


# The Importance of Parameters in Ranking Functions

Christoph Standke ✉ 


RWTH Aachen University, Germany

Nikolaos Tziavelis ✉ 

University of California Santa Cruz, CA, USA

Wolfgang Gatterbauer ✉ 

Northeastern University, Boston, MA, USA

Benny Kimelfeld ✉ 

Technion, Haifa, Israel

RelationalAI Inc., Berkeley, CA, USA

---

## Abstract

How important is the weight of a given column in determining the ranking of tuples in a table? To address such an *explanation question about a ranking function*, we investigate the computation of SHAP scores for column weights, adopting a recent framework by Grohe et al. [ICDT'24]. The exact definition of this score depends on three key components: (1) the *ranking function* in use, (2) an *effect function* that quantifies the impact of using alternative weights on the ranking, and (3) an underlying *weight distribution*. We analyze the computational complexity of different instantiations of this framework for a range of fundamental ranking and effect functions, focusing on probabilistically independent finite distributions for individual columns.

For the ranking functions, we examine lexicographic orders and score-based orders defined by the summation, minimum, and maximum functions. For the effect functions, we consider *global*, *top-k*, and *local* perspectives: global measures quantify the divergence between the perturbed and original rankings, *top-k* measures inspect the change in the set of *top-k* answers, and local measures capture the impact on an individual tuple of interest. Although all cases admit an additive fully polynomial-time randomized approximation scheme (FPRAS), we establish the complexity of exact computation, identifying which cases are solvable in polynomial time and which are  $\#P$ -hard. We further show that all complexity results, lower bounds and upper bounds, extend to a related task of computing the Shapley value of whole columns (regardless of their weight).

**2012 ACM Subject Classification** Information systems → Retrieval models and ranking

**Keywords and phrases** Ranking, Explanation, Shapley value, SHAP scores

**Digital Object Identifier** 10.4230/LIPIcs.ICDT.2026.7

**Related Version** *Full Version*: <https://arxiv.org/abs/2601.06001> [33]

**Funding** *Christoph Standke*: ERC grant 101054974 (SymSim) and German Research Foundation grant GRK 2236 (UnRAVeL). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council. Neither the European Union nor the granting authority can be held responsible for them.

*Wolfgang Gatterbauer*: NSF Career Award IIS-1762268.

*Benny Kimelfeld*: German Research Foundation grant KI 2348/1-1.

**Acknowledgements** This work has been initiated in Dagstuhl Seminar 24032: *Representation, Provenance, and Explanations in Database Theory and Logic*.



© Christoph Standke, Nikolaos Tziavelis, Wolfgang Gatterbauer, and Benny Kimelfeld; licensed under Creative Commons License CC-BY 4.0

29th International Conference on Database Theory (ICDT 2026).

Editors: Balder ten Cate and Maurice Funk; Article No. 7; pp. 7:1–7:18

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

## 1 Introduction

A *ranking function* takes as input a set of records and produces a permutation over the set based on the entry values (commonly referred to as *features*) of the records. As ranking is abundant in decision making, it is natural to look for an *explanation* for the outcome of the ranking of a given dataset. Explanations for ranking functions have received significant attention in the area of Information Retrieval (IR) over recent years [24, 25, 32, 38] and before, as well described in surveys on “explainable IR” [2, 30]. Explanations for ranking can be largely categorized into two types: A *value-based* explanation aims to elucidate how the values of a certain record led to its being ranked in its position (or why it is included in, or excluded from, the top- $k$  answers); this is typically done by quantifying the contribution of every entry value to the status of the record [1, 9, 25]. On the other hand, a *function-based* explanation aims to investigate what aspect in the ranking function has led to the outcome, from the point of view of a specific record or the entire permutation; this is typically done by quantifying the contribution of function components to the outcome [14, 19, 32, 37]. This work belongs to the latter kind.

Typically, the ranking function involves *parameters*. For example, if the ranking is determined by a score function that is a linear combination of the attributes of the record, then the parameters are the *weights* attached to the features of the record. Previous work in the database community has investigated ways of selecting parameters so that the ranking function satisfies certain criteria [4, 8]. In this work, we study the contribution of the parameter choices to the outcome of the ranking. For that, we adopt the framework of Grohe et al. [18] for measuring the importance of parameters in the context of database queries, where the contribution of a parameter is determined by its SHAP score [23]. In contrast to previous work on the explanation of ranking functions, we focus on the computational complexity of calculating the contribution.

**SHAP scores and Shapley values.** The SHAP score [23] is an instantiation of the Shapley value that, in turn, is used to attribute a share to each player in a cooperative game, where each coalition (set of players) gains some utility [31]. The Shapley value, named after its inventor, is unique up to some axioms of rationality (e.g., the sum of shares adds up to the utility of the entire set) [28, 31] and, besides applications in a plethora of domains, has been used for explanations in data-centric fields such as Machine Learning [11, 23, 29], IR [9, 19], and databases [6, 10, 21]. The SHAP score was originally proposed for the purpose of explaining machine-learned models, and particularly attributing responsibility for the outcome of the model to the feature values of a particular instance [20, 23].

Specifically, SHAP is the Shapley value in the cooperative game where the feature values are the players, and the utility of a coalition  $C$  is the expected effect (model’s output) of the instance where the feature values of  $C$  are used and the rest chosen randomly. In the framework of Grohe et al. [18], the parameter values play the same role that feature values play in the application of SHAP to machine-learning explanations [20, 23] and that feature values play in the application of SHAP to value-based explanations of ranking [9, 26]. Moreover, the Shapley value has recently been used in function-based explanations [19].

Connecting to this work, consider a particular parameterized ranking function, along with a particular choice of values for its parameters. These define a ranking over the given a set of records. To apply the SHAP score, two additional components need to be determined. The first is a *distribution* from which parameter choices are assumed to be drawn. The second is an *effect function* that determines how different the ranking obtained from a

specific (random) choice of parameters is from the base ranking with the original choice of parameters. The effect function captures different interpretations of what Heuss et al. [19] refer to as “listwise feature attribution.” Similarly to the study of Grohe et al. [18], we restrict the discussion to simple distributions of parameters, namely finite and probabilistically independent parameters (i.e., “fully factorized” distributions [36]).

From the computational perspective, the SHAP score entails an expectation (w.r.t. a random choice of players) over an expectation (over the random parameter choices). Nevertheless, it was shown by Grohe et al. [18] that the SHAP score can also be viewed as a single expectation over an efficiently samplable space; therefore, the SHAP score of a parameter can be computed in polynomial time via sampling if we settle for a randomized additive approximation (FPRAS), under the mild assumption that the effect function is computable in polynomial time. This technique allows additive approximations also for more general classes of probability distributions, including fully factorized continuous distributions that allow efficient sampling. Moreover, if we make a *data complexity* assumption of a fixed number of parameters, then the SHAP score can be computed in polynomial time via an explicit enumeration of the probability space. We investigate the ability to compute the SHAP score exactly (and deterministically), for a given (non-fixed) number of parameters, in polynomial time. Van den Broeck et al. [36] have established that there are polynomial-time reductions in both directions between the computation of the SHAP score and the computation of the *expected effect*, which is arguably a simpler notion. Hence, following their result, we focus mainly on the expected effect throughout the paper.

**Studied ranking and effect functions.** Our study considers a variety of basic ranking functions and effect functions. For ranking, we consider the score functions of **Sum**, **Min**, and **Max**, and the lexicographic ordering (descending or ascending). In all of these functions, the parameters are coefficients (weights) over the record attributes. We view the dataset simply as a matrix (where every row corresponds to a record), and the parameters as per-column multiplicative weights. For the effects, we consider three types: *global* measures determine an effect over the entire ranking, *top- $k$*  measures determine the impact on the set of top- $k$  answers, and *local* measures determine the impact on a single record. The global measures we consider are standard distances between permutations [5, 27], and specifically Kendall’s tau, maximum displacement, and Hamming distances between the random ranking (due to the random choice of parameters) and the base ranking. The top- $k$  measures include the symmetric difference between the top- $k$  sets (for the base and random rankings) and the binary indicator of whether there is *any* difference between the top- $k$  sets. The local measures include the change in the record’s position and the change in its status of membership in the set of top- $k$  answers. A detailed example is presented at the end of Section 3.

**Contributions.** We begin with algorithms for a simpler task: given two records  $r_1$  and  $r_2$ , compute the probability (for randomly chosen parameters) that  $r_1$  precedes  $r_2$  in the ranking. This problem is tractable for all the ranking functions we consider, while for **Sum** we make the necessary assumption that the numbers are in unary representation (as the problem is  $\text{FP}^{\#P}$ -hard<sup>1</sup> for the binary representation). Our results (summarized in Table 1) show

---

<sup>1</sup> Recall that  $\text{FP}^{\#P}$  is the class of functions computable in polynomial time using an oracle to some function in  $\#P$ . A function  $F$  is  $\text{FP}^{\#P}$ -hard if there is a polynomial-time Turing reduction from every function in  $\text{FP}^{\#P}$  to  $F$ . Such a problem is at least as hard as every problem in the polynomial hierarchy [34].

## 7:4 The Importance of Parameters in Ranking Functions

■ **Table 1** Summary of the complexity results. The complexity for the Hamming distance is the same as MD (maximum displacement). The complexity results for the effects of the top- $k$  perspective are the same as the top- $k$  membership. All problems are solvable in polynomial time when the number of attributes is fixed. They are also solvable in polynomial time when the number of records is fixed, with the exception of **Sum** with binary numerical representation. All  $\text{FP}^{\#P}$ -hardness results are actually  $\text{FP}^{\#P}$ -completeness, due to standard techniques probability computation in  $\text{FP}^{\#P}$  [13, 17].

