AdaNDV: Adaptive Number of Distinct Value Estimation via Learning to Select and Fuse Estimators

Xianghong Xu
ByteDance
Beijing, China
xuxianghong@bytedance.com

Tieying Zhang*
ByteDance
San Jose, USA
tieying.zhang@bytedance.com

Xiao He
ByteDance
Hangzhou, China
xiao.hx@bytedance.com

Haoyang Li
ByteDance
Beijing, China
lihaoyang.cs@bytedance.com

Rong Kang
ByteDance
Beijing, China
kangrong.cn@bytedance.com

Shuai Wang
ByteDance
Beijing, China
wangshuai.will@bytedance.com

Linhui Xu
ByteDance
Beijing, China
xulinhui@bytedance.com

Zhimin Liang
ByteDance
Beijing, China
liangzhimin@bytedance.com

Shangyu Luo
ByteDance
San Jose, USA
shangyu.luo@bytedance.com

Lei Zhang
ByteDance
Shenzhen, China
zhanglei.michael@bytedance.com

Jianjun Chen
ByteDance
San Jose, USA
jianjun.chen@bytedance.com

ABSTRACT

Estimating the Number of Distinct Values (NDV) is fundamental for numerous data management tasks, especially within database applications. However, most existing works primarily focus on introducing new statistical or learned estimators, while identifying the most suitable estimator for a given scenario remains largely unexplored. Therefore, we propose AdaNDV, a learned method designed to adaptively select and fuse existing estimators to address this issue. Specifically, (1) we propose to use learned models to distinguish between overestimated and underestimated estimators and then select appropriate estimators from each category. This strategy provides a complementary perspective by integrating overestimations and underestimations for error correction, thereby improving the accuracy of NDV estimation. (2) To further integrate the estimation results, we introduce a novel fusion approach that employs a learned model to predict the weights of the selected estimators and then applies a weighted sum to merge them. By combining these strategies, the proposed AdaNDV fundamentally distinguishes itself from previous works that directly estimate NDV. Moreover, extensive experiments conducted on real-world datasets, with the number of individual columns being several orders of magnitude larger than in previous studies, demonstrate the superior performance of our method.

PVLDB Reference Format:
Xianghong Xu, Tieying Zhang, Xiao He, Haoyang Li, Rong Kang, Shuai Wang, Linhui Xu, Zhimin Liang, Shangyu Luo, Lei Zhang, and Jianjun Chen. AdaNDV: Adaptive Number of Distinct Value Estimation via Learning to Select and Fuse Estimators. PVLDB, 18(4): 1104 - 1117, 2024.

*Tieying Zhang corresponds to this work.

doi:10.14778/3717755.3717769

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/bytedance/adandv.

1 INTRODUCTION

Identifying the Number of Distinct Values (NDV) in a data column is a fundamental task across numerous data management scenarios, particularly within the database domain. However, directly obtaining NDV is often impractical in real-world scenarios due to the prohibitive overheads of processing massive data volumes or data access restrictions. Therefore, estimating NDV on limited samples has been a critical and longstanding research topic, explored for over seven decades [27]. For instance, in the realm of Biology, a critical task is to estimate unseen species [15, 53, 54]. Similarly, in Statistics, a recurring engagement involves quantifying the number of distinct categories within a given population [18, 27]. Moreover, in the domain of Networks, assessing the quantity of virtualized devices is a significant challenge [23, 40]. In Database, some widely used systems (e.g., Spark and PostgreSQL) directly rely on NDV to compute cardinality, a metric that is subsequently utilized by the query optimizer [9, 10]. Besides, NDV affects the join order selection in MySQL as well [8]. Furthermore, recent studies show

Proceedings of the VLDB Endowment, Vol. 18, No. 4 ISSN 2150-8097.
doi:10.14778/3717755.3717769

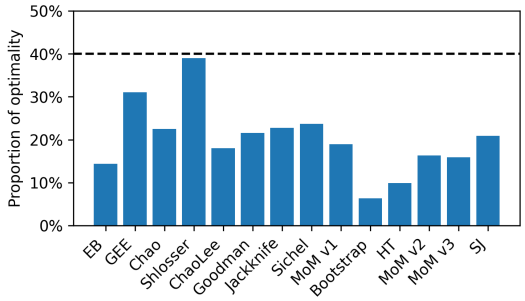


Figure 1: Evaluation of fourteen statistical estimators on 25,159 test columns, where the bar represents the proportion of each estimator achieving the optimality (lowest estimation error) among the fourteen estimators. No single estimator achieves optimality on more than 40% of test cases.

that precise NDV estimation can generate better query plans that bring significant SQL query execution latency reductions [29, 36].

Prolonged research efforts in NDV estimation have accumulated numerous NDV estimators, the majority of which are statistical methods relying on manually designed polynomials computing or equation-solving [15, 18–21, 27, 28, 28, 42, 47–52]. Each statistical estimator is grounded in distinct hypotheses regarding the underlying distribution. As a result, their efficacy markedly declines when the actual distribution does not conform to their assumptions. Recently, some studies have introduced learned estimators based on Machine Learning (ML) techniques [34, 57] to solve this issue, demonstrating better performance than statistical estimators. Nevertheless, despite the significant progress achieved, the domain of NDV estimation is still beset by the following difficulties:

(1) Selection dilemma. Although a plethora of estimators exist, there frequently remains ambiguity regarding the optimal choice for practical application. The question of *which estimator is most suitable for a specific data column* has received scarce attention over time. As new estimators continue to be introduced, this understudied question becomes increasingly substantial. To intuitively show the selection dilemma, we depict the performance of fourteen statistical estimators on a large dataset comprising 25,159 test cases (for additional details, please refer to Section 4.1) in Figure 1. The results illustrate that no single estimator consistently achieves the lowest estimation error across all test cases, indicating that no individual estimators can consistently beat others. For instance, the top-performing estimator, Shlosser, only manages to outperform others in roughly 40% of the cases. This observation highlights the intricate nature of selecting a suitable estimator from a set of available options.

(2) Underexploitation issue. Most studies have focused on exploring new estimators, including the recently proposed learned estimator [34, 57], while exploiting existing estimators to improve estimations has been largely overlooked. To better illustrate the issue, we conceptualize a *hypothetical* estimator, which involves picking one of the aforementioned fourteen statistical estimators, under the hypothetical condition that we know the actual estimation errors in advance. Precisely, we name the hypothetical estimator "Hypo-optimal" since we select the estimator with the lowest actual estimation error for every test. The performance of the hypothetical estimator and a state-of-the-art (SOTA) learned estimator [57] is shown in Table 1. From the results in the table, it is evident that the Hypo-optimal estimator considerably outperforms the SOTA learned estimator. The comparison illustrates the substantial potential held by statistical estimators, simultaneously underscoring the critical significance of judiciously exploiting estimators.

Table 1: Experiments on hypothetical estimators. The numbers represent estimation errors, with lower values indicating better performance.

Estimator	Mean	50%	75%	90%	95%	99%
Hypo-optimal	**1.20**	**1.08**	**1.28**	**1.56**	**1.83**	**2.36**
SOTA learned estimator [57]	2.24	1.72	2.28	3.20	4.11	10.46

Inspired by the above observations, in this paper, we introduce AdaNDV, an Adaptive NDV estimation method learning to select the proper estimators from existing ones and to fuse their estimation results to enhance estimation precision for different scenarios. Selecting the optimal estimator with high accuracy is quite challenging because it is difficult to extract adequate features and explore appropriate ML models. On the contrary, selecting the k estimators that are most likely to approach the ground truth is a relatively easier task for ML models [37]. Moreover, we distinctively propose to address the issue by distinguishing whether an estimator is overestimated or underestimated, allowing us to utilize them accordingly. Specifically, if one estimator overestimates and another underestimates for a test case, there exists a set of weights such that their weighted sum performs better than either estimator individually. This approach inherently leverages the complementary nature of overestimations and underestimations to reduce estimation error. In addition, in our initial experiments, we found that certain base estimators tend to make more overestimations or underestimations, indicating that distinguishing between overestimating and underestimating estimators is a comparatively easier task. Subsequently, we select the estimators with leading overestimation and underestimation performance respectively. Further, we introduce a learned model to predict the weights of the chosen estimators and then establish the ultimate estimation by applying a weighted sum to fuse them. Different from the previous works that directly estimate NDV, our method offers a novel approach to enhance the accuracy of NDV estimation by merging existing estimators. This allows our method to adaptively select appropriate estimators for specific scenarios and allocate proper weights to fuse them for end-to-end estimation. Finally, extensive experiments are carried out on a voluminous dataset from the real world. Specifically, the number of individual test columns exceeds tens of thousands, while previous works were tested on at most about two hundred columns. This orders of magnitude increase in individual test columns enables a thorough evaluation encompassing existing estimators alongside our novel approach.

