assigned a weight that derives from the weights of the parent lists. The process continues by merging the two closest lists at each step, until a single output list is produced. As an agglomerative method, the introduced time complexity is cubic, whereas the space complexity is quadratic to the number of the lists. Moreover, while in clustering the objective is, indeed, to discover the most similar elements, in rank aggregation we are primarily interested in reducing the distance from a consensus ranking.

From the perspective of clustering, it would be more suitable to adopt an algorithm that minimizes the distances from a temporary artificial point that progressively moves towards an "optimal" place. Such a strategy would be in agreement with the Kemeny optimal aggregation criterion (Ailon et al., 2008; Coppersmith et al., 2010) and the optimization problem of Pihur et al. (2007). One may easily distinguish the similarity between this description and k-Means. Our proposed algorithm imitates k-Means to a degree, since it creates a temporary aggregate list that is modified in an iterative fashion, until its distances from all the input lists converge. Of course, in clustering, the primary goal is to group together similar elements, and, to achieve that purpose, multiple clusters are created. In rank aggregation, we build a single list based on the agreement or disagreement degree among the individual voters.

This paper is an extension of the preliminary unsupervised distance-based model that has been introduced by Akritidis et al. (2019). This preliminary model is further enhanced in this work by the following aspects:

- The learned weights are applied not only to modify the rankings of the input elements, but also to determine the number of results per voter that shall be utilized to build the aggregate list. The experiments on the effectiveness of this pruning method indicate substantial gains in retrieval effectiveness.

- It utilizes a variant of the popular cosine similarity measure to determine the list-wise distances in a more robust manner. Initially, the ranked lists are converted to weighted vectors according to several criteria, and, in the sequel, the cosine measure is employed to compute their distance.

- The iterative weight computation is improved by checking the convergence of the weight of each voter individually. Therefore, if the weight of a voter has converged, the execution flow ignores him/her, and continues with the next one. This action saves the redundant weight calculations of Akritidis et al. (2019), thus leading to enhanced efficiency.

- Additional experiments are conducted to attest the usefulness of the introduced model. The validity of the presented results is verified and strengthened by the selection of the test datasets. Some of them are synthetic ones, while the rest originate from the well-known TREC conference and are accompanied by human judgments on the relevance of the submitted items; so, effectiveness measurements are rendered both reliable and reproducible.

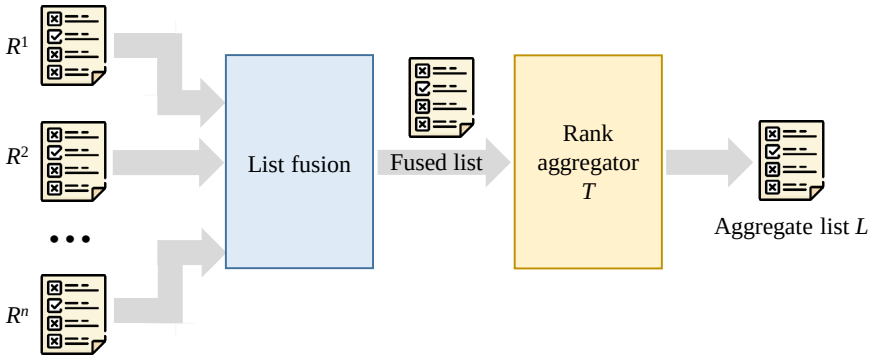

Figure 1: The execution flow of an unweighted rank aggregation algorithm.

3. Overview and Notation

In this section, we present the basic theoretical elements of the article, and we review the most popular distance metrics for full and partial lists. We also introduce a novel measure that treats the ranked lists as weighted vectors, and employs cosine similarity to determine their distance.

3.1. Preliminary Elements

The problem of rank aggregation considers a set V of n sources, also called *voters*, or *rankers*. Each voter $v \in V$ accepts a query q and generates an ordered list $R^v = \{r_1^v, \ldots, r_k^v\}$ of $k = |R^v|$ items in response to q. We call the R^v lists of the voters as *input lists*. The items in the input lists are arranged in decreasing importance, or relevance order. In other words, the first item is considered as the most important (or the most relevant) one, the second one is less important than the first one, and so on.

Apart from the items themselves, a voter may submit an additional score for each item. This value reflects the importance of an item, and justifies its position in the list. In this work, we consider that the items in the input lists are not accompanied by any score and that their ranking is the only available information. In this context, the proposed method is more general compared to the score-based methods, or the methods that require this information.

The notation r_i^v denotes the i^{th} item in the input list of v. In addition, we assume that all the items of the input lists are retrieved from a common universe U. A list that contains all the items of U is called a *full* list, or *permutation*; otherwise, it is a *partial* list. A top-k list is a partial list that contains the k most important items of U w.r.t. q. Notice that the proposed model operates on both full and partial lists.

Fig. 1 depicts an non-weighted rank aggregation method $\mathcal{T}$ that is fed with n ranked lists. Initially, $\mathcal{T}$ merges the input lists by eliminating their overlapping elements. The output of this process is a single aggregate list $L = \{r_1, r_2, \ldots\}, r_i \in U$ that has its items ordered according to the fusion algorithm and several additional criteria. On its simplest form, $\mathcal{T}$ considers that all voters are equivalent, therefore, their lists are processed in an identical manner. Then, L is determined by the following operation:

$$L = \mathcal{T}\big(R^{v_1}, R^{v_2}, \ldots, R^{v_n}\big). \tag{1}$$

In contrast, the weighted methods assign a weight w_v to each voter v. This value reflects the significance of the voter, and/or his/her trustworthiness, and/or his/her degree of expertise on a particular subject. In this case, $\mathcal{T}$ accepts both the input lists R^v and the aforementioned weights, and generates the aggregate list L in the following fashion:

$$L = \mathcal{T}\big(w_{v_1}, w_{v_2}, \ldots, w_{v_n}, R^{v_1}, R^{v_2}, \ldots, R^{v_n}\big). \tag{2}$$

The execution flow of a typical weighted rank aggregation method that implements Eq. 2 is illustrated in Figure 2.

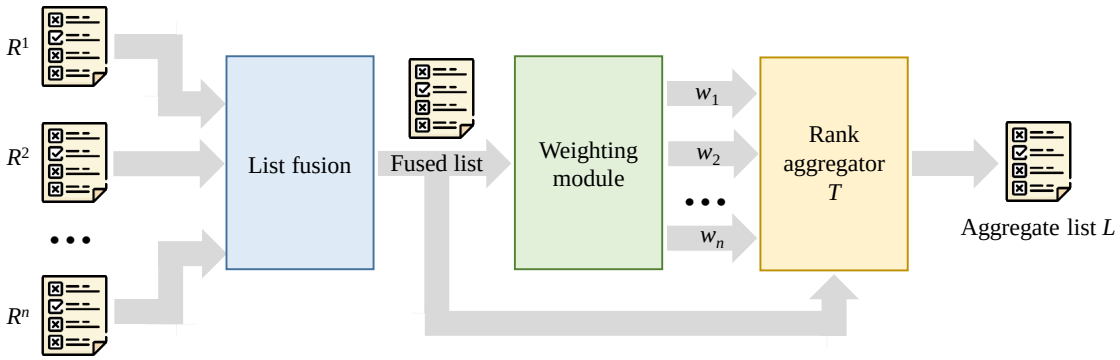

Figure 2: The execution flow of a typical weighted rank aggregation algorithm. The weighting module determines the voter weights and passes them to the aggregator $\mathcal{T}$ to satisfy Eq. 2.

3.2. Research Questions, Objectives, and Framework

The primary objective of rank aggregation research is the design of algorithms that improve the quality of the aggregate list L. Combining this goal with Fig. 2, the following research questions can be stated:

- How can the weighting module of Fig. 2 learn the importance of each voter in an unsupervised manner?
- Which mechanism better quantifies the voter importance while it simultaneously avoids the disadvantages of the current solutions, namely: i) the pairwise item comparisons of the majoritarian methods (Desarkar et al., 2016), ii) the pairwise list merges of the agglomerative algorithms (Chatterjee et al., 2018), and iii) the usage of the scores of the individual elements (which are frequently unknown)?
- How can the aggregation method $\mathcal{T}$ exploit the learned voter weights to improve the quality of the consensus ranking?
- Does the current literature provide all the necessary tools to achieve our research objectives?

These issues establish the research objectives of this article. In summary, the proposed method is inspired by the Kemeny optimal aggregation criterion and the optimization problem of Pihur et al. (2007) that determine an aggregate list by minimizing its distance from the input lists. Apparently, the distance of an input list from the aggregate list is indicative of its quality. Nonetheless, the optimization is frequently a very expensive procedure, whereas the algorithm of Pihur et al. (2007) requires knowledge of not only the the item rankings, but also their individual scores. Consequently, a more robust solution is required.

The research questions and objectives outline the contributions of this paper and have been used as a guideline for designing our research framework. Initially, since the list distances are a central tool in our approach, we examine the properties of the traditional functions, and we set an additional question/objective:

- Are the established list distance functions sufficient to achieve our objectives? Can we discover their weak points and introduce a more robust metric?

In this work, we introduce CODRA, a novel measure that quantifies the distance between two ranked lists. In contrast to the existing metrics, CODRA treats the ranked lists as vectors, and employs well-established vector similarity metrics such as cosine similarity.

In the sequel, we develop an unsupervised model that assigns distance-dependent weights to the voters. During this procedure, it is imperative that we avoid the three drawbacks that have been mentioned in the second question. The research framework requires the introduction of a converging kernel function that will progressively assign large (small) weights to the lists that are close to (away from) the aggregate list.

In the next phase, we examine additional strategies to exploit the learned weights. To the best of our knowledge, this work is the first to investigate such strategies. More specifically, the rationale is:

Domain	Aggregation of	Implications/Benefits
Politics, society	Ballots	Fair elections
E-commerce	Recommender systems	Increased profits, sales, traffic, user satisfaction
E-commerce, society	User reviews/opinions	Correct & targetted purchases, user satisfaction
Bioinformatics	Gene sequences	Enhanced models, more effective treatments
Machine learning	Classifier outputs	Ensemble models, enhanced classification accuracy
Information retrieval	Search engine results	Improved retrieval effectiveness, user satisfaction

Table 1: Broad implications and benefits of rank aggregation.

- Should a weak voter contribute the same number of elements in the aggregate list as a strong one?

For this purpose, we introduce a simple yet effective weight-dependent list pruning strategy that implements the aforementioned idea. According to it, the important voters (i.e., voters with high weights) deserve to contribute more elements to the aggregate list compared to the irrelevant ones. This policy increases the probability that more qualitative elements will be present in the consensus ranking, thus improving the performance of the proposed algorithm.

The above discussion is aligned to the theoretical and methodological contributions enlisted in Section 1. Now, let us elaborate the broader implications of this work. Nowadays, rank aggregation methods are utilized extensively in numerous scientific, societal and industrial applications. Undoubtedly, the introduction of a novel method that improves the current state-of-the-art leads to significant implications to these domains. Table 1 includes some indicative areas that will benefit from the application of the proposed method.

3.3. Distance Measures

The core of the proposed method includes the computation of the distance $d(R^v, L)$ between each input list R^v from the aggregate list L. Here, we review the most popular metrics that have been proposed in the relevant literature for computing these distances. For simplicity reasons, and without loss of generality, we shall drop the exponent v from R^v in this subsection. The simplified notation R refers to a ranked list submitted by any voter.

If the input lists are permutations of k items, then L is also a full list, and the distance between R and L can be computed by using standard metrics. One such metric is the Spearman footrule distance D_F, defined by the following equation:

$$d_F(R, L) = \sum_{j=1}^{k} |j - l_j|, \tag{3}$$

where j and l_j are the rankings of r_j in R and L, respectively. The distance of Eq. 3 is usually used on its normalized form, which derives by dividing $d_F(R, L)$ by $k^2/2$.

Another popular distance metric is Kendall's τ, which counts the number of pairwise disagreements between two lists. In other words, it is the minimum number of pairwise adjacent transpositions that are required to transform one permutation into the other:

$$\tau = d_K(R, L) = \frac{2}{n(n-1)} \sum_{i<j} \text{sgn}(i - j)\text{sgn}(l_i - l_j), \tag{4}$$

where $\text{sgn}(x)$ is the sign function with outputs 1, 0, and -1, when $x > 0$, $x = 0$, and $x < 0$, respectively.

If the individual lists R of the voters are not permutations but top-k lists, then only L is a full list. In this case, R and L have different lengths, and the aforementioned distance metrics are used in their *scaled* versions, based on the sizes of the involved lists. For example, the scaled footrule distance (SFD) is computed by using the following formula:

$$d_{SFD}(R, L) = \sum_{j=1}^{k} \left| \frac{j}{k} - \frac{l_j}{|L|} \right|, \tag{5}$$

that can be normalized by dividing the distance of Eq. 5 by $k/2$.

3.4. Weighted Cosine Distance

The aforementioned distance metrics are well-established and have been utilized by many researches on rank aggregation. Nevertheless, in several cases, their performance may become problematic for reasons that are explained below. The following list provides some illustrative examples.

- When it is applied on input lists of different lengths, SFD favors the short lists. In other words, it erroneously considers that the short lists are closer to L compared to similar but longer lists. This issue mostly occurs when the voters submit partial lists of variable lengths, a situation that is quite common in applications related to Web metasearch.

  Example 1. *Consider a list $L = \{a, b, c, d, e\}$ that has been generated by the aggregation of multiple input lists. Let $R^{(1)} = \{c, d, e\}$ and $R^{(2)} = \{c, d, e, a, b\}$ be two of these lists. Observe that $R^{(1)}$ comprises only 3 items. On the other hand, $R^{(2)}$ enlists the same 3 items in the same positions as $R^{(1)}$, and, moreover, it accommodates two additional elements (a and b), that also exist in L. In other words, $R^{(2)}$ has all its items in common with L and, intuitively, it is closer to L compared to $R^{(1)}$. According to Eq. 5, the scaled footrule distances of the two lists from L are $d_{SFD}(R^{(1)}, L) = |1/3 - 3/5| + |2/3 - 4/5| + |3/3 - 5/5| = 0.4$ and $d_{SFD}(R^{(2)}, L) = (|1-3| + |2-4| + |3-5| + |4-1| + |5-2|)/5 = 2.4$. That is, $d_{SFD}(R^{(2)}, L) > d_{SFD}(R^{(1)}, L)$. Even if we employ the normalized forms of these distances by dividing them with $k/2$, we get $d_{SFD}(R^{(1)}, L) = 0.267$, and $d_{SFD}(R^{(2)}, L) = 0.96$. Consequently, the issue is not resolved by normalization.*

- The previous example highlights another problem of SFD: namely, the assignment of very high distances to seemingly similar lists. For instance, the normalized SFD of $R^{(2)}$ from L was found equal to 0.96, a value that indicates that $R^{(2)}$ and L are almost completely unrelated. Nevertheless, this is not true according to the discussion of Example 1.

For these reasons, we devised a different strategy for the computation of the distance between two ranked lists L and R. More specifically, each item of L and R is initially assigned a ranking-dependent weight β; we shall provide more details on this procedure shortly. Then, the ranked lists can be expressed as vectors of these weights. Specifically, $\boldsymbol{\lambda} = \left(\beta_1^{(L)}, \beta_2^{(L)}, \dots\right)$, and $\boldsymbol{\rho} = \left(\beta_1^{(R)}, \beta_2^{(R)}, \dots\right)$, where $\beta_i^{(v)}$ represents the weight of the i^{th} element of list v. Notice that each element in a ranked list is transformed to a vector component which lies in a dimensional space that identifies U. Therefore, the number of the non-zero components of the resulting vector equals the number of the corresponding list items.