Effect	Global		Local		
	Kendall's $\tau$	MD	Position	Top- $k$ membership	
				Fixed $k$	Given $k$
Sum unary	P		P	$\text{FP}^{\#P}$ -h.	
Sum binary	$\text{FP}^{\#P}$ -h.		$\text{FP}^{\#P}$ -h.	$\text{FP}^{\#P}$ -h.	
Max asc / Min dsc	P	$\text{FP}^{\#P}$ -h.	P	$\text{FP}^{\#P}$ -h.	$\text{FP}^{\#P}$ -h.
Max dsc / Min asc	P		P	P	
Lexicographic	P		P	$\text{FP}^{\#P}$ -h.	

that, in general, this pairwise case suffices for the tractability of some of the effect functions, namely Kendall's tau and the position and top- $k$  membership of a row for small  $k$ ; for the rest, we prove hardness results. We also consider the case where the number of records is bounded, thus, we have a small number of competitors. There, we prove that all problems become tractable, again with the exception of **Sum** with binary numeric representation.

Finally, we consider another variation of a function-based explanation for ranking, now parameter-free. The goal is to compute the contribution of whole columns (attributes) to the ranking. More precisely, we consider the Shapley value of the cooperative game where the players are the columns, and the utility of a set of columns is the effect of the ranking obtained by considering only the columns in the set, while ignoring the rest. This task has been presented by Heuss et al. [19] where they refer to the action of ignoring a column as “masking the feature vector,” yet with no complexity analysis. We show that this problem reduces in polynomial time to the problem in the focus of this paper, namely computing the SHAP score of parameter choices. Hence, our algorithms for the SHAP score of parameters can be used for the Shapley value of columns. Moreover, we show that the hard cases of computing the SHAP scores are hard already for the Shapley value of columns.

**Organization.** We begin with preliminary definitions in the next section. In Section 3, we describe the formal framework and computational problems that we study, namely the SHAP score of parameters, the expected effect, and the Shapley value of columns. We give algorithms for the expectation in Section 4 and establish lower bounds in Section 5. Finally, we discuss extensions to the Shapley value of columns in Section 6, and conclude in Section 7. For space limitations, some of the proofs are given in the archive version of this paper [33].

## 2 Preliminaries

We begin with preliminary concepts and terminology that we use throughout the paper.

**Matrices and permutations.** For a natural number  $n$ , we denote by  $[n]$  the set  $\{1, \dots, n\}$ , and by  $\mathcal{S}_n$  the set of permutations over  $[n]$  (i.e., bijective functions  $\pi : [n] \rightarrow [n]$ ). By  $\pi_{\text{id}}$  we denote the *identity* permutation defined by  $\pi_{\text{id}}(i) = i$  for all  $i \in [n]$ . By  $\mathcal{S}$  we denote the set of all permutations, that is,  $\mathcal{S} := \cup_n \mathcal{S}_n$ . We denote by  $\mathcal{M}^{n \times m}$  the set of all  $n \times m$  matrices,

over the rational numbers, with  $n$  rows and  $m$  columns, and by  $\mathcal{M}$  the set of all matrices of all dimensions (i.e.,  $\mathcal{M} := \cup_{n,m} \mathcal{M}^{n \times m}$ ). If  $\mathbf{M} \in \mathcal{M}^{n \times m}$  and  $C \subseteq \{1, \dots, m\}$  represents a set of columns, then we denote by  $\mathbf{M}|_C$  the  $n \times |C|$  matrix obtained from  $\mathbf{M}$  by removing all columns except those in  $C$ . In other words,  $\mathbf{M}|_C$  is the projection of  $\mathbf{M}$  on  $C$  under bag semantics. We may refer to a row of a matrix as a *tuple*. We denote such a row (and every numeric vector)  $\mathbf{v}$  in boldface and its  $j$ th entry with  $\mathbf{v}[j]$ .

Computation-wise, we assume every number is represented as a pair  $(a, b)$ , standing for the rational number  $a/b$ , where  $a$  and  $b$  are integers in a binary representation. When we refer to a *unary* representation, we mean an integer  $a$  encoded as a string of length  $a$ .

**Ranking functions.** By a *ranking function* we refer to a function  $r : \mathcal{M} \rightarrow \mathcal{S}$  that maps every matrix  $\mathbf{M} \in \mathcal{M}^{n \times m}$  to a permutation in  $\mathcal{S}_n$ ; hence,  $r$  ranks the rows of its input matrix  $\mathbf{M}$ . Specifically, we will focus on several ranking functions  $r$ :

- Ranking by decreasing/increasing score of a row  $(a_1, \dots, a_m)$ , with the score being the sum of the  $a_j$ , denoted **Sum**, the maximum among the  $a_j$ , denoted **Max**, or the minimum among the  $a_j$ , denoted **Min**. For each scoring function  $s$ , we denote by  $r_s^{\text{dsc}}$  and  $r_s^{\text{asc}}$  the rankings by decreasing and increasing  $s$ , respectively. For example, we have  $r_s^{\text{dsc}}(i) < r_s^{\text{dsc}}(j)$ , meaning that the  $i$ th row  $\mathbf{t}_i$  precedes the  $j$ th row  $\mathbf{t}_j$ , whenever  $s(\mathbf{t}_i) > s(\mathbf{t}_j)$ .
- Ranking by the lexicographic ordering (left to right) over the rows, denoted  $r_{\text{Lex}}$ .

In our analysis, the direction of the ranking with respect to the score (increasing/decreasing) is important for ranking by **Min** and **Max**, as shown in Table 1. This direction is not important for ranking by **Sum** since we can switch between the directions by multiplying the entries by a negative number (or subtracting each value from the maximum value in the matrix); hence, we simply write  $r_{\text{Sum}}$  as standing for  $r_{\text{Sum}}^{\text{asc}}$  (since  $r_{\text{Sum}}^{\text{dsc}}$  is equivalent). In the lexicographic order, we assume for the same reason that the order of each column is ascending (i.e., lower numbers precede higher numbers). Also for the same, we will not consider ranking by **Min** explicitly since it is the same as ranking by **Max** of the negated matrix in reversed direction.

For the framework to be well-defined, we need to handle tie-breaking. For that, we will use the ordering by the row indices; that is, if the rows  $i$  and  $i'$  are tied by the ordering and  $i < i'$ , then we give precedence to the  $i$ th row.

**Shapley value.** A *cooperative game* is a pair  $(P, \nu)$  where  $P$  is a finite set of players and  $\nu : 2^P \rightarrow \mathbb{R}$  is a *utility function* that associates with every coalition  $C \subseteq P$  a value  $\nu(C)$ , so that  $\nu(\emptyset) = 0$ . The *Shapley value* of a player  $p \in P$  is defined by the following formula [28,31].

$$\text{Shapley}(p, \nu) = \sum_{C \subseteq P \setminus \{p\}} \frac{|C|!(|P| - |C| - 1)!}{|P|!} \cdot (\nu(C \cup \{p\}) - \nu(C)) \quad (1)$$

Intuitively, we consider the situation where we select players iteratively without replacement, starting with the empty set; the Shapley value of a player  $p \in P$  is the mean increase in utility when adding  $p$ .

**SHAP score.** Let  $f : D^m \rightarrow \mathbb{R}$  be an  $m$ -ary function over some domain  $D$ . Let  $\Pi$  be a discrete distribution over  $D^m$ , and let  $\mathbf{w} = (w_1, \dots, w_m)$  be a tuple from  $\Pi$  with a nonzero probability. For  $j = 1, \dots, m$ , the *SHAP score* of  $j$  with respect to (w.r.t.)  $\mathbf{w}$ , denoted  $\text{SHAP}(j, f, \mathbf{w})$ , is the value  $\text{Shapley}(j, \nu)$  for the cooperative game  $(P, \nu)$  defined as follows [23].

- $P := \{1, \dots, m\}$ ;
- $\nu(C) := \mathbb{E}_{\mathbf{u} \sim \Pi} [f(\mathbf{u}) \mid u_\ell = w_\ell \text{ for all } \ell \in C]$ .

That is, the utility of a subset  $C$  of parameters is the expectation of  $f$  over the distribution  $\Pi$ , conditioned on every parameter in  $C$  having its specific value from  $\mathbf{w}$ .

### 3 Framework and Computational Problems

Our goal is to compute the contribution of columns and column weights to the ranking of the rows of a given matrix. For that, we need to reason about how different the ranking would be had we eliminated certain columns or changed their weights. Therefore, we need to adopt a measure of difference between permutations. We refer to such measures as *effect functions*, as they determine the effect of the column alteration.