To sum up, the main contributions are shown as follows:

- We propose AdaNDV, an adaptive NDV estimation method learning to select and fuse appropriate estimators for specific scenarios. To the best of our knowledge, we are the first to combine existing estimators with ML techniques to improve NDV estimation.
- We introduce an innovative overestimation-underestimation complementary perspective for estimator selection and exploitation to reduce estimation error.
- We develop a novel learned weighted sum strategy to fuse the estimation results to obtain the ultimate estimation, which is significantly different from directly estimating NDV.
- Extensive experiments, conducted on a rich, real-life dataset with tens of thousands of individual columns, significantly larger than at most hundreds of columns used in past studies, demonstrate the superiority of AdaNDV.

2 PRELIMINARIES

2.1 Problem Statement

Existing NDV estimation methods can be categorized into *sketch-based* [25, 26, 31] and *sampling-based* [27, 34, 57], while most of the former require scanning all the data, making them impractical in many scenarios. Here, we focus on sampling-based NDV estimation.

NDV Estimation Definition. Given a data column C with N rows, let D be the NDV of column C. The task is to estimate D by uniformly sampling n ($n \leq N$) rows (denoted S, and $S \subseteq C$) from C. Let $r = n/N$ be the sampling rate, and we assume N is known. We define d as the NDV of S.

Two features are widely discussed in sampling-based NDV estimation: *frequency* and *frequency profile*.

Frequency. The frequency of a value x in C is the number of times it appears in C. Denote $N_x = \sum_{i \in C} \mathbb{1}_x(i)$, where N_x is the frequency of value x in C, $\mathbb{1}_x(\cdot)$ is the indicator function that returns 1 if the input equals to x and 0 otherwise. Similarly, $n_x = \sum_{i \in S} \mathbb{1}_x(i)$ is the frequency of value x in S.

Frequency Profile. Frequency profile is the *frequency of frequency*. Let the frequency profile of C be $F = (F_j)_{j=1,2,\ldots,N}$, where $F_j = |\{i \in C | N_i = j\}|$. Similarly, the frequency profile of S be $f = (f_j)_{j=1,2,\ldots,n}$, where $f_j = |\{i \in S | n_i = j\}|$.

Feature Relations and Examples. The two features are closely related to NDV and the number of rows. We can get $D = |\{i \in C | N_i > 0\}| = \sum_{j=1}^{N} F_j$ and $d = |\{i \in S | n_i > 0\}| = \sum_{j=1}^{n} f_j$. Besides, the total number of rows can be expressed as $N = \sum_{j=1}^{N} j \cdot F_j$, and $n = \sum_{j=1}^{n} j \cdot f_j$.

For instance, suppose the sample data is $S = \{a, a, a, b, b, b, c, c, d\}$, we can observe that the sample size is $|S| = 9$, and $d = 4$ (there are a, b, c, d four distinct values). The frequency of S is $\{n_a = 3, n_b = 3, n_c = 2, n_d = 1\}$, and the frequency of frequency is $\{f_1 = 1, f_2 = 1, f_3 = 2, f_i = 0, i = 4, \ldots, 9\}$. Based on these features, we can get $d = \sum_{i=1}^{9} f_i = 4$, and $|S| = \sum_{i=1}^{9} i \cdot f_i = 9$.

2.2 NDV Estimators

We use several representative estimators to demonstrate how they use the frequency profile of sample data to estimate NDV.

Traditional Estimators. Goodman [27] is a representative *linear polynomial* estimator with a sophisticated expression:

$$D_{\text{Goodman}} = d + \sum_{i=1}^{n} (-1)^{i+1} \frac{(N-n+i-1)!(n-i)!}{(N-n-1)!n!} f_i. \quad (1)$$

Chao [18, 42] estimator has a *nonlinear polynomial* expresion:

$$D_{\text{Chao}} = d + \frac{f_1^2}{2f_2}. \quad (2)$$

Besides, some estimators need to solve sophisticate *non-linear equations* constructed by the frequency profiles [15, 20, 49–51]. For instance, Sichel [49–51] estimator needs to solve the following non-linear equations:

$$\begin{aligned} &(1+g)\ln g - Ag + B = 0, \frac{f_1}{n} < g < 1, A = \frac{2n}{d} - \ln\frac{n}{f_1}, \\ &B = \frac{2f_1}{d} + \ln\frac{n}{f_1}, \hat{b} = \frac{g \ln\frac{ng}{f_1}}{1-g}, \hat{c} = \frac{1-g^2}{ng^2}, D_{\text{Sichel}} = \frac{2}{\hat{b}\hat{c}}. \end{aligned} \quad (3)$$

Learned Estimators. Recently, ML techniques have been introduced into NDV estimation [34, 57]. Wu et al. [57] proposed a learned statistician (LS in short) to estimate NDV. It constructs a multi-layer perception (MLP) as the estimator, which takes the cut-off of the frequency profile and some features as input and outputs the estimated NDV. It is trained in the regression paradigm, which minimizes the L_2 loss between the estimated NDV and the ground truth.

Evaluation Protocol. Ratio-error, also known as q-error [38], is widely used to evaluate the performance of an estimator in database applications:

$$\text{q-error} = \max\left(\frac{\hat{D}}{D}, \frac{D}{\hat{D}}\right), \quad (4)$$

where $\hat{D}$ is the estimated NDV and D is the ground truth NDV. The lower error represents the better performance.

3 METHODOLOGY

3.1 Model Architecture

3.1.1 Overview of AdaNDV. The architecture of AdaNDV is shown in Figure 2 and there are four components.

(1) Base estimators collection. We collect fourteen representative statistical estimators as our base estimators, detailed in Section 4.1. We do not include learned estimators due to the limited availability of such estimators and nontrivial overhead associated with training them. Additionally, statistical estimators offer higher efficiency and are free of training.

(2) Leading estimator selection. "OverEst" and "UnderEst" represent overestimation and underestimation. Estimator selection is designed to prioritize the base estimators by their overestimation and underestimation errors. Essentially, the prioritization of a set of base estimators entails assigning them scores that reflect their performance in terms of overestimation and underestimation errors. Specifically, we use two identical learned models with different training objectives to prioritize the two types of estimators. The loss functions of the two models are denoted as $\mathcal{L}_{\text{over}}$ and $\mathcal{L}_{\text{under}}$, respectively. This component will be detailed in Section 3.2.

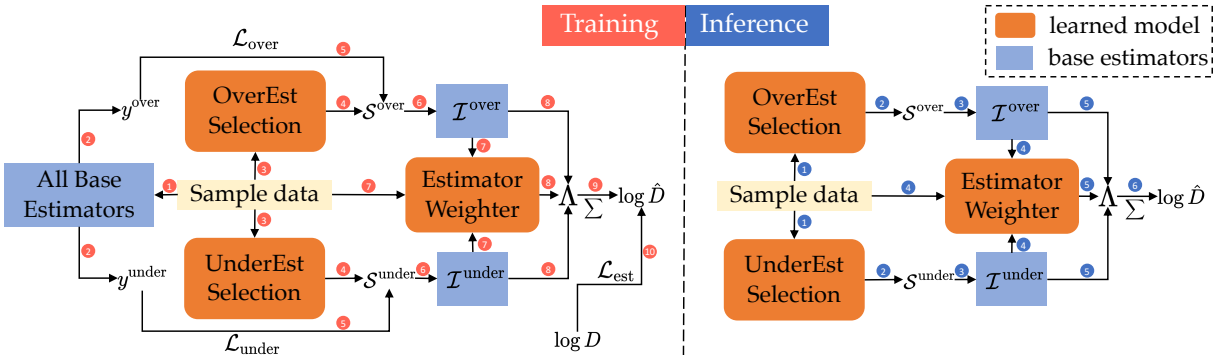


Figure 2: Overview of AdaNDV on NDV estimation including training and inference data pipelines.

