Post-hoc Diversity-aware Curation of Rankings

Vassilis Markos[1] and Loizos Michael[1,2]
[1]*Open University of Cyprus, Nicosia, Cyprus*
[2]*CYENS Center of Excellence, Nicosia, Cyprus*

Keywords: Diversity, Preferences, Ranking, Shuffling.

Abstract: We consider the problem of constructing rankings that exhibit a prescribed level of diversity. We take a black-box view of the ranking procedure itself, and choose, instead, to post-hoc curate any given ranking, deviating from the original ranking only to the extent allowed by any given constraints. The curation algorithm that we present is oblivious to how diversity is measured, and returns shuffled versions of the original rankings that are optimal in terms of their exhibited level of diversity. Our empirical evaluation on synthetic data demonstrates the effectiveness and efficiency of our methodology, across a number of diversity metrics from the literature.

1 INTRODUCTION

The over-abundance of options in practically any facet of modern life makes it unrealistic for any individual to meaningfully consider them all. An often-used technology-mediated solution is to present these options in a ranked manner, as in the case of receiving a ranked list of results when posing a web search query.

Although these rankings might capture some form of preference that is centrally defined by the technology provider (e.g., ranking higher a well-connected web page), it is generally the case that the preferences expressed in rankings are adapted, through machine learning, to the individual who is consuming the rankings. Since the learning itself happens by observing the choices that the individual makes through those rankings, it is not surprising that this process leads to echo chambers, where the individual and the learning algorithm reinforce each other in a narrower and narrower view of what is to be ranked (or ranked higher).

Although such echo chambers are already inherently problematic, they can be even more so when what is being ranked are other individuals. Such could be the case, for instance, in social networking sites, where the linear manner in which one's "friends" might be presented might lead, through the reinforcement effect mentioned above, to one interacting with only a "selected few", exacerbating social stereotyping, biases against minorities, and discrimination.

Such considerations are particularly relevant in the context of the EU-funded project "WeNet: The Internet of Us", which seeks to design and develop a machine-mediated platform that connects a requester with potential volunteers who can help the former address a given task. To reduce the risk of the aforementioned issues arising, the project embraces diversity-awareness as a core principle in machine-mediation.

Diversity-awareness should not be taken to imply that maximum diversity is always desirable; instead, it implies that diversity should be carefully balanced in each particular context. Indeed, deciding when diversity should be promoted to ensure inclusion, and when it should be restricted to ensure protection, is an ethically-challenging question (Helm et al., 2021).

Accordingly, this work takes the position that how diversity is measured, and the level of diversity that should be exhibited by a ranking, are externally determined and given. Our proposed approach seeks, then, to demonstrate how that diversity can be achieved by *curating a given ranking*. The ranking itself is also assumed to be externally determined, which in the case of the WeNet project would correspond to a ranking expressing the user's preferences, as learned by observing the user interact within the WeNet platform. Since deviating considerably from this given ranking (while promoting or restricting the level of diversity with respect to what is already present) would antagonize the purpose of having a user-specific ranking to begin with, we assume that we are provided with constraints on what constitutes an acceptable deviation.

Given the above, our proposed curation algorithm proceeds to shuffle the given ranking, while respecting the given constraints, towards achieving a level of diversity that is as close to the given specifications as

Markos, V. and Michael, L.
Post-hoc Diversity-aware Curation of Rankings.
DOI: 10.5220/0010839600003116
In *Proceedings of the 14th International Conference on Agents and Artificial Intelligence (ICAART 2022) - Volume 2*, pages 323-334
ISBN: 978-989-758-547-0; ISSN: 2184-433X

possible. We prove that the algorithm returns optimal rankings, in a formally specified sense, and we empirically demonstrate its effectiveness and efficiency.

2 RELATED WORK

In what follows we review some of the most widely used ranking methodologies and metrics of diversity.

2.1 Ranking Methodologies

An early but popular learning-to-rank methodology is RankNet (Burges et al., 2005), where a Neural Network learns a ranking function from observed pairs (A,B) that are labeled by target probabilities $\overline{P}_{AB}$ that A is preferred over B. RankNet uses a probabilistic ranking cost function based on the target probabilities, which it seeks to minimize using gradient descent.

A well-known variant of RankNet is DirectRank (Köppel et al., 2019), which seeks to learn a quasi-order (i.e., reflexive, anti-symmetric, transitive) relation over an attribute space by observing pairwise comparisons of attributes. Since this is beyond what a standard Neural Network could do, DirectRank employs two identical Neural Networks that are trainable on pairwise comparisons, and appropriately enhances them and combines their outputs via an extra neuron.

In (Cakir et al., 2019), another Neural Network based learning-to-rank methodology, FastAP, is presented. FastAP is a Deep Learning method that is based on an algorithm that computes Average Precision (AP) significantly faster than other approaches — average precision is defined as the (continuous or discrete, accordingly) convolution of ranking precision with ranking recall. The architecture of FastAP relies on learning a Deep Neural Network that maps entities into a d-dimensional Euclidean space and is optimized for AP. The contribution of the above work is mostly focused on two axes: (i) a new significantly faster way to compute AP given a ranking is provided, based on discretizing the underlying convolution and using a histogram-based approach and; (ii) introducing a new learning strategy regarding mini-batching, were batches are collected so as to form difficult problems — i.e., they contain, among others, some highly similar entities — so as to reduce time complexity (namely, introducing harder learning examples helps avoiding larger batches during the training phase).

In (Pfannschmidt et al., 2018), there are presented two neural-based context-sensitive ranking methodologies that allow for a ranking to take into account information regarding the ranked items and possible underlying relations. The two methodologies presented, First-Aggregate-Then-Evaluate-Net (FATE-Net) and First-Evaluate-Then-Aggregate-Net (FETA-Net), both rely on computing a ranking score function which maps pairs of the form (a,A), where a is an entity and A a set of entities (i.e., a context), to a ranking score. The difference between FETA and FATE is that in FETA all scores for each entity are computed using a ranking function that takes into account pairwise (in the simplest case) dependencies between entities, while in FATE, a representative for each context is computed first, followed by the computation of the ranking scores, according again to a context dependent ranking (utility) function. (Oliveira et al., 2018) present a ranking/comparison methodology which relies on the analysis of paired comparison data. The novelty introduced by (Oliveira et al., 2018) is that, in contrast to other works, the comparison function — i.e., the function used to infer whether a is more preferred than b — is assumed to be only partially known. Namely, it is assumed that the comparison function belongs in a class of functions and make no other assumptions regarding its form — in the experiments presented in (Oliveira et al., 2018), this class consists of all the functions whose inverse is a polynomial of some certain degree with a bounded support. Given the above assumptions, a new comparison/ranking methodology is presented, PolyRank, which transforms the comparison learning problem into an efficiently solvable Least-Squares problem.