**Effect functions.** We are given a matrix  $\mathbf{M}$  and a ranking function  $r$  with *original order*  $\pi_0 := r(\mathbf{M})$ . As a result of applying the ranking function with different weights on the columns (or only on a subset of columns), we get a different permutation  $\pi$ . We will focus on three classes of effect functions:

- **Global perspective:** *How far is  $\pi$  from the original permutation?* We can use several notions of distance between permutations. We give here three conventional ones, where the first two were used for ranking explanation in the RankSHAP work [9].
  - Kendall’s tau:  $e_{k\tau}(\pi) := \sum_{1 \leq i < j \leq n} \mathbb{1}_{\pi_0(i) < \pi_0(j) \wedge \pi(i) > \pi(j)}$  determines the number of pairwise disagreements (swaps) between  $\pi$  and the original order. This effect function has the range  $[\frac{n(n-1)}{2}]$ .
  - Maximum Displacement (MD) distance:  $e_{\text{md}}(\pi) := \max_{1 \leq i \leq n} |\pi(i) - \pi_0(i)|$  is the maximum difference of an item’s position between the two permutations. This difference is a number in  $[n - 1]$ .
  - Hamming distance:  $e_{\text{Ham}}(\pi) := \sum_{1 \leq i \leq n} \mathbb{1}_{\pi(i) \neq \pi_0(i)}$  determines the number of positions where the tuple is different between the original and permuted orders. It can attain values in  $[n]$ .
- **Top- $k$  perspective:** *What is the impact on the set of top- $k$  answers?* For that, let  $T = \{\pi^{-1}(1), \dots, \pi^{-1}(k)\}$  and  $T_0 = \{\pi_0^{-1}(1), \dots, \pi_0^{-1}(k)\}$  we will consider two functions.
  - Top- $k$  difference:  $e_{\Delta}^k(\pi) := |T \cup T_0| - |T \cap T_0|$  is the size of the symmetric difference between the sets of top- $k$  elements.
  - Top- $k$  any-change:  $e_{\text{any}}^k(\pi) := \mathbb{1}_{T \neq T_0}$  determines whether there is *any* change in the top- $k$  elements.
- **Local perspective:** *What is the impact on a specific row  $i$ ?* We will consider two functions:
  - Position:  $e_{\text{pos}}(i, \pi) := \pi(i) - \pi_0(i)$  is the change of position of the  $i$ th row and is in the range  $\{-n + 1, \dots, n - 1\}$ .
  - Top- $k$  membership:  $e_{\text{top}}^k(i, \pi) := \mathbb{1}_{\pi(i) \leq k} - \mathbb{1}_{\pi_0(i) \leq k}$  determines how the  $i$ th row changes its membership in the top- $k$  tuples. Its range is either  $\{-1, 0\}$  or  $\{0, 1\}$ .

In the analysis we conduct in the remainder of this paper, we focus on four effects: Kendall’s tau, MD, position, and top- $k$  membership. The complexity results for the Hamming distance are the same as those of MD. The complexity results for the impacts of the top- $k$  perspective are the same as the top- $k$  membership. Details are in the archive version [33].

**Computing SHAP scores.** Next, we assume that the columns of the input matrix  $\mathbf{M}$  are weighted, and the goal is to compute the SHAP score of a given weight. To make it precise, we assume that our data is a pair  $(\mathbf{M}, \mathbf{w})$ , where  $\mathbf{M} \in \mathcal{M}^{n \times m}$  and  $\mathbf{w} = (w_1, \dots, w_m) \in \mathbb{Q}^m$  is a sequence of column weights. The pair  $(\mathbf{M}, \mathbf{w})$  represents the pair  $\mathbf{M} \circ \mathbf{w}$ , which is the matrix  $\mathbf{M}'$  obtained by multiplying the  $j$ th column by  $w_j$  for  $j = 1, \dots, m$ ; that is, we apply element-wise multiplication by  $\mathbf{w}$  to each row of  $\mathbf{M}$ , or, in other words,  $\mathbf{M}' = \text{diag}(\mathbf{w}) \cdot \mathbf{M}$  where  $\text{diag}(\mathbf{w})$  is the diagonal matrix whose diagonal elements are given by  $\mathbf{w}$ .

We will also assume that, in addition to  $\mathbf{M}$  and  $\mathbf{w}$ , we are given a finite probability distribution  $\Pi$  over  $\mathbb{Q}^m$ . We restrict our complexity study to column-wise independent distributions (i.e., *fully factorized* distributions [36]), hence,  $\Pi$  is represented by finite distributions  $\Pi_1, \dots, \Pi_m$  over  $\mathbb{Q}$  for each weight parameter, each distribution given as a list of value-probability-pairs; the total number of such pairs is denoted by  $|\Pi|$ .

Let  $r$  be a ranking function and  $e$  an effect function. Given the pair  $(\mathbf{M}, \mathbf{w})$  and a column number  $j$ , our goal is to compute  $\text{SHAP}(j, f, \mathbf{w})$ , where  $f(\mathbf{u}) := -e(r(\mathbf{M} \circ \mathbf{u}))$ , that is, the inverse of the effect on the ranking of replacing the weights of  $\mathbf{w}$  with those of  $\mathbf{u}$ . We denote this value by  $\text{SHAP}\langle r, e \rangle(j, \mathbf{M}, \mathbf{w}, \Pi)$ . The reason for the inverse (using minus) is that the closer we are to the original ranking order  $\pi_0$  (i.e., the less effect), the higher we deem the contribution to the the actual  $\pi_0$  in place.

<b>Problem:</b>	$\text{SHAP}\langle r, e \rangle$ : SHAP score computation
<b>Fixed:</b>	Ranking function $r$ and effect function $e$
<b>Input:</b>	Matrix $\mathbf{M}$ , weight vector $\mathbf{w}$ , distribution $\Pi$ , and column number $j$ of $\mathbf{M}$
<b>Goal:</b>	Compute $\text{SHAP}\langle r, e \rangle(j, \mathbf{M}, \mathbf{w}, \Pi)$

In this and the other problems we consider, when the effect function  $e$  is local, the input also includes the row number  $i$  of the effect. In addition, when  $e$  is the top- $k$  membership, then the input also includes  $k$  (unless we explicitly state that  $k$  is assumed to be fixed).

It has been established by Van den Broeck et al. [36] that, for finite and probabilistically independent parameters, calculating the SHAP score is computationally equivalent to calculating the expected value of  $f$ , which is arguably simpler to handle.

► **Theorem 1** ([36, Theorem 2]). *Let  $f$  be a function that can take any number of numerical arguments and can be computed in polynomial time. The following problems are polynomially (Turing) reducible to each other.*

1. Compute  $\text{SHAP}(j, f, \mathbf{w})$ , given  $\mathbf{w}$ ,  $j$  and  $\Pi$ .
2. Compute  $\mathbb{E}_{\mathbf{u} \sim \Pi}[f(\mathbf{u})]$ , given  $m$  and  $\Pi$ .

Hence, we will also study the following problem.

<b>Problem:</b>	$\text{EXP}\langle r, e \rangle$ : Expectation computation
<b>Fixed:</b>	Ranking function $r$ and effect function $e$
<b>Input:</b>	Matrix $\mathbf{M}$ and distribution $\Pi$
<b>Goal:</b>	Compute $\mathbb{E}_{\mathbf{u} \sim \Pi}[e(r(\mathbf{M} \circ \mathbf{u}))]$

**Computing Shapley values.** Let  $r$  be a ranking function and  $e$  an effect function. Given a matrix  $\mathbf{M} \in \mathcal{M}^{n \times m}$  and a column number  $j \in \{1, \dots, m\}$ , the Shapley value of the  $j$ th column is defined as  $\text{Shapley}(i, \nu)$  for the game  $(P, \nu)$  defined as follows:

- $P := \{1, \dots, m\}$ .
- $\nu(C) := -e(r(\mathbf{M}|_C))$ , that is, the inverse of the effect on the ranking we obtain by applying  $r$  to only the columns of  $C$  (with the same rationale for inverse as SHAP).<sup>2</sup>

We denote this value by  $\text{Shapley}\langle r, e \rangle(j, \mathbf{M})$ .

<sup>2</sup> The ranking function  $r$  applied to the empty set of columns always yields the matrix order  $\pi_{\text{id}}$  since all tuples of  $\mathbf{M}|_{\emptyset}$  are identical. In particular, this definition does not depend on the value of the scoring function on the empty set.

## 7:8 The Importance of Parameters in Ranking Functions

<b>Problem:</b>	Shapley $\langle r, e \rangle$ : Shapley value computation
<b>Fixed:</b>	Ranking function $r$ and effect function $e$
<b>Input:</b>	Matrix $\mathbf{M}$ and a column number $j$ of $\mathbf{M}$
<b>Goal:</b>	Compute Shapley $\langle r, e \rangle(j, \mathbf{M})$

**Approximation.** Before moving to exact algorithms, we want to emphasize that each of the three problems EXP $\langle r, e \rangle$ , SHAP $\langle r, e \rangle$ , and Shapley $\langle r, e \rangle$  can be efficiently additively approximated whenever  $r$  and  $e$  can be computed in polynomial time and the range of  $e$  is polynomial in the number of rows. This is the case for all ranking and effect functions introduced in this section. As we will show in Section 5.2, this is not always the case for multiplicative approximations. The additive approximation can be done via basic Monte-Carlo-sampling since all three problems can be expressed as the expectation of a random variable over a probability space that allows efficient sampling:  $\Pi$  for EXP $\langle r, e \rangle$ ,  $\mathcal{S}_m$  for Shapley $\langle r, e \rangle$ , and  $\Pi \times \mathcal{S}_m$  for SHAP $\langle r, e \rangle$ . We can use this insight for efficient additive approximations also in the case of not necessarily independent discrete parameter distributions under very mild assumptions (see [18]) or independent continuous parameter distributions that can be sampled efficiently.<sup>3</sup> This simple idea was further improved to allow for faster additive approximations (e.g. [7]). Hence, our framework could be applied in practice, even in cases where we show that exact computation is hard.