After the transformation of the ranked lists to vectors, their distance can be computed by using any of the well-established vector similarity/distance metrics, such as cosine similarity, Jaccard index, and Sørensen-Dice coefficient. For example, in the case of cosine similarity, the distance is calculated by applying the following equation:

$$d_{CS}(R, L) = 1 - \frac{\boldsymbol{\rho} \cdot \boldsymbol{\lambda}}{||\boldsymbol{\rho}||\ ||\boldsymbol{\lambda}||} = 1 - \frac{\sum_{i=1}^{k} \beta_i^{(R)} \beta_y^{(L)}}{\sqrt{\sum_{i=1}^{k} \left(\beta_i^{(R)}\right)^2} \sqrt{\sum_{i=1}^{|L|} \left(\beta_i^{(L)}\right)^2}}, \tag{6}$$

where y denotes the ranking of the i^{th} item of R in the aggregate list L. We name this distance metric *CODRA (Cosine Distance for Rank Aggregation)*.

Now, we focus on the determination of the weights of the components of $\boldsymbol{\rho}$ and $\boldsymbol{\lambda}$. A first approach to this problem is to use the inverse ranking of each item as a weight, i.e., $\beta_i^{(v)} = 1/i$. This approach rewards the highly ranked elements, since the higher their ranking, the greater their weight. With this setting, the vectors of the lists of Example 1 are $\boldsymbol{\rho}^{(1)} = \widehat{\mathbf{j_c}} + 0.5\widehat{\mathbf{j_d}} + 0.33\widehat{\mathbf{j_e}} = (0, 0, 1, 0.5, 0.33)$, $\boldsymbol{\rho}^{(2)} = 0.25\widehat{\mathbf{j_a}} + 0.2\widehat{\mathbf{j_b}} + \widehat{\mathbf{j_c}} + 0.5\widehat{\mathbf{j_d}} + 0.33\widehat{\mathbf{j_e}} = (0.25, 0.2, 1, 0.5, 0.33)$, and $\boldsymbol{\lambda} = \widehat{\mathbf{j_a}} + 0.5\widehat{\mathbf{j_b}} + 0.33\widehat{\mathbf{j_c}} + 0.25\widehat{\mathbf{j_d}} + 0.2\widehat{\mathbf{j_e}} = (1, 0.5, 0.33, 0.25, 0.2)$, with $\widehat{\mathbf{j_x}}$ the basis vector corresponding to x.

However, in practice, this assignment method is proved rather inappropriate. The reason is that it will assign high cosine similarities (namely, small distances) to pairs of lists that exhibit numerous disagreements but have in common only a few highly ranked elements. The following example highlights this problem.

Example 2. *Consider three lists, $R^{(1)} = \{a, b, c, d\}$, $R^{(2)} = \{e, f, g, h\}$, and $L = \{a, f, g, h\}$. $R^{(1)}$ shares only its top element with L, whereas $R^{(2)}$ has all its items in common with L apart from the first one. Apparently, $R^{(2)}$ is closer to L than $R^{(1)}$ is. If we applied the aforementioned strategy for setting the item weights (i.e., $\beta_i^{(v)} = 1/i$), the corresponding vectors of these lists would written as $\boldsymbol{\rho}^{(1)} = (1, 0.5, 0.33, 0.25, 0, 0, 0, 0)$, $\boldsymbol{\rho}^{(2)} = (0, 0, 0, 0, 1, 0.5, 0.33, 0.25)$, and $\boldsymbol{\lambda} = (1, 0, 0, 0, 0, 0.5, 0.33, 0.25)$. From Eq. 6 we compute the distances $d_{CS}(R^{(1)}, L) = 1 - 1/2.08 \simeq 0.52$, and $d_{CS}(R^{(2)}, L) = 1 - 0.42/2.08 \simeq 0.80$. Consequently, we erroneously find that $R^{(1)}$ is much closer to L than $R^{(2)}$ actually is.*

To confront the problems mentioned in the above examples, we apply a different strategy for the assignment of weights to the items of R in L. More specifically, we set:

$$\beta_i^{(R)} = \begin{cases} 1/i, & \text{if item} \in R \cap L \\ 0, & \text{otherwise} \end{cases} \quad (7), \qquad \text{and} \qquad \beta_i^{(L)} = \begin{cases} \log(10 + i - 1), & \text{if item} \in R \cap L \\ 0, & \text{otherwise} \end{cases} \quad (8)$$

In this way, we assign large weights to the highly ranked items of R; the higher the ranking of an item in R, the greater its weight is. In contrast, for the elements of L, the weights decay slowly, in a logarithmic fashion, as the rankings increase. Now, let us examine how this approach cures the problems that were presented in the Examples 1 and 2.

Example 3. *Example 1: Following our previous discussion, the vectors of the lists of this example are written as $\boldsymbol{\rho}^{(1)} = \widehat{\mathbf{j_c}} + 0.5\widehat{\mathbf{j_d}} + 0.33\widehat{\mathbf{j_e}} = (0, 0, 1, 0.5, 0.33)$, $\boldsymbol{\rho}^{(2)} = 0.25\widehat{\mathbf{j_a}} + 0.2\widehat{\mathbf{j_b}} + \widehat{\mathbf{c}} + 0.5\widehat{\mathbf{j_d}} + 0.33\widehat{\mathbf{j_e}} = (0.25, 0.2, 1, 0.5, 0.33)$, and $\boldsymbol{\lambda} = \log 10\widehat{\mathbf{j_a}} + \log 11\widehat{\mathbf{j_b}} + \log 12\widehat{\mathbf{c}} + \log 13\widehat{\mathbf{j_d}} + \log 14\widehat{\mathbf{e}} \simeq (1, 1.04, 1.08, 1.11, 1.15)$. Their corresponding magnitudes are $||\boldsymbol{\rho}^{(1)}|| \simeq 1.17$, $||\boldsymbol{\rho}^{(2)}|| \simeq 1.21$, and $||\boldsymbol{\lambda}|| \simeq 2.41$. According to Eq. 6, the cosine distances of the two lists $R^{(1)}$ and $R^{(2)}$ from L are $d_{CS}(R^{(1)}, L) = 1 - 2.01/2.82 \simeq 0.29$ and $d_{CS}(R^{(2)}, L) = 1 - 2.47/2.91 \simeq 0.15$. Consequently, the proposed distance metric correctly identifies that $R^{(2)}$ is more proximal to L than $R^{(1)}$ is, in contrast to SFD.*

Example 2: Here the distances according to the proposed metric are $d_{CS}(R^{(1)}, L) = 1 - 1/2.4 \simeq 0.58$ and $d_{CS}(R^{(2)}, L) = 1 - 2.235/2.4 \simeq 0.07$. That is, $R^{(2)}$ is much closer to L than $R^{(1)}$ is. This outcome demonstrates the capability of our metric to effectively handle the lists that present many disagreements, but have in common only a few highly ranked elements.

We conclude the presentation of CODRA with two useful remarks. First, notice that the values returned by Eq. 6 fall into the range $[0, 1]$. Consequently, there is no need for normalizing the computed distances. The second remark concerns the integer argument of the logarithm in Eq. 8. This was a deliberate choice, since the computation of the logarithm during runtime is an expensive operation. To improve the efficiency of this metric, in our implementation we pre-computed the values of several thousands of logarithms and stored them in a perfect hash table H, so that $H(i) = \log(10+i-1)$. With this table, we are able to quickly retrieve the weight of the i^{th} element of L by simply accessing $H(i)$.

4. Learning the Voter Weights

In this section, we present our distance-based model for the unsupervised computation of the voter weights in rank aggregation problems. Initially, a preliminary 2-phase approach is introduced in Subsection 4.1. In the sequel, Subsection 4.2 extends this approach and transforms it into a generic iterative algorithm for the determination of converging values for the weights. In the following analysis, we shall restore the exponent v in the R^v notation to distinguish the ranked lists submitted by different voters.

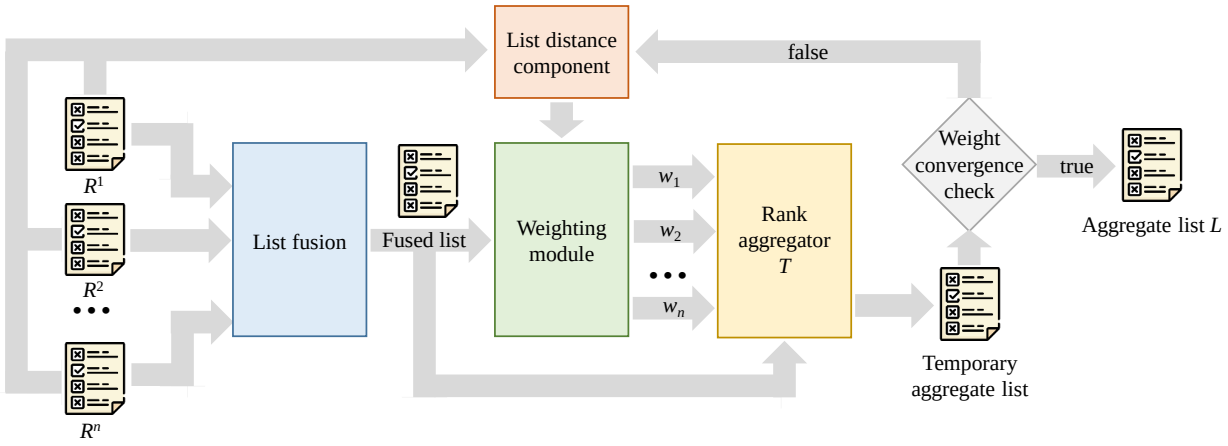

Figure 3: The execution flow of our proposed iterative weighted algorithm. The temporary aggregate list is repeatedly "corrected" by recomputing its distances from the input lists. Then, the voter weights are updated accordingly to construct a new temporary list. When the voter weights converge, the temporary aggregate list becomes the output of the algorithm.

4.1. Preliminary 2-phase Approach

Let us consider that a query q is submitted to the previously introduced set V of n voters. The voters respond by generating n ranked lists of answers, which are later aggregated by a method $\mathcal{T}$, and a consensus list L is eventually formed. According to our previous discussion, if the distance between a single input list R^v and L is small, then these lists are similar. This means that many of the top answers of v have also been included near the top of L, and, also, these answers have been submitted by multiple other voters. This, in turn, reveals that the answers of v are of high informational value and, consequently, v is either an expert on this specific subject or at least has a satisfactory perception of this particular query.

The aforementioned observations establish the basis of the proposed distance-based model. The idea is to assign a voter v a weight w_v according to the distance $d(R^v, L)$ between the list of answers R^v and L. More specifically, our method operates by assigning higher weights to the important voters, i.e., those whose input lists are in close proximity to L, and lower weights to those who are less important.

This logic is implemented by a preliminary approach that consists of two phases. Initially, a draft output list L_0 is created by applying any of the well-known rank aggregation methods encountered in the relevant literature. This method operates in a non weighted fashion. So, it treats all voters equally without considering any weights (Eq. 1). Next, during the second phase, the distances $d(R^v, L_0)$ between all input lists R^v and L_0 are calculated by using one of the distance metrics presented in Subsection 3.3. The computed distances are then utilized to modify the weight of each voter. Finally, $\mathcal{T}$ is re-executed on its weighted form (Eq. 2), and another list L_1 is obtained.

The changes in the voter weights are determined by a kernel function f that incorporates the aforementioned distances. We shall shortly discuss its form and its desired properties. In short, the equation that employs f to quantify the weights modification is defined as follows:

$$w_{v,1} = w_{v,0} + f\big(d(R^v, L_0)\big), \tag{9}$$

where $w_{v,0}$ and $w_{v,1}$ denote the initial and the updated weights of v, respectively. The weights $w_{v,0}$ can be initialized by employing an arbitrary fixed value (e.g., 0, 1, $1/n$, etc.). In the experimental evaluation we set $w_{v,0} = 1$, since this initial value leads to a slightly improved performance.

4.2. Iterative Approach

The interesting point in the aforementioned 2-phase approach is that it can be extended and transformed into a generic, iterative process. Figure 3 depicts the block diagram of the proposed architecture. In this

context, the weights of Eq. 9 are rewritten in the following form:

$$w_{v,i} = w_{v,i-1} + f\big(d(R^v, L_{i-1})\big), \quad i \in \mathbb{N}^*, \tag{10}$$

where i denotes the i^{th} iteration. Similarly to the preliminary 2-phase approach, the voters are initially assigned equal weights, $w_{v,0}$. Therefore, $\mathcal{T}$ is initially applied in a standard non weighted fashion.

Regarding the kernel function f, it must fulfil two requirements. The first one dictates that $f\big(d(R^v, L)\big) \leq f\big(d(R^{v'}, L)\big)$ for $d(R^v, L) \geq d(R^{v'}, L)$, and vice versa. In other words, f must return a large value for small distances between an input list R^v and L, and vice versa. Under this condition, it follows that Eq. 10 will assign higher (lower) weights to the lists having smaller (greater) distances from the aggregate list L. The second requirement concerns the limitation of the returned value of f by an upper bound. Without this limitation, the weights of Eq. 10 would grow infinitely large. To avoid the introduction of a hyper-parameter, or setting a hard limit in an ad-hoc manner, we opted for a converging form for f so that its returned values are asymptotically upper bounded.

We experimented with a large number of functions that possess these two properties. We concluded to the simple exponential function because: i) this form has been previously used by other popular distance-based methods such as the Mallows model (Mallows, 1957) and its extensions (Lebanon & Lafferty, 2002); and ii) it achieved the highest performance amongst multiple candidate functions. Consequently, we define f as follows:

$$f\big(d(R^v, L)\big) = \exp\big(-i \cdot d(R^v, L)\big), \tag{11}$$

where, as previously, i represents the i^{th} iteration. Its presence in the exponent of Eq. 11 guarantees the convergence of the weights, because, as it increases over time, the value of f decreases exponentially.

Now, if we plug Eq. 11 into Eq. 10, we acquire the final iterative form for the weights of the voters:

$$w_{v,i} = w_{v,i-1} + \exp\big(-i \cdot d(R^v, L_{i-1})\big), \quad i \in \mathbb{N}^*. \tag{12}$$

Eq. 12 can be easily rewritten as an equivalent form of partial sums, so that the voter weight at the i^{th} iteration $w_{v,i}$ is connected to the initial weight $w_{v,0}$:

$$w_{v,i} = w_{v,0} + \sum_{j=1}^{i} \exp\big(-j \cdot d(R^v, L_{j-1})\big), \quad i \in \mathbb{N}^*. \tag{13}$$

After the computation of the weights of Eq. 12 during an iteration i, the distance values $d(R^v, L_i)$ may increase for some voters and decrease for some others. Regarding the latter, this decrease may lead the algorithm to perform more iterations to reach convergence. To make Eq. 12 more stable, instead of using the raw distance values $d(R^v, L_i)$, we employ their normalized versions, as described in Subsection 3.3. On the other hand, recall that CODRA is free of this requirement, since its produced distances fall into the range $[0, 1]$ by default.

The execution of the proposed method is described in Algorithm 1. The process employs a small set of temporary, auxiliary variables that control the flow of the algorithm. Specifically, *converged* is a bit array of length n, whose bits are all initialized to zero. During the iterative procedure, the v^{th} bit of *converged* is set equal to 1 if and only if the weight of the v^{th} voter has converged. The identification of convergence is achieved by employing two temporary variables w'_v and w_v that store the weights of v during the previous and the current iteration, respectively. If their difference gets smaller than a predefined threshold *prec*, then we assume that the weight of v has converged, and we set $converged[v] = 1$. The last auxiliary variable is *allconverged*, and determines whether *all* the voter weights have converged to their final values. It becomes true if $converged[v] = 1, \forall v \in V$.

The algorithm begins by initializing all bits of *converged* to 0 and assigning all voters identical weights equal to $w'_v = 1/n, \forall v \in V$ (steps 2–5). Next, a non weighted rank aggregation method $\mathcal{T}$ is employed to generate the initial aggregate list L (step 6). The current iteration is monitored with the aim of the assistant counter i; its value is updated at the beginning of each iteration (step 10) and it is plugged later into the equation that determines the voter weights (step 15).