(3) Estimator fusion. We first select estimators with the top-k overestimated and underestimated performance, denoted as $\mathcal{I}^{\text{over}}$ and $\mathcal{I}^{\text{under}}$. Next, we use a learned model to assign weights to each selected estimator, and then employ a weighted sum to compute the NDV. For instance, suppose k is 1, the ground truth NDV D is 10,000, and the estimation results of the selected estimators are $\hat{D}_1 = 11,000$ and $\hat{D}_2 = 9,000$, our estimation is formulated as $\hat{D} = \Lambda_1\hat{D}_1 + \Lambda_2\hat{D}_2, 0 \le \Lambda_1, \Lambda_2 \le 1, \Lambda_1 + \Lambda_2 = 1$. This fusion method can reduce estimation errors based on existing estimators. The loss function of this component is denoted as $\mathcal{L}_{\text{est}}$ and its details will be elaborated in Section 3.3.

(4) Model training. There are three objectives in AdaNDV, and we can derive the end-to-end loss function to train our method:

$$\mathcal{L}_{\text{AdaNDV}} = \mathcal{L}_{\text{over}} + \mathcal{L}_{\text{under}} + \beta\mathcal{L}_{\text{est}}, \tag{5}$$

where β is a hyperparameter that modulates the trade-offs between different kinds of training objectives. Our proposed method can be trained by minimizing the $\mathcal{L}_{\text{AdaNDV}}$ loss function.

Training pipeline. ①All base estimators use the sample data to estimate NDV. ②It is straightforward to distinguish the overestimated and underestimated estimators on the training data, and then we construct training labels y^{over} and y^{under}. ③-④The estimator selection models will respectively generate the scores that prioritize the estimators by the predicted overestimation and underestimation performance. $\mathcal{S}^{\text{over}} \in \mathbb{R}^m$ represents the scores based on overestimation, where m is the number of base estimators, with a higher value indicating better overestimation performance. $\mathcal{S}^{\text{under}} \in \mathbb{R}^m$ is similar to $\mathcal{S}^{\text{over}}$. ⑤The selection loss functions ($\mathcal{L}_{\text{over}}$ and $\mathcal{L}_{\text{under}}$) of the two models take the scores and labels as input. ⑥Then we respectively select top base estimators with high scores and the selected estimators are $\mathcal{I}^{\text{over}}$ and $\mathcal{I}^{\text{under}}$. ⑦The learned estimator weighter takes the sample data and the estimations of the selected estimators as input and predicts the weight (Λ). ⑧-⑨Finally, we employ a weighted sum on the estimation results of the selected estimators to fuse them into the ultimate NDV estimation $\hat{D}$, ⑩deriving fusion-based estimation loss function $\mathcal{L}_{\text{est}}$.

Inference pipeline. ①-②The learned leading estimator selection models take the sample data as input to generate the scores of each estimator. ③Then, we select the top estimators with the highest scores for overestimation and underestimation, respectively.

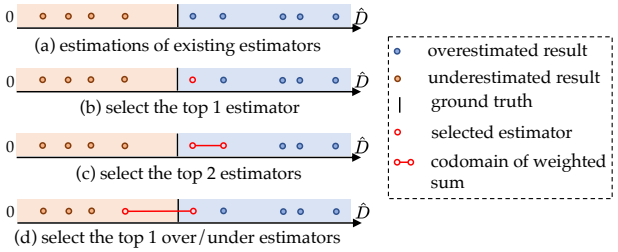


Figure 3: Intuition behind leveraging the properties of overestimation and underestimation.

④Next, we input the sample data and the estimations of the selected estimator into the learned estimator weighter to obtain the weights. ⑤-⑥Finally, the ultimate estimation is fused by a weighted sum on the estimations of the selected estimators.

3.1.2 Properties of overestimation and underestimation. We illustrate the intuition behind our method in leveraging the properties of overestimation and underestimation in Figure 3. Each circle in the figure represents the estimation result from a base estimator. The ground truth, indicated by a vertical bar in the middle, separates these results into two sections: overestimation and underestimation, which are marked with two different colors. The performance of selecting the optimal estimator from existing ones (as shown in Figure 3b) is challenging and limited by both selection accuracy and the base estimators. While it is straightforward to ensemble estimators to alleviate these issues, selecting estimators with the lowest errors and ensemble them may not bring performance improvement. Figure 3c illustrates such a scenario where two overestimation results are selected, and the corresponding codomain cannot cover the ground truth bar. Therefore, differentiating the results into overestimations and underestimations is essential for improving the performance. Although the estimation errors of the selected results may be substantial, complementing overestimations and underestimations enables their weighted sum to robustly encompass the ground truth, as shown in Figure 3d. This allows the model to learn a set of parameters, resulting in the weighted sum of the selected results performing better than any individual estimator, potentially even approaching the ground truth.

3.2 Leading Estimator Selection

We first describe the features extracted from the sample data, then we comprehensively illustrate the ranking paradigm used in estimator selection, and finally, we delineate the construction of the objective function for this component.

3.2.1 Feature Engineering. Formally, we denote the features extracted from the sample data as $x \in \mathbb{R}^H$, where H denotes the number of dimensions in the feature space. Frequency profiles f are widely used features, but their sizes varies across different column test cases. Since a learned model needs a fixed number of input features, we apply a cut-off to the frequency profile, similar to previous works [34, 57]. This operation is based on the assumption that the predictive power of f_i decreases as i increases, an assumption that is widely used in previous works [18, 20, 28, 34, 57]. In addition, according to Section 2.1, cutting off the frequency profile will make computing the number of sample data n and the NDV of sample data d inapplicable. Therefore, we use n and d as directly as features. Furthermore, we include the original column size N as another feature.

Since the length of input features must be a fixed number H, if the size of f is smaller than the required length, we pad it with zeros. Specifically, the input feature is formulated as:

$$x = [f_1, f_2, \cdots, f_{H-3}, \log n, \log d, \log N]. \tag{6}$$

We apply the logarithm operation on n, d, and N to mitigate the skewness of input features.

3.2.2 Estimator Ranking. The ranking paradigm has demonstrated substantial superiority in solving item prioritization tasks [14, 37, 44, 55, 59]. We adapt SOTA ranking techniques in estimator selection, which encompasses the elaborated construction of ranking labels and the training of the learned ranker.

Ranking Label Construction. Constructing the ranking label is a complex task [56] due to the potentially infinite number of label values, even when the ranking order is fixed. For instance, given two estimators e_1 and e_2, where e_1 has a lower q-error. We denote $e_1 \succ e_2$, indicating that e_1 is better than e_2. Let y_1 and y_2 be the ranking labels. Any values of y_1 and y_2 that satisfy the condition $y_1 > y_2$ are eligible to serve as ranking labels, because the ranking paradigm focuses solely on the relative orders rather than specific values.

There are no predefined estimator ranking labels available, and the labels in the estimator selection context require the following attributes: (1) A *higher* value of the label reflects the *higher* priority, while also indicating a *lower* q-error in NDV estimation; (2) The value domain of the label needs to be constrained, as applying ranking techniques usually involves exponentiation operation on the label values [37]. Unconstrained values could potentially lead to computational overflow; (3) The label needs to distinctly differentiate between overestimated and underestimated estimators, as required by our designed objectives.

To this end, we propose an efficient ranking label construction strategy. Firstly, we use the ranking position to constrain the value of labels to be no greater than m. Then, we construct the labels by reversing the ranking positions based on the lowest overestimated or underestimated q-error. Finally, we mask the overestimated or underestimated estimators to differentiate them. Through these steps, the constructed labels satisfy the required attributes.

Algorithm 1: Overestimation ranking label construction.

Input: $\hat{\mathcal{D}}, D$
Output: y^{over}

```
UnderEstSet ← ∅; i ← 1; D̂_max ← max_{1≤j≤m} D̂_j;
for i ← 1; i ≤ m; i ← i + 1 do
    if D̂_i ≤ D then
        UnderEstSet ← i ;
        D̂_i ← D̂_i + D̂_max;
    end
end
for i ← 1; i ≤ m; i ← i + 1 do
    y_i^over ← m − π_{D̂_i};
end
for i in UnderEstSet do
    y_i^over ← 0; // mask the underestimate estimators
end
return y^over;
```

To prioritize the estimators with low overestimation q-errors, the process of constructing ranking labels y^{over} is shown in Algorithm 1. Specifically, we first record the maximum estimated result among all base estimators as $\hat{\mathcal{D}}_{\max}$. Then, to separate underestimations from overestimations, we add $\hat{\mathcal{D}}_{\max}$ to the result of each underestimated estimator to ensure that their estimations are higher than those of any overestimated estimators. Next, we obtain the ranking position π of each base estimator by sorting the estimation results, where $\pi_{\hat{\mathcal{D}}} = \text{argsort}(\hat{\mathcal{D}})$. Specifically, the argsort operation yields the indices required to sort the data in ascending order, and $\pi_{\hat{\mathcal{D}}_i}$ represents the index of $\hat{\mathcal{D}}_i$. In the context of overestimation, the transformed estimation is considered better when it is smaller. Therefore, the estimator with better overestimation performance has a higher value of the ranking label y_i^{over} by reversing its ranking position. Finally, we mask the underestimated estimators by setting their ranking labels as zero to differentiate them from the overestimated ones.