**A detailed example.** Before we start our complexity analysis, let us first look at a small example that demonstrates the framework in detail. Consider the following matrix  $\mathbf{M}$ :

id	$a_1$	$a_2$
1	20	26
2	30	13
3	40	0
4	0	39

Assume that we rank the row by Sum in descending order with a scoring parameter  $\mathbf{u} \in \{1, 2\} \times \{1, 2\}$ . We want to measure the impact of choosing the weight vector  $\mathbf{w} = (1, 1)$  among the possible choices for  $\mathbf{u}$  (where we assume a uniform distribution).

To measure this, we first determine the rankings for each  $\mathbf{u}$ : For  $\mathbf{u} = \mathbf{w} = (1, 1)$ , the scores are (46, 43, 40, 39), so we obtain the ranking (1, 2, 3, 4). In the same way, we obtain (4, 1, 2, 3) for  $\mathbf{u} = (1, 2)$ , (3, 2, 1, 4) for  $\mathbf{u} = (2, 1)$ , and again (1, 2, 3, 4) for  $\mathbf{u} = (2, 2)$ .

$\mathbf{u}$	scores	permutation $\pi$	$e_{k\tau} = d_{k\tau}(\pi, \pi_0)$	$e_{\text{pos}}(4, \pi)$
$\mathbf{w} = (1, 1)$	(46, 43, 40, 39)	$\pi_0 = (1, 2, 3, 4)$	0	4
(1, 2)	(72, 56, 40, 78)	(4, 1, 2, 3)	3	1
(2, 1)	(66, 73, 80, 39)	(3, 2, 1, 4)	3	4
(2, 2)	(92, 86, 80, 78)	(1, 2, 3, 4)	0	4

Now, we need to choose an effect function  $e$  that measures the distance between two rankings. We will look at the Kendall's tau distance  $e_{k\tau}$  and the position of the fourth row  $e_{\text{pos}}(4, \pi)$ . We observe that  $d_{k\tau}((1, 2, 3, 4), (4, 1, 2, 3)) = d_{k\tau}((1, 2, 3, 4), (3, 2, 1, 4)) = 3$  and that the row 4 is in the last position of all the rankings except for (4, 1, 2, 3) where it is first.

<sup>3</sup> For continuous distributions, the definition of the SHAP score via conditional representations is not well-defined since the set we condition on has measure 0. In the case of independent distributions, we can derive a compatible definition via disintegration, that is, fixing the parameters in  $C$  to their reference values and choosing the others at random.

For the SHAP score calculation, the valuation of a set of columns  $C$  is the expected value of the negated effect function when we fix the parameter value for the columns in  $C$  to the value of  $\mathbf{w}$  and choose the rest at random. For example, for  $C = \{2\}$ , we obtain for  $e = e_{k\tau}$

$$\begin{aligned} \nu_{k\tau}(\{2\}) &= -\frac{e_{k\tau}(r_{\text{Sum}}(\mathbf{M} \circ (1, 1))) + e_{k\tau}(r_{\text{Sum}}(\mathbf{M} \circ (2, 1)))}{2} \\ &= -\frac{d_{k\tau}((1, 2, 3, 4), (1, 2, 3, 4)) + d_{k\tau}((1, 2, 3, 4), (3, 2, 1, 4))}{2} = -\frac{3}{2}, \end{aligned}$$

and for  $e = e_{\text{pos}}(4, \pi)$ , we obtain  $\nu_{\text{pos}(4)}(\{2\}) = -\frac{(4-4)+(4-4)}{2} = 0$ . Calculating the valuation  $\nu$  of every  $C \subseteq \{1, 2\}$  gives the following. For  $e = e_{k\tau}$ , we obtain  $\text{SHAP}(1) = \frac{1}{2}(\nu_{k\tau}(\{1, 2\}) - \nu_{k\tau}(\{2\})) + \frac{1}{2}(\nu_{k\tau}(\{2\}) - \nu_{k\tau}(\emptyset)) = \frac{1}{2}(0 - (-\frac{3}{2})) + \frac{1}{2}(-\frac{3}{2} - (-\frac{3}{2})) = \frac{3}{4}$  and  $\text{SHAP}(2) = \frac{3}{4}$  as well, so both parameter choices have the same impact w.r.t. global changes in the ranking measured by  $d_{k\tau}$ . This reflects the symmetry in the 4th column of table above. In contrast, for  $e = e_{\text{pos}}(4, \pi)$ , we obtain  $\text{SHAP}(1) = \frac{3}{8}$  and  $\text{SHAP}(2) = -\frac{9}{8}$ . This matches our intuition that fixing the second weight to 1 is bad for the ranking of row 4, since the only good weighting vector for row 4 is  $\mathbf{u} = (1, 2)$ .

In the remainder of the paper, we study the complexity of exactly solving the computational problems we defined in this section.

## 4 Exact Algorithms for Expectation (and SHAP Scores)

In this section, we show the tractability results of Table 1 for  $\text{EXP}\langle r, e \rangle$  – computing the expected effect. Combined with Theorem 1, these give the tractability results of the table for the SHAP score. We begin a more basic problem that we later use for expectation.

### 4.1 Algorithms for the Pairwise Case

We begin with the following problem: *Given tuples  $x$  and  $y$ , what is the probability that  $x$  precedes  $y$  in the ranking?* Since we only care about their relative order, we can ignore all the other tuples in the table and assume  $n = 2$ . Although this pairwise case may seem highly restricted, we will see in the next section that it forms the foundation of all our algorithms.

<b>Problem:</b>	$\text{PREC}\langle r \rangle$ : Precedence probability computation
<b>Fixed:</b>	Ranking function $r$
<b>Input:</b>	Matrix $\mathbf{M}$ with two rows, distribution $\Pi$
<b>Goal:</b>	Compute $\mathbb{P}_{\mathbf{u} \sim \Pi}[s(1) < s(2)]$ where $s$ stands for $r(\mathbf{M} \circ \mathbf{u})$

A naive way to compute the probability of precedence is to iterate through the exponentially many choices of weights  $\mathbf{u}$  and check whether tuple precedence holds in the resulting order. We exploit the fact that many of these choices can be grouped together, allowing us to consider only a polynomial number of possibilities. This can be achieved for all ranking functions discussed in this paper, except for Sum where our algorithm is pseudo-polynomial, that is, polynomial in the magnitude of the given values.

► **Theorem 2.** *The problems  $\text{PREC}\langle r_{\text{Max}}^{\text{asc}} \rangle$ ,  $\text{PREC}\langle r_{\text{Max}}^{\text{dsc}} \rangle$ , and  $\text{PREC}\langle r_{\text{Lex}} \rangle$  are solvable in polynomial time. The problem  $\text{PREC}\langle r_{\text{Sum}} \rangle$  is solvable in polynomial time if the matrix  $\mathbf{M}$  and the weights in  $\Pi$  are integers encoded in unary.*

We now give the algorithm for each ranking function in Theorem 2. For readability, we use  $\mathbf{x}$  and  $\mathbf{y}$  for the two tuples  $\mathbf{t}_1$  and  $\mathbf{t}_2$  in the matrix. Our goal is to compute the probability that  $\mathbf{x}$  precedes  $\mathbf{y}$ ; without loss of generality, we assume that this is false in the event of a tie.

## 7:10 The Importance of Parameters in Ranking Functions

**Sum ranking.** For Sum, our algorithm relies on dynamic programming and follows a similar approach to the standard pseudo-polynomial algorithm for the counting knapsack problem [16]. Specifically, we consider a subproblem where the matrix  $\mathbf{M}$  is restricted to columns  $j, \dots, m$  and compute the probability that tuple  $\mathbf{y}$  surpasses tuple  $\mathbf{x}$  by (strictly) more than  $s$  in total score. This can be expressed recursively by considering how the score difference changes when removing the  $j$ -th column for each assignment to the weight  $u_j$ :

$$R[j, s] = \sum_{v_j \in \text{supp}(\Pi_j)} \mathbb{P}[u_j = v_j] \cdot R[j+1, s - v_j(\mathbf{y}[j] - \mathbf{x}[j])] \quad (2)$$

$$R[m+1, s] = 1 \text{ if } s > 0, \text{ and } R[m+1, s] = 0 \text{ if } s \leq 0 \quad (3)$$

The final answer is  $R[1, 0]$ . The running time is  $\mathcal{O}(|\Pi| \cdot \|s\|)$  where  $\|s\|$  is the number of possible values for  $s$  in our dynamic program. If the input consists of integers given in unary, then  $\|s\|$  is polynomial in the input size. Indeed, suppose the matrix  $\mathbf{M}$  contains integers in the range  $[-c, c]$  and the weights in  $\Pi$  are in the range  $[-b, b]$ . Then  $\|s\|$  will be bounded by  $\sum_{j=1}^m b \cdot 2c$ , which is in  $\mathcal{O}(mbc)$ .

**Max ranking.** For the Max ranking function, the key idea is that the “winner” in the comparison between the two tuples is determined by a single column – the one with the maximum value. Thus, we can break the problem into distinct cases, each corresponding to a specific column determining the outcome.