Algorithm 1: Distance-based iterative algorithm for weight computation

```
1  initialize an empty aggregate list L;
2  for each voter v ∈ V do
3  |  w'_v ← 1/|V|;
4  |  converged[v] ← false;
5  end
6  L ← T(R^{v_1}, R^{v_2}, ..., R^{v_n}) (Eq. 1);
7  i ← 0;
8  allconverged ← false;
9  while not allconverged do
10 |  i ← i + 1;
11 |  allconverged ← true;
12 |  for each voter v ∈ V do
13 |  |  if converged[v] = false then
14 |  |  |  compute d(R^v, L) (Eq. 3, 4, 5);
15 |  |  |  w_v ← w'_v + exp( − i · d(R^v, L)) (Eq. 12);
16 |  |  |  if w_v − w'_v > prec then
17 |  |  |  |  allconverged ← false;
18 |  |  |  else
19 |  |  |  |  converged[v] ← true;
20 |  |  |  end
21 |  |  end
22 |  end
23 |  L ← T(w_{v_1}, ..., w_{v_n}, R^{v_1}, ..., R^{v_n}) (Eq. 2);
24 |  for each voter v ∈ V do
25 |  |  w'_v ← w_v;
26 |  end
27 end
```

The main part of the algorithm consists of two nested loops. The main loop (steps 9–27) is responsible for performing the iterations, and controls the main execution flow. Its exit condition is triggered when *allconverged* is set equal to true, that is, when the weights of all voters have converged to their final values and no change in L is possible hereafter. Before its start, *allconverged* is set to false (step 8). Then, it is immediately switched to true before each iteration; it will remain true if and only if all the values of bit array *converged* are equal to 1. At that instance, the algorithm assumes that convergence has been achieved, and the execution flow exits the outer loop.

The inner loop (steps 12–22) modifies the weights, and updates the aforementioned bit array *converged*. In particular, for each voter $v \in V$, the normalized distance $d(R^v, L)$ of the input list R^v from the current aggregate list L is computed (steps 14–15). In the sequel, the new weight w_v of the voter v is calculated according to Eq. 12. In steps 16–20, the algorithm checks the difference between the newly computed weight of v with the previous one ($w_v - w'_v$). If it is found smaller than *prec*, *converged*$[v]$ is set equal to 1; the condition of step 13 guarantees that w_v will never be modified again. When the weights of all voters have converged, the weighted version of $\mathcal{T}$ is employed, and a new aggregate list L derives according to Eq. 2 (step 24). The current iteration ends by overwriting the previous voter weights with the newly computed ones (steps 24–26) to allow a new check for convergence during the next iteration.

In overall, Algorithm 1 needs $n_{it}|V|$ distance and $(n_{it}+1)$ aggregate list computations, where n_{it} denotes the number of iterations. As it is exhibited in Section 6, n_{it} primarily depends on the selected distance metric and, on average, it takes values that can be characterized from small to moderate.

Notice how the proposed algorithm satisfies the research objectives that were set in Subsection 3.2.

First, in contrast to the score-based methods, it is independent of the scores that an item may (or may not) be assigned by its voter (Pihur et al., 2007). Second, it operates by computing the distances of *all* input lists from the aggregate list. Therefore, contrary to the agglomerative method of Chatterjee et al. (2018) that performs pairwise list merges, it complies with the spirit of Kemeny optimal aggregation and the optimization problem of Pihur et al. (2007). It also maintains another advantage over these approaches, since it does not solve expensive optimization problems. Instead, it adopts an iterative style. Finally, it is not a majoritarian method, so it avoids the quadratic complexity of these algorithms (Desarkar et al., 2016).

5. Using the Computed Voter Weights

In this section, we explore two strategies for exploiting the computed weights, with the aim of improving the effectiveness of rank aggregation. The first one dictates that the weights directly modify the ranking, and, consequently, the score of each element according to the aggregation method $\mathcal{T}$. The second strategy determines the number of elements from each input list that will be considered during aggregation.

5.1. Ranking Modification

The issue of weights utilization in rank aggregation problems is not new and has been previously investigated by multiple researchers. The relevant works have shown that. if such weights are assigned to the voters correctly, then considerable benefits in retrieval effectiveness can be achieved (Desarkar et al., 2016).

In this work we also adopt this strategy with the aim of implementing the essence of a weighted method, as stated by Eq. 2. Initially, the weights can be used in $\mathcal{T}$, either in their raw form (namely, as they derive by Eq. 12), or normalized by using a standard normalization technique like min-max scaling:

$$w'_v = \frac{w_v - w_{v,min}}{w_{v,max} - w_{v,min}}. \tag{14}$$

Then, if $\mathcal{T}$ is a positional method (e.g., Borda Count), its weighted version merely replaces the plain ranking j of an item r_j^v within a list R^v by the product $j \cdot w'_v$. For example, if an item has been ranked 2^{nd}, 5^{th}, and 7^{th} in three top-10 lists, then its non-weighted Borda Score is $8 + 5 + 3 = 16$. Now, if the normalized weights of the first, second, and third voters are 0.2, 0.3, and 0.4, respectively, then the weighted Borda score becomes $0.2 \cdot 8 + 0.3 \cdot 5 + 0.4 \cdot 3 = 4.3$. The same modification is applicable to other rank aggregation methods, like the Outranking Approach of Farah & Vanderpooten (2007).

On the other hand, in order-based methods, such as the Condorcet method, the weights can be used to modify the number of wins of each element. That is, instead of increasing the number of wins by 1 each time an element dominates over another, the weighted version increases it by a quantity that depends on the weight w_v of the voter v. A straightforward approach would be to increase the number of wins not by 1, but by w_v.

Regarding the probabilistic methods for rank aggregation, in some cases, the weight of a voter is incorporated within the proposed models themselves, whereas in others, it can be used as a parameter to these models. For instance, the Mallows model defines the probability that a voter v generates a permutation π of the elements of U according to the following equation (Mallows, 1957):

$$p(\pi | w_v, R^v) = \frac{1}{Z(w_v, R^v)} \exp(w_v d(\pi, R^v)), \tag{15}$$

where $Z(w_v, R^v)$ is a normalizing factor, and $w_v \leq 0$ is called the dispersion parameter. As it increases, the probability distribution becomes more concentrated at R^v. Therefore, it could represent the weight of v. Similar approaches for the weights are also valid for some of the extended Mallows models, including the one presented by Lebanon & Lafferty (2002). In this method, the free parameters $\boldsymbol{\theta}$ represent the degree of expertise of the voters.

These remarks establish one of the basic properties of the proposed model: by directly modifying the rankings of the elements or the number of wins, the proposed algorithm can be applied in conjunction with the majority of the existing rank aggregation methods. In addition, the computed weights can be applied as parameters to the most successful probabilistic models and enhance their performance.

Algorithm 2: Input list pruning

1 **for** *each voter* $v \in V$ **do**
2 | initialize an empty list R_P^v;
3 | compute C_v (Eq. 16);
4 | **for** *each element* $r_i^v \in R^v$ **do**
5 | | **if** $i \leq C_v$ **then**
6 | | | $R_P^v \leftarrow R_P^v + \{r_i^v\}$;
7 | | **end**
8 | **end**
9 **end**
10 $L \leftarrow \mathcal{T}\big(w'_{v_1}, \ldots, w'_{v_n}, R_P^{v_1}, \ldots, R_P^{v_n}\big)$ (Eq. 2);

5.2. Input List Pruning

Now, let us study a new possibility for exploiting the weights of the voters to further enhance the aggregation quality. In short, the idea is to discard a certain amount of the low ranked elements of the voters, and perform the aggregation on the resultant lists. The rationale is that the elements with low rankings are assumed to be of smaller importance compared to those with higher rankings. To extend this rationale, the elements that both have low rankings and are proposed by voters with small weights are of even smaller importance. Consequently, their integration in the aggregate list may degrade its quality.

The aforementioned logic provides the motivation for the application of a list pruning technique with respect to the voter weights. This process is initiated after the computation of the final, converged voter weights; that is, after Algorithm 1 has been completed. At that point, a cut-off value is set for each input list R^v, according to the following equation:

$$C_v = (\delta_1 + \delta_2 w'_v)k, \tag{16}$$

where $k = |R^v|$ is the number of elements in R^v, and $w'_v \in [0, 1]$ represents the normalized weight of v. Furthermore, $\delta_1 \in [0, 1]$ and $\delta_2 \in [0, 1 - \delta_1]$ are two constant parameters; their usefulness will be explained shortly. The computed cut-off point also serves as a reference point. More specifically, the elements $r_i^v \in R^v$ whose ranking i exceeds C_v are removed from R^v. The result of this process is a new, pruned list R_P^v. In the sequel, the weighted aggregation method $\mathcal{T}$ is reapplied on these pruned lists, and the final aggregate list is constructed, as indicated by Eq. 2.

Before we proceed with the detailed presentation of the pruning algorithm, let us examine the roles of the two parameters δ_1 and δ_2. The first one determines the minimum number of items of R^v that will be copied to R_P^v. In other words, δ_1 represents the minimum length of R_P^v. Its usefulness is to prevent the pruned lists of the voters who have been assigned very small weights from including no, or very few items. Regarding δ_2, its role is to regulate the importance of the weights of the voters in the amount of pruning. In particular, if δ_2 is selected to be large, then C_v shall also be large; therefore, the number of the pruned items will be small. The opposite statement is also valid.

According to Eq. 16, the value of the cut-off point in an input list R^v depends on the importance of the corresponding voter v. For important voters, who have been assigned high weights, the value of C_v will be greater compared to the cut-off points of the unimportant voters. In addition, notice that the parenthesized quantity always falls into the range $[0, 1]$. This means that, essentially, Eq. 16 computes a percentage of the length of the original input list. In the extreme case of $\delta_1 = 1$, we get that $\delta_2 = 0$; consequently, no pruning takes place.

The steps of the pruning process are presented in Algorithm 2. Initially, for each voter $v \in V$, an empty pruned list R_P^v is initialized (step 2) and the cut-off point C_v is computed by utilizing Eq. 16 (step 3). Next, all the elements of the original input list R^v, whose rankings are lower than C_v, are inserted to R_P^v (steps 4–8). Finally, after the construction of the pruned lists of all voters, the weighted rank aggregation method $\mathcal{T}$ is applied on them, and a final aggregate list L is obtained (step 10).

6. Experimental Evaluation

In this section the experimental evaluation of the proposed methods is analyzed. The presentation is organized as follows: In Subsection 6.1 the experimental setup is described, including the state-of-the-art methods that are used for comparison, the utilized datasets, and the employed precision evaluation metrics. Subsection 6.2 provides a statistical proof of the correctness of the key idea, which states that "the distance between the ranked list of a voter and the aggregate list is a strong indication of the importance of this voter". A performance analysis against the fluctuations of the parameters δ_1 and δ_2 is conducted in Subsection 6.3. The statistical significance of the obtained results through the execution of the Friedman and Wilcoxon tests is discussed in Subsection 6.4, whereas the effectiveness of the model is attested in Subsection 6.5.

All the experiments have been conducted on a workstation equipped with 32GB of RAM, and a single CoreI7 7700 processor with frequency equal to 3.6GHz.

6.1. Datasets, Competitive Methods, and Evaluation Metrics

Text Retrieval Conference[1] (TREC) is a reputable and reliable source of datasets for a wide range of tasks, related to the broad research areas of IR and Data Mining. In this work, we have utilized 9 datasets originating from TREC. To minimize the random effects that would distort our results, we gathered data from 3 different TREC tasks, spanning 9 years (i.e., from 2009 to 2017). In summary, we used data from:

- The Adhoc tasks of the Web Tracks of 6 TREC conferences, organized between 2009 and 2014. These datasets are abbreviated *WA-XX*, where *XX* denotes the two trailing digits of the corresponding year;
- Two *Clinical Decision Support (CDS-XX)* tasks from TREC 2014 and 2015; and
- The *Abstracts* task of the *Precision Medicine* Track of TREC 2017 *(PMA-17)*.

In the aforementioned tasks, the organizers release a set of queries and ask from the participant groups to submit ranked lists of results as responses to these queries. The submissions of the groups are subsequently evaluated by computing several measures, based on relevance judgments made by human judges. The relevance judgments are made publicly available by the organizers. In most cases, the participant groups are free to submit arbitrarily long lists. Nevertheless, in this work, we have restricted the size of all input lists to 1000 elements to keep the running times of all methods within reasonable limits. This constraint was necessary especially for the majoritarian methods that perform pairwise comparisons between all the included list elements, such as the Condorcet method.

The TREC datasets satisfy most of the requirements for fairly evaluating rank aggregation methods. Their main characteristics, along with their abbreviation in this article, are presented in Table 2. The third, fourth, and fifth columns represent the number of topics, voters (n), and items in each input list (k), respectively. In the last column, we briefly annotate each dataset. From these annotations it becomes apparent that, although the TREC datasets are fair and reliable, they only cover scenarios where the input lists are long and the number of voters is moderate to large.

To enhance the generalizability of the experiments, we created 6 additional synthetic datasets that represent diverse scenarios[2]. The names of the synthetic datasets are in the form of *XXYY*, where *XX* and *YY* denote the number of voters and the length of the participating lists, respectively.

More specifically, FELO (Few Long lists), FESO (Few Short lists), and FEMO (Few Moderate lists) represent the cases where a small number of voters submit lists with many, few, and moderate number of elements, respectively. FELO and FEMO are indicative in ensemble machine learning models, where a limited number of component algorithms are evaluated by using different measures and the best algorithm is chosen. On the other hand, MASO (Many Short lists) and MAMO (Many Moderate lists) are common in scenarios where a large number of voters submit short, or medium-sized lists. Such cases are frequent

[1] http://trec.nist.gov

[2] A Github repository of the dataset generator tool can be found at https://github.com/lakritidis/RASDaGen

Track/Task	Abbr.	Topics	n	k	Attributes
Web/Adhoc - TREC 2009	WA-09	50	61	1000	Moderate number of long lists
Web/Adhoc - TREC 2010	WA-10	50	56	1000	Moderate number of long lists
Web/Adhoc - TREC 2011	WA-11	50	62	1000	Moderate number of long lists
Web/Adhoc - TREC 2012	WA-12	50	48	1000	Moderate number of long lists
Web/Adhoc - TREC 2013	WA-13	50	61	1000	Moderate number of long lists
Web/Adhoc - TREC 2014	WA-14	50	30	1000	Moderate number of long lists
Clinical Decision Support - TREC 2014	CDS-14	30	102	1000	Many long lists
Clinical Decision Support - TREC 2015	CDS-15	30	103	1000	Many long lists
Precision Medicine Abstr. - TREC 2017	PMA-17	30	125	1000	Many long lists
MASO - Synthetic	MASO	20	1000	30	Many short lists
MAMO - Synthetic	MAMO	20	1000	100	Many lists of moderate length
FESO - Synthetic	FESO	20	10	30	Few short lists
FEMO - Synthetic	FEMO	20	10	100	Few lists of moderate length
FELO - Synthetic	FELO	20	10	1000	Few long lists
MOSO - Synthetic	MOSO	20	50	30	Moderate number of short lists

Table 2: The characteristics of the utilized datasets.

in recommendation systems and crowdsourcing platforms, where a huge number of users express their opinion on a fixed set of subjects (products, events, etc.). Moreover, the datasets that include long lists (i.e. FELO, CDS-14/15, PMA-17) are representative of bioinformatics applications. Consequently, in comparison to other competitive approaches that are targetted to only specific domains (e.g. recommender systems, metasearching, or bioinformatics), the performance of our proposed algorithms is tested on a much broader field of applications.

Now, let us describe the aggregation methods that have been implemented for the requirements of the tests. Three of the most popular rank aggregation methods were employed to evaluate the performance of the proposed model, namely: i) *Borda Count (BC)*, ii) the *Condorcet method (CM)*, and iii) the *Outranking approach (OA)* of Farah & Vanderpooten (2007). For each of these methods we created four test cases:

- The original non-weighted aggregation methods, namely, BC, CM, and OA. These constitute the baseline methods to which the proposed model is compared.
- The application of Algorithm 1 to each baseline method, in combination with the CODRA distance metric, introduced in Subsection 3.4, and without list pruning. We will refer to these methods as *YY-CODRA*, where *YY* is replaced by BC, CM, and OA, w.r.t. the employed aggregation method.
- The application of Algorithm 1 to each baseline method, in combination with the scaled footrule distance metric (SFD), also without list pruning. These scenarios provide the bases for evaluating the performance of CODRA, and will be called as *YY-SFD*, similarly to the previous methods.
- The application of Algorithm 1 to each baseline method, in combination with the CODRA distance metric and accompanied by the proposed list pruning strategy (Algorithm 2). These methods are abbreviated as *YY-LP*, where *YY* receives the aforementioned values BC, CM, or OA, whereas *LP* stands for *"List Pruning"*.