The process of constructing labels y^{under} is similar to that of y^{over}, and we omit it for conciseness.

Train the Learned Ranker. We use a multi-layer perceptron (MLP) as the backbone of our ranking model: $\mathcal{S} = \text{MLP}(x)$, where $\mathcal{S} \in \mathbb{R}^m$ represents the ranking scores of the estimators. The higher score indicates the higher priority of the corresponding estimator. We adapt the SOTA ranking techniques [14, 44, 55, 59] into estimator selection to train the ranking models. In short, the learned ranker can be trained by:

$$\begin{aligned} \mathcal{L}_{\text{rank}}(\mathcal{S}, y) &= -\sum_{i=1}^{m} \frac{2^{y_i} - 1}{\log_2(1 + \pi(i))}, \\ \pi(i) &= 1 + \sum_{j, j \neq i}^{m} \frac{1}{1 + e^{-\alpha(\mathcal{S}_j - \mathcal{S}_i)}}, \end{aligned} \tag{7}$$

where α is the hyperparameter in the training framework, $\mathcal{L}_{\text{rank}}$ is the estimator ranking loss function, and y is the ranking label. The learned ranker can similarly achieve different objectives by assigning the labels constructed by Algorithm 1.

3.2.3 Leading Estimator Selection. In this subsection, we show the training objectives of complementary estimator selection of this component.

Overestimated Estimators Selection. We construct the training labels y^{over} according to Algorithm 1, and we use an MLP that has two hidden layers with 128 and 64 dimensions to compute the ranking scores $\mathcal{S}^{\text{over}}$. The loss function for overestimated estimator selection is:

$$\mathcal{L}_{\text{over}} = \frac{1}{\mathcal{N}} \sum_{i=1}^{\mathcal{N}} \mathcal{L}_{\text{rank}}(\mathcal{S}_i^{\text{over}}, y_i^{\text{over}}), \tag{8}$$

where $\mathcal{N}$ is the number of training samples, and $\mathcal{L}_{\text{rank}}$ is the loss function defined in Equation (7).

Underestimated Estimators Selection. Similarly, we use another MLP with the identical model architecture to compute $\mathcal{S}^{\text{under}}$ and the training labels y^{under}. The loss function for underestimated estimator selection is:

$$\mathcal{L}_{\text{under}} = \frac{1}{\mathcal{N}} \sum_{i=1}^{\mathcal{N}} \mathcal{L}_{\text{rank}}(\mathcal{S}_i^{\text{under}}, y_i^{\text{under}}). \tag{9}$$

3.3 Estimator Fusion

3.3.1 Feature Engineering. The features described in Section 3.2.1 are also used in this component. Moreover, we incorporate the estimated results of the chosen base estimators as the additional features.

Specifically, the leading estimator selection component provides the priority scores $\mathcal{S}^{\text{over}}$ and $\mathcal{S}^{\text{under}}$ of the base estimators. We select k estimators with the top-k highest scores for both both overestimated and underestimated estimators. The chosen estimators are denoted as $\mathcal{I}^{\text{over}} = \text{argmax}_k \mathcal{S}^{\text{over}}$ and $\mathcal{I}^{\text{under}} = \text{argmax}_k \mathcal{S}^{\text{under}}$. The features are defined as $x' = [x, \hat{\mathcal{D}}|_{\mathcal{I}^{\text{over}}}, \hat{\mathcal{D}}|_{\mathcal{I}^{\text{under}}}], x' \in \mathbb{R}^{H+2k}$.

3.3.2 Estimator Fusion. We use an MLP to compute the weights for the chosen base estimators: $\Lambda = \text{MLP}(x')$, where $\Lambda \in \mathbb{R}^{2k}$ is the weight vector concatenated by two k-dimentional vectors corresponding to the chosen leading estimators. Λ is not fixed but depends on sample data and selected estimators. To restrict the output of estimated NDV, we limit $\sum_{j=1}^{2k} \Lambda_j = 1$ and estimate NDV by exploiting the base estimators:

$$\log \hat{D} = \sum_{j=1}^{k} (\Lambda_j \cdot \log \hat{\mathcal{D}}|_{\mathcal{I}_j^{\text{over}}} + \Lambda_{k+j} \cdot \log \hat{\mathcal{D}}|_{\mathcal{I}_j^{\text{under}}}), \tag{10}$$

where $\hat{\mathcal{D}}|_{\mathcal{I}_j^{\text{over}}}$ is the estimated NDV of the j-th chosen overestimated base estimator, and the output estimated NDV $\hat{D} = e^{\log \hat{D}}$. The logarithm is applied to limit the estimation of base estimators so that they do not exceed the range of a 32-bit floating-point number, thus preventing potential impacts on model training. The learned model solely generates the weight vector, different from previous works that directly estimate NDV.

3.3.3 Fusion Component Training. Denote D_i as the ground truth NDV of the i-th training sample, and the learned model can be trained using:

$$\mathcal{L}_{\text{est}} = \frac{1}{\mathcal{N}} \sum_{i=1}^{\mathcal{N}} (\log \hat{D}_i - \log D_i)^2 + \lambda ||W||_2, \tag{11}$$

where we apply L_2 regularization on model parameters W for better generalization. The regularization parameter λ is tuned based on the validation loss.

4 EXPERIMENTS

4.1 Experimental Setup

Dataset Selection. A fundamental test case for evaluating estimators is applying them to an individual data column, where the underlying data distribution is manifested. Therefore, the diversity of evaluation scenarios hinges on the variety of individual columns with various data distributions, rather than the quantity of data tuples in the data column. Most traditional statistical estimators [15, 18–21, 27, 28, 28, 42, 47–52] are tested on a few synthetic columns that satisfy their heuristics and assumptions. Recent studies of learned estimators [35, 57] primarily evaluate the performance on manually crafted standard data distribution (e.g. Zipfian and Poisson) and a limited number of columns of some open-source datasets (SSB [43], Campaign [1], NCVR [2], et. al.). In conclusion, the previous works used limited data distributions and data columns to evaluate the performance of NDV estimators, which may lead to insufficient evaluations.

In recent years, the research community has proposed open-source large-scale tabular datasets [24, 32]. TabLib [24], which collects 627M individual tables totaling 69 TiB, is the largest one all over the world. In this paper, we select TabLib sample version [11], which contains 0.1% of the full version (69 GB), as our dataset for evaluation. TabLib exhaustively contains tabular data from the real world (GitHub [6] and Common Crawl [5]) with diverse domains, which can better reflect the data distribution in practice.

Data Preprocess. TabLib sample version contains 77 parquet files, we remove three of them (2d7d54b8, 8e1450ee, and dc0e820c) for the memory issue. Then we divide the remaining 74 files into train, test, and validation sets to avoid potential data leaks. Each parquet file contains thousands of tables, where for each column we independently sample 1% of data tuples uniformly to construct evaluation cases. Previous works solely focus on large tables with millions of rows [34, 57]. We expand the evaluation cases by assessing the methods on table sizes ranging from tens of thousands to millions of rows. The statistics of preprocessed data are shown in Table 2, where "# Columns" represents the number of Train/Validation/Test cases used for evaluation.

Evaluation Criteria. To comprehensively evaluate the performance of NDV estimators, we use mean q-error (as defined in Equation (4)) and the distribution (50%, 75%, 90%, 95%, and 99% quantiles) of q-error on a large data volume.

Table 2: Statistics of preprocessed TabLib sample data.

	Train	Validation	Test
# Columns	89,283	30,418	25,159

Implementation Details. We implement our model in PyTorch, and the implementation details are shown as follows: the optimizer is Adam [33] with an initial learning rate of 0.001, α is 1, β is 0.5, the number of input features H is 100, the number of selected leading estimators k is 2. The training epoch is 100, we save the model by 99% quantile of q-error on the validation set and report the performance on the test set. All the experiments in this paper are conducted on an NVIDIA A100 GPU.