► **Example 3.** Let  $\mathbf{x} = (3, 5, 2)$  and  $\mathbf{y} = (4, 1, 6)$  be two tuples, and suppose that the weights  $\mathbf{u}$  are drawn uniformly from  $\{0, 1\}$ , and the ranking function is  $r_{\text{Max}}^{\text{asc}}$ . For  $\mathbf{x}$  to be ranked before  $\mathbf{y}$ , we need  $\max(u_1 \cdot 3, u_2 \cdot 5, u_3 \cdot 2) < \max(u_1 \cdot 4, u_2 \cdot 1, u_3 \cdot 6)$ . We consider three distinct cases where this event occurs: (Case 1) The first  $\mathbf{y}$ -coordinate  $u_1 \cdot 3$  is the overall maximum. We iterate over possible values of  $u_1$ , and for each one, we determine the valid weights for  $u_2$  and  $u_3$  individually. If  $u_1 = 0$ , then  $u_1 \cdot 3$  cannot be the winner. If  $u_1 = 1$ , then the valid sets of weights are:  $u_1 \in \{1\}$ ,  $u_2 \in \{0\}$ ,  $u_3 \in \{0\}$ . This occurs with probability  $1/8$ . (Case 2) The second  $\mathbf{y}$ -coordinate  $u_2 \cdot 1$  is the overall maximum. This cannot happen. (Case 3) The third  $\mathbf{y}$ -coordinate  $u_3 \cdot 6$  is the overall maximum. Here, we obtain the sets  $u_1 \in \{0, 1\}$ ,  $u_2 \in \{0, 1\}$ ,  $u_3 \in \{1\}$ . All combinations of these weights are valid, thus the probability is  $4/8$ . Summing up the probabilities from the valid cases, we obtain the final result  $5/8$ . ◻

Algorithm 1 builds on the logic of our example, decomposing the comparison of two maximum predicates into  $m$  distinct cases, and computing their probabilities efficiently. Some care is required in the event of ties among columns of  $\mathbf{y}$ ; in such cases, we elect the first occurring maximum to be the winner, ensuring that the  $m$  cases remain distinct. The decomposition can be expressed as follows:

$$(\max(\mathbf{y} \circ \mathbf{u}) > \max(\mathbf{x} \circ \mathbf{u})) \equiv \bigvee_{j \in [m]} \bigvee_{v_j \in \text{supp}(\Pi_j)} W_{j, v_j} \quad (4)$$

$$W_{j, v_j} = (v_j \cdot \mathbf{y}[j] > v_j \cdot \mathbf{x}[j]) \wedge \left( \bigwedge_{k \notin [j]} v_j \cdot \mathbf{y}[j] > u_k \cdot \mathbf{x}[k] \right) \wedge \quad (5)$$

$$\left( \bigwedge_{k \in [j-1]} v_j \cdot \mathbf{y}[j] > u_k \cdot \mathbf{y}[k] \right) \wedge \left( \bigwedge_{k \in [j+1, m]} v_j \cdot \mathbf{y}[j] \geq u_k \cdot \mathbf{y}[k] \right)$$

The algorithm constructs these distinct events  $W_{j, v_j}$  and for each one, it determines sets of valid weights for  $\mathbf{u}$ , except for position  $j$  where the weight has been fixed to  $v_j$ . Finally, the probability of these events is computed efficiently, since the distributions  $\Pi_j$  are independent. The running time of the algorithm is  $\mathcal{O}(|\Pi|^2)$ .

■ **Algorithm 1** Precedence computation for ranking by Max ascending.

---

```

1 Input: tuples  $\mathbf{x}$  and  $\mathbf{y}$ , distribution  $\Pi$  //Assuming  $\mathbf{y}$  is before  $\mathbf{x}$  in the tie-breaking scheme
2 Output: probability that  $\mathbf{x}$  appears before  $\mathbf{y}$  under  $r_{\text{Max}}^{\text{asc}}$  ranking
3 result := 0
4 for column  $j$  in  $[m]$  do
5   for weight  $v_j \in \text{supp}(\Pi_j)$  do
6     //Find cases where  $v_j \cdot \mathbf{y}[j]$  is the winner
7      $U := \{\emptyset, \dots, \emptyset\}$  //Data structure with one list of satisfying weights per column
8     for column  $k$  in  $[m]$  do
9       //Build the list of weights for column  $k$ 
10      if  $k = j$  and  $v_j \cdot \mathbf{y}[j] > v_j \cdot \mathbf{x}[j]$  then  $U[j] := \{v_j\}$ 
11      if  $k \neq j$  then
12         $U_1 := \{v_k \in \text{supp}(\Pi_k) \mid v_j \cdot \mathbf{y}[j] > v_k \cdot \mathbf{x}[k]\}$  //Strictly beat  $\mathbf{x}$  on column  $k$ 
13        if  $k < j$  then
14           $U_2 := \{v_k \in \text{supp}(\Pi_k) \mid v_j \cdot \mathbf{y}[j] > v_k \cdot \mathbf{y}[k]\}$  //Strictly beat  $\mathbf{y}$  on column  $k$ 
15        else
16           $U_2 := \{v_k \in \text{supp}(\Pi_k) \mid v_j \cdot \mathbf{y}[j] \geq v_k \cdot \mathbf{y}[k]\}$  //Beat  $\mathbf{y}$  on column  $k$ 
17         $U[j] := U_1 \cap U_2$ 
18      result +=  $\prod_{k \in [m]} \sum_{v_k \in U[k]} \mathbb{P}[u_k = v_k]$ 
19 return result

```

---

**Lexicographic ranking.** We now move on to ranking by lexicographic orders. Our algorithm again follows a decomposition into non-overlapping events, each corresponding to a specific column and its weight determining the outcome. For this to happen, all preceding columns need to be tied between the two tuples. The decomposition works as follows:

$$((\mathbf{y} \circ \mathbf{u}) >_{\text{Lex}} (\mathbf{x} \circ \mathbf{u})) \equiv \bigvee_{j \in [m]} \bigvee_{v_j \in \text{supp}(\Pi_j)} W'_{j,v_j} \quad (6)$$

$$W'_{j,v_j} = (v_j \cdot \mathbf{y}[j] > v_j \cdot \mathbf{x}[j]) \wedge \left( \bigwedge_{k \in [j-1]} v_j \cdot \mathbf{y}[j] = u_k \cdot \mathbf{x}[k] \right) \quad (7)$$

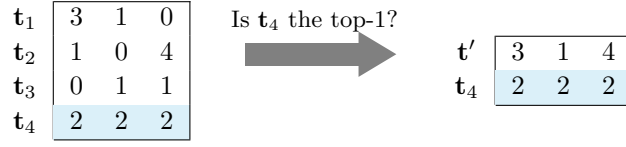
Similarly to the case of Max, we construct these events, determine the valid weight assignments  $\mathbf{u}$  for each one, and calculate their probability. The running time is  $\mathcal{O}(|\Pi|^2)$ .

**Recovering the satisfying weights.** In some cases, we need not only the probability that tuple  $\mathbf{x}$  is before tuple  $\mathbf{y}$ , but also a succinct representation of the weights for which this is true. Fortunately, our decomposition-based algorithms for Max and Lex provide this directly. We simply need to replace summation with union and multiplication with set product in line 18 of Algorithm 1.

► **Observation 4.** *The algorithms for  $\text{PREC}\langle r_{\text{Max}}^{\text{asc}} \rangle$ ,  $\text{PREC}\langle r_{\text{Max}}^{\text{dsc}} \rangle$ , and  $\text{PREC}\langle r_{\text{Lex}} \rangle$  can additionally return the weights  $\mathbf{u}$  for which precedence holds as a disjoint list, where each element consists of a set of valid weights per column, i.e.,  $\times_{j \in m} \bigcup (u_j = v_j)$ .*

## 4.2 From Pairwise Impact to Ranking Impact

We now extend the polynomial-time algorithms for the pairwise case to the expected effect on the overall ranking. When the effect function measures the position of a tuple or Kendall's tau for the entire permutation, we can reduce the problem to a polynomial number of calls to  $\text{PREC}\langle r \rangle$ . For both measures, we can write the expected effect as the expectation of a sum of indicator variables for tuple precedence and then use the linearity of expectation.



■ **Figure 1** Example 6: The transformation to the pairwise case for  $r_{\text{Max}}^{\text{dsc}}$  and top- $k$  membership.

► **Theorem 5.** For ranking function  $r \in \{r_{\text{Max}}^{\text{asc}}, r_{\text{Max}}^{\text{dsc}}, r_{\text{Lex}}\}$  and effect function  $e \in \{e_{k\tau}, e_{\text{pos}}\}$ , the problem  $\text{EXP}\langle r, e \rangle$  is solvable in polynomial time. For ranking function  $r = r_{\text{Sum}}$  and effect function  $e \in \{e_{k\tau}, e_{\text{pos}}\}$ , the problem  $\text{EXP}\langle r, e \rangle$  is solvable in polynomial time if the matrix  $\mathbf{M}$  and the weights in  $\Pi$  are integers encoded in unary.

The final case that admits a polynomial-time algorithm is the expected top- $k$  membership for fixed  $k$  and ranking by  $r_{\text{Max}}^{\text{dsc}}$ . Interestingly, our approach only applies to a descending order, and in fact, we prove in the next section that the problem is hard when the order is ascending. The key idea of the algorithm is that, for this specific ranking function, we can compute the probability that the tuple of interest beats a small subset of other tuples.