In addition, we implemented two recent unsupervised weighted rank aggregation algorithms with the aim of comparing our model to the state-of-the-art solutions of the relevant literature. In particular, we implemented the *Preference Relation Method (PRF)* of Desarkar et al. (2016) and the *Agglomerative Aggregation Method (AAM)*, a variant of the method of Chatterjee et al. (2018). Regarding the hyper-parameters of the competitive methods, we applied the values suggested in the respective studies, namely: i) for Outranking Approach, the preference and veto thresholds were tuned to the 0% and 75% of the aggregate list lengths, ii) for PRF, we set $\alpha = \beta = 0.5$, and iii) for AAM, we set $c_1 = 2.5$ and $c_2 = 1.5$. Notice that Chatterjee et al. (2018) describe a weight initialization technique based on the comparison of each voter opinion with

the pairwise majority votes. However, the quadratic space complexity of this strategy renders AAM not applicable to the TREC datasets, because the total number of items in all input lists is in the scale of many tens of thousands. For this reason, we implemented a variant that assigns equal initial weights to the voters.

We close this subsection with a brief description of the metrics that were used for evaluating the retrieval effectiveness of the involved methods. Precision is among the most popular metrics, and it is measured at specific points of the aggregate list. Therefore, the notation $P@X$ denotes the Precision at the X^{th} element of the list; that is, the number of relevant items that were ranked in the first X positions of the list, divided by X. Mean Average Precision (MAP) is another measure that has been broadly used in the literature. Given a set of $|Q|$ queries, MAP is defined as the mean of the average precision scores:

$$\text{MAP} = \frac{1}{|Q|}\sum_{q=1}^{|Q|}\overline{P}(q) = \frac{1}{|Q|}\sum_{q=1}^{|Q|}\frac{1}{|\text{Rel}_q|}\sum_{k=1}^{|L|}\text{rel}(l_k)P@k, \tag{17}$$

where $\overline{P}(q)$ and $|\text{Rel}_q|$ represent the average precision and the total number of relevant documents of a query q, respectively, $P@k$ is the precision value at the k^{th} position of the aggregate list L, and $\text{rel}(l_k)$ is a binary function which returns 1 if and only if $l_k \in L$ is relevant to q, else, its value is 0.

Furthermore, the normalized discounted cumulative gain (nDCG) is another popular evaluation metric, defined by the ratio:

$$\text{nDCG@}k = \frac{\text{DCG@}k}{\text{iDCG@}k}, \tag{18}$$

where

$$\text{DCG@}k = \sum_{k=1}^{|L|}\frac{2^{\text{rel}(l_k)}-1}{\log_2(k+1)}, \tag{19}$$

and $\text{iDCG@}k$ is the value of the ideal $\text{DCG@}k$, which is obtained by sorting the aggregate list L in decreasing relevance scores of its elements and computing DCG at the k^{th} position of that ideal list.

6.2. Proof of Concept

The experimental analysis of the model begins with a statistical study on the correctness of its fundamental element. Namely, the statement that *the distance between an input ranked list of a voter and the aggregate list indicates the importance of this voter*. Since the proposed model is designed on top of this statement, the examination of its significance is crucial.

For each query of each dataset of Table 2, our methodology dictates the construction of two ranked lists:

- The first one contains all the voters sorted in decreasing MAP order. We call this list as the *Real Experts List (REL)*, because, for each query, it enlists the voters according to how relevant their submitted results were. REL was constructed with the assistance of the TREC relevance judgments. Thus, it reflects the real performance of the voters, and this justifies the term *"real"*.
- The second list is called the *Computed Experts List (CEL)*, and, for each query, it includes the same voters as the corresponding REL, ordered in increasing distance of their lists distance from the aggregate list L. The list L was obtained by using any of the aforementioned aggregation methods, whereas the distances were computed by utilizing either SFD or CODRA. Hence, CEL has on its top the voter whose submitted list has the smallest distance from L, followed by the rest of the voters.

To examine the correctness of the model, we investigate the correlation between REL and CEL. If REL and CEL are indeed correlated, then that would provide an adequate evidence of the relation between MAP (namely, the ranking criterion of REL), and the distance of R from L (namely, the ranking criterion of CEL). A commonly employed method to measure the correlation between two ranked lists is Spearman's ρ, defined by the following equation:

$$\rho = 1 - \frac{6}{n(n^2-1)}\sum_{\forall v \in V}\left(z^v_{REL} - z^v_{CEL}\right)^2, \tag{20}$$

Dataset	Borda Count		Condorcet Method		Outranking Approach	
	SFD	**CODRA**	**SFD**	**CODRA**	**SFD**	**CODRA**
WA-09	0.238 (0.203)	0.476 (0.098)	0.201 (0.231)	0.457 (0.091)	0.228 (0.207)	0.537 (0.064)
WA-10	0.333 (0.087)	0.471 (0.094)	0.227 (0.179)	0.467 (0.098)	0.240 (0.178)	0.482 (0.082)
WA-11	0.254 (0.170)	0.337 (0.106)	0.211 (0.258)	0.373 (0.101)	0.114 (0.226)	0.175 (0.141)
WA-12	0.061 (0.353)	0.245 (0.120)	0.028 (0.355)	0.254 (0.111)	0.017 (0.350)	0.196 (0.165)
WA-13	0.088 (0.306)	0.164 (0.192)	0.053 (0.304)	0.172 (0.140)	0.031 (0.309)	0.081 (0.258)
WA-14	0.237 (0.266)	0.269 (0.197)	0.196 (0.290)	0.274 (0.188)	0.141 (0.302)	0.214 (0.268)
CDS-14	0.479 (0.041)	0.510 (0.055)	0.496 (0.026)	0.503 (0.060)	0.455 (0.055)	0.478 (0.042)
CDS-15	0.397 (0.079)	0.470 (0.028)	0.380 (0.075)	0.487 (0.030)	0.357 (0.082)	0.437 (0.019)
PMA-17	0.491 (0.002)	0.617 (0.003)	0.473 (0.005)	0.627 (0.007)	0.422 (0.019)	0.606 (<0.001)

Table 3: Ranking correlation between REL, which arranges the voters in decreasing MAP order, and CEL, which enlists the authors in increasing distance of their submitted ranked list from the aggregate list. The correlation is measured by using Spearman's ρ. High values of ρ indicate a statistically significant connection between distance and performance. The parenthesized numbers denote the respective p-values. Namely, the probability of obtaining these test results given that the null hypothesis ($\rho = 0$) is correct.

where z^v_{REL} and z^v_{CEL} denote the ranking of v in REL and CEL, respectively. In short, if $\rho \to 1$, then the two variables (namely, MAP and distance) are highly correlated; that is, the precision tends to increase when the distance from the aggregate list L decreases. In contrast, when $\rho \to -1$, the two variables are fully opposed, whereas a value of $\rho \to 0$ indicates that there is no tendency for the performance to either increase or decrease when the list distance increases.

Table 3 contains the values of ρ for all datasets and aggregation methods. The `cor.test` method of the R language was employed to obtain these values. For each method, two values of ρ were measured, according to the metric that was used to compute the distance between an input list and the aggregate list. The parenthesized numbers denote the respective p-values, that quantify the statistical significance of our measurements[3].

The results demonstrate the ability of the proposed model to identify the most important voters. In particular, for Borda Count in combination with CODRA, ρ received values between roughly 0.16 and 0.62. In 5 out of the 9 datasets, ρ was near or above 0.5, a value that is considered satisfactory, given that the respective p-values are all lower than 0.1. In contrast, in the WA-12, WA-13, and WA-14 datasets, the value of ρ was considerably lower. This indicates a clear tendency for the performance to increase when the distance from the aggregate list L decreases. Consequently, the introduced strategy to assign high weights to the voters whose ranked lists are proximal to L is proven solid.

Furthermore, the values of ρ that were obtained by using SFD were lower than those achieved by using CODRA. The greatest difference was observed in the WA-12 dataset, where the value of ρ of CODRA was more than 4 times greater than that of SFD. However, the statistical significance of this measurement was not so high in this case, since p-value was roughly 0.353. Other remarkable differences were also observed in WA-09 and WA-10, where CODRA outperformed SFD by more than two times. Consequently, our introduced distance metric was proved to be more effective than the well-established footrule distance in identifying the important voters. As we will discuss shortly in Subsection 6.5, this effectiveness leads to substantial improvements in the quality of the generated aggregate list.

Similar results were obtained by applying the other two methods, i.e., the Condorcet method and the Outranking approach. Regarding the first method, the measured values of ρ for CM-CODRA were very close to those of BC-CODRA. This is also valid for the associated p-values. Nevertheless, in some cases, ρ for CM-SFD was much lower than that of BC-SFD. This was particularly true for the WA-12 and WA-13

[3] p-value represents the probability of observing the same results as the real ones, under the assumption that the null hypothesis is valid. In our case, the null hypothesis states the statistical independence of the voters' performance and the distance of their submitted lists from the computed aggregate list. Consequently, the lower the p-value, the stronger the statistical significance of the measured ρ values.

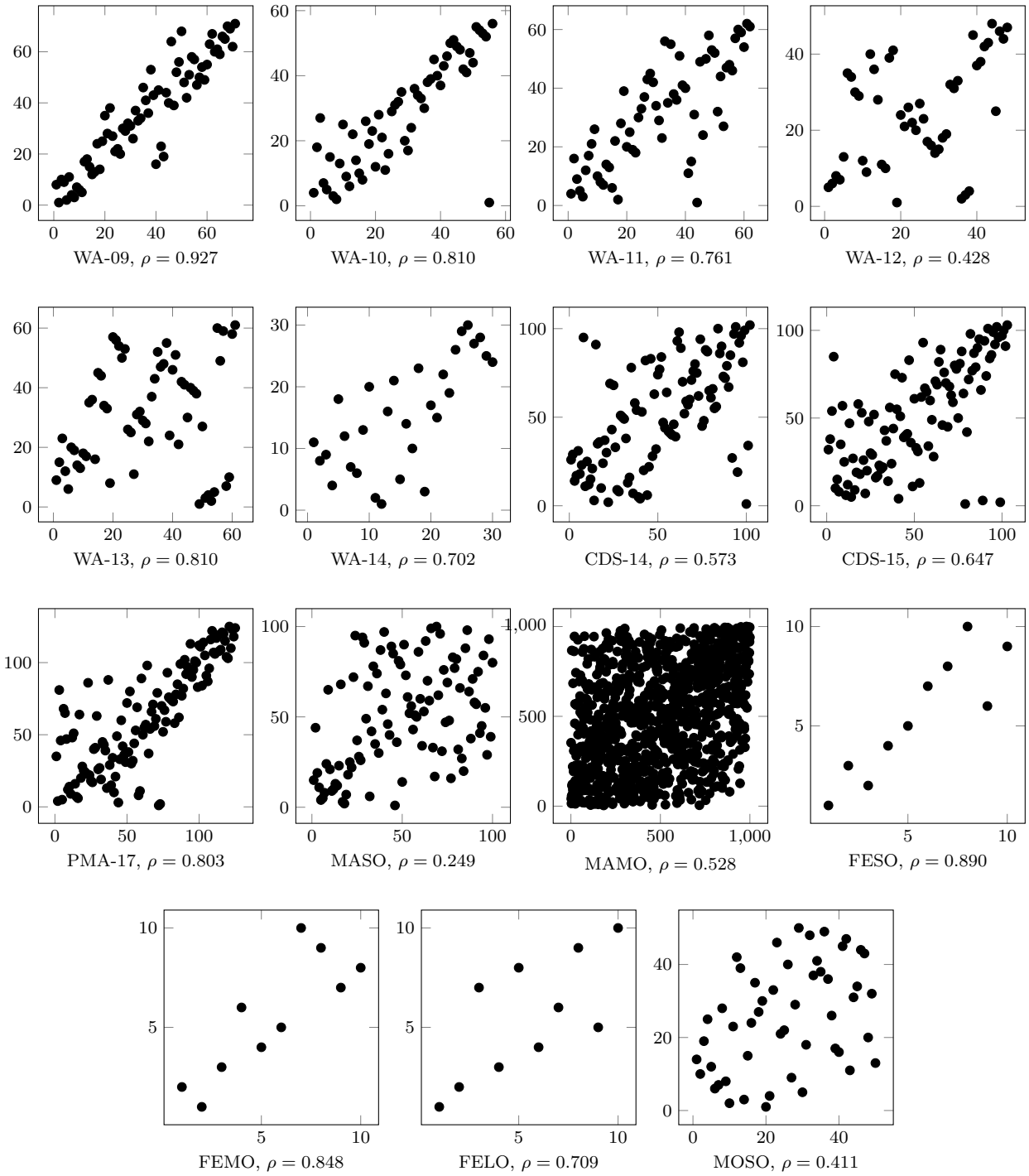

Figure 4: Scatter plots of the real vs. computed ranking of the voters for the 9 TREC and the 6 synthetic datasets of Table 2. The plots concern the 12th query from each dataset. Each dot represents a voter; the vertical and horizontal coordinates of each dot denote the real and the computed rankings of the corresponding voter respectively. The values of the Spearman's ρ is reported in the title of each diagram.

datasets. In contrast, the values of ρ for OA-CODRA were in general considerably lower than the respective of BC-CODRA and CM-CODRA, with only two exceptions, namely, WA-09 and WA-10. This partially justifies the worse retrieval effectiveness of OA compared to BC and CM, as we discuss in Subsection 6.5.

Figure 4 provides a visual sample of the correlation between the performance of the voters and the distance of their submitted lists from the aggregate list. More specifically, it contains 15 scatter plots of the real vs. the computed rankings of the voters for one random query selected from each one of the 9 TREC, and the 6 synthetic datasets. Unfortunately, due to space restrictions,it is impossible to present such scatter plots for all six involved aggregation methods; that is, 3 YY-CODRA and 3 YY-SFD and for all 510 queries of the 15 datasets. Nevertheless, to provide an indicative visual sample, we have randomly selected the 12^{th} query from each dataset and executed BC-CODRA to generate the voter rankings.

The top-left scatter plot of Figure 4 concerns the twelfth query of WA-09. Each dot in the plot represents a voter, and, according to Table 2, there are 61 such dots in the diagram. The vertical co-ordinate of each dot denotes the real ranking of the voter, as this derives from the calculation of MAP in this query. On the other hand, the horizontal co-ordinate of each dot is the ranking of the corresponding voter, obtained by the computation of the distance of the submitted input list from the aggregate list. The aggregate list was constructed by using BC, whereas the distance was measured with CODRA. On the horizontal axis, we also report the value of Spearman's ρ for this particular query.

The scatter plots, in general, reveal a significant association between distance and performance, visualizing a monotonic relation of these two variables. Regarding the TREC datasets, the positive monotone increasing relation is obvious on six out of nine cases (namely, WA-09, WA-10, WA-11, CDS-14, CDS-15, and PMA-17). Therefore, the tendency of retrieval effectiveness, in terms of MAP, to increase as the distance from the aggregate list decreases is also apparent. Especially for the WA-09, WA-10, and PMA-17 datasets, the high value of Spearman's ρ is remarkable. Regarding the rest 3 diagrams that correspond to the WA-12, WA-13, and WA-14 datasets, the correlation between the performance and the distance of the input lists from L is not equally strong, although the differences in the values of Spearman's ρ are not very high. This indicates that, for these three queries, the distance of the input lists from L alone may not be sufficient to precisely predict the performance of the voters.

Similar results were obtained for the six synthetic datasets. More specifically, on FESO, FEMO, and FELO (diagrams 12, 13, and 14 of Fig. 4 respectively), the Spearman's ρ value was particularly high and the correlation between performance and the distance of the input lists from L was visually verified. On the other three datasets, the strength of this correlation was also significant, albeit smaller than the one that was observed on the first three cases. On the MAMO dataset, although the value of $\rho = 0.528$ was satisfactory, the existence of 1000 dots on the respective diagram renders the visual inspection quite hard.