Perhaps the most close in spirit to our work is the learning-to-rank approach PRM (Personalized Re-ranking Model) (Pei et al., 2019). PRM is a methodology that relies on re-ranking ranked lists according to personalized criteria for a certain user. Like our work, PRM can be appended on top of any ranking methodology. Unlike our work, however, which simply receives a ranked list, PRM expects to receive ranking scores for each entity in an attribute space. It takes into account the output of the ranking process as well as relations among the ranked entities and the user's preferences/history (explicitly described) and re-computes ranking scores, thus yielding a new ranking that adheres to the user's personal preferences.

2.2 Diversity Metrics

In this subsection we shall present in detail ways in which diversity may be quantified. To begin with, measures of diversity have been widely studied, especially within the context of Ecosystem Biology as well as in other fields, such as business human resources management. Hence, there are numerous metrics that measure diversity from certain

viewpoints and under certain assumptions. Here we will discuss many already existing metrics as well as present ways in which they could be utilized during a ranking process with diversity in mind.

Starting with basic metrics, the most common one is *richness* (Colwell, 2009), defined as the number of different classes present in a certain sample of entities S. More formally, richness(S), is defined as follows:

$$\text{richness}(S) := \#\{p \in P : p \cap S \neq \varnothing\}. \quad (1)$$

A normalized version of richness, where we divide by the total number of classes in X, is the following one:

$$\overline{\text{richness}}(S) := \frac{\text{richness}(S)}{\#P}. \quad (2)$$

As one may observe, richness captures only vaguely one of the many meanings diversity has in everyday language as well as in several applications. A simple way to obtain more meaningful information regarding a sample's diversity is through the so-called Berger-Parker Index, $BP(S)$, (Berger and Parker, 1970), which corresponds to the maximum abundance ratio of S, i.e.:

$$BP(S) := \max_{i=1,\ldots,n} p_i, \quad (3)$$

where S is assumed to be partitioned into classes $\{C_1,\ldots,C_n\}$ and $p_i := {}^{\#C_i}/_{\#S}$, $i = 1,\ldots,n$, are the corresponding relative frequencies. In order to facilitate comparison with other metrics we will discuss later on, we shall use the following version of $BP(S)$ which is increasing with respect to sample diversity:

$$\overline{BP}(S) := 1 - BP(S). \quad (4)$$

Another way to measure diversity is to introduce more sophisticated metrics, such as Simpson's Index (Simpson, 1949) — often referred to as Herfindahl-Hirschmann Index (HHI) in economics (Herfindahl, 1950). Simpson's index, given a sample, S, partitioned into classes $\{C_1,\ldots,C_n\}$, represents the probability two randomly chosen entities to belong to the same class. So, if p_i, $i = 1,\ldots,n$, are the relative frequencies of all classes, C_i, as above, then Simpson's Index, $\lambda(S)$, equals:

$$\lambda(S) := \sum_{i=1}^{n} p_i^2. \quad (5)$$

In the same probabilistic view of diversity, one may also define Shannon's Index (Spellerberg and Fedor, 2003), which coincides with Shannon's entropy, i.e.:

$$H(S) := -\sum_{i=1}^{n} p_i \ln p_i. \quad (6)$$

The idea of Shannon's Index is that, the more uniformly distributed sample entities are, the more diverse a sample should be considered, since it is more difficult to guess to which class a randomly drawn entity would belong (Shannon, 1948). Indeed, Shannon's Index is equal to 0 in cases where all classes but one are empty, while it attains its maximum value, $\ln n$, for $p_i = {}^1/_n$, $i = 1,\ldots,n$.

A natural normalization would be to divide $H(S)$ by its maximum value, $\ln n$, obtaining the following normalized version of Shannon's Index:

$$\overline{H}(S) := \frac{H(S)}{\ln n}. \quad (7)$$

Other more complex metric of diversity are the so-called Hill numbers (Hill, 1973). Hill numbers are given by the following formula, where q is a parameter of the metric:

$${}^qD(S) := \left(\sum_{i=1}^{n} p_i^q\right)^{{}^1/_{1-q}}. \quad (8)$$

Observe that for $q = 1$, equation (8) cannot be computed, in which case we compute the corresponding limit as $q \to 1$, as shown in equation (9)

$${}^1D(S) := e^{H(S)}. \quad (9)$$

The above metric, also referred to as *effective number of species*, *true diversity* or *gamma-diversity* in Biology, corresponds to the number of equally abundant classes required to achieve the same average proportional abundance as in S (Chao et al., 2010; Colwell, 2009).

While Hill numbers are used in Biology to measure species diversity at a global scale, there also other metrics that allow for computing diversity at a more local scale — e.g. in smaller natural habitats or in local departments of businesses. One of the most popular choices is *alpha-diversity*, (Whittaker, 1960), which is defined as follows:

$${}^qD_\alpha(S) := \left(\sum_{i=1}^{k}\sum_{j=1}^{n} p_i p_{j|i}^{q-1}\right)^{{}^1/_{1-q}}. \quad (10)$$

In the above, n is the total number of classes, as previously, while k corresponds to the total number of local components — e.g., local habitats. Also, $p_{j|i}$ is the relative frequency of class C_j at component i while p_i is the relative size of component i in S. Based on alpha diversity, which is of local nature, another popular diversity metric is *beta-diversity*, (Whittaker, 1960; Tuomisto, 2010), which is defined as the ratio between total and local diversity, namely:

$${}^qD_\beta(S) = \frac{{}^qD(S)}{{}^qD_\alpha(S)}. \quad (11)$$

Another conceptualization of diversity is understanding diversity as the opposite of similarity. In this

sense, a natural choice could be to define diversity of a given sample S as:

$$\mathrm{div}(S) := \sum_{1 \le i < j \le \#S} w_{ij} \, \mathrm{sim}(x_i, x_j)^2. \quad (12)$$

In the above, sim is any (normalized) similarity metric defined on S and w_{ij} are non-negative weights that sum to 1, while $x_i, x_j \in S$. Some choices for sim, among others, could be L_p metrics, cosine similarity, etc.

3 POST-HOC SHUFFLING

In this section we present and discuss an abstract framework that shuffles a given ranking to improve its diversity according to a given metric μ. We start by presenting an unconstrained version of the algorithm where the shuffled ranking can differ arbitrarily from the initial ranking, and then show how to extend that to obey given constraints.

Let $X = \{x_1, \ldots, x_n\}$ be a finite set and consider a ranking function r, i.e., a function that maps X to a vector $r(X)$. Now, let us assume that μ is a certain metric we would like to monitor in $r(X)$ — e.g., the diversity of some part(s) of the returned ranked list. More precisely, let without loss of generality μ take values in $[0,1]$ and let $D \in \mathcal{P}([0,1])^n$ be a sequence of sets of μ's desired values for each initial part of a ranking $r(X)$ of the elements of a set X. For instance, D in case of a set $X = \{x_1, x_2, x_3, x_4\}$ could have the following form:

$$D = (\{0.47\}, [0.33, 0.65], \{0.56, 0.89\}, \{1\}). \quad (13)$$

So, if $r(X) = (x_3, x_2, x_4, x_1)$ is a ranking of X, D informs us that it is desired that $\mu(r(X)_1) = 0.47$, $\mu(r(X)_2) \in [0.33, 0.65]$, $\mu(r(X)_3) \in \{0.56, 0.89\}$ and $\mu(r(X)_4) = 1$, where by $r(X)_i$ we denote the list containing the first i elements in $r(X)$.