Baseline Models. In our evaluation, we include statistical estimators, hybrid estimators that integrate statistical ones, and learned estimators as our baselines.

Statistical estimators (base estimators). There are many statistical estimators, we select representative ones as our baselines as well as our base estimators.

- Goodman [27] is the seminal work in NDV estimation, and we use the expression in Equation (1).
- GEE [20] provides a theoretical lower bound of ratio error and it uses geometric mean as scale factor: $D_{\text{GEE}} = \sqrt{N/n} f_1 + \sum_{j=2}^{n} f_j$.
- Error Bound (EB) [21] is proposed to estimate NDV in sampling-based histogram construction, and $D_{\text{EB}} = \sqrt{N/n} f_1^+ + \sum_{j=2}^{n} f_j$, $f_1^+ = \max(1, f_1)$.
- Chao [18, 42] assumes the size of C is infinity. We use the expression in Equation (2). If f_2 is zero, we will return d.
- Shlosser [48] is based on the assumption that the frequency profile of sample data is approximately the frequency profile of the original column. $D_{\text{Shlosser}} = d + \frac{f_1 \sum_{i=1}^{n} (1-r)^i f_i}{\sum_{i=1}^{n} i r (1-r)^{i-1} f_i}$.
- ChaoLee [19] adds another estimator in Chao to treat data skew, we refer to [7] to implement it.
- Jackknife [16, 17] assumes d_n be the NDV of the sample and numbers the tuples from 1 to n in the sample data. Denote $d_{n-1}(k), 1 \le k \le n$, $d_{n-1}(k) = d_n - 1$ if the attribute value for tuple k is unique; otherwise $d_{n-1}(k) = d_n - 1$. The first-order Jackknife estimator is: $D_{\text{Jackknife}} = d_n - (n-1)(d_{n-1} - d_n)$.
- Sichel [49–51] estimator needs to solve non-linear equations, as shown in Equation (3).
- Bootstrap [52]: $D_{\text{Boot}} = d + \sum_{j:n_j>0} (1 - n_j/n)^n$. It may perform worse when D is large and n is small because $D_{\text{Boot}} \le 2d$.
- Horvitz-Thompson (HT) [47] has a sophisticated expression, it defines $h_n(x) = \frac{\Gamma(N-x+1)\Gamma(N-n+1)}{\Gamma(N-n-x+1)\Gamma(N+1)}$, where Γ is the gamma function, and $D_{HT} = \sum_{j:n_j>0} \frac{1}{1-h_n(\hat{N}_j)}$, $\hat{N}_j = N(n_j/n)$.
- Method of Movement (MoM) [15] has three versions. MoM v1 assumes the frequencies are equal ($N_1 = N_2 = \ldots = N_D$) and an infinite population, it needs to solve the equation: $d = D_{\text{MoM1}}(1 - e^{-n/D_{\text{MoM1}}})$. MoM v2 assumes the population size is finite, and the estimator is $d = D_{\text{MoM2}}(1 - h_n(N/D_{\text{MoM2}}))$. MoM v3 assumes the frequencies are unequal and it has a sophisticated expression, we refer to [7] to implement it.
- Smoothed Jackknife (SJ) [28]. $D_{\text{SJ}} = d_n - K((d_{n-1} - d_n))$, there is a extremely sophisticated approximation expression for K, we omit its expression in the paper and refer to [7] to implement it.

Hybird estimators. Existing hybrid estimators are commonly constructed by using SJ [28], GEE [20], Shlosser [48] estimators. Specifically, they use χ^2_{n-1} test [4] to pick estimators. Define

$$u = \sum_{j,n_j>0} \left(\frac{(n_j - \bar{n})^2}{\bar{n}}\right), \bar{n} = \frac{n}{d}. \tag{12}$$

- HYBSkew [28] uses SJ estimator if $u \le \chi^2_{n-1,0.975}$, otherwise it takes Shlosser estimator.
- HYBGEE [20] uses SJ estimator if $u \le \chi^2_{n-1,0.975}$, otherwise it takes GEE estimator.

Learned estimators. In addition, we compare our method with the open-sourced SOTA learned estimator LS [57]. We consider three variants of LS for a fair comparison:

- $\text{LS}_{\text{general}}$: since LS claimed can directly apply to any data distribution [57], we use its open-source checkpoint as a baseline.
- $\text{LS}_{\text{scratch}}$: we train LS from scratch on the same training set as AdaNDV.
- LS_{FT}: we fine-tune (FT) $\text{LS}_{\text{general}}$ as described in [57] on the same training set as AdaNDV.

Besides, we construct two learned estimator baselines as follows:

- Select-Optimal (SO): it selects one optimal base estimator using an MLP with the same architecture as the estimator selection model in AdaNDV, designed to accomplish the objective as Hypo-optimal.
- Learnable Ensemble (LE): it integrates the results of all fourteen base estimators by a learnable weighted sum, where the number of parameters is equal to the number of base estimators.

4.2 Statistical Estimators Analysis

In this section, we present a detailed analysis of the performance of statistical and hybrid estimators in large-scale real-world scenarios.

Overall Performance. The performance is shown in Table 3. According to the results, we can draw the following conclusions:

- There is no single traditional estimator that always outperforms the other ones across all the evaluated metrics.
- The majority of traditional estimators exhibit subpar performance across most metrics. In the most adverse scenarios, the q-error of the Goodman estimator even surpasses the upper limit that a 32-bit floating-point value can represent due to factorial operations in Equation (1).
- Using a comprehensive assessment of the performance of NDV estimators is desired. For example, although the Shlosser estimator has a mean q-error that is higher but close to that of EB, its 50% quantile of q-error is considerably lower, while its 99% quantile of q-error is significantly higher. A single metric can not reflect the performance of an estimator.
- HYBSkew does not outperform Shlosser, and HYBGEE does not surpass GEE. On the one hand, they rarely select SJ, indicating hybrid estimators may mitigate poor results to some extent. On the other hand, the performance of traditional hybrid estimators is limited by the selected single estimator.

Table 3: Mean and quantiles of q-error of baselines and our method, where ∞ indicates that the number exceeds the representation limits of a 32-bit floating-point type. Each best-performing metric is emphasized in boldface.

Estimator	Mean	50%	75%	90%	95%	99%
Goodman	∞	4.14	37.91	100.78	1.11e11	∞
GEE	3.71	1.97	3.86	9.91	10.10	11.70
EB	3.98	2.62	6.00	9.98	10.12	11.00
Chao	21.98	1.85	6.50	99.99	100.04	100.19
Shlosser	4.25	1.80	4.53	10.41	15.14	25.14
ChaoLee	23.28	4.35	29.39	96.00	99.36	100.51
Jackknife	12.60	2.80	16.24	49.04	50.38	51.71
Sichel	152.93	2.49	99.70	365.10	1.03e3	2.20e3
MoM v1	2.51e4	2.00	6.60	22.70	66.51	1.11e6
MoM v2	8.60	3.14	9.96	18.84	30.61	86.09
MoM v3	4.29e3	5.95	61.92	818.18	3.30e3	4.30e4
Bootstrap	70.01	10.99	47.91	95.42	224.09	1.25e3
HT	2.83e3	41.56	354.42	5.07e3	1.03e4	4.41e5
SJ	231.67	2.17	8.12	34.03	90.25	1.81e3
HYBSkew	4.25	1.80	4.53	10.41	15.14	25.14
HYBGEE	3.71	1.97	3.86	9.91	10.10	11.70
$LS_{general}$	2.24	1.72	2.28	3.20	4.11	10.46
$LS_{scratch}$	1.91	1.36	1.80	2.57	3.53	10.80
LS_{FT}	1.96	1.39	1.86	2.64	3.63	11.44
SO	∞	1.29	2.08	3.29	4.42	13.03
LE	2.30	1.71	2.22	3.44	4.63	11.88
AdaNDV	**1.62**	**1.22**	**1.60**	**2.34**	**3.24**	**6.79**

Intrinsic Weaknesses. The intrinsic weaknesses of statistical estimators are evident and have been discussed in previous works [34, 57]: the assumptions and conditions of traditional estimators are infrequently satisfied in practice.

We expand the number of evaluation cases by several orders of magnitude to provide a more comprehensive assessment that has not been investigated before, and we can conclude that the weaknesses exposed by prior studies remain valid.