► **Example 6.** Suppose the matrix  $\mathbf{M}$  contains tuples  $\mathbf{t}_1, \mathbf{t}_2, \mathbf{t}_3, \mathbf{t}_4$  as shown in Figure 1, the ranking function is  $r_{\text{Max}}^{\text{dsc}}$ , and our goal is to compute the probability that  $\mathbf{t}_4$  is ranked top-1. We can merge the competitor tuples  $\mathbf{t}_1, \mathbf{t}_2, \mathbf{t}_3$  into a single competitor  $\mathbf{t}'$  that retains the largest value per column. Now,  $\mathbf{t}_4$  is top-1 precisely when it precedes  $\mathbf{t}'$  in the ranking. This merging has effectively reduced the problem to the pairwise problem  $\text{PREC}\langle r_{\text{Max}}^{\text{dsc}} \rangle$ , which we have already established can be solved efficiently.  $\lrcorner$

The example illustrates how the algorithm works when  $k = 1$ . For  $k > 1$ , we apply the principle of inclusion-exclusion to express the event that our tuple is in the top- $k$  as the intersection of events where it ranks above specific subsets of other tuples.

► **Theorem 7.** The problem  $\text{EXP}\langle r_{\text{Max}}^{\text{dsc}}, e_{\text{top}}^k \rangle$  with  $k$  as a fixed parameter is solvable in polynomial time.

Later, we will show that the assumption of a fixed  $k$  is necessary since the problem becomes intractable otherwise (Theorem 12).

### 4.3 Bounding the Matrix Dimensions

We conclude this section by investigating the consequences of bounding one of the two dimensions of our matrix by a fixed constant. This allows us to obtain a clearer picture of the parameters that make the problem hard. We find that such a restriction makes the problem significantly easier and, in most cases (with the exception of Sum with arbitrary values), both dimensions need to be non-fixed for the problem to be hard.

► **Theorem 8.** The following hold.

1. If the number  $m$  of columns of the matrix  $\mathbf{M}$  is bounded by a constant, then the problem  $\text{EXP}\langle r, e \rangle$  is solvable in polynomial time for any ranking function  $r$  and effect function  $e$  computable in polynomial time.
2. If the number  $n$  of rows of  $\mathbf{M}$  is bounded by a constant, then the problem  $\text{EXP}\langle r, e \rangle$  is solvable in polynomial time for any effect function  $e$  computable in polynomial time and
  - a. the ranking function  $r$  is in  $\{r_{\text{Max}}^{\text{asc}}, r_{\text{Max}}^{\text{dsc}}, r_{\text{Lex}}\}$ , or
  - b.  $r$  is the ranking function  $r_{\text{Sum}}$  and, additionally, the matrix  $\mathbf{M}$  and weights in  $\Pi$  are integers encoded in unary.

We note that the proof is straightforward for the cases where  $m$  (the number of columns) is fixed, since we can then materialize the entire probability space in an explicit representation. The proof of tractability for a fixed number of  $n$  of rows is more involved.

## 5 Intractable Cases for Expectation (and SHAP Scores)

We now show that the collection of tractability results in Section 4 is complete in the sense that expectation (and SHAP) computations are  $\text{FP}^{\#\text{P}}$ -hard for all remaining combinations of  $r$  and  $e$ .

### 5.1 Ranking based on Summation

We start with the observation that precedence probability is hard for ranking by Sum.

► **Theorem 9.**  $\text{PREC}\langle r_{\text{Sum}} \rangle$  is  $\text{FP}^{\#\text{P}}$ -hard, if the input matrix  $\mathbf{M}$  is encoded in binary.

The proof of this theorem is given in the archive version [33]. We use the following lemma to prove this claim by reducing from the counting knapsack problem that, given natural numbers  $b_1, \dots, b_\ell$  and a number  $d$  all encoded in binary, asks for the number of subsets  $S \subseteq [\ell]$  with  $\sum_{i \in S} b_i \leq d$ . This problem is known to be  $\#\text{P}$ -hard (see, e.g., [12, 16]).

► **Lemma 10.** Let  $b_1, \dots, b_\ell$  and  $d$  be an input to the counting knapsack problem. Then, there is a matrix  $\mathbf{M} \in \mathcal{M}^{2 \times (\ell+1)}$  with the following property: For each set  $C$  of columns, we have  $r_{\text{Sum}}(\mathbf{M}|_C)(2) \leq r_{\text{Sum}}(\mathbf{M}|_C)(1)$  if and only if  $C$  contains  $\ell + 1$  and  $\sum_{i \in C \setminus \{\ell+1\}} b_i \leq d$ .

**Proof.** Consider the following matrix  $\mathbf{M} \in \mathcal{M}^{2 \times (\ell+1)}$ :

id	$a_1$	$\dots$	$a_n$	$a_{n+1}$
1	0	$\dots$	0	$d + 1$
2	$b_1$	$\dots$	$b_n$	0

It is easy to verify this has the claimed property. ◀

We can use this theorem to show the hardness of expectation computation (and hence SHAP score computation) whenever we rank by Sum.

► **Corollary 11** (from Theorem 9). For each of the effect functions  $e \in \{e_{k\tau}, e_{\text{Ham}}, e_{\text{pos}}, e_{\text{top}}^k\}$ , the problem  $\text{EXP}\langle r_{\text{Sum}}, e \rangle$  is  $\text{FP}^{\#\text{P}}$ -hard if the input matrix  $\mathbf{M}$  is encoded in binary.

**Proof.** We observe that for  $n = 2$ , expectation computation of the effect functions  $e \in \{e_{k\tau}, e_{\text{Ham}}, e_{\text{pos}}\}$  is equivalent to precedence probability computation, as they are all of the form  $a \cdot \mathbb{P}(\pi(1) > \pi(2)) + b \cdot \mathbb{P}(\pi(2) > \pi(1))$  with  $a \neq b$ .

For  $\mathbf{M} \in \mathcal{M}^{2 \times m}$ ,  $k \geq 1$  and  $e = e_{\text{top}}^k$ , we consider a matrix  $\mathbf{M}' \in \mathcal{M}^{(k+1) \times m}$  where  $\mathbf{t}'_1 = \mathbf{t}_1$  and  $\mathbf{t}'_2 = \dots = \mathbf{t}'_{k+1} = \mathbf{t}_2$ , so all the tuples  $\mathbf{t}'_2$  up to  $\mathbf{t}'_{k+1}$  are identical. Now,  $\mathbf{t}'_1$  is in the top- $k$  of  $\mathbf{M}'$  is equivalent to  $\mathbf{t}_1$  being ranked better than  $\mathbf{t}_2$  in  $\mathbf{M}$ . This yields the claim. ◀

### 5.2 Top-k Membership

In Section 4, we saw a polynomial-time algorithm for  $\text{EXP}\langle r_{\text{Max}}^{\text{dsc}}, e_{\text{top}}^k \rangle$  when  $k$  is a parameter. We now show that this problem is hard for all other ranking functions we consider.

► **Theorem 12.** For each ranking function  $r \in \{r_{\text{Max}}^{\text{asc}}, r_{\text{Sum}}, r_{\text{Lex}}\}$  the problem  $\text{EXP}\langle r, e_{\text{top}}^k \rangle$  is  $\text{FP}^{\#\text{P}}$ -hard, when  $k$  is a parameter and also when  $k$  is part of the input. Furthermore,  $\text{EXP}\langle r_{\text{Max}}^{\text{dsc}}, e_{\text{top}}^k \rangle$  is  $\text{FP}^{\#\text{P}}$ -hard if  $k$  is part of the input. These claims remain true if the entries of  $\mathbf{M}$  are restricted to  $\{0, 1\}$  and the distribution  $\Pi$  is the uniform distribution on  $\{0, 1\}^m$ .

<b>u</b>	1	0	0	1
<b>t<sub>4</sub></b>	0	0	0	0
<b>t<sub>1</sub></b>	1	1	0	1
<b>t<sub>2</sub></b>	1	0	1	0
<b>t<sub>3</sub></b>	0	1	1	1

<b>u</b>	0	0	0	1
<b>t<sub>2</sub></b>	1	0	1	0
<b>t<sub>4</sub></b>	0	0	0	0
<b>t<sub>1</sub></b>	1	1	0	1
<b>t<sub>3</sub></b>	0	1	1	1

■ **Figure 2** Example 6:  $\mathbf{t}_4$  is top-1 under  $r_{\text{Max}}^{\text{asc}}$  iff  $\mathbf{u}$  gives a satisfying assignment for  $(X_1 \vee X_2 \vee X_4) \wedge (X_1 \vee X_3) \wedge (X_2 \vee X_3 \vee X_4)$ . The left is a satisfying assignment, while the right one is non-satisfying.

We prove this theorem (in the archive version [33]) using a reduction from counting satisfying assignments to positive CNF formulas. This problem is known to be  $\#P$ -hard [35]. Before we continue, we fix some notation for the remainder of this section. Let  $X_1, \dots, X_m$  be Boolean variables. A vector  $\mathbf{u} \in \{0, 1\}^m$  defines an assignment  $\alpha[\mathbf{u}]: [m] \rightarrow \{\text{true}, \text{false}\}$  by interpreting 1 as **true** and 0 as **false**. A set  $C \subseteq [m]$  defines the assignment  $\alpha[C] = \alpha[\mathbb{1}_C]$ , where  $\mathbb{1}_C \in \{0, 1\}^m$  is the incidence vector of  $C$ .