Then, given a set $X = \{x_1, \ldots, x_n\}$ and a ranking, $r(X)$, of it, we would like to shuffle $r(X)$ or, equivalently, provide an alternative ranking r^* such that each initial part $r^*(X)_i$ of $r^*(X)$ satisfies the condition $\mu(r^*(X)) \in D$. However, it is not difficult to see that this is not possible in most cases. So, a more realistic goal would be to demand that $r^*(X)$ is not globally inferior to any other ranking $r(X)$. Formally, we first define D-loss, $L_D(r^*)$, or simply $L(r^*)$ when D is obvious from the context, as follows:

$$L(r^*) := (\mathrm{dist}(\mu(r^*(X)_i), D_i))_{i=1}^n. \quad (14)$$

That is, $L(r^*)$ is the sequence of distances of μ computed at each initial part of $r^*(X)$ from D_i (the i-th set in D). Given that, we shall say that a ranking r_1 dominates another ranking r_2 if the following two conditions hold:

$$L(r_1)(i) \le L(r_2)(i), \text{ for every } i \in \{1, \ldots, n\} \quad (15)$$

$$L(r_1)(i_0) < L(r_2)(i_0), \text{ for some } i_0 \in \{1, \ldots, n\} \quad (16)$$

Given the above definition, a reasonable goal would be to compute a shuffled ranking r^* that is not dominated by any other shuffled ranking r' in the above sense; i.e., the shuffled ranking r^* is Pareto optimal.

A simple method to determine r^* is given by Algorithm 1, where at each iteration we compute the next element of $r^*(X)$ by sorting all unranked elements of X by ascending distance from D_i, $\mathrm{dist}(\mu(X \setminus r^*(X)), D_i)$, and appending the first element of that list to $r^*(X)$. Sorting is performed according to Algorithm 2.

Algorithm 1: Diversity-aware shuffling.

```
1: procedure SHUFFLE(X, μ, D)
2:     μRank ← ∅
3:     for i = 1, ..., X.length do
4:         x* ← SORTBY(X, μRank, μ, D_i)[0]
5:         μRank[i] ← x*
6:     end for
7:     return μRank
8: end procedure
```

Algorithm 2: SORTBY sorting algorithm.

```
1: procedure SORTBY(X, μRank, μ, D_i)
2:     dists ← ∅                          ▷ Empty dictionary.
3:     for x ∈ X \ μRank do
4:         dists[x] ← dist(μ(μRank ∪ {x}), D_i)
5:     end for
6:     return dists.sortByValue()         ▷ Sort keys.
7: end procedure
```

Algorithm 1 performs a greedy search for the best candidate to append next to the generated ranking as described above. At this point we should remark that, while SHUFFLE does not take the ranking function r into account, is utilized in SORTBY at each iteration as a tie-breaker. That is, from all the possible $x \in X$ that may satisfy $x = \arg\min_{x \in X \setminus \mu Rank} \mathrm{dist}(\mu(\mu Rank \cup \{x\}), D)$, we prefer the one ranked higher by r.

In the following lemma we state two obvious, yet important, properties that the returned ranking, $\mu Rank$ has:

Lemma 1. *Given a set $X = \{x_1, \ldots, x_n\}$, a ranking function r, a metric μ and a sequence of desired values of μ, D,* SHUFFLE *as described in Algorithm 1 returns a ranking $r^*(X)$ which satisfies the following*

two conditions:

$$\text{dist}(\mu(r^*(X)_1), D) = \min_{x \in X} \text{dist}(\mu(\{x\}), D_1) \quad (17)$$

$$\text{dist}(\mu(r^*(X)_i), D) = \min_{\substack{Y \subseteq X, \#Y = i \\ r^*(X)_{i-1} \subseteq Y}} \text{dist}(\mu(Y), D_i) \quad (18)$$

for each $i = 2, \ldots, n$.

Proof. Regarding $r^*(X)_1$, it is chosen in order to satisfy (17). Now, for some $i \geq 2$ we observe that if $Y \subseteq X$ is such that $\#Y = i$ and $r^*(X)_{i-1} \subseteq Y$, then, again, $r^*(X)_{i-1} \cup \{x^*\}$ — where x^* is determined as in Algorithm 1 — satisfies the following inequality:

$$\text{dist}(\mu(r^*(X)_{i-1} \cup \{x^*\}), D_i) \leq \text{dist}(\mu(Y), D_i). \quad (19)$$

So, r^* satisfies both (17) as well as (18). □

Using the above, we may now prove the following:

Theorem 1. *Given a set* $X = \{x_1, \ldots, x_n\}$*, a ranking function* r*, a metric* μ *and a sequence of desired values of* μ*,* D*,* SHUFFLE *returns a ranking* $r^*(X)$ *that is Pareto optimal.*

Proof. Follows directly from Lemma 1. □

As one may easily observe, the above method favors μ over r. That is, we only utilize our initial ranking as a tie-breaker — which could naturally lead to completely ignoring r in case no ties occur — while we focus on keeping μ as close as possible to the set of desired values, D. While in some scenarios this may be our target behavior, there are also cases where preserving μ is only a secondary objective compared to yielding an accurate ranking according to some given ranking function, r. A natural solution to this issue is to allow for a *strict* constrain capturing how $r^*(X)$ can deviate from $r(X)$, let us denote it by $R(r^*, r)$ or simply R.

Before we proceed with the rest of our framework, we shall remark at this point that we consider restrictions R that are monotonic with respect to the involved rankings' initial parts in the following sense: if R is violated for some initial part $r'(X)_i$ of $r'(X)$, then it is also violated for each initial part $r'(X)_j$ for $j > i$. This demand, while restricting the possible choices for R, is aligned with the idea that when it comes to ranking, it is the first positions that matter the most, so we would like to exclude the case where a ranking violates R in some initial positions while it does not entirely do so.

So, instead of returning a ranking, r^*, that is Pareto optimal among all the possible rankings, we seek to compute a ranking, r^*, that is Pareto optimal among all possible rankings that respect R.

In this direction, we can design a variation of Algorithm 1 that yields a shuffled ranking that respects R and at the same time is Pareto optimal, in the above setting. Namely, given a set X, a ranking function r, a metric μ, a sequence of sets of desired values of μ, D, as well as a deviation restriction R, we aim to find a ranking $r^*(X)$ such that $\mu(r^*(X))$ is close to D — where "close" is understood in the context of Pareto optimality — and at the same time r, r^* do not violate R. This is achieved by Algorithm 3, which is actually a backtracking search for r^*, with R serving as the backtracking condition.