Undiscovered Strengths. However, we argue that the strengths of statistical estimators are significantly eclipsed by their weaknesses. Although real-world data distributions often fail to meet their presupposed conditions, it is possible to identify estimators that deliver the estimated result with an acceptable level of q-error. Our argument is supported by the Hypo-optimal estimator shown in Table 1. Meanwhile, *it suggests that accurately selecting appropriate estimators could significantly reduce q-error.*

4.3 Results of Learned Estimators

Effectiveness. The overall performance of learned estimators is shown in Table 3, and we observe the following findings:

- *Performance of the proposed learned estimator.* AdaNDV significantly outperforms all estimators across all the metrics. These results substantially demonstrate the superiority of AdaNDV.

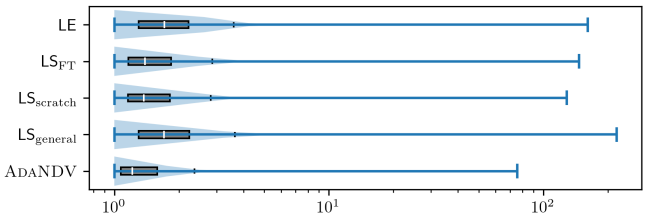


Figure 4: Error distribution of learned estimators on the test set. The violin plot is in blue. The boxplot is in black, the gray box contains 50% of data points, and the white line in the gray box represents the median. We exclude the SO estimator due to its extremely large mean error.

- *Performance of SOTA learned estimators.* $LS_{general}$ consistently outperforms the statistical estimators across all the metrics, which exhibits the advantages of learned estimators that can adapt to data shifting over statistical estimators. In addition, $LS_{scratch}$ and LS_{FT} have better performance compared to $LS_{general}$ in general. However, by looking at the different quantiles of q-error, training or fine-tuning may decline the robustness, where they perform worse than $LS_{general}$ for 99% quantile of q-error.
- *Performance of constructed learned estimators.* The performance of the SO and LE estimators reveals the necessity of investigating the historically overlooked issues in NDV estimation. Firstly, selecting one optimal estimator by a learned model can outperform individual estimators in most scenarios, but it is challenging to achieve high accuracy, which can lead to worse cases. Besides, the LE estimator outperforms individual statistical ones, showcasing the potential of estimator fusion.
- *Error distribution discussion.* In Table 3, we also observe that AdaNDV exhibits a more robust error distribution: about 90% of the test cases show a q-error below 2.30, and approximately 99% of test cases exhibit the q-error do not exceed 7. Moreover, we depict the boxplot [3] and violin plot [12] of q-error distributions of learned estimators in Figure 4, where we omit the SO estimator for its overflow issue. From the figure, we observe that the q-error distribution for the AdaNDV shows a higher concentration around values closer to the minimum error, indicating its advantages. Besides, the maximal q-error of AdaNDV is much smaller than other learned estimators, representing better performance in the worst cases.

Efficiency. The efficiency of an estimator determines its practicality. Thus we also evaluate the efficiency of AdaNDV and the learned estimators in terms of both time and space consumption. The computing overhead of AdaNDV involves the neural networks and the selected base estimators. Since some base estimators involve solving non-linear equations, it is not easy to provide a rigorous time complexity analysis. Therefore, for simplicity, we record the end-to-end latency of the training and testing stages for them. Specifically, in the training stage, we record the execution time of the base estimators as well as the time reaching convergence for the learned estimators. In the inference stage, we capture the time required for processing all test cases. In addition, we compare the learnable parameters between the learned estimators to show their space efficiency. We exclude $LS_{general}$ in this comparison since it is

Table 4: Efficiency comparison of the methods.

	$LS_{scratch}$	LS_{FT}	AdaNDV	SO	LE
Train (s)	929	2,963	712	1,261	1,337
Inference (s)	25.42	25.42	51.44	32.79	84.85
# Params	62,435	62,435	55,328	22,294	14

pre-trained and has the same inference efficiency and parameters as its variants. The time and space consumptions are shown in Table 4, and we derive the following conclusions.

- *Training efficiency of learned estimators.* We observe that LS_{FT} is the most time-consuming method in that it requires about three times of epochs than $LS_{scratch}$ to converge. One possible reason may be that adapting the general model to the training domain requires more time than learning from scratch since the fine-tuning process may need to maintain the features learned in the pre-training stage. AdaNDV requires minimal time for convergence. The SO and LE estimators have longer training time than AdaNDV and $LS_{scratch}$, indicating their objectives are more difficult to converge.
- *Inference efficiency.* The inference overhead of baseline learned estimators solely involves a neural network model inference and the total time on the test set (25, 159 samples) is 25.42s, the average inference time of a test case is about **1ms**. In contrast, AdaNDV consists of three neural models and they need 27.46s to finish the inference on the test set. Besides, AdaNDV additionally requires the base estimator computation which needs 23.97s. In total, AdaNDV spends 51.44s to finish the inference on all the test samples with an average estimation time of about **2ms**. The elaborated components bring performance improvement and efficiency decline in AdaNDV. For the constructed estimators, the SO estimator requires a selected base estimator employment for each test case, resulting in a longer inference time compared to LS models. LE needs the estimation results of all base estimators, leading to the longest inference time among the learned estimators.
- *Space efficiency comparison.* We observe that the variants of LS possess approximately 12.85% more parameters compared to AdaNDV. Its advantage can likely be attributed to the elaborated integration of base estimators. This strategic utilization enhances the performance of learned estimator AdaNDV with less representation ability (number of learnable parameters). For the constructed estimators, the SO estimator has fewer parameters than AdaNDV and LS models, since its architecture is the same as a component in AdaNDV. LE solely has 14 learnable parameters, making it the most lightweight among the learned estimators, which may indicate the potential for lightweight optimization in AdaNDV.

4.4 Ablation Study

The outstanding performance of AdaNDV mainly stems from the effective collaboration between its leading estimator selection and the fusion-based estimation. To investigate the contributions of each module, we develop several variants of AdaNDV. The symbol "w/o" indicates removing the component or strategy from our method.

Table 5: Ablation study, where "w/o" indicates removing a component from AdaNDV.

Estimator	Mean	50%	75%	90%	95%	99%
AdaNDV	**1.62**	**1.22**	**1.60**	**2.34**	**3.24**	**6.79**
w/o select	2.30	1.71	2.22	3.44	4.63	11.88
w/o fusion	2.42	1.35	2.01	3.31	4.75	19.30
w/o log	2.72e7	41.54	671.06	2.67e3	5.87e4	1.00e10
w/o base	1.67	1.27	1.66	2.39	3.34	7.42
w/o over	2.20	1.37	1.92	3.10	4.27	11.97
w/o under	2.49	1.82	2.88	5.50	6.00	11.00
w/o comp	2.05	1.26	1.76	2.68	3.81	11.68

Primary Components. We individually eliminate each of the two principal constituents within AdaNDV. Firstly, we drop the leading estimator selection component to deliberately make the model use all base estimators to estimate NDV, the variant is named "**w/o select**". We then remove the estimator fusion component to select a single estimator by the ranking paradigm from the base estimators to present NDV, the variant is named "**w/o fusion**". The performances of the two variants are shown in Table 5. We can observe that the performance of AdaNDV significantly declines across all the metrics if we remove one of the two components. Based on the observation, we can derive the following conclusions.

- Directly estimating NDV by fusing all base estimators may make the task difficult because it has to leverage poor-performing estimators. Besides, the performance of "w/o select" surpasses that of LE, highlighting the advantages of adaptively adjusting the weights of fused estimators for different test cases.
- The performance of "w/o fusion" shows that selecting a single estimator with high accuracy is challenging because of the representation ability (sparse features and the MLP architecture) of the model. On the contrary, it performs better than the SO estimator in most metrics and can alleviate the ∞ of mean q-error. It demonstrates the effectiveness of the ranking paradigm ("w/o fusion") over the classification paradigm (SO), and further investigation will be conducted in Section 4.6.
- Each component has a significant contribution to AdaNDV, and the combination of two components leads to the superiority of AdaNDV.

Detailed Strategies. We further study the other strategies utilized in AdaNDV and the performance of dropping each strategy is shown in Table 5. Firstly, we remove the log scale training technique by transforming Equation (10) to $\hat{D} = \prod_{j=1}^{k} (\hat{\mathcal{D}}|_{\mathcal{I}_j^{\text{over}}}^{\Lambda_j} \cdot \hat{\mathcal{D}}|_{\mathcal{I}_j^{\text{under}}}^{\Lambda_{k+j}})$, the variant is denoted as "**w/o log**". The performance of "w/o log" significantly declined, which indicates the effectiveness of logarithm operation.