► **Example 13.** We illustrate how the satisfiability of a CNF formula can be modeled using the top-1 membership problem. Let our formula be  $(X_1 \vee X_2 \vee X_4) \wedge (X_1 \vee X_3) \wedge (X_2 \vee X_3 \vee X_4)$ . We construct the matrix shown in Figure 2, with columns corresponding to formula variables and tuples  $\mathbf{t}_1, \mathbf{t}_2, \mathbf{t}_3$  corresponding to clauses. A value of one indicates that the corresponding variable appears in the clause. Now, let the column weights be  $\{0, 1\}$ , representing whether a variable is true or false, and let the ranking function be  $r_{\text{Max}}^{\text{asc}}$ . By our construction,  $\mathbf{t}_4$  is top-1 if and only if all clauses are satisfied.  $\dashv$

We are now ready to state our main tool for the proof of Theorem 12.

► **Lemma 14.** Let  $\phi = \bigwedge_{i=1}^{\ell} D_i$  be a positive CNF formula with variable set  $X_1, \dots, X_m$  and clauses  $D_i = \bigvee_{j=1}^{r_i} X_{j_i}$ . Furthermore, let  $k \in \mathbb{N}$ . Then, there is a matrix  $\mathbf{M} \in \mathcal{M}^{(\ell+k) \times m}$  with entries in  $\{0, 1\}$  and the following property: For each set  $C$  of columns and each ranking function  $r \in \{r_{\text{Max}}^{\text{asc}}, r_{\text{Sum}}, r_{\text{Lex}}\}$ , we have  $r(\mathbf{M}|_C)(k + \ell) \leq k$  if and only if the assignment  $\alpha[C]$  models  $\phi$ .

► **Remark 15.** Using [3, Corollary 10], the previous lemma directly shows that we cannot approximate SHAP scores for  $r \in \{r_{\text{Max}}^{\text{asc}}, r_{\text{Sum}}, r_{\text{Lex}}\}$  and  $e = e_{\text{top}}^k$  unless  $\text{RP} = \text{NP}$ .

### 5.3 Maximum Displacement Distance

In this section, we prove that expectation computation is a hard problem if we measure the effect using the maximum displacement distance  $d_{\text{md}}(\pi_1, \pi_2) = \max_{1 \leq i \leq n} |\pi_1(i) - \pi_2(i)|$ .

► **Theorem 16.** For  $e = e_{\text{md}}$  and  $r \in \{r_{\text{Max}}^{\text{asc}}, r_{\text{Max}}^{\text{dsc}}, r_{\text{Sum}}, r_{\text{Lex}}\}$ , the problem  $\text{EXP}(r, e)$  is  $\text{FP}^{\#P}$ -hard. This remains true if the entries of  $\mathbf{M}$  are restricted to  $\{0, 1\}$  and the distribution  $\Pi$  is the uniform distribution on  $\{0, 1\}^m$ .

We prove this theorem (in the archive version [33]) via a reduction from counting satisfying assignments to positive CNF formulas. The proof is similar to that of Theorem 12, but with a small twist. Since  $d_{\text{md}}$  is a global measure, we cannot define a matrix  $\mathbf{M}$  where we separate satisfying assignments from non-satisfying ones. Instead, we will define two matrices  $\mathbf{M}_1$  and  $\mathbf{M}_2$  in a way that the difference of the effect functions allows this distinction.

► **Lemma 17.** *Let  $\phi = \bigwedge_{i=1}^{\ell} D_i$  be a positive CNF formula with variable set  $X_1, \dots, X_m$  and clauses  $D_i = \bigvee_{j=1}^{r_i} X_{j_i}$ . Then, there exist two matrices  $\mathbf{M}_1 \in \mathcal{M}^{(2\ell+1) \times m}$  and  $\mathbf{M}_2 \in \mathcal{M}^{(2\ell+2) \times m}$  with entries in  $\{0, 1\}$  and the following property: For each set  $C$  of columns and each ranking function  $r \in \{r_{\text{Max}}^{\text{asc}}, r_{\text{Sum}}, r_{\text{Lex}}\}$ , we have*

$$d_{\text{md}}(r(\mathbf{M}_2), r(\mathbf{M}_2|_C)) - d_{\text{md}}(r(\mathbf{M}_1), r(\mathbf{M}_1|_C)) = \begin{cases} 0, & \text{if } \alpha[C] \models \phi \\ 1, & \text{otherwise.} \end{cases} \quad (8)$$

**Proof sketch of Lemma 17.** The construction of the matrices is similar to Example 6 and Lemma 14 with more copies of the all-zero tuple, which occupy the top positions whenever the formula is satisfied, yielding zero displacement. Matrix  $\mathbf{M}_2$  contains one additional copy of the all-zero tuple compared to  $\mathbf{M}_1$ , so the two matrices behave similarly, but the maximum displacement in  $\mathbf{M}_2$  is higher than that of  $\mathbf{M}_1$  by exactly 1 whenever the formula is not satisfied. ◀

## 6 Exact Computation of the Shapley Value

In this section, we turn our attention to the importance of columns via the Shapley value instead of the importance of scoring parameters via the SHAP score. We will first show that all our algorithms from Section 4 for SHAP score computation can be used to determine Shapley values and then complete the picture by extending our results from Section 5.

► **Theorem 18.** *For the ranking functions  $r$  and effect functions  $e$  that we introduced in Section 3, computational complexity of  $\text{Shapley}(r, e)$  is the same as the computational complexity of  $\text{SHAP}\langle r, e \rangle$  given in Table 1.*

### 6.1 Algorithms

Gilad et al. [15] give a simple algorithm based on interpolation that computes Shapley values in polynomial time when having access to an oracle to a special instance of SHAP score computation called binarySHAP. Translated into our setting, their result is the following:

► **Theorem 19** ([15, Theorem 6.15]). *Let  $r$  be a ranking function and  $e$  be an effect function with the property that for each set  $C$  of columns, we have  $e(r(\mathbf{M}|_C)) = e(r(\mathbf{M} \circ \mathbb{1}_C))$ . Then, we can solve  $\text{Shapley}\langle r, e \rangle$  in polynomial time with  $m$  oracle calls to  $\text{SHAP}\langle r, e \rangle$ .*

While the condition of the theorem is always true for ranking by Sum and Lex, it does not necessarily hold for Max ranking if  $\mathbf{M}$  contains negative numbers. Fortunately, we can make  $\mathbf{M}$  non-negative by adding the minimal value in  $\mathbf{M}$  to each entry without changing the rankings of each set of columns. Hence, we can use the previous theorem to show that all our algorithms carry over to Shapley value computation.

### 6.2 Intractable Cases

To show hardness results from Section 5, we reduced the counting knapsack problem (Lemma 10) and the problem of counting satisfying assignments to positive CNF formulas (Lemmas 14 and 17) to SHAP score computation via the equivalence to expectation computation (Theorem 1). Since this equivalence does not hold in general for Shapley value computation, we need an alternative approach to extend our results. Instead, we will define two auxiliary games and show the hardness of Shapley value computation for each of them. The proofs are given in the archive version [33]. They follow the technique that is used for

most hardness proofs of Shapley value computation [18, 22]: For a given instance to a game, define a set of related instances and use their Shapley values to count valid knapsack sets or satisfying assignments, respectively, via interpolation.

The first game is the *knapsack game*: For natural numbers  $b_1, \dots, b_\ell$  and  $b_{\ell+1} = d$  given in binary, consider the game with player set  $[\ell + 1]$  and valuation function  $\nu_{\text{knap}}$  where  $\nu_{\text{knap}}(C) := 1$  if  $\ell + 1 \in C$  and  $\sum_{i \in C \setminus \{\ell\}} b_i \leq d$ , and  $\nu_{\text{knap}}(C) := 0$  otherwise.

► **Lemma 20.** *Shapley value computation for the players of the knapsack game is  $\text{FP}^{\#P}$ -hard.*

The second game is the *positive CNF game*: Given a positive CNF formula  $\phi$  with variable set  $X_1, \dots, X_m$ , consider the game with player set  $[m]$  and valuation function  $\nu_{\text{CNF}}$  where  $\nu_{\text{CNF}}(C) := 1$  if  $\alpha[C] \models \phi$ , and  $\nu_{\text{CNF}}(C) := 0$  otherwise.

► **Lemma 21.** *Shapley value computation is  $\text{FP}^{\#P}$ -hard for the positive CNF game.*

With these lemmas and the constructions from Section 5, we can now easily prove Theorem 18. The details are in the archive version of the paper [33].

## 7 Conclusions

We studied the computational complexity of measuring the importance of parameter choices in ranking functions. Specifically, we investigated the SHAP score of parameters under a collection of specific (yet basic) ranking functions and effect functions. For that, we studied the complexity of calculating the expected effect over random parameter values, focusing on finite and probabilistically independent distributions. We also studied the related problem of computing the Shapley value of columns for measuring their contribution to the ranking.

We view this work as a first step in the rigorous analysis of the complexity of function-based explanations for rankings. As such, many directions are left for future investigation. We would like to generalize our results into broad *classes* of ranking and effect functions, rather than considering specific cases separately. In particular, we aim to analyze the implication of the different choices made in popular ranking schemes; examples include sports ranking such as WBSC<sup>4</sup> and FIFA/Coca-Cola World Ranking,<sup>5</sup> which have the shape of workflow diagrams with linear combinations of features, and the Computer Science Rankings<sup>6</sup> that accounts for a pre-determined choice of publication venues. We also want to extend our analysis to other classes of probability distributions, including uniform distributions over intervals, and to characterize the cases where we can efficiently approximate SHAP multiplicatively. Moreover, we would like to understand the complexity of general types of effect functions that include alternatives to those studied here, such as Spearman’s footrule and the Cayley distance.