Algorithm 3: Constrained diversity-aware shuffling.

```
1:  procedure SHUFFLE2(X, r, μ, D, R)
2:      rank ← r(X)
3:      μRank ← ∅
4:      i ← 0
5:      front ← SORTBY(X, μRank, μ, D_i)   ▷ Stack.
6:      while front ≠ ∅ do
7:          next ← front.pop()
8:          μRank.append(next)
9:          i ← i + 1
10:         if R(μRank, rank) is violated then
11:             μRank.pop()
12:             i ← i − 1                  ▷ Backtrack.
13:         else if X \ μRank = ∅ then
14:             return μRank               ▷ Found shuffling.
15:         else
16:             children ← SORTBY(X, μRank, μ, D_i)
17:             front.push(children.reverse())
18:         end if
19:     end while
20:     return False                       ▷ Failed to find shuffling.
21: end procedure
```

We shall now make some remarks regarding Algorithm 3. To begin with, observe that, when expanding our partial ranking, $\mu Rank$, we push all elements returned by SORTBY in *reverse* order so as to ensure that the most fitting item computed by SORTBY is examined first. Also, SHUFFLE2 is actually an extension of SHUFFLE since, in case we set R to be a restriction that may never be violated, then Algorithm 3 returns the same shuffling of X that Algorithm 1 would return.

As one may expect, we can prove that, indeed, $\mu Rank$ lies on the Pareto Frontier. Namely, we have the following:

Theorem 2. *Given a set* $X = \{x_1, \ldots, x_n\}$*, a ranking function* r*, a metric* μ *and a sequence of desired values of* μ*,* D*,* SHUFFLE2 *returns a ranking* $r^*(X)$ *such that* r^* *is Pareto optimal among all rankings* $r' \in \mathcal{R}(X)$ *that respect* $R(r', r)$.

Proof. We will proceed with reductio ad absurdum. Let a ranking r' such that all the above three conditions hold and, specifically, let i_0 be the *minimum* index such that:

$$\text{dist}(\mu(r'(X)_{i_0}), D_{i_0}) < \text{dist}(\mu(r^*(X)_{i_0}), D_{i_0}). \quad (20)$$

Since we also have that:

$$\text{dist}(\mu(r'(X)_i), D_i) \leq \text{dist}(\mu(r^*(X)_i), D_i) \quad (21)$$

for every $i = 1, \ldots, n$ and i_0 is the minimum index such that the inequality in (21) is strict, we have that for every $i = 1, \ldots, i_0 - 1$ it holds:

$$\text{dist}(\mu(r'(X)_i), D_i) = \text{dist}(\mu(r^*(X)_i), D_i) \quad (22)$$

Equation (22) means that calling SORTBY with $X, r^*(X)_{i_0-1}, \mu$ and D_{i_0} as arguments yields a list where all elements of $r^*(X)_{i_0-1} \cup r'(X)_{i_0-1}$ are ranked below the i_0-th element in $r'(X)$, let x'. Furthermore, Equation (20) ensures that x^* is ranked higher in the returned list than $x^* \neq x'$, where by x^* we denote the i_0-th element of $r^*(X)$. However, this means that (see Algorithm 3, line 7) x' is processed prior to x^*, so, since x' is not the i_0-th element of the returned ranking, $r^*(X)$, we infer that R is violated by $r'(X)_{i_0}$. which is a contradiction, since $R(r', r)$ is assumed to hold. So, r^* satisfies indeed all three conditions. □

So, SHUFLE2 computes, indeed, a shuffling of $r(X)$ that, given R, is optimal with respect to all other soft restrictions about μ.

4 EXPERIMENTS

In this section we shall present some experiments conducted on synthetic data with our main purpose being studying Algorithm 3 and its behavior with respect to its hyperparameters on the special case where μ is some diversity metric. Namely, we examine the scalability of Algorithm 3 as well as how different choices of D and R affect its behavior — either combined or separately. At first we present our data generation protocol and then we proceed with explaining which specifications we decided to make to our framework. At last, we present and discuss the results of our experiments.

4.1 Data Construction

In these experiments, we study our diversity aware ranking framework with two goals in mind: (i) to examine the effect the choice of our framework's hyperparameters has on its behavior and; (ii) to examine any differences the diversity metrics we have presented may have in practice. For the purposes of these experiments, we have generated two synthetic datasets, P_1 and P_2. Regarding P_1, it consists of N randomly generated samples of points in $[0,1]^2$. and each sample is generated under the following protocol:

1. At first, a set of M points in $[0,1]^2$, call it C, is generated — the points in C will serve as cluster centers.
2. Then, for each cluster center, c_k, in C, a sample of n_k points is generated, under a uniform distribution on the ℓ_1 sphere with center c_k and radius $r_k := \frac{1}{2}\min_{i \neq k} \ell_\infty(c_k, c_i)$. Also, we choose n_k randomly from a triangular distribution $Trig(n_{\min}, n_{\max})$ with mode $\delta = \frac{n_{\min}+n_{\max}}{2}$.

We chose for n_k to be a random variable since having clusters of fixed size would have direct implications for our dataset's diversity according to most metrics. Specifically, in our experiments, we set $M = 4$, $n_{\min} = 0$ and $n_{\max} = 10$. That is, P_1 contains 10 clusters with each cluster containing from 0 to 40 points. Now, given the above data, we assign to each sample S of D a randomly chosen target point $t(S) \in [0,1]^2$, which will serve as the target point of each ranking, as we shall present below.

Regarding P_2, it is significantly larger than P_1, since it consists of 17 sets, with each set containing $N = 10$ samples, as in P_1. However, while in P_1 we did not monitor the total number of points in each sample — i.e., each sample in P_1 may contain from 0 to 400 points — we did so in all samples in P_2. Namely, every set of samples in P_2 contains samples with exactly K elements, where $K = 15, 20, \ldots, 95$. Then, given K we determine how many elements each of the five clusters of a sample contains by randomly selecting a partition of K into five non-negative integers.

4.2 Empirical Setting

Our shuffling framework, as described above is quite abstract, so, we needed to make some specifications in order to implement and test it. To begin with, we chose our deviation restriction, R, to be a simple expression of the form $d(r^*, r) \leq d_{\max}$, where d is a metric defined on permutations (rankings) and $d_{\max}$ is some fixed threshold provided to our algorithm. Namely, we chose the ℓ_1 metric, which is defined as follows:

$$\ell_1(r_1, r_2) := \sum_{k=1}^{n} |r_1(k) - r_2(k)|. \quad (23)$$

So, for instance, for $n = 5$, let two rankings of $X = \{1,2,3,4,5\}$ be $r_1 = (3,5,1,4,2)$ and $r_2 = (3,5,2,4,1)$. Then, have $\ell_1(r_1, r_2) = 2$ while, if $r_3 =$

$(4,1,3,2,5)$, we have $\ell_1(r_1,r_3) = 12$. We preferred ℓ_1 over other permutation metrics, mostly due to its simplicity and intuitive interpretation in the context of ranking — i.e., the total "displacement" of each entity.