In the estimator fusion component, we add the estimated NDV of the selected leading estimators as features. We remove the features of estimated results, and the variant is named "**w/o base**". Its performance slightly declined on all metrics but performs better than all baselines, which indicates that taking the estimated results

as input features may make the model aware of the input value domain and then improve the weights allocation to get more accurate estimations.

Finally, we study whether discerning overestimation and underestimation estimators work. We remove the two kinds of leading estimators and name the two variants as "**w/o over**" and "**w/o under**", respectively. We can observe the two variants significantly decrease on all metrics and they cannot beat baseline learned estimators. The performance illustrates that individually utilizing the two kinds of leading estimators can not obtain satisfactory results but the combination of them can bring a significant approximation to the ground truth NDV. In addition, we construct a new variant "**w/o comp**" that directly selects $2k$ estimators without deliberately utilizing the overestimation-underestimation complementary error correction strategy. It outperforms all variants in most metrics but is consistently worse than AdaNDV. On the one hand, it illustrates the benefit of our proposed complementary error correction strategy. On the other hand, although not intentionally distinguished, it is rare for the selected estimators to be exclusively overestimated or underestimated. This further demonstrates the effectiveness of our strategy combined with the experimental conclusions of "w/o over" and "w/o under".

4.5 Impact of Hyperparameters

There are three hyperparameters in AdaNDV: α, β, k, and we investigate the sensitivity of the model. We demonstrate the performance of mean and 99% percentile of q-error in Figure 5 to show the impact of hyperparameters.

Effect of α in the Ranking Model. We can observe different α values lead to some fluctuations in the performance, but AdaNDV is not sensitive to different values of α. We set α to 1 because it achieves the best performance in most metrics. Extensively discussing the effect of this knob is out of the scope of this paper, refer to [14, 55] for more details.

Effect of Multi-objective Optimization Knob β. As shown in Figure 5, we observe that the performance in mean q-error remains unchanged across different values of the knob β. However, there is a notable difference in 99% q-error. For each experimented value of β, AdaNDV exhibits relatively stable performance. The knob β balances the estimator ranking and NDV prediction tasks, we set β as 0.5 in this paper because it achieves better results in most metrics than in other settings.

The Number of Leading Estimators k. The number of selected leading estimators directly affects the input of the estimation fusion component. With the variation of k, the performance of the model does not improve beyond that achieved with $k = 2$ on the mean and the 99% percentile of q-error. This finding illustrates that selecting more estimators does not consistently benefit the fusion component, as demonstrated in the analysis of the LE estimator and the variant of AdaNDV "**w/o select**". Additionally, increasing k involves incorporating more base estimators during the inference stage, which introduces additional computational overhead, as shown in Table 4. Considering both effectiveness and efficiency, we set k as 2.

Table 6: Performance comparison of different variants of AdaNDV.

	Mean	50%	75%	90%	99%
AdaNDV	1.62	1.22	1.60	2.34	6.79
AdaNDV(C)	1.86	1.34	1.98	3.04	8.42
AdaNDV(base-EB)	1.64	1.24	1.62	2.39	7.11
AdaNDV(base+$LS_{general}$)	1.62	1.21	1.58	2.33	7.18
AdaNDV(Hypo)	1.18	1.10	1.24	1.43	2.20

4.6 Flexibility Discussion

Selected Estimator Discussion. Since we leverage the property of overestimation and underestimation, we demonstrate the overestimation and underestimation ratios of base estimators in Figure 6a. We count the estimators selected by the overestimated and underestimated selection models of AdaNDV, and the proportion of each estimator is shown in Figure 6b. We can observe the following findings based on the results.

- Base estimators can hardly derive the ground truth, as they always either overestimate or underestimate. Besides, no estimators can consistently overestimate or underestimate, but certain estimators tend to underestimate on the dataset.
- EB is the most frequently selected by both models, ChaoLee has not been selected by either model, and some estimators are exclusively selected by a single model. It suggests that most estimators are beneficial and the overestimation-underestimation complementary perspective is utility.
- Figure 1 illustrates that all base estimators can achieve the lowest q-error in certain scenarios compared to others, but some are selected infrequently. This highlights that collecting the set of base estimators may be an open research question.

Precisions of Estimator Selection. We have constructed the SO estimator as our baseline by categorizing the estimators into two classes for training: the optimal overestimated or underestimated estimator and otherwise. To further investigate the differences between the ranking and classification paradigms, we develop another variant of AdaNDV by transforming the SO estimator as the estimator selection component in AdaNDV, and the method is named AdaNDV(C). The performance is shown in Table 6, and we can derive the following conclusion. On the one hand, AdaNDV(C) consistently outperforms all baseline estimators, demonstrating the necessity of treating the selection dilemma problem. On the other hand, AdaNDV surpasses AdaNDV(C) on all metrics. The two variants have identical estimator fusion components, indicating that selected estimators directly affect the ultimate estimations.

We use the metric *Precision@K* (P@K) to evaluate the accuracy of estimator selection, representing the partition of the optimal estimator(s) amongst the top-K estimators, and the results are shown in Figure 7a. We can observe the model trained in the ranking paradigm exhibits substantially higher P@1 and P@2 than that trained in the classification paradigm. The sensitive analysis results in Figure 5 show that the ultimate performance is closely related to the top estimators and we set k as 2 for AdaNDV, so P@1 and

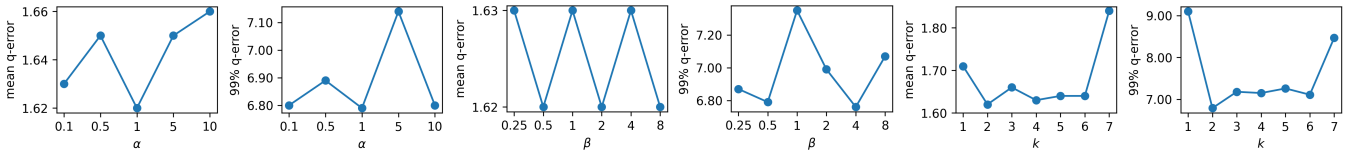
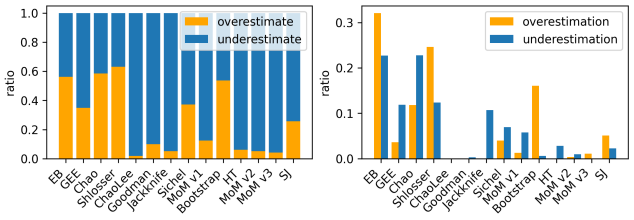
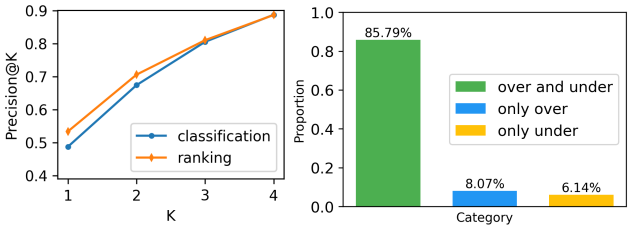


Figure 5: Performance on mean and 99% percentile of q-error of AdaNDV with different hyperparameters.



(a) Performance of base estimators. **(b) Proportion of selected estimators.**

Figure 6: Illustration of overestimation and underestimation properties.



(a) Selection precisions. **(b) Type of selected estimators.**

Figure 7: Analysis of estimator selection.

P@2 of leading estimator selection directly affect the performance of our exploitation.

Selected Estimators for Fusion. We summarize the fusion scenarios of over/under estimation properties of the $2k$ ($k = 2$) selected estimators in Figure 7b. The term "over and under" indicates that both overestimated and underestimated results exist among the $2k$ estimators, meaning their weighted sum can encompass the ground truth. Conversely, "only over" and "only under" refer to the selected estimators that exhibit only overestimation or underestimation, respectively. In most test cases, AdaNDV chose both overestimated and underestimated results, leveraging the properties of overestimation and underestimation to reduce errors in the fusion process. In addition, the P@2 for estimator selection is 71%, as shown in Figure 7a, indicating that the selected overestimating and underestimating estimators are also among the top estimators. This demonstrates the effectiveness of the top-k selection strategy of AdaNDV.