6.3. Parameter Tuning

In this subsection, we examine the assignment of values to the parameters that govern the proposed model. We begin with *prec*, a tolerance parameter that checks the convergence of the voter weights in Algorithm 1. Recall that, at each iteration, all the weights are compared with the respective ones from the previous iteration. In our experiments, we consider that the convergence is achieved as long as all the weights agree at the third decimal digit. Consequently, we set $prec = 10^{-3}$. Lower values of *prec* lead to a greater number of iterations, with practically infinitesimal performance gains.

We proceed with the examination of the performance of the proposed pruning method against the fluctuations of the parameters δ_1 and δ_2 of Eq. 16. Recall that these parameters control the placement of the cutoff points in the input ranked lists according to Algorithm 2. Therefore, they essentially determine the size of the pruned lists. The goal of this experiment was to investigate the effectiveness of the proposed pruning policy and its dependence on the values of these parameters.

More specifically, we measured the retrieval effectiveness of the model by simultaneously modifying δ_1 and δ_2 as follows: Initially, δ_1 was gradually increased from 0.1 to 0.9 by taking steps of 0.1 each time. At each step, δ_2 was also modified in the range $[0.1, 1 - \delta_1]$, and for each pair of (δ_1, δ_2) values, the MAP achieved by the model was recorded. Three TREC and one synthetic datasets were randomly selected for this purpose, namely, WA-12, WA-14, CDS-15, and FESO. Then, the pruning algorithm was applied in combination with either BC-DOBRA, or CM-DOBRA, or OA-DOBRA.

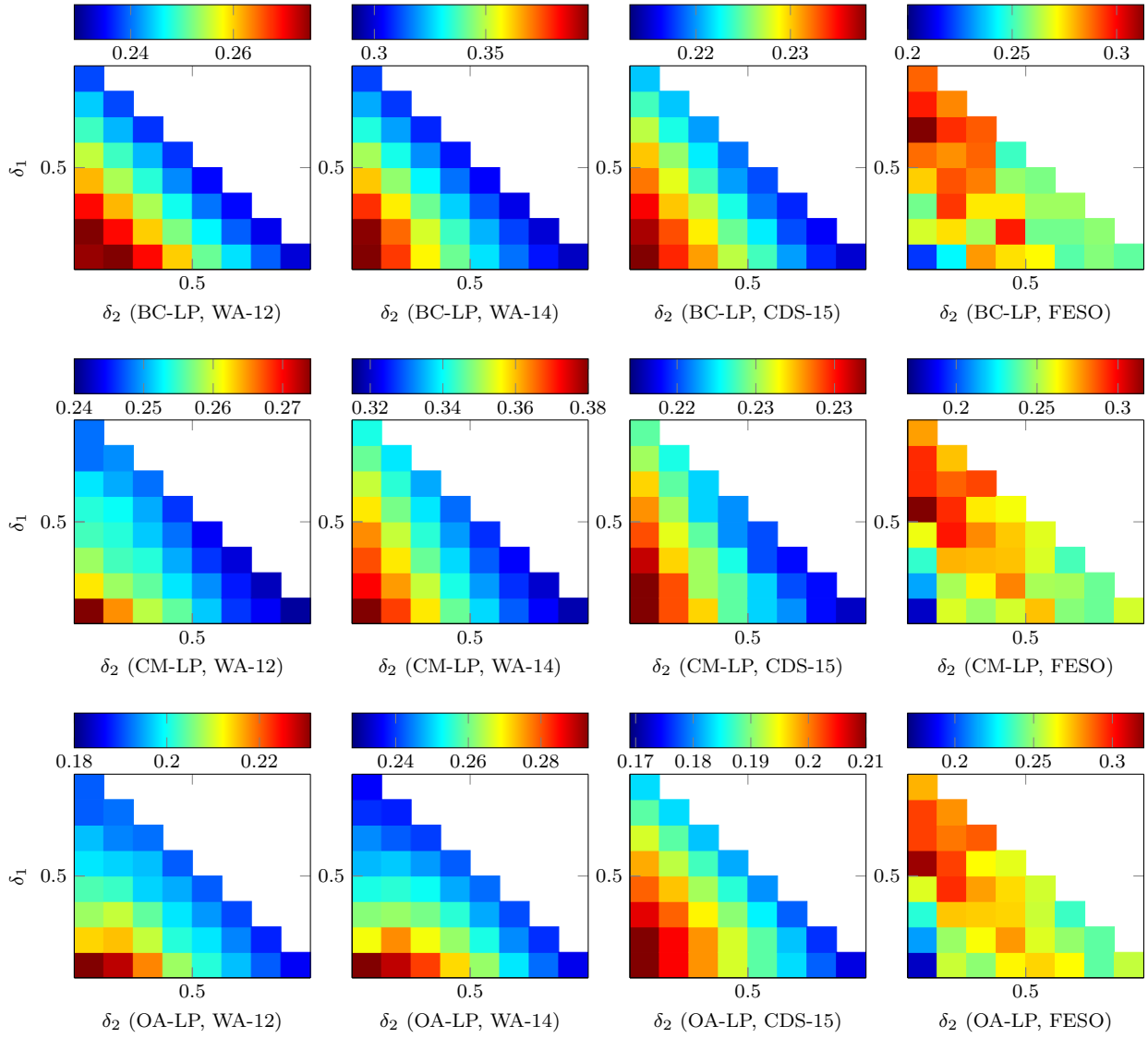

Figure 5: Performance fluctuation (in terms of Mean Average Precision) of the model's pruning policy against the parameters δ_1 and δ_2, for various methods and datasets (reported in the horizontal axis). The "hotter" points indicate better performance.

The results are illustrated in the 12 triangular "heat maps" of Fig.5. Each row in the figure represents the execution of a different rank aggregation method; specifically, BC-LP, CM-LP, and OA-LP, for columns 1–4, respectively. The details are shown in the label of the horizontal axis of each diagram.

Regarding the 3 TREC datasets, the 9 heat maps of the first 3 columns reveal that for each dataset and each aggregation method, the best performance was achieved for small values of δ_1 and δ_2. More specifically, the "hotter" areas were observed for $\delta_1 = 0.1$ to 0.2, and, similarly, $\delta_2 = 0.1$ to 0.2. In some cases, MAP is also high for values of δ_1 and δ_2 that are equal to 0.3, or even 0.4. Nevertheless, the results were different for the "small" FESO dataset. In this case, the highest precision was achieved for $\delta_1 = 0.5$ and $\delta_2 = 0.1$.

The first conclusion that derives from this experiment is that on the examined TREC datasets, large amounts of pruning lead to better results, independently of the employed aggregation method. More specifi-

	AAM	BC	BC-COD	BC-LP	BC-SFD	CM	CM-COD	CM-LP	CM-SFD	OA	OA-COD	OA-LP	OA-SFD
BC	7.2e-4												
BCCODRA	6.1e-5	6.1e-5											
BCLP	6.1e-5	6.1e-4	2.1e-2										
BCSFD	6.1e-5	2.7e-2	1.2e-3	2.0e-3									
CM	7.2e-4	6.0e-1	2.1e-3	1.5e-3	1.6e-1								
CMCODRA	6.1e-5	4.1e-3	4.7e-1	4.1e-3	3.8e-1	3.4e-3							
CMLP	6.1e-5	3.3e-2	5.9e-1	2.7e-3	2.1e-1	4.3e-3	1.4e-1						
CMSFD	6.1e-5	7.3e-2	4.3e-3	6.1e-4	8.8e-1	6.9e-2	1.6e-2	6.4e-2					
OA	6.1e-5	4.7e-3	6.1e-5	8.5e-4	2.2e-3	5.5e-2	2.6e-3	1.3e-2	1.1e-2				
OACODRA	6.1e-5	3.5e-2	1.3e-3	2.2e-3	4.9e-3	1.9e-1	7.6e-3	4.8e-2	5.0e-2	2.1e-2			
OALP	6.1e-5	2.7e-1	3.1e-4	3.1e-4	6.5e-2	2.5e-1	1.5e-3	4.3e-3	2.5e-2	4.5e-2	7.3e-2		
OASFD	6.1e-5	4.5e-2	2.2e-3	3.8e-3	3.4e-3	2.1e-1	1.0e-2	6.5e-2	2.6e-2	6.8e-2	2.2e-1	6.5e-2	
PRF	1.5e-1	3.8e-2	1.8e-2	1.5e-2	3.3e-2	4.8e-2	2.9e-2	3.5e-2	4.1e-2	3.8e-2	3.8e-2	1.1e-2	4.8e-2

Table 4: Post hoc statistical analysis of the performance of the attested algorithms with the Wilcoxon signed-rank test.

cally, the setting $\delta_1 = 0.1$ guarantees that all input lists will retain at least 10% of their top ranked elements, as dictated by Eq. 16. Additional elements with lower rankings may also enter the pruned list, according to the weight of the corresponding voter. The number of these items may not exceed the next 10% of the list size, since $\delta_2 = 0.1$. In total, the selected values of δ_1 and δ_2 indicate that the pruned list shall contain at least 10% and at most 20% of the elements of the original ranked list.

This in turn means that the lower ranked elements of all voters are both non-relevant and harmful for a rank aggregation method, evidently introducing the requirement for a list pruning strategy. According to the proposed method, the placement of cutoff points at the top rankings of the input lists eventually leads to substantial improvements in retrieval effectiveness, as we demonstrate in the next part of our experiments.

On the other hand, the FESO synthetic dataset has different characteristics, since it includes a few very short input lists ($k = 30$, Table 2). Thus, by setting $\delta_1 = 0.1$, we will perform a rather aggressive pruning process, because the minimum number of elements that will be preserved is just 3. Consequently, we run the risk of discarding multiple relevant items with this setting. This explains why the setting $\delta_1 = 0.5$ is ideal for datasets that are similar to FESO. Remarkably, the best setting for δ_2 in FESO was also equal to 0.1.

The results presented in the following subsections have been obtained by setting $\delta_1 = 0.1$ in the TREC datasets, and $\delta_1 = 0.5$ in the synthetic ones. Additionally, the value of δ_2 was fixed to 0.1 in all tests. The convergence tolerance *prec* was set equal to 10^{-3}. Of course, similarly to all unsupervised methods, the selected hyper-parameter values concern the utilized datasets. For other types of datasets, with more or fewer voters and longer or shorter input lists, the optimal values of these parameters may vary significantly.

6.4. Statistical Significance Tests

Before we present the performance measurements of the proposed algorithm against the competitive ones, we conduct an analysis of the statistical significance of the results of the next subsection. In this case, the null hypothesis states that there are no significant differences among the 12 examined methods. To test its validity, we performed the Friedman test, by employing the `friedman.test` method of R, including all the 12 methods. The output of the test was a p-value of 5.559e-14. Consequently, we can safely reject the null hypothesis (with significance level < 0.05).

Since the output of the Friedman test was promising, we performed pairwise comparisons to estimate the statistical significance of the observed performance discrepancies between the various methods. For this purpose, we employed the two-tailed Wilcoxon signed-rank test (for a 5% significance level), and we report its output in Table 4. The results indicate that, in some cases, the null hypothesis can be rejected, whereas in others, it cannot.

However, here we are primarily interested in two issues: Firstly, to estimate the significance of the discrepancies between our proposed model and the respective baseline methods. In this context, the performance difference between all YY-CODRA models and all YY baseline methods was always found statistically significant (recall that YY is replaced by either BC, CM, or OA). The same applies to the list pruning policy YY-LP, since the measured p-values for the three YY vs. YY-LP comparisons were all lower than 0.05. Regarding the usage of the SFD distance metric, only the null hypothesis on the equivalence of CM and CM-SFD cannot be rejected (p-value $\simeq 0.07$).

The second point of interest is the evaluation of the significance of the performance comparisons between our proposed model and the competitive PRF and AAM methods. Interestingly, the first column and the last row of Table 4 contain encouraging p-values. Namely, none of them exceeds the critical significance level of 0.05.

6.5. Performance Evaluation

In this subsection we investigate the effectiveness of the proposed model. Two primary points of research are set here: i) to evaluate the magnitude of the benefits in comparison to the ones achieved by the baseline approaches, and ii) to estimate the cost at which these benefits derive, in terms of consumed execution times or iteration cycles.

Tables 5, 6, 7, 8, and 9 contain the results of this double evaluation. Each of them is divided into three subtables so that, in total, there are 15 subtables that depict the performance of the various methods on the 15 datasets of Table 2. Each subtable is in turn divided into three groups of 4 rows: the first row denotes the performance of a baseline method (i.e. BC, CM, and OA), whereas the next two concern the performance of the proposed model accompanied by either SFD (YY-SFD, row 2), or CODRA (YY-CODRA, row 3). The fourth row illustrates the performance of the proposed model in combination with CODRA and the proposed list pruning algorithm (YY-LP). At the bottom of each subtable we report the performance of the adversary PRF and AAM methods. Regarding the vertical organization, the tables include 12 columns, of which, the first two contain the aggregation method and the number of the executed iterations, respectively. Finally, the next 10 columns contain the effectiveness measurements with the metrics of Subsection 6.1.

The second column of the tables reveals a remarkable property of CODRA: in all TREC datasets and in all the attested YY-CODRA methods, the voter weights required 6 iterations to converge. In fact, in some queries, that number was either 5 or 7, however these cases were rare. Consequently, only a few iterations are sufficient for the model to produce its results. In contrast, SFD required a significantly larger number of iterations to produce stable weights, especially when it was used in combination with Borda Count. For example, in all datasets apart from CDS-15, BC-SFD performed more than 20 iterations to generate the voter weights (in WA-13 that number was 26.4). CM-SFD and OA-SFD required significantly fewer iterations (i.e, 11–16) than BC-SFD, but they were still 2–3 times slower than the respective CM-CODRA and OA-CODRA. Consequently, the utilization of CODRA increases the efficiency of the model in comparison to the well-established SFD metric.

The explanation of this small and constant number of iterations is based on multiple elements, starting from the number of voters, which in the 9 TREC datasets is, on average, 72. This is rather a high number of voters, and leads to aggregate lists that are much longer than the individual input lists. Hence, the denominator of Eq. 6 becomes much greater than the nominator, and, eventually, the distances between the input and the aggregate lists become roughly equal to 1 for all voters. This is turn means that, according to Eq. 6, at each iteration, the weight of each voter is modified by a quantity that exponentially depends on the number of this iteration almost solely (recall that the weights are increased by $\exp\left(-i \cdot d(R^v, L)\right)$. The exponential dependence of the weights almost only on the number of iteration explains their fast convergence.