At this point we should mention that across all experiments we chose to use a normalized version of the ℓ_1 metric. Namely, for given n, ℓ_1 in S_n attains a maximum value of $\left\lceil \frac{n^2-1}{2} \right\rceil$, so the normalized ℓ_1 distance, denoted by $\hat{\ell}_1$, is defined follows:

$$\hat{\ell}_1(r_1,r_2) := \frac{1}{\left\lceil \frac{n^2-1}{2} \right\rceil} \ell_1(r_1,r_2). \tag{24}$$

Regarding the metric μ used, we used the diversity metrics shown in Table 1. We did not include α and β diversities in our study since our datasets did not have a structure that would allow us to do so. We also chose to exclude Hill numbers since in the most used cases in bibliography, q is set to either 0, or 1 or 2, (Chao et al., 2010), which in each case reduces to a function[1] of some of the metrics presented in Table 1.

Table 1: Diversity metrics used in A Series Experiments.

Method	Tag
Richness	R
Berger-Parker Index	BP
Simpson's Index	λ
Shannon's Index	H

Another parameter that needed to be specified was D, for which we considered two different scenarios: (i) each set of D to be a singleton containing the same value and; (ii) each set of D to be a singleton containing a randomly chosen value, with all choices being pairwise independent.

Lastly, we chose a simple ranking function r across all experiments, which ranked all points in a sample by ascending distance to each sample's target point. As our distance in this case we preferred the usual Euclidean Distance.

Given the above specifications, we performed the following sets of experiments:

1. Set A consisted of experiments using dataset P_1 where we allowed for a single value for D for each sample win $[0,1]$ with a 0.05 step while we allowed for $d_{\max}$ to similarly vary from 0 to 1.
2. Set B consisted of experiments using dataset P_1 where D took random sequences as values and $d_{\max}$ took values from 0 to 1 with a 0.05 step.
3. Set C consisted of experiments using dataset P_2 where D took random sequences as values and $d_{\max}$ took values from 0 to 1 with a 0.05 step.

Experiments of Set A where used to study the behavior of Algorithm 3 in a general setting and verify that its behavior is the expected one, as described by Theorem 2. Regarding experiments in Set B, we wanted again to test the behavior of Algorithm 3 but, this time, under a more general protocol of determining D, so as to further support the results found in Set A. Regarding Set C, we wanted to study properties of Algorithm 3 related to scalability as well as determine whether trends observed in smaller datasets — sets A and B — were also present at a larger scale.

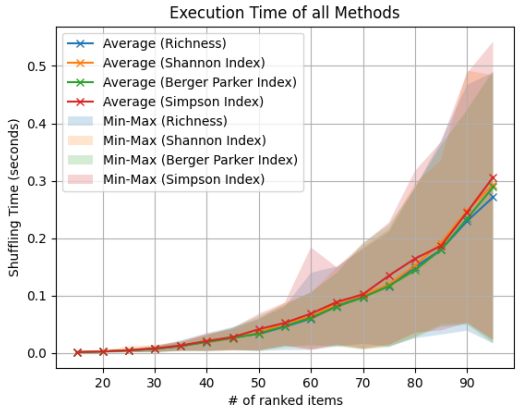
4.3 Results

We shall now present the results of our experiments. To begin with, in Figure 1a we observe that the choice of the diversity metric does not seem to affect the overall computation time since average computation times seem to be identical while minimum and maximum values also do not seem to differ significantly. This was more or less expected since computing the diversity of a set is not time-demanding when compared to SHUFFLE2's backtrack search while also all four diversity metrics have similar time complexities. As a result we choose to aggregate over all diversity quantification methods in the rest three plots presented in Figure 1.

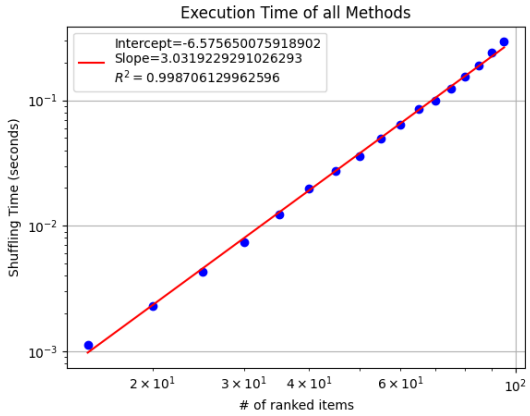
In Figure 1b we observe that the average execution time is quite well fitted by a straight line in log-log axes ($R^2 > 0.99$), which implies that execution time grows polynomially with respect to input size, n. Namely, as shown in Figure 1c, the average execution time of SHUFFLE2 in Set C was asymptotically cubic with respect to n. Since, the worst-case complexity of our backtrack searching algorithm is $\Theta(n!)$, having an implementation that has a significantly lower average time complexity is quite surprising and would require further investigation.

One fact that could contribute to this unexpected efficiency is that in all experiments we allowed for $d_{\max}$ to vary significantly, so, one may argue that the low observed average time complexity is due to e.g., lower execution time of searches for higher values of $d_{\max}$. However, as shown in Figure 1d, this is not true since, while there is some expected variation of the execution time of SHUFFLE2 with respect to $d_{\max}$, it is not significant enough to account for a notable decrease of average execution time.

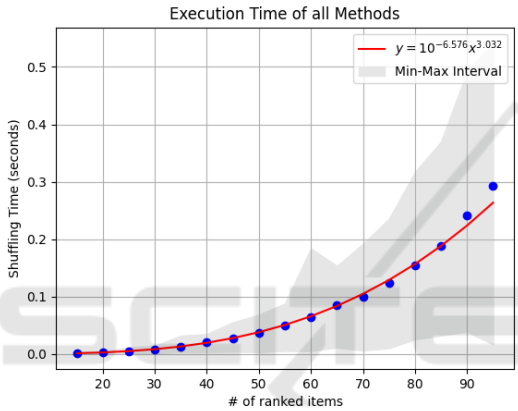
[1] Namely, for $q = 0$, ${}^0D$ coincides with richness, for $q = 1$, ${}^1D$ is just a function of Shannon's Index, namely ${}^1D = \exp(-H)$ while for $q = 2$, ${}^2D$ is the inverse of Simpson's Index (Chao et al., 2010).



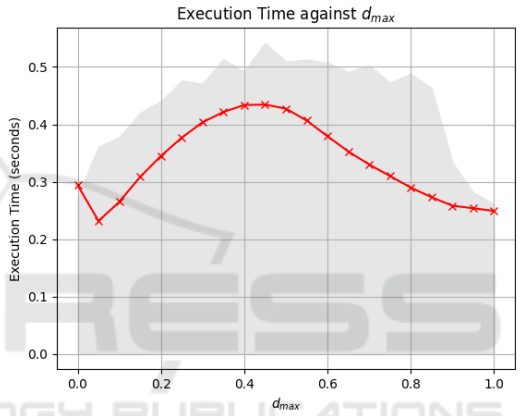
(a) Scalability of Shuffle2 with respect to all diversity metrics.