Finally, an important direction for future research is to study the practicality of the framework in real-life scenarios, and particularly understand how well the attribution functions of SHAP and Shapley capture people’s intuition on the contribution of the components of a ranking function.

---

## References

- 1 Hadis Anahideh and Nasrin Mohabbati-Kalejahi. Local explanations of global rankings: insights for competitive rankings. *IEEE Access*, 10:30676–30693, 2022. doi:10.1109/ACCESS.2022.3159245.

---

<sup>4</sup> <https://www.wbsc.org/en/rankings>

<sup>5</sup> <https://inside.fifa.com/fifa-world-ranking>

<sup>6</sup> <https://csrankings.org/>

- 2 Avishek Anand, Lijun Lyu, Maximilian Idahl, Yumeng Wang, Jonas Wallat, and Zijian Zhang. Explainable information retrieval: A survey, 2022. doi:10.48550/arXiv.2211.02405.
- 3 Marcelo Arenas, Pablo Barcelo, Leopoldo Bertossi, and Mikael Monet. On the complexity of shap-score-based explanations: Tractability via knowledge compilation and non-approximability results. *Journal of Machine Learning Research*, 24(63):1–58, 2023. URL: <http://jmlr.org/papers/v24/21-0389.html>.
- 4 Abolfazl Asudeh, H. V. Jagadish, Gerome Miklau, and Julia Stoyanovich. On obtaining stable rankings. *PVLDB*, 12(3):237–250, 2018. doi:10.14778/3291264.3291269.
- 5 Alexander Barg and Arya Mazumdar. Codes in permutations and error correction for rank modulation. *IEEE Trans. Inf. Theory*, 56(7):3158–3165, 2010. doi:10.1109/TIT.2010.20484.
- 6 Leopoldo E. Bertossi, Benny Kimelfeld, Ester Livshits, and Mikael Monet. The Shapley value in database management. *SIGMOD Record*, 52(2):6–17, 2023. doi:10.1145/3615952.361595.
- 7 Javier Castro, Daniel Gómez, Elisenda Molina, and Juan Tejada. Improving polynomial estimation of the shapley value by stratified random sampling with optimum allocation. *Computers & Operations Research*, 82:180–188, 2017. doi:10.1016/j.cor.2017.01.019.
- 8 Zixuan Chen, Panagiotis Manolios, and Mirek Riedewald. Why not yet: Fixing a top-k ranking that is not fair to individuals. *PVLDB*, 16(9):2377–2390, 2023. doi:10.14778/3598581.3598606.
- 9 Tanya Chowdhury, Yair Zick, and James Allan. RankSHAP: Shapley value based feature attributions for learning to rank. In *ICLR*, 2025. URL: <https://openreview.net/forum?id=4011PUI9vm>.
- 10 Daniel Deutch, Nave Frost, Benny Kimelfeld, and Mikael Monet. Computing the Shapley value of facts in query answering. In *SIGMOD*, pages 1570–1583. ACM, 2022. doi:10.1145/3514221.3517912.
- 11 Alexandre Duval and Fragkiskos D. Malliaros. GraphSVX: Shapley value explanations for graph neural networks. In *ECML/PKDD (2)*, volume 12976 of *LNCS*, pages 302–318. Springer, 2021. doi:10.1007/978-3-030-86520-7\_19.
- 12 Martin E. Dyer, Alan M. Frieze, Ravi Kannan, Ajai Kapoor, Ljubomir Perkovic, and Umesh V. Vazirani. A mildly exponential time algorithm for approximating the number of solutions to a multidimensional knapsack problem. *Comb. Probab. Comput.*, 2:271–284, 1993. doi:10.1017/S0963548300000675.
- 13 Ronald Fagin, Benny Kimelfeld, and Phokion G. Kolaitis. Probabilistic data exchange. *J. ACM*, 58(4):15:1–15:55, 2011. doi:10.1145/1989727.1989729.
- 14 Abraham Gale and Amélie Marian. Explaining monotonic ranking functions. *PVLDB*, 14(4):640–652, 2020. doi:10.14778/3436905.3436922.
- 15 Amir Gilad, Martin Grohe, Benny Kimelfeld, Peter Lindner, and Christoph Standke. The importance of parameters in database queries, 2024. doi:10.48550/arXiv.2401.04606.
- 16 Parikshit Gopalan, Adam Klivans, Raghu Meka, Daniel Štefankovic, Santosh Vempala, and Eric Vigoda. An FPTAS for #knapsack and related counting problems. In *FOCS*, pages 817–826, 2011. doi:10.1109/FOCS.2011.32.
- 17 Erich Grädel, Yuri Gurevich, and Colin Hirsch. The complexity of query reliability. In *PODS*, pages 227–234. ACM Press, 1998. doi:10.1145/275487.29512.
- 18 Martin Grohe, Benny Kimelfeld, Peter Lindner, and Christoph Standke. The importance of parameters in database queries. In *ICDT*, volume 290 of *LIPICs*, pages 14:1–14:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPICs.ICDT.2024.14.
- 19 Maria Heuss, Maarten de Rijke, and Avishek Anand. RankingSHAP - listwise feature attribution explanations for ranking models, 2024. doi:10.48550/arXiv.2403.16085.
- 20 Uday Kamath and John Liu. *Explainable Artificial Intelligence: An Introduction to Interpretable Machine Learning*. Springer, 2021. doi:10.1007/978-3-030-83356-5.
- 21 Ahmet Kara, Dan Olteanu, and Dan Suciu. From Shapley value to model counting and back. *PACMOD*, 2(2):79, 2024. doi:10.1145/3651142.

- 22 Ester Livshits, Leopoldo E. Bertossi, Benny Kimelfeld, and Moshe Sebag. The shapley value of tuples in query answering. *Log. Methods Comput. Sci.*, 17(3), 2021. doi:10.46298/LMCS-17(3:22)2021.
- 23 Scott M. Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *NIPS*, pages 4765–4774, 2017. URL: <https://proceedings.neurips.cc/paper/2017/hash/8a20a8621978632d76c43dfd28b67767-Abstract.html>.
- 24 Lijun Lyu and Avishek Anand. Listwise explanations for ranking models using multiple explainers. In *ECIR*, volume 13980 of *LNCS*, pages 653–668. Springer, 2023. doi:10.1007/978-3-031-28244-7\_41.
- 25 Gustavo Penha, Eyal Krikon, and Vanessa Murdock. Pairwise review-based explanations for voice product search. In *CHIIR*, pages 300–304. ACM, 2022. doi:10.1145/3498366.350582.
- 26 Venetia Pliatsika, Joao Fonseca, Kateryna Akhynko, Ivan Shevchenko, and Julia Stoyanovich. ShaRP: A novel feature importance framework for ranking, 2024. arXiv:2401.16744.
- 27 Ariel D. Procaccia, Nisarg Shah, and Yair Zick. Voting rules as error-correcting codes. *Artificial Intelligence*, 231:1–16, 2016. doi:10.1016/j.artint.2015.10.003.
- 28 Alvin E. Roth, editor. *The Shapley value : essays in honor of Lloyd S. Shapley*. Cambridge University Press, 1988. URL: <http://www.loc.gov/catdir/samples/cam031/88002983.html>.
- 29 Benedek Rozemberczki, Lauren Watson, Péter Bayer, Hao-Tsung Yang, Olivér Kiss, Sebastian Nilsson, and Rik Sarkar. The Shapley value in machine learning. In *IJCAI*, pages 5572–5579, 2022. doi:10.24963/ijcai.2022/778.
- 30 Sourav Saha, Debapriyo Majumdar, and Mandar Mitra. Explainability of text processing and retrieval methods: A critical survey, 2022. doi:10.48550/arXiv.2212.07126.
- 31 Lloyd S Shapley. A value for n-person games. In Harold W. Kuhn and Albert W. Tucker, editors, *Contributions to the Theory of Games II*, pages 307–317. Princeton University Press, Princeton, 1953. doi:10.1515/9781400829156-012.
- 32 Jaspreet Singh and Avishek Anand. Model agnostic interpretability of rankers via intent modelling. In *FAT\**, pages 618–628. ACM, 2020. doi:10.1145/3351095.33752.
- 33 Christoph Standke, Nikolaos Tziavelis, Wolfgang Gatterbauer, and Benny Kimelfeld. The importance of parameters in ranking functions, 2026. arXiv:2601.06001.
- 34 Seinosuke Toda and Mitsunori Ogiwara. Counting classes are at least as hard as the polynomial-time hierarchy. *SIAM J. Comput.*, 21(2):316–328, 1992. doi:10.1137/0221023.
- 35 Leslie G Valiant. The complexity of enumeration and reliability problems. *SIAM Journal on Computing*, 8(3):410–421, 1979. doi:10.1137/0208032.
- 36 Guy Van den Broeck, Anton Lykov, Maximilian Schleich, and Dan Suciu. On the tractability of SHAP explanations. *Journal of Artificial Intelligence Research*, 74:851–886, June 2022. doi:10.1613/jair.1.13283.
- 37 Ke Yang, Julia Stoyanovich, Abolfazl Asudeh, Bill Howe, HV Jagadish, and Gerome Miklau. A nutritional label for rankings. In *SIGMOD*, pages 1773–1776, 2018. doi:10.1145/3183713.319356.
- 38 Puxuan Yu, Razieh Rahimi, and James Allan. Towards explainable search results: A listwise explanation generator. In *SIGIR*, pages 669–680. ACM, 2022. doi:10.1145/3477495.3532067.