Now, there is another interesting question that requires explanation: *"If all the distances from the aggregate list are almost equal to 1 and all the weights are increased by a quantity that depends almost solely on the number of the current iteration, then are all the voter weights approximately equal?"*. The answer to this question is negative for two reasons. First, the quantity at which the weights are modified depends *almost* solely on the number of the current iteration. In other words, although these distances are all close to 1, they are still different. These differences may be small at each iteration. However, their accumulation

WA-09	n_{it}	Precision					NDCG				
		MAP	P@5	P@10	P@20	P@100	Mean	N@5	N@10	N@20	N@100
BC	-	0.204	0.504	0.498	0.462	0.296	0.134	0.395	0.397	0.376	0.352
BC-SFD	20.8	0.215	0.552	0.516	0.458	0.305	0.136	0.422	0.406	0.368	0.355
BC-CODRA	6.0	0.221	0.552	0.516	0.461	0.309	0.140	0.434	0.422	0.384	0.359
BC-LP	7.0	0.234	0.624	0.557	0.482	0.311	0.147	0.444	0.420	0.401	0.369
CM	-	0.190	0.424	0.438	0.441	0.287	0.117	0.322	0.344	0.345	0.334
CM-SFD	10.7	0.203	0.488	0.478	0.432	0.296	0.128	0.379	0.373	0.345	0.342
CM-CODRA	6.0	0.209	0.500	0.508	0.464	0.307	0.131	0.385	0.387	0.358	0.354
CM-LP	7.0	0.188	0.468	0.436	0.400	0.273	0.106	0.343	0.326	0.307	0.306
OA	-	0.192	0.544	0.506	0.445	0.289	0.118	0.390	0.369	0.342	0.333
OA-SFD	15.6	0.199	0.549	0.508	0.450	0.291	0.121	0.392	0.369	0.339	0.330
OA-CODRA	6.0	0.206	0.568	0.516	0.450	0.291	0.123	0.399	0.373	0.346	0.334
OA-LP	7.0	0.214	0.566	0.528	0.477	0.294	0.128	0.396	0.386	0.355	0.339
PRF	-	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a
AAM	-	0.047	0.038	0.034	0.032	0.032	0.033	0.032	0.031	0.031	0.031

WA-10	n_{it}	Precision					NDCG				
		MAP	P@5	P@10	P@20	P@100	Mean	N@5	N@10	N@20	N@100
BC	-	0.218	0.496	0.458	0.408	0.277	0.144	0.268	0.260	0.257	0.289
BC-SFD	22.2	0.230	0.504	0.472	0.417	0.291	0.149	0.277	0.271	0.264	0.303
BC-CODRA	6.0	0.248	0.500	0.468	0.443	0.306	0.160	0.268	0.264	0.270	0.316
BC-LP	7.0	0.252	0.524	0.507	0.462	0.311	0.167	0.324	0.288	0.299	0.323
CM	-	0.200	0.432	0.398	0.390	0.281	0.115	0.195	0.212	0.229	0.274
CM-SFD	10.7	0.231	0.452	0.440	0.421	0.295	0.136	0.229	0.240	0.250	0.298
CM-CODRA	6.0	0.241	0.492	0.454	0.431	0.313	0.144	0.248	0.244	0.258	0.314
CM-LP	7.0	0.229	0.460	0.410	0.382	0.309	0.124	0.220	0.216	0.222	0.301
OA	-	0.215	0.476	0.456	0.418	0.282	0.136	0.245	0.251	0.255	0.290
OA-SFD	16,1	0.222	0.488	0.460	0.415	0.282	0.138	0.247	0.254	0.260	0.299
OA-CODRA	6.0	0.229	0.490	0.464	0.413	0.284	0.143	0.253	0.259	0.263	0.299
OA-LP	7.0	0.213	0.480	0.400	0.387	0.272	0.107	0.236	0.221	0.229	0.282
PRF	-	0.054	0.114	0.094	0.094	0.085	0.041	0.077	0.070	0.071	0.083
AAM	-	0.041	0.030	0.033	0.031	0.030	0.032	0.030	0.030	0.033	0.033

WA-11	n_{it}	Precision					NDCG				
		MAP	P@5	P@10	P@20	P@100	Mean	N@5	N@10	N@20	N@100
BC	-	0.229	0.408	0.360	0.336	0.233	0.154	0.266	0.242	0.239	0.306
BC-SFD	21.1	0.233	0.440	0.356	0.325	0.222	0.156	0.283	0.241	0.237	0.299
BC-CODRA	6.0	0.241	0.444	0.370	0.328	0.239	0.161	0.282	0.247	0.245	0.313
BC-LP	7.0	0.290	0.440	0.400	0.385	0.273	0.183	0.282	0.263	0.272	0.372
CM	-	0.246	0.452	0.376	0.361	0.249	0.171	0.294	0.271	0.270	0.335
CM-SFD	11.7	0.247	0.468	0.390	0.358	0.246	0.170	0.317	0.272	0.262	0.331
CM-CODRA	6.0	0.258	0.476	0.396	0.375	0.248	0.183	0.303	0.279	0.280	0.342
CM-LP	7.0	0.280	0.448	0.400	0.369	0.269	0.191	0.301	0.278	0.274	0.372
OA	-	0.186	0.412	0.346	0.301	0.193	0.131	0.276	0.238	0.218	0.258
OA-SFD	15.0	0.188	0.416	0.342	0.302	0.192	0.131	0.279	0.238	0.218	0.259
OA-CODRA	6.0	0.193	0.416	0.352	0.309	0.195	0.135	0.277	0.241	0.221	0.261
OA-LP	7.0	0.230	0.432	0.372	0.350	0.229	0.157	0.278	0.255	0.258	0.318
PRF	-	0.054	0.082	0.070	0.078	0.068	0.041	0.071	0.062	0.066	0.083
AAM	-	0.056	0.071	0.065	0.081	0.030	0.036	0.045	0.035	0.051	0.060

Table 5: Comparison of the retrieval effectiveness of the examined rank aggregation methods with various precision evaluation measures, on the WA-12, WA-13, and WA-14 datasets.

WA-12	n_{it}	Precision					NDCG				
		MAP	P@5	P@10	P@20	P@100	Mean	N@5	N@10	N@20	N@100
BC	-	0.224	0.400	0.380	0.342	0.237	0.356	0.155	0.161	0.163	0.253
BC-SFD	20.2	0.230	0.428	0.382	0.346	0.239	0.360	0.172	0.155	0.169	0.252
BC-CODRA	6.0	0.239	0.448	0.382	0.348	0.236	0.400	0.175	0.170	0.179	0.259
BC-LP	7.0	0.275	0.442	0.410	0.393	0.282	0.476	0.174	0.182	0.181	0.275
CM	-	0.239	0.472	0.452	0.396	0.245	0.486	0.200	0.207	0.205	0.275
CM-SFD	11.5	0.239	0.448	0.444	0.400	0.246	0.475	0.187	0.189	0.202	0.274
CM-CODRA	6.0	0.246	0.472	0.452	0.398	0.251	0.494	0.203	0.208	0.211	0.282
CM-LP	7.0	0.275	0.468	0.444	0.402	0.270	0.509	0.185	0.192	0.193	0.294
OA	-	0.178	0.312	0.296	0.288	0.203	0.303	0.145	0.133	0.148	0.208
OA-SFD	14.2	0.177	0.308	0.284	0.290	0.202	0.282	0.135	0.130	0.148	0.205
OA-CODRA	6.0	0.185	0.320	0.324	0.300	0.212	0.308	0.147	0.139	0.149	0.218
OA-LP	7.0	0.231	0.392	0.394	0.367	0.249	0.392	0.157	0.162	0.176	0.272
PRF	-	0.057	0.094	0.086	0.106	0.076	0.055	0.050	0.056	0.056	0.068
AAM	-	0.123	0.158	0.142	0.134	0.115	0.132	0.075	0.077	0.087	0.122

WA-13	n_{it}	Precision					NDCG				
		MAP	P@5	P@10	P@20	P@100	Mean	N@5	N@10	N@20	N@100
BC	-	0.255	0.488	0.462	0.419	0.272	0.160	0.237	0.251	0.268	0.314
BC-SFD	26.4	0.247	0.504	0.478	0.426	0.275	0.155	0.237	0.252	0.266	0.315
BC-CODRA	6.0	0.267	0.524	0.472	0.437	0.278	0.171	0.255	0.260	0.277	0.324
BC-LP	7.0	0.328	0.524	0.470	0.440	0.319	0.210	0.257	0.267	0.309	0.379
CM	-	0.248	0.460	0.422	0.409	0.268	0.150	0.220	0.239	0.264	0.310
CM-SFD	12.2	0.262	0.496	0.470	0.430	0.287	0.174	0.248	0.261	0.277	0.336
CM-CODRA	6.0	0.277	0.500	0.486	0.444	0.290	0.185	0.257	0.274	0.291	0.346
CM-LP	7.0	0.305	0.484	0.476	0.443	0.311	0.203	0.248	0.272	0.294	0.371
OA	-	0.209	0.496	0.442	0.387	0.244	0.133	0.235	0.231	0.243	0.285
OA-SFD	26.0	0.210	0.488	0.452	0.396	0.244	0.133	0.231	0.235	0.245	0.285
OA-CODRA	6.0	0.213	0.500	0.448	0.395	0.245	0.134	0.239	0.235	0.248	0.287
OA-LP	7.0	0.259	0.504	0.470	0.430	0.287	0.170	0.252	0.264	0.283	0.349
PRF	-	0.046	0.050	0.054	0.071	0.085	0.034	0.037	0.040	0.047	0.056
AAM	-	0.077	0.110	0.086	0.091	0.082	0.047	0.057	0.058	0.062	0.088

WA-14	n_{it}	Precision					NDCG				
		MAP	P@5	P@10	P@20	P@100	Mean	N@5	N@10	N@20	N@100
BC	-	0.284	0.596	0.582	0.546	0.338	0.198	0.265	0.282	0.302	0.337
BC-SFD	22.1	0.282	0.576	0.568	0.507	0.340	0.198	0.262	0.270	0.280	0.339
BC-CODRA	6.0	0.306	0.592	0.560	0.536	0.354	0.209	0.269	0.278	0.299	0.351
BC-LP	7.0	0.396	0.620	0.594	0.571	0.426	0.269	0.293	0.303	0.325	0.415
CM	-	0.298	0.600	0.580	0.572	0.371	0.212	0.282	0.288	0.318	0.360
CM-SFD	11.7	0.290	0.620	0.590	0.573	0.369	0.208	0.291	0.289	0.319	0.366
CM-CODRA	6.0	0.320	0.624	0.594	0.564	0.368	0.224	0.287	0.296	0.319	0.369
CM-LP	7.0	0.380	0.608	0.594	0.569	0.419	0.262	0.297	0.300	0.325	0.416
OA	-	0.225	0.516	0.508	0.453	0.298	0.159	0.234	0.239	0.249	0.304
OA-SFD	15.1	0.229	0.524	0.504	0.464	0.302	0.166	0.235	0.240	0.259	0.304
OA-CODRA	6.0	0.239	0.556	0.512	0.465	0.306	0.163	0.243	0.238	0.258	0.308
OA-LP	7.0	0.293	0.600	0.578	0.556	0.344	0.212	0.276	0.279	0.305	0.348
PRF	-	0.064	0.090	0.098	0.099	0.081	0.041	0.055	0.055	0.061	0.073
AAM	-	0.127	0.230	0.234	0.225	0.157	0.066	0.100	0.115	0.124	0.143

Table 6: Comparison of the retrieval effectiveness of the examined rank aggregation methods with various precision evaluation measures, on the WA-12, WA-13, and WA-14 datasets.

CDS-14	n_{it}	Precision					NDCG				
		MAP	P@5	P@10	P@20	P@100	Mean	N@5	N@10	N@20	N@100
BC	-	0.197	0.460	0.407	0.368	0.233	0.153	0.385	0.345	0.314	0.276
BC-SFD	23.7	0.206	0.460	0.417	0.353	0.238	0.158	0.397	0.359	0.313	0.289
BC-CODRA	6.0	0.207	0.473	0.427	0.378	0.242	0.160	0.402	0.366	0.317	0.288
BC-LP	7.0	0.216	0.433	0.423	0.357	0.249	0.157	0.357	0.341	0.306	0.292
CM	-	0.196	0.447	0.427	0.365	0.236	0.152	0.370	0.351	0.309	0.280
CM-SFD	12.2	0.201	0.440	0.400	0.372	0.235	0.154	0.360	0.333	0.314	0.281
CM-CODRA	6.0	0.202	0.460	0.423	0.368	0.237	0.157	0.384	0.355	0.318	0.284
CM-LP	7.0	0.212	0.447	0.417	0.362	0.246	0.158	0.368	0.345	0.310	0.291
OA	-	0.180	0.393	0.377	0.313	0.227	0.134	0.319	0.306	0.273	0.267
OA-SFD	19.7	0.180	0.393	0.373	0.317	0.229	0.133	0.310	0.295	0.273	0.267
OA-CODRA	6.0	0.180	0.393	0.373	0.317	0.229	0.134	0.312	0.300	0.272	0.267
OA-LP	7.0	0.168	0.427	0.380	0.347	0.225	0.128	0.346	0.323	0.299	0.272
PRF	-	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a
AAM	-	0.038	0.041	0.037	0.040	0.044	0.035	0.050	0.052	0.050	0.043

CDS-15	n_{it}	Precision					NDCG				
		MAP	P@5	P@10	P@20	P@100	Mean	N@5	N@10	N@20	N@100
BC	-	0.211	0.487	0.453	0.420	0.292	0.125	0.368	0.343	0.312	0.290
BC-SFD	16.4	0.218	0.473	0.440	0.418	0.297	0.128	0.366	0.333	0.312	0.291
BC-CODRA	6.0	0.221	0.493	0.443	0.418	0.299	0.132	0.375	0.336	0.306	0.292
BC-LP	7.0	0.238	0.553	0.497	0.418	0.313	0.139	0.380	0.358	0.312	0.314
CM	-	0.214	0.540	0.487	0.430	0.300	0.128	0.379	0.347	0.321	0.298
CM-SFD	11.5	0.219	0.527	0.490	0.413	0.297	0.133	0.390	0.359	0.319	0.297
CM-CODRA	6.0	0.216	0.540	0.487	0.428	0.298	0.128	0.376	0.351	0.320	0.299
CM-LP	7.0	0.232	0.553	0.493	0.432	0.310	0.133	0.371	0.348	0.316	0.311
OA	-	0.197	0.447	0.407	0.370	0.281	0.111	0.315	0.299	0.278	0.271
OA-SFD	13.2	0.197	0.454	0.403	0.368	0.281	0.113	0.315	0.299	0.278	0.272
OA-CODRA	6.0	0.199	0.440	0.410	0.368	0.281	0.113	0.310	0.301	0.278	0.272
OA-LP	7.0	0.206	0.500	0.463	0.427	0.298	0.116	0.352	0.334	0.312	0.306
PRF	-	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a
AAM	-	0.044	0.092	0.090	0.074	0.057	0.029	0.070	0.075	0.064	0.057

PMA-17	n_{it}	Precision					NDCG				
		MAP	P@5	P@10	P@20	P@100	Mean	N@5	N@10	N@20	N@100
BC	-	0.272	0.620	0.560	0.492	0.315	0.242	0.619	0.558	0.507	0.445
BC-SFD	22.4	0.289	0.620	0.590	0.500	0.326	0.251	0.607	0.575	0.512	0.456
BC-CODRA	6.0	0.294	0.627	0.573	0.497	0.331	0.254	0.612	0.567	0.511	0.462
BC-LP	7.0	0.298	0.627	0.550	0.480	0.344	0.259	0.632	0.563	0.507	0.482
CM	-	0.271	0.613	0.557	0.493	0.324	0.246	0.621	0.560	0.519	0.462
CM-SFD	13.0	0.287	0.627	0.560	0.500	0.338	0.255	0.620	0.562	0.516	0.468
CM-CODRA	6.0	0.282	0.613	0.570	0.502	0.333	0.252	0.616	0.571	0.522	0.468
CM-LP	7.0	0.291	0.607	0.550	0.487	0.334	0.252	0.608	0.549	0.504	0.464
OA	-	0.255	0.580	0.553	0.458	0.302	0.217	0.547	0.528	0.462	0.412
OA-SFD	18.4	0.255	0.580	0.555	0.460	0.306	0.218	0.548	0.529	0.462	0.412
OA-CODRA	6.0	0.255	0.580	0.557	0.465	0.308	0.218	0.548	0.529	0.462	0.414
OA-LP	7.0	0.231	0.600	0.570	0.502	0.294	0.225	0.609	0.566	0.514	0.435
PRF	-	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a
AAM	-	0.035	0.057	0.057	0.052	0.053	0.032	0.044	0.044	0.041	0.046

Table 7: Comparison of the retrieval effectiveness of the examined rank aggregation methods with various precision evaluation measures, on the CDS-14, CDS-15, and PMA-17 datasets.