(b) Average execution time across all diversity metrics (Log-Log).



(c) Average execution time across all diversity metrics alongside the fitting polynomial, $y = 10^{-6.576}x^{3.032}$.



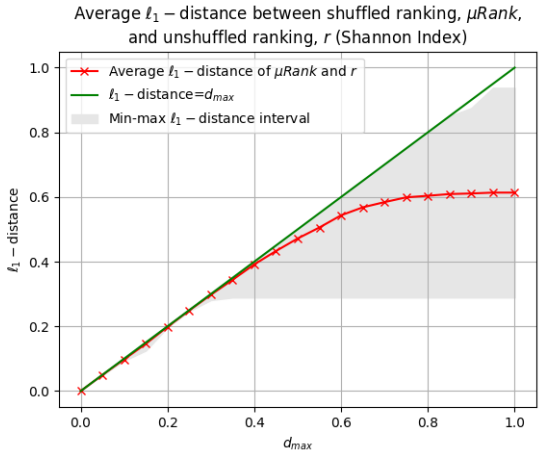
(d) Average execution time vs deviation threshold, $d_{\max}$.

Figure 1: Scalability results regarding SHUFFLE2.

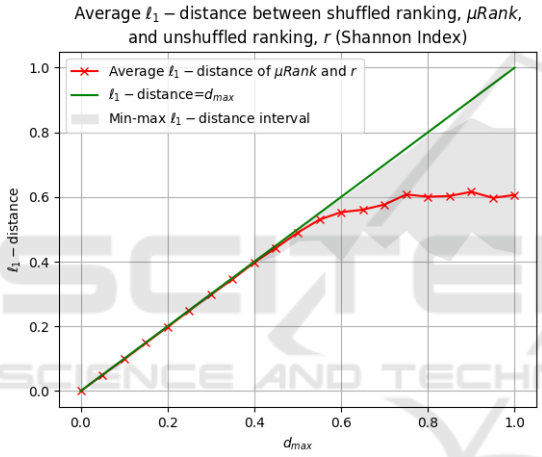
Another interesting feature observed in Figure 1d is an initial spike for $d_{\max} = 0$. We can provide two reasons for this spike — possibly among others. At first, $d_{\max} = 0$ means that we have to look for the initial ranking r throughout the entire search space, so even in this case we have to explore some possibly significant part of the search tree. On the other hand, in each experiment conducted $d_{\max} = 0$ was the first case examined after the dataset was loaded, so this initial spike might be a result of caching the dataset in memory for the rest values of $d_{\max}$, as well.

Next, we examine how much the ranking returned by SHUFFLE2 deviates from the initial ranking, r — see Figure 2. Starting from Set A, we observe that using all four metrics, the algorithm initially returns a shuffled ranking r^* that seems to take as much distance allowed from r as possible, while after around $d_{\max} = 0.4$ in all four cases of Set A the distance between r and r^* converges to a value around 0.6 — Figure 2a demonstrates the case of Shannon Entropy, while using any of the rest metrics yielded similar results. Since SHUFFLE2 for $d_{\max} = 1$ is actually Algorithm 1, we also infer from the above results that an empirical estimation of the maximum average deviation from the initial ranking r using Algorithm 1 is slightly above 60% as quantified using ℓ_1 as our metric. Similarly, in the case of Set B — Figure 2b again demonstrates the case of Shannon Entropy, while using any of the rest metrics yielded similar results — we observe the very same behavior, but for the maximum and minimum values of $\ell_1(r^*, r)$ which are less diverse than those in Set A. This may be explained by the different way we have chosen D in each case, since in Set A the values of D were ordered and strictly determined — D was constant across all initial parts, $r^*(X)_i$ — while in Set B it was allowed to vary randomly with i.

At this point, observe that in both plots in Figure 2 the distance between r^* and r is virtually increasing even past the 0.4 limit we described above.



(a) ℓ_1 distance between r^* and r using Shannon's Index as diversity index (Set A).



(b) ℓ_1 distance between r^* and r using Shannon's Index as diversity index (Set B).

Figure 2: Results from experiment sets A and B regarding the deviation of the returned ranking, r^*, from the initial ranking, r, as measured using the ℓ_1 ranking metric.

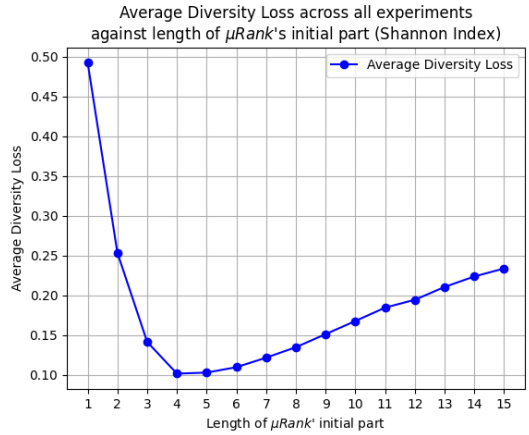
One may have expected that a virtually unconstrained search for the Pareto optimal shuffling would terminate in more or less the same time, however, as $d_{\max}$ increases, so does the part of the search graph that SHUFFLE2 is allowed to search, which can explain why execution time is increasing with $d_{\max}$, even if at a slower rate after some certain point.

Since in all experiments we defined μ to be a diversity metric, we will refer to D-loss as defined in Equation (14) as *diversity loss*. In Figure 3 we present, at first, the average diversity loss in the first set of experiments (Set A) — Figure 3a — where we observe a similar trend across all diversity metrics — the corresponding curves for the rest metrics were similar to the one shown in Figure 3a. Namely, initially L is quite high, then it approaches zero for some of the first initial parts of the returned ranking and, eventually, it grows again away from zero. This behavior is replicated at a larger scale in Set C — Figure 3b — where we allowed for rankings to contain each 95 elements — instead of 15 we present in Figure 3a. Again, average diversity loss drops for the first initial parts for each metric while then it rises to some higher value. The more vivid variation of average diversity loss we observe in the results of Set C is attributed mostly to D taking random values compared to Set A where D took a single value across each sample. We also produced plots for the rest subsets of Set C — namely, for rankings of length $l = 15, 20, \ldots, 95$ — but they all were more or less truncated versions of Figure 3b.