MASO	n_{it}	Precision					NDCG				
		MAP	P@5	P@10	P@20	P@100	Mean	N@5	N@10	N@20	N@100
BC	-	0.315	0.290	0.280	0.263	-	0.294	0.291	0.284	0.273	-
BC-SFD	18.4	0.324	0.310	0.325	0.285	-	0.298	0.298	0.315	0.292	-
BC-CODRA	8.0	0.333	0.280	0.310	0.300	-	0.305	0.303	0.316	0.312	-
BC-LP	9.0	0.330	0.290	0.285	0.303	-	0.299	0.288	0.284	0.301	-
CM	-	0.313	0.270	0.270	0.275	-	0.290	0.257	0.261	0.270	-
CM-SFD	21.0	0.317	0.280	0.290	0.275	-	0.295	0.279	0.285	0.279	-
CM-CODRA	9.0	0.317	0.230	0.270	0.287	-	0.294	0.245	0.267	0.286	-
CM-LP	10.0	0.309	0.180	0.210	0.273	-	0.285	0.184	0.205	0.255	-
OA	-	0.320	0.260	0.265	0.273	-	0.294	0.256	0.261	0.271	-
OA-SFD	21.0	0.321	0.260	0.265	0.275	-	0.294	0.256	0.261	0.271	-
OA-CODRA	8.8	0.320	0.260	0.265	0.273	-	0.294	0.256	0.261	0.271	-
OA-LP	9.8	0.303	0.210	0.230	0.218	-	0.270	0.210	0.221	0.249	-
PRF	-	0.324	0.290	0.300	0.302	-	0.300	0.295	0.301	0.306	-
AAM	-	0.198	0.360	0.325	0.318	-	0.278	0.380	0.349	0.339	-

MAMO	n_{it}	Precision					NDCG				
		MAP	P@5	P@10	P@20	P@100	Mean	N@5	N@10	N@20	N@100
BC	-	0.342	0.270	0.275	0.302	0.335	0.261	0.253	0.264	0.288	0.324
BC-SFD	9.3	0.343	0.320	0.280	0.295	0.330	0.264	0.342	0.305	0.308	0.329
BC-CODRA	5.0	0.344	0.340	0.300	0.307	0.336	0.265	0.353	0.322	0.319	0.335
BC-LP	6.0	0.349	0.300	0.345	0.367	0.349	0.266	0.300	0.332	0.353	0.346
CM	-	0.342	0.300	0.290	0.315	0.336	0.262	0.282	0.281	0.302	0.327
CM-SFD	9.4	0.340	0.270	0.270	0.277	0.330	0.261	0.268	0.269	0.274	0.320
CM-CODRA	5.0	0.344	0.320	0.305	0.310	0.334	0.264	0.339	0.326	0.321	0.334
CM-LP	6.0	0.349	0.280	0.320	0.365	0.343	0.266	0.292	0.318	0.351	0.342
OA	-	0.342	0.280	0.315	0.318	0.328	0.262	0.265	0.294	0.304	0.321
OA-SFD	10.2	0.342	0.300	0.310	0.333	0.321	0.263	0.295	0.307	0.325	0.320
OA-CODRA	5.0	0.345	0.320	0.315	0.320	0.326	0.264	0.313	0.311	0.316	0.325
OA-LP	6.0	0.345	0.350	0.345	0.340	0.333	0.265	0.349	0.346	0.342	0.335
PRF	-	0.347	0.340	0.320	0.325	0.339	0.265	0.344	0.331	0.331	0.338
AAM	-	0.254	0.310	0.283	0.313	0.330	0.207	0.324	0.305	0.298	0.199

FESO	n_{it}	Precision					NDCG				
		MAP	P@5	P@10	P@20	P@100	Mean	N@5	N@10	N@20	N@100
BC	-	0.243	0.140	0.155	0.153	-	0.288	0.164	0.246	0.373	-
BC-SFD	19.8	0.249	0.160	0.185	0.158	-	0.291	0.169	0.277	0.395	-
BC-CODRA	10.2	0.252	0.180	0.160	0.170	-	0.292	0.192	0.262	0.406	-
BC-LP	11.2	0.260	0.180	0.170	0.155	-	0.297	0.194	0.276	0.392	-
CM	-	0.243	0.140	0.165	0.158	-	0.287	0.161	0.258	0.382	-
CM-SFD	16.4	0.237	0.150	0.185	0.165	-	0.280	0.143	0.268	0.391	-
CM-CODRA	10.4	0.235	0.140	0.155	0.160	-	0.285	0.160	0.243	0.376	-
CM-LP	11.4	0.250	0.150	0.140	0.163	-	0.293	0.178	0.246	0.398	-
OA	-	0.236	0.140	0.165	0.158	-	0.284	0.152	0.247	0.375	-
OA-SFD	12.4	0.262	0.160	0.155	0.163	-	0.301	0.194	0.266	0.416	-
OA-CODRA	11.7	0.241	0.160	0.175	0.155	-	0.287	0.170	0.266	0.375	-
OA-LP	12.7	0.195	0.170	0.160	0.138	-	0.277	0.196	0.267	0.384	-
PRF	-	0.225	0.160	0.170	0.148	-	0.277	0.159	0.240	0.353	-
AAM	-	0.159	0.170	0.145	0.110	-	0.163	0.186	0.241	0.366	-

Table 8: Comparison of the retrieval effectiveness of the examined rank aggregation methods with various precision evaluation measures, on the MASO, MAMO, and FESO synthetic datasets.

FEMO	n_{it}	Precision					NDCG				
		MAP	P@5	P@10	P@20	P@100	Mean	N@5	N@10	N@20	N@100
BC	-	0.343	0.390	0.370	0.352	0.341	0.275	0.380	0.372	0.358	0.353
BC-SFD	8.8	0.341	0.450	0.425	0.362	0.332	0.276	0.417	0.411	0.370	0.346
BC-CODRA	6.0	0.350	0.410	0.400	0.365	0.342	0.280	0.427	0.415	0.386	0.362
BC-LP	7.0	0.348	0.390	0.380	0.345	0.340	0.284	0.410	0.396	0.366	0.375
CM	-	0.334	0.380	0.380	0.345	0.341	0.274	0.377	0.378	0.353	0.353
CM-SFD	7.8	0.341	0.380	0.405	0.368	0.331	0.276	0.376	0.394	0.372	0.349
CM-CODRA	6.0	0.345	0.410	0.385	0.372	0.339	0.279	0.438	0.410	0.393	0.361
CM-LP	7.0	0.346	0.280	0.285	0.348	0.331	0.280	0.311	0.303	0.342	0.354
OA	-	0.344	0.420	0.365	0.352	0.338	0.276	0.416	0.380	0.366	0.353
OA-SFD	9.5	0.336	0.340	0.335	0.360	0.327	0.272	0.324	0.326	0.346	0.336
OA-CODRA	6.0	0.351	0.420	0.385	0.352	0.343	0.279	0.418	0.395	0.370	0.359
OA-LP	7.0	0.357	0.390	0.425	0.380	0.350	0.287	0.424	0.437	0.402	0.391
PRF	-	0.337	0.300	0.355	0.322	0.336	0.271	0.278	0.325	0.312	0.339
AAM	-	0.275	0.360	0.365	0.333	0.332	0.246	0.370	0.372	0.346	0.344

FELO	n_{it}	Precision					NDCG				
		MAP	P@5	P@10	P@20	P@100	Mean	N@5	N@10	N@20	N@100
BC	-	0.246	0.200	0.235	0.220	0.232	0.258	0.201	0.225	0.219	0.402
BC-SFD	11.4	0.248	0.270	0.215	0.220	0.226	0.263	0.291	0.245	0.238	0.404
BC-CODRA	7.9	0.247	0.260	0.250	0.235	0.225	0.061	0.255	0.247	0.238	0.399
BC-LP	8.9	0.234	0.200	0.230	0.222	0.227	0.054	0.181	0.207	0.211	0.397
CM	-	0.241	0.170	0.210	0.233	0.232	0.255	0.170	0.196	0.217	0.397
CM-SFD	7.1	0.244	0.220	0.235	0.263	0.220	0.259	0.224	0.233	0.254	0.389
CM-CODRA	6.9	0.248	0.230	0.235	0.217	0.231	0.258	0.211	0.221	0.213	0.400
CM-LP	7.9	0.237	0.210	0.210	0.240	0.225	0.257	0.237	0.205	0.229	0.402
OA	-	0.239	0.150	0.200	0.217	0.231	0.252	0.132	0.172	0.194	0.387
OA-SFD	6.4	0.241	0.155	0.213	0.217	0.231	0.252	0.132	0.172	0.194	0.387
OA-CODRA	7.1	0.241	0.155	0.213	0.217	0.231	0.252	0.132	0.172	0.194	0.387
OA-LP	8.1	0.223	0.140	0.180	0.197	0.202	0.233	0.114	0.156	0.187	0.365
PRF	-	0.255	0.350	0.280	0.215	0.238	0.266	0.340	0.294	0.244	0.428
AAM	-	0.135	0.240	0.245	0.245	0.219	0.140	0.238	0.243	0.245	0.401

MOSO	n_{it}	Precision					NDCG				
		MAP	P@5	P@10	P@20	P@100	Mean	N@5	N@10	N@20	N@100
BC	-	0.483	0.430	0.460	0.467	0.463	0.436	0.434	0.454	0.463	0.789
BC-SFD	24.5	0.487	0.490	0.425	0.440	0.463	0.433	0.472	0.430	0.440	0.785
BC-CODRA	5.4	0.492	0.550	0.485	0.467	0.463	0.445	0.543	0.501	0.484	0.801
BC-LP	6.4	0.493	0.500	0.485	0.495	0.463	0.443	0.499	0.489	0.495	0.799
CM	-	0.478	0.420	0.475	0.475	0.458	0.435	0.426	0.465	0.469	0.789
CM-SFD	24.2	0.484	0.460	0.425	0.445	0.457	0.435	0.476	0.444	0.452	0.791
CM-CODRA	5.5	0.491	0.550	0.495	0.490	0.460	0.445	0.545	0.508	0.500	0.802
CM-LP	6.5	0.485	0.530	0.475	0.477	0.458	0.441	0.517	0.482	0.482	0.796
OA	-	0.480	0.440	0.485	0.445	0.463	0.435	0.434	0.467	0.446	0.786
OA-SFD	23.4	0.481	0.460	0.435	0.430	0.463	0.439	0.491	0.463	0.449	0.793
OA-CODRA	5.1	0.494	0.460	0.475	0.478	0.463	0.444	0.485	0.487	0.485	0.801
OA-LP	6.1	0.500	0.540	0.520	0.493	0.463	0.448	0.540	0.528	0.505	0.807
PRF	-	0.462	0.350	0.405	0.430	0.458	0.423	0.335	0.382	0.407	0.768
AAM	-	0.241	0.460	0.455	0.468	0.224	0.380	0.444	0.447	0.457	0.788

Table 9: Comparison of the retrieval effectiveness of the examined rank aggregation methods with various precision evaluation measures, on the FEMO, FELO, and MOSO synthetic datasets.

across six iterations will eventually produce weights with significant discrepancies. The second reason is that, according to the discussion of Subsection 5.1 and Eq. 14, the weights are normalized with min-max scaling before they are plugged into a rank aggregation method. Consequently, even the smallest of the differences among them is enlarged.

The performance of PRF was remarkably low in all TREC datasets, since its precision was hugely smaller than all the other methods, with the exception of AAM. A different selection for the values of its hyper-parameters, other than the default ones, would probably improve its effectiveness; here, we used $\alpha = \beta = 0.5$, as it was suggested by Desarkar et al. (2016). Moreover, PRF failed to complete some experiments in our 32GB workstation. The reason is that this method creates a preference aggregate graph in its last stage. Each node in the graph is connected to every other node by 1 or 2 edges. Consequently, its space complexity is quadratic to the number of elements in the aggregate list, leading to "out of memory" conditions when that list is very long. Furthermore, as a majoritarian method, PRF was very slow on these tests.

The performance of AAM was even worse. However, as mentioned earlier, it is not comparable to the algorithm of Chatterjee et al. (2018) due to the different weight initialization approach. The bad results is an indication of the ineffectiveness of the greedy nature of AAM that immediately merges the two closest input lists. Although this interesting approach originates from the hierarchical clustering theory, it was found inferior to other majority-based methods (e.g., the Condorcet method), or the techniques that try to minimize the distances between the input lists and the aggregate list (e.g., Kemeny optimal aggregation).

The effectiveness of our distance-based technique was verified again by the experiments with the six synthetic datasets (Tables 8 and 9). Specifically, the proposed model enhanced the precision of BC in all cases. The gains were larger in MASO ($\simeq 5.7\%$), and smaller in FELO ($< 1\%$). This observation combined with the results on the TREC datasets indicates that the impact of our method is magnified in the presence of many voters, because the aggregate list is generated by multiple rankings. The situation was similar on CM and OA, with only a few individual exceptions (e.g. CM-SFD and CM-CODRA in FESO, CM-SFD in the MAMO, and OA-SFD in FEMO).

On the other hand, the application of the list pruning method led to mixed results. In some cases, the improvement was substantial, for example, BC-LP and CM-LP in FESO, whereas, in others, the precision degradation was significant, e.g., the OA-LP methods in 4 out of 6 synthetic datasets. Apparently, the list pruning strategy maximizes the obtained benefits when it is applied to long input lists. In the opposite case, the omission of specific list elements may hurt performance, because the probability that these elements are indeed relevant increases.

The performance of PRF was hugely improved in the experiments with the synthetic datasets. More specifically, this algorithm outperformed all its adversaries in the FELO dataset, whereas on MASO, its precision was inferior only compared against BC-CODRA. In contrast, PRF was outperformed by all of its counterparts, both weighted and non-weighted, on FELO and on MOSO.

Regarding AAM, its measured precisions were improved, albeit in terms of MAP, they were all still significantly smaller than those of the other attested methods. Nevertheless, in several cases, the local Precision and nDCG measurements were rather satisfactory. Indicatively, the values of P@5, P@10, N@5 and N@10 of AAM on the MOSO dataset were higher than those of PRF, and of those of the non-weighted methods.

7. Discussion

Undoubtedly, there is a large number of measurements presented in Tables 5, 6, 7, 8, and 9. For this reason, we attempt to simplify and interpret their reading by summarizing the percentile performance benefits of the various methods in Table 10. More specifically, in the second column we report the average number of iterations performed by each method, whereas the next four columns contain the average and the maximum improvements in MAP and Mean nDCG.

The primary conclusion that derives from the numbers of Table 10 is that the distance between an input list and the aggregate list is indeed indicative of the importance of the corresponding voter. Hence, the introduction of distance-based voter weights is a reasonable approach that enhanced all the baseline methods, regardless of the utilized distance metric.

Specifically, the MAPs and Mean nDCGs of BC-SFD were on average 2.9% and 1.5% higher than those of the original Borda Count method, respectively. These gains were magnified by employing CODRA, since BC-CODRA was more effective than BC by 7.2% and 6.7% in terms of MAP and Mean nDCG, respectively. Similar results were obtained for the Condorcet method, where the application of CODRA instead of SFD was beneficial in terms of both effectiveness and number of iterations. Compared to the baseline CM, the MAPs and Mean nDCGs of CM-CODRA were on average 7.3% and 9% higher, respectively. The Outranking approach was the hardest method to improve. Thus, the application of the proposed model in combination with SFD yielded small improvements of 0.7% and 0.4% in terms of MAP and Mean nDCG, respectively. The usage of CODRA improved the situation and, on average, it led to an improvement of 3% for MAP, and 2.2% for Mean nDCG.

Remarkably, the list pruning algorithm led to substantial gains over all baseline methods. For Borda Count, the MAPs and Mean nDCGs were increased on average by 20% and 18.5%, respectively. The maximum performance of BC-LP was measured in the WA-14 dataset, where MAP and Mean nDCG were boosted by almost 40% and 36%, respectively. Regarding CM-LP, MAP and Mean nDCG were also substantially enhanced by an average margin of 27.5% and 35.3%, respectively. In OA, the average MAP and Mean nDCGs were improved on average by 12.7% and 11.3%, respectively. Their maximum values were measured in the WA-14 dataset and approached 30.2% and 33.3%, respectively.

Compared to the recent state-of-the-art solutions, all our weighted models outperformed both the Preference Relation (PRF) method of Desarkar et al. (2016), and the variant of the agglomerative approach (AAM) of Chatterjee et al. (2018) in all TREC datasets. These methods were proved both ineffective and expensive in these datasets. PRF failed to complete several tests due to memory shortage. For the same reason, we could not apply the original AAM method, and we were forced to implement a variant that assigns equal initial weights to the voters. On the other hand, the effectiveness of these approaches was substantially improved in the synthetic datasets. PRF was also able to win all its adversary methods (including our own) in the MASO dataset, where numerous voters submit short ranked lists.

In some isolated cases, the application of our model led to negative results. Namely, the achieved performance was lower than that of the corresponding baseline methods. For example, in the WA-13 and WA-14 datasets, the measured MAPs and nDCGs of BC-SFD were lower than those achieved by the non-weighted BC. Similarly, in WA-14, the effectiveness of CM-SFD was also negative compared to that of CM. However, notice that the employment of CODRA cured most of these problems, and the gains were rendered positive. These results indicate the importance of employing a robust technique to measure the list distances, and highlight the usefulness of CODRA.

Moreover, the computation of the voter weights using SFD requires on average more iterations than the introduced CODRA. Therefore, BC-SFD, CM-SFD, and OA-SFD performed or average 21.7, 11.7, and 17 iterations, respectively. On the other hand, CODRA rendered our model significantly faster and more stable, since six iterations were adequate to generate converged voter weights on the TREC datasets. A

Method	$\overline{n}_{it}$	MAP		Mean nDCG	
		avg (%)	max (%)	avg (%)	max (%)
BC-SFD	21.7	2.9	6.3 (PMA-17)	1.5	3.7 (PMA-17)
BC-CODRA	6.0	7.2	13.8 (WA-10)	6.7	12.4 (WA-12)
BC-LP	7.0	20.0	39.4 (WA-14)	18.5	35.9 (WA-14)
CM-SFD	11.7	4.1	15.5 (WA-10)	5.3	18.3 (WA-10)
CM-CODRA	6.0	7.3	20.5 (WA-10)	9.0	25.2 (WA-10)
CM-LP	7.0	13.0	27.5 (WA-14)	9.4	35.3 (WA-13)
OA-SFD	17.0	0.7	3.7 (WA-09)	0.4	4.4 (WA-14)
OA-CODRA	6.0	3.0	7.3 (WA-09)	2.2	5.2 (WA-10)
OA-LP	7.0	12.7	30.2 (WA-14)	11.3	33.3 (WA-14)

Table 10: Average and maximum percentile improvements in MAP and nDCG of the proposed methods over the corresponding baseline approaches.

similar improvement was also observed in the three synthetic datasets.

Having commented adequately on the average and maximum MAP and Mean nDCG, we annotate some additional indicative numbers. Regarding the measured values of Precision at several points of the aggregate list, the average beneficial effects of our model on P@5 were similar to those of MAP. Hence, BC-SFD and BC-CODRA improved this metric by margins, which, on average, were roughly equal to 2.6% and 4.8%, respectively. These values were slightly greater in the case of CM, since the gains of CM-SFD and CM-CODRA approached 3.1% and 6%, respectively. On the other hand, P@5 was not significantly affected by the application of our model in combination with OA; the enhancements were about 0.5% for OA-SFD and 2% for OA-CODRA.

As expected, list pruning enhanced P@5 further. Therefore, BC-LP, CM-LP, and OA-LP outperformed their respective baselines by 7.6%, 6.0% and 8.6%, respectively. Benefits of similar or slightly greater magnitudes were observed for the other Precision values, namely, P@10, P@20, and P@100. Indicatively, CM-CODRA enhanced P@10 by 16% and 15.2% in WA-09 and WA-13, respectively, while the average increase in all nine datasets was 6%. Regarding list pruning, BL-LP increased P@10 by 11.9% and 11.1% in WA-09 and WA-11, respectively, CM-LP boosted P@10 by 12.8% in WA-13, and OA-LP achieved a substantial improvement of approximately 33% in WA-12.

We conclude our discussion by summarizing the limitations of the proposed method. As mentioned earlier, a common feature of the TREC datasets is that the input lists are long. Therefore, there is a large number of elements that can be discarded. Nevertheless, when the input lists are short, then the erroneous removal of even a single element may substantially degrade the performance. This behavior was exhibited in the MASO synthetic dataset. A strategy to minimize this effect is to increase the values of the hyper-parameters δ_1 and δ_2. This manual solution will prevent the algorithm from pruning too many elements from the input lists. This is in fact a subject of future work, and a new line of research can be set from this point.

Similarly to the vast majority of the unsupervised learning methods, the effectiveness of the proposed algorithm depends on the values of several hyper-parameters. More specifically, the list pruning strategy includes two such hyper-parameters, δ_1 and δ_2. According to the analysis of Subsection 6.3, their values may have a significant impact on the accuracy of the algorithm, so an amount of manual fine-tuning may be necessary with respect to the underlying dataset. Of course, the competitive unsupervised aggregation methods of Desarkar et al. (2016) and Chatterjee et al. (2018) also introduce their own hyper-parameters that require manual fine tuning.

The proposed method is neither majoritarian, as the Condorcet method or the unsupervised algorithm of Desarkar et al. (2016), nor agglomerative, i.e., it does not perform repeated pairwise list merges as the technique of Chatterjee et al. (2018) does. Instead, its running times depend on the complexity of the non-weighted method that is used to perform the initial aggregation. Consequently, if it is applied in combination with an expensive algorithm, such as the Outranking Approach of Farah & Vanderpooten (2007), its iterative nature will decelerate the execution even further. In such cases, the size of the data will determine which non-weighted algorithm should be employed. If the data is large, e.g., we have very long input lists, then Borda Count is the preferred method to utilize, because it was proved much faster than the majoritarian CM and OA, especially in the large TREC datasets. Alternatively, the number of iterations can be reduced by lowering the convergence precision of the voter weights.

8. Conclusions and Future Work

In this paper we introduced a new unsupervised weighted rank aggregation method that automatically determines the importance of the voters. The relevant literature includes several such methods. A portion of them set and solve optimization problems that are computationally very expensive, whereas some others require not only the item rankings, but also their individual scores that are often unavailable. Another category of works are majoritarian, focusing on specific applications, e.g., Web metasearch, where the number of voters is small and/or the input lists are short.

Motivated by these disadvantages, we developed our model by applying the concept that *the importance of the voters is reflected by the distances between their submitted lists and the list that derives after the*

aggregation of multiple input lists. We conducted an analysis by using 9 real and 3 synthetic datasets, and we verified the statistical significance of the aforementioned idea. More specifically, the correlation between the importance of a voter and the distance of its input list from the aggregate list was proved statistically significant.

Avoiding the computationally expensive optimization problems and the majority votes, the introduced model initially applies a typical non-weighted method to generate a temporary aggregate list. Then, it repeatedly enhances this temporary list by measuring its distances from all the input lists. The voter weights are determined by a converging kernel function that assigns high weights to the voters whose submitted lists are more proximal to the aggregate list, and vice versa. This process is repeated until all the weights converge to their final values and the aggregate list becomes stable.

The experimental evaluation of the model indicated a statistically significant improvement of the achieved precision of the generated aggregate list. Compared to its non-weighted counterparts and two recent state-of-the-art unsupervised methods, the proposed model exhibited superior performance in terms of both precision and nDCG. The benefits in performance depend on the employed rank aggregation method. The average improvement for MAP was approximately 7-8%, and for nDCG was in the 7-9% range. In some datasets, the benefits approached 25%.

The experiments have also revealed that the role of the metric that is employed to measure the list-wise distances is crucial. It was shown that, although the traditional Spearman's footrule distance (SFD) yields decent results, other techniques can improve the aggregation quality even further. For this reason, we introduced a new distance measure, named *CODRA*, that treats the input lists as weighted vectors. This approach provides the ability to employ standard vector distance/similarity measures such as Cosine similarity and the Jaccard co-efficient. Compared to SFD, we demonstrated that CODRA is both faster and more effective. More specifically, our model in combination with this metric required fewer iterations to produce stable weights for the voters.

Moreover, we investigated additional ways of exploiting the learned weights. To the best of our knowledge, this work is the first to apply the voter weights not only to assign improved scores to the involved list elements, but also to determine their population. In this context, we introduced a pruning algorithm that discards the low ranked elements from the input lists of the weaker voters. Despite its simplicity, this method has been proved very effective, especially when the input lists are long. At the cost of only one additional iteration, it led to large improvements in performance, which, on average, approached 18.5–20% for Borda Count, and 10–13% for the rest of our attested methods.

Nevertheless, the effectiveness of this pruning strategy was limited, or even reversed, when it was applied to short input lists. In such cases, an aggressive removal of elements from the input lists may hurt the overall performance. Consequently, one of the most interesting topics of future work is to develop an efficient strategy that will address this issue. Furthermore, we intend to investigate additional scenarios for exploiting the learned weights, such as identifying and discarding the irrelevant voters (or the spammers) completely from the aggregation process, instead of just pruning their respective lists. We are also planning to study more forms for the converging kernel function that determines the user weights. CODRA is also included in this future work. The objective of this investigation is double: first, to reduce the number of iterations even further, and, second, to enhance the precision of our model.

References

Ailon, N., Charikar, M., & Newman, A. (2008). Aggregating inconsistent information: Ranking and clustering. *Journal of the ACM*, *55*, 23.

Akritidis, L., Fevgas, A., & Bozanis, P. (2019). An iterative distance-based model for unsupervised weighted rank aggregation. In *Proceedings of the 2019 IEEE/WIC/ACM International Conference on Web Intelligence* (pp. 358–362). ACM.

Akritidis, L., Katsaros, D., & Bozanis, P. (2008). Effective ranking fusion methods for personalized metasearch engines. In *Proceedings of the 12th Panhellenic Conference on Informatics* (pp. 39–43). IEEE.

Akritidis, L., Katsaros, D., & Bozanis, P. (2011). Effective rank aggregation for metasearching. *Journal of Systems and Software*, *84*, 130–143.

Aledo, J. A., Gámez, J. A., & Rosete, A. (2021). A highly scalable algorithm for weak rankings aggregation. *Information Sciences*, *570*, 144–171.

Amin, G. R., & Emrouznejad, A. (2011). Optimizing search engines results using linear programming. *Expert Systems with Applications*, *38*, 11534–11537.

Aslam, J. A., & Montague, M. (2001). Models for metasearch. In *Proceedings of the 24th ACM International Conference on Research and Development in Information Retrieval* (pp. 276–284).

Bartholdi, J., Tovey, C. A., & Trick, M. A. (1989). Voting schemes for which it can be difficult to tell who won the election. *Social Choice and welfare*, *6*, 157–165.

Bhowmik, A., & Ghosh, J. (2017). LETOR methods for unsupervised rank aggregation. In *Proceedings of the 26th International Conference on World Wide Web* (pp. 1331–1340).

de Borda, J. C. (1781). Mémoire sur les élections au scrutin. *Histoire de l'Academie Royale des Sciences*, (pp. 657–665).

Caragiannis, I., Chatzigeorgiou, X., Krimpas, G. A., & Voudouris, A. A. (2019). Optimizing positional scoring rules for rank aggregation. *Artificial Intelligence*, *267*, 58–77.

Chatterjee, S., Mukhopadhyay, A., & Bhattacharyya, M. (2018). A weighted rank aggregation approach towards crowd opinion analysis. *Knowledge-Based Systems*, *149*, 47–60.

Chen, J., Long, R., Wang, X.-l., Liu, B., & Chou, K.-C. (2016). PdRHP-PseRA: Detecting remote homology proteins using profile-based pseudo protein sequence and rank aggregation. *Scientific reports*, *6*, 32333.

Clémençon, S., & Jakubowicz, J. (2010). Kantorovich distances between rankings with applications to rank aggregation. In *Proceedings of Joint European Conference on Machine Learning and Knowledge Discovery in Databases* (pp. 248–263).

Cohen, W. W., Schapire, R. E., & Singer, Y. (1998). Learning to order things. In *Advances in Neural Information Processing Systems* (pp. 451–457).

Coppersmith, D., Fleischer, L. K., & Rurda, A. (2010). Ordering by weighted number of wins gives a good ranking for weighted tournaments. *ACM Transactions on Algorithms (TALG)*, *6*, 55.

De Condorcet, N. (1785). *Essai sur l'application de l'analyse à la probabilité des décisions rendues à la pluralité des voix*. Paris: Imprimerie Royale.

Desarkar, M. S., Sarkar, S., & Mitra, P. (2016). Preference relations based unsupervised rank aggregation for metasearch. *Expert Systems with Applications*, *49*, 86–98.

Díaz, I., Domínguez, M., Cuadrado, A. A., & Fuertes, J. J. (2008). A new approach to exploratory analysis of system dynamics using som. applications to industrial processes. *Expert Systems with Applications*, *34*, 2953–2965.

Dwork, C., Kumar, R., Naor, M., & Sivakumar, D. (2001). Rank aggregation methods for the Web. In *Proceedings of the 10th International Conference on World Wide Web* (pp. 613–622).

Emerson, P. (2013). The original Borda Count and partial voting. *Social Choice and Welfare*, *40*, 353–358.

Farah, M., & Vanderpooten, D. (2007). An outranking approach for rank aggregation in information retrieval. In *Proceedings of the 30th ACM Conference on Research and Development in Information Retrieval* (pp. 591–598).

Kilgour, D. M. (2010). Approval balloting for multi-winner elections. In *Handbook on approval voting* (pp. 105–124). Springer.

Klementiev, A., Roth, D., & Small, K. (2008). Unsupervised rank aggregation with distance-based models. In *Proceedings of the 25th International Conference on Machine Learning* (pp. 472–479).

Klementiev, A., Roth, D., Small, K., & Titov, I. (2009). Unsupervised rank aggregation with domain-specific expertise. In *Proceedings of the 21st International Joint Conferences on Artificial Intelligence* (pp. 1101–1106).

Lebanon, G., & Lafferty, J. (2002). Cranking: Combining rankings using conditional probability models on permutations. In *Proceedings of the 19th International Conference on Machine Learning* (pp. 363–370). volume 2.

Li, X., Wang, X., & Xiao, G. (2019). A comparative study of rank aggregation methods for partial and top ranked lists in genomic applications. *Briefings in bioinformatics*, *20*, 178–189.

Mallows, C. L. (1957). Non-null ranking models. *Biometrika*, *44*, 114–130.

Montague, M., & Aslam, J. A. (2002). Condorcet fusion for improved retrieval. In *Proceedings of the 11th ACM International Conference on Information and Knowledge Management* (pp. 538–548).

Omidi, L., Salehi, V., Zakerian, S., & Nasl Saraji, J. (2021). Assessing the influence of safety climate-related factors on safety performance using an integrated entropy-topsis approach. *Journal of Industrial and Production Engineering*, (pp. 1–10).

Onan, A., & Korukoğlu, S. (2017). A feature selection model based on genetic rank aggregation for text sentiment classification. *Journal of Information Science*, *43*, 25–38.

Pihur, V., Datta, S., & Datta, S. (2007). Weighted rank aggregation of cluster validation measures: A Monte Carlo cross-entropy approach. *Bioinformatics*, *23*, 1607–1615.

Reyes Ayala, B., Knudson, R., Chen, J., Cao, G., & Wang, X. (2018). Metadata records machine translation combining multi-engine outputs with limited parallel data. *Journal of the Association for Information Science and Technology*, *69*, 47–59.

Rosti, A.-V., Ayan, N. F., Xiang, B., Matsoukas, S., Schwartz, R., & Dorr, B. (2007). Combining outputs from multiple machine translation systems. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics* (pp. 228–235).

Samanlioglu, F., & Ayağ, Z. (2021). Concept selection with hesitant fuzzy anp-promethee ii. *Journal of Industrial and Production Engineering*, *38*, 547–560.

Tseng, M.-L., Tran, T. P. T., Ha, H. M., Bui, T.-D., & Lim, M. K. (2021). Sustainable industrial and operation engineering trends and challenges toward industry 4.0: A data driven analysis. *Journal of Industrial and Production Engineering*, *38*, 581–598.

Wang, M., Li, Q., Lin, Y., & Zhou, B. (2017). A personalized result merging method for metasearch engine. In *Proceedings of the 6th International Conference on Software and Computer Applications* (pp. 203–207).