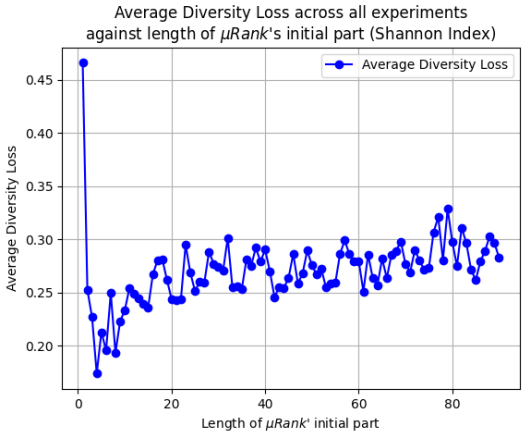
4.4 Diversity Loss

This trend observed in Figure 3 is of special interest so we shall further elaborate on it. As we have discussed in the description of our framework — see Section 3 for more details — adhering to the restrictions imposed by D is treated as a *soft* restriction throughout Algorithm 3. That is, we do not expect that the shuffled ranking returned by SHUFFLE2 will be *globally* optimal with respect to diversity loss, L, but, as we proved in Theorem 2, each shuffled ranking returned is Pareto optimal. However, as we observe in Figure 3, in all cases of diversity metrics we have a similar trend. Namely, L is initially relatively high, it then drops close to zero for the first initial parts of the returned ranking and then it rises again away from the desired levels of diversity determined by D for the last parts of the returned shuffled ranking.

The above behavior is expected, given the metrics we have at our disposal. To begin with, given a set, X, and a a diversity metric, μ, each ranking of X, let $r(X)$, has the same diversity as X since none of the metrics we used is sensitive to the *order* in which elements appear in X — this is a natural feature of a diversity metric since it quantifies the diversity of a population/sample at a given time/time interval. Also, as we include more of X's elements into our ranking we expect $\mu(r(X)_i)$ to be approximately $\text{div}(X)$ since, for large sets, introducing another element does not affect diversity dramatically — at least not when quantifying it using the metrics discussed in Section 2.2. As a result, we are not able to guarantee adherence to D_i as far as large values of i are concerned, which is clearly depicted in all figures in Figure 3. Also, the same applies to singletons, since the diversity of a singleton takes some trivially constant value under all the metrics presented in Section 2.2.



(a) Average diversity loss (L) using Shannon's Index as diversity metric (Set A).



(b) Average diversity loss (L) using Shannon's Index as diversity metric (Set C).

Figure 3: Average Diversity Loss (L) across experiments of sets A and C.

So, the only values of i for which we could reasonably expect that $\mu(r^*(X)_i) \approx D_i$ are those closer to bur larger than 1. As shown in Figure 3, this is indeed the case since, using any metric, we achieve a minimum diversity loss for such values of i. At this point we should also observe that the fact that each ranked sampled contains elements from (at most) four classes plays an important role. Namely, with less than four — or richness(X), in general — items the values diversity may take — with any metric but for richness in mind — are more restricted than when there are more — but not much more — items in the sample. Intuitively, this happens because, especially when richness(X) is not large, modifying the diversity of a sample boils down to adding a few items to it. On the contrary, when there are too few items in a sample, diversity values are more "sparsely" distributed in $[0,1]$ while when the cardinality of a sample is much larger than richness(X), adding a few items does not significantly affect its diversity. Bearing these in mind, it is expected that a sample is more flexible in manipulations of its diversity when it contains roughly as many items as the classes available. Less items lead to less flexibility as well as much more.

This trend is also reflected in the distance the ranking r^* returned by Algorithm 3 has from r, as shown in Figure 2. Namely, for tighter values of $d_{\max}$ we observe that r^* takes as much distance as possible in order to minimize diversity loss as much as possible. In these cases there is an expected trade-off between adherence to r and diversity, which is attributed mostly to the following two factors: (i) first, the ranking methodology we chose in all our experiments was promoting similarity in the first positions of the returned rankings while, at the same time it compromised diversity and; (ii) second, diversity is inherently opposite to similarity in the sense that a set that is considered diverse should contain elements that are dissimilar to each other above some certain threshold. Bearing in mind the above, we can easily explain the observed trade-off between diversity and adherence to the initial ranking we provide to SHUFFLE2.

To shed more light on the above behavior of SHUFFLE2 let us consider a setting where $X = \{a_1, a_2, a_3, b_1, b_2, b_3\}$, where a, b denote the classes of the corresponding elements and $r(X) = (a_1, a_2, a_3, b_1, b_2, b_3)$. Also, let us consider that we measure diversity using Richness and that our desired diversity values are $D = (1,1,1,0,0,0)$, i.e., we care about the first half of the list being highly diverse while we have no demands regarding the list's last elements. Then, it is not difficult to see that any ranking, $r^*(X)$, containing at least one a and at least one b would yield a diversity loss vector $L(r^*) = (1,0,0,0,0,0)$. However, a ranking such as $r^*(X) = (a_1, b_1, a_2, a_3, b_2, b_3)$ has an ℓ_1 distance from $r(X)$ equal to $\ell_1(r^*, r) = \frac{1+2+3}{17} \approx 0.353$. That is, in order to achieve a highly diverse shuffling, we had to deviate from our initial ranking at about 35%.

5 DISCUSSION

Our work raises certain issues and suggests possible directions for future work that we discuss below.

5.1 Determining D in Practice

Our developed framework assumes that D, the desired level of diversity, is externally provided, with each of

its components, associated to an initial part of the constructed ranking, being a set or interval of values. Although we have not fully explored this flexibility in our empirical setting, both of these features could be useful in practical applications, where one would attempt to actually determine what D should be.

A first approach to determining D could be to measure the diversity $\mu(X)$ of the set X of options that are available, and set each component of D to equal $\{\mu(X)\}$. This would effectively ask for some form of uniformity of the diversity exhibited in any initial part of a ranking presented to the user. Thus, the user would not only wish for the entire list of options presented to them to be diverse, but also for the first handful of results in that ranking — which practically is what a user gets to see when making their choice — to also be equally diverse.

Variations are, of course, possible. Instead of insisting on uniformity, the user could ask that each initial part of the ranking is at least as diverse as the entire list by setting each component of D to equal the interval $[\mu(X), 1]$, so that the shuffling algorithm will not seek to demote diversity of a part of the ranking if this happens to be higher than $\mu(X)$. Analogously, the user may wish to relax the minimum bound on the diversity and replace the interval $[\mu(X), 1]$ with, say, $[\mu(X)/2, 1]$, insisting that some diversity is minimally and uniformly required in each part of the ranking.

An interesting direction for future research would be to determine D using actual user feedback in real scenarios. Namely, it would be interesting to see to what extent and under what conditions users desire ranked lists presented to them to be diverse and, on top of that, what part of the list is of significant importance — for instance, in typical web search tasks, the first page of the returned results is usually of the highest significance to a user.

One approach to learning the diversity that a user may be seeking in their ranked options is to track the user's past choices from ranked lists. The precise choice from each specific ranking can, naturally, be used as a signal of how the user wishes the options to be ranked to begin with. But in addition, looking at the choices C^t of the user after t rounds of being presented with a ranking and making a choice, one could consider the diversity $\mu(C^t)$ of C^t as an indication of the level of diversity that the user would wish to see in subsequent rounds of rankings. Setting each component of D to equal $\mu(C^t)$, and presumably updating D over time as C^t is computed for larger and larger values of t would capture this intuition.

A more involved solution would be to maintain a list of user choices as above, but partitioned based on the position in the ranking of each choice made. So, if during round t a user chooses option at position i in their presented ranking, then that option would be added to list C^t_i. D would then be defined by setting its i-th component to equal $\mu(C^t_i)$. Further accounting for the amount of data within each list C^t_i of choices, one could insist that each component of D is some confidence interval around $\mu(C^t_i)$, whose size is determined by the size of C^t_i. This would then be able to capture situations where a user might signal that different levels of diversity are desired for different initial parts of the ranking; e.g., no diversity preference for the first handful of results (so that the original ranking is maintained), but then maximum diversity in the second handful of results (so that the user can explore alternatives that might be less preferred according to the original ranking, but are highly diverse).

5.2 Measuring Shuffling Deviation

A choice that we made for the purposes of our empirical evaluation was to implement the deviation restriction, R, required by Algorithm 3, through the use of ℓ_1 as a simple to compute yet informative way to quantify distance between two ranked lists. As we explained in Section 4.2, ℓ_1 is naturally interpreted as the total "displacement" required to transit from a ranking r_1 to another ranking r_2. However, what matters in most occasions a ranking is used, is the first part of the returned ranked list. So, ℓ_1 may not be the most appropriate choice since it does not make a distinction between changes in the first, middle or last places of a ranking r. A natural adaptation of ℓ_1 accommodating the above is to introduce weights, w_k, $\sum_{k=1}^{n} w_k = 1$, such that moving the first, say, element of $r(X)$ three places down the list matters more than moving the 15$^{\text{th}}$ element to position 18.

Other than distance-based deviation constraints that apply globally on a ranked list, it would also be meaningful to explore more complex restrictions R, including ones that are not necessarily based on distance metrics. For instance, in some scenarios a user may wish to explicitly state that the initial handful of options in the original ranking should not be shuffled and that subsequent options could be shuffled more the later in the ranking they appear.

5.3 Efficient Search of Shufflings

In the implementation of Algorithm 3, as used in the empirical evaluation presented in Section 4, we adopted a greedy breadth-first strategy to search the space of all shufflings of $r(X)$. The only heuristic we have implemented in order to alter the order nodes are expanded is, as mentioned in Section 3, that when

sorting any remaining items using SORTBY we break any ties that may occur using r, expecting that an item ordered at a higher position in $r(X)$ will lead to a shuffled ranking $r^*(X)$ with smaller ℓ_1 distance from r.

However, there is plenty of space for further improvement in this direction. To begin with, SHUFFLE2's performance may be enhanced by introducing Forward Checking. Since after each expansion of the generated ranking, *μRank*, the remaining children nodes are fewer than in the previous iteration, Forward Checking may lead to significant parts of the search tree to be pruned. Moreover, one may also deploy more sophisticated search techniques such as A^* or other heuristic or stochastic search methods in order to further reduce the running time of SHUFFLE2.

6 CONCLUSIONS

Motivated by the widespread use of ranking to facilitate the choices of individuals, and the real danger of ending up creating echo chambers with potentially grave implications in social settings, this work has proposed and empirically evaluated a post-hoc approach for the diversity-aware curation of rankings. Our next steps include the integration of our methodology in the context of the WeNet platform, and its evaluation on data coming from the platform's users.

ACKNOWLEDGMENTS

This work was supported by funding from the EU's Horizon 2020 Research and Innovation Programme under grant agreements no. 739578 and no. 823783, and from the Government of the Republic of Cyprus through the Deputy Ministry of Research, Innovation, and Digital Policy.

REFERENCES

Berger, W. H. and Parker, F. L. (1970). Diversity of Planktonic Foraminifera in Deep-Sea Sediments. *Science*, 168(3937):1345–1347.

Burges, C., Shaked, T., Renshaw, E., Lazier, A., Deeds, M., Hamilton, N., and Hullender, G. (2005). Learning to Rank Using Gradient Descent. In *Proceedings of the 22nd International Conference on Machine Learning*, ICML '05, page 89–96, New York, NY, USA. Association for Computing Machinery.

Cakir, F., He, K., Xia, X., Kulis, B., and Sclaroff, S. (2019). Deep Metric Learning to Rank. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1861–1870.

Chao, A., Chiu, C.-H., and Jost, L. (2010). Phylogenetic Diversity Measures Based on Hill Numbers. *Philosophical transactions of the Royal Society of London. Series B, Biological sciences*, 365:3599–609.

Colwell, R. (2009). *Biodiversity: Concepts, Patterns, and Measurement*, pages 257–263. Princeton Univ. Press.

Helm, P., Michael, L., and Schelenz, L. (2021). Diversity by Design: Balancing Protection and Inclusion in Social Networks.

Herfindahl, O. (1950). Concentration in the US Steel Industry. In *Unpublished doctoral dissertation*.

Hill, M. O. (1973). Diversity and Evenness: A Unifying Notation and Its Consequences. *Ecology*, 54(2):427–432.

Köppel, M., Segner, A., Wagener, M., Pensel, L., Karwath, A., and Kramer, S. (2019). Pairwise Learning to Rank by Neural Networks Revisited: Reconstruction, Theoretical Analysis and Practical Performance. *CoRR*, abs/1909.02768.

Oliveira, I. F. D., Ailon, N., and Davidov, O. (2018). A New and Flexible Approach to the Analysis of Paired Comparison Data. *Journal of Machine Learning Research*, 19(60):1–29.

Pei, C., Zhang, Y., Zhang, Y., Sun, F., Lin, X., Sun, H., Wu, J., Jiang, P., Ou, W., and Pei, D. (2019). Personalized Context-aware Re-ranking for E-commerce Recommender Systems. *CoRR*, abs/1904.06813.

Pfannschmidt, K., Gupta, P., and Hüllermeier, E. (2018). Deep Architectures for Learning Context-dependent Ranking Functions.

Shannon, C. E. (1948). A Mathematical Theory of Communication. *The Bell System Technical Journal*, 27(3):379–423.

Simpson, E. H. (1949). Measurement of Diversity. *Nature*, 163(4148):688–688.

Spellerberg, I. and Fedor, P. (2003). A Tribute to Claude Shannon (1916–2001) and a Plea for More Rigorous use of Species Richness, Species Diversity and the 'Shannon–Wiener' Index. *Global Ecology & Biogeography*, 12:177–179.

Tuomisto, H. (2010). A Consistent Terminology for Quantifying Species Diversity? Yes, It Does Exist. *Oecologia*, 164(4):853–860.

Whittaker, R. H. (1960). Vegetation of the Siskiyou Mountains, Oregon and California. *Ecological Monographs*, 30(3):279–338.