Adaptivity of Base Estimators. AdaNDV contains fourteen traditional base estimators and it is adaptive to add or remove base estimators. In addition, our method is available for learned estimators. To show the flexibility of base estimators in AdaNDV, we respectively remove the EB estimator and add the $\text{LS}_{\text{general}}$ estimator in AdaNDV. The two variants are respectively named AdaNDV(base-EB) and AdaNDV(base+$\text{LS}_{\text{general}}$), and their performance is demonstrated in Table 6. We can derive the following conclusions based on the results: (1) AdaNDV(base-EB) and AdaNDV(base+$\text{LS}_{\text{general}}$) consistently outperform the individual estimators within their base estimator sets, which indicates the effectiveness of our method remains in changing the base estimators. (2) Removing EB results in a consistent performance decline, indicating that estimators that are frequently selected, such as EB, play a crucial role in maintaining the overall effectiveness of the model. Adding $\text{LS}_{\text{general}}$ will not consistently improve the performance: some metrics have decreased, while others have increased. The possible reason is that the leading estimator is relative, based on the current set of base estimators. Changing the base estimators will affect estimator selection.

Figure 7a has shown that accurately identifying the optimal base estimator from the set of base estimators is a challenging task. In the subsequent study, we explore the potential impact on our estimation performance by considering the hypothetical scenario in which this challenge is addressed.

Upper Bound of Estimator Selection. To further investigate the upper bound of the estimator selection component, we suppose we can exploit the optimal estimators, where P@1 is 100%. This variant of the model is denoted as AdaNDV(Hypo), and its performance is shown in Table 6. The results can derive the following findings: (1) If we can accurately select the optimal estimator with the lowest overestimated and underestimated q-error, the performance of AdaNDV will significantly improve. It shows the promising performance upper bound of our AdaNDV framework. (2) Compared to the Hypo-optimal estimator in Table 1, the performance of AdaNDV(Hypo) is better in most metrics except for 50% quantile q-error. It further demonstrates the effectiveness of AdaNDV and indicates that our proposed complementary estimator exploitation strategy is beneficial.

The P@1 of AdaNDV stands at 53%, indicating that there is significant scope for enhancement within our proposed method. It is feasible to explore the estimator selection methods beyond the classification and ranking paradigms in the future.

4.7 Further Evaluation

In this section, we show the performance of our method under more evaluation scenarios.

Performance on Artificial Distributions. AdaNDV is trained and evaluated on the data distributions from the real world, as illustrated in Section 4.1. However, most previous works are evaluated on artificial data distributions, so it is not clear will baseline learned methods outperform AdaNDV with artificial data distribution. We conduct experiments on Zipfian distribution with skew factors (s)

Table 7: Q-error of learned estimators when sampling 1% data from Zipfian distribution with skew factors (s) of {1.2, 1.5, 2.0} with column size (N) of 1e5 and 1e6.

N	1e5			1e6		
s	1.2	1.5	2.0	1.2	1.5	2.0
$LS_{general}$	5.14	2.93	2.47	3.76	5.94	2.5
$LS_{scratch}$	**1.27**	1.47	1.36	1.53	**2.67**	**1.06**
LS_{FT}	1.62	1.56	**1.28**	1.23	2.93	1.15
AdaNDV	1.77	**1.20**	2.36	**1.13**	5.86	1.75

1.2, 1.5, and 2.0, consistent with previous work [35]. We freeze the parameters of $LS_{scratch}$, LS_{FT}, and AdaNDV trained on the TabLib training set, and evaluate them when sampling 1% of data from Zipfian distributions with data size of 1e5 and 1e6. The results are shown in Table 7 and we can draw the following conclusions. No single estimator achieves optimal results under the Zipfian distributions with different skew factors and AdaNDV consistently beats $LS_{general}$, demonstrating that our method does not fail on standard artificial distributions. Besides, $LS_{general}$ is pre-trained on an artificial dataset containing 7.2×10^5 data points [57], in which the original columns follow specific data distributions, but it processes the worst performance in each metric. This demonstrates the effectiveness of training models on real-world data.

Performance under Different Sampling Rates. We depict the performance of 75% quantile q-error of AdaNDV, base estimators, and the representative learned baseline $LS_{general}$ under different sampling rates in Figure 8. AdaNDV shows consistent performance improvement with increasing sample rates, while some base estimators decline in performance, possibly due to practical scenarios not aligning with their assumptions. Besides, the advantages of AdaNDV persist across different sample rates, demonstrating that it is not sensitive to the variations of base estimators.

5 RELATED WORKS

5.1 Sketch-based NDV Estimation

Sketch-based NDV estimation [25, 26, 31] represents an orthogonal approach to sampling-based NDV estimation. This line of research requires scanning all the data to maintain a memory-efficient sketch for NDV estimation, which may bring an unaffordable overhead [35]. Furthermore, real-world databases may have data access restrictions, which makes sketch-based NDV estimation not applicable in many applications.

5.2 Sampling-based NDV Estimation

Traditional NDV Estimators. Traditional methods explore statistical techniques to summarize heuristic rules to estimate NDV and they have been studied for over seven decades in Biology [15, 53, 54], Statistics [18, 27], Networks [23, 40], and Databases [8–10]. Representative traditional estimators make different assumptions, for example, they assume infinity population size [15], certain data distribution [15, 39], and data skewness [20, 28]. Based on the assumptions, numerous estimators have been proposed to utilize the frequency profile of sample data to build linear polinomials [20, 21, 27], non-linear polynomials [16–19, 42, 47, 48], and solving non-linear

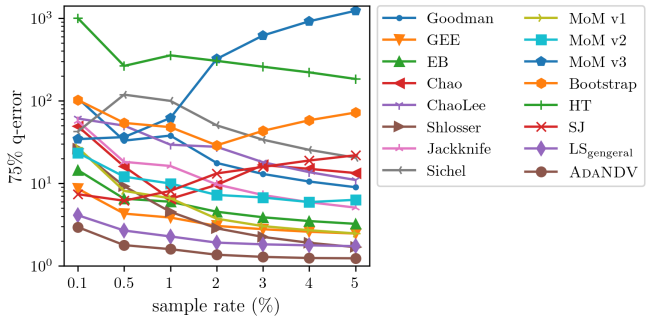


Figure 8: Performance under different sampling rates.

equations [15, 28, 49–52] to estimate NDV, which have been intensively discussed in Section 4.1. In addition, some works focused on the relation between the sampling size and the errors [22, 53, 58].

Since representative traditional estimators are based on different heuristics, so it is difficult for them to adapt to distribution shifting.

Learned NDV Estimators. The introduction of ML techniques for NDV estimation has recently emerged. Wu et al. [57] are the first to leverage ML models as a Learned Statistician (LS) for NDV estimation. They improve profile maximum likelihood [13, 41, 45] methods and use neural networks to take data profiles of the sampled data as inputs to estimate NDV. Li et al. [34] introduced polynomial approximation techniques [30, 58] to learn the parameters of linear polynomials of frequency profile to estimate NDV.

5.3 Method Selection in Databases

Selecting an optimal model from a fixed model set, as well as the ensembling multiple models, for specific database task scenarios, has emerged and garnered significant attention in recent years. Examples include identifying the proper learned cardinality estimation model for different datasets [60], allocating a suitable budget for each data sampler [46], and choosing the optimal knob tuning optimizer for each iteration [61]. However, few studies have attempted to investigate how to select or ensemble existing NDV estimators to acquire better results.

6 CONCLUSION AND FUTURE WORK

In this paper, we propose AdaNDV to address the historically neglected selection dilemma and underexploitation issue of NDV estimators. We propose a complementary perspective of overestimated and underestimated estimators for estimation error correction. Besides, we propose fusing the estimations of the selected estimators to improve the estimation precision through a learned weighted sum, rather than directly estimating NDV. Extensive experiments on a large-scale real-life dataset exhibit the superior performance of our method.

We reveal that using existing estimators can bring promising results, however, the representation ability and the implementation paradigm of the estimator selection and fusion may limit the estimator selection precisions. In the future, we plan to develop more powerful feature extraction methods and explore more beneficial model architectures. Moreover, improving the time and space efficiency is another important direction for future work.

REFERENCES

[1] 2020. Campaign finance data. https://www.fec.gov/data/

[2] 2020. Voter Registration Statistics. https://www.ncsbe.gov/results-data/voterregistration-data

[3] 2024. Box plot. https://en.wikipedia.org/wiki/Box_plot

[4] 2024. Chi-squared test. https://en.wikipedia.org/wiki/Chi-squared_test

[5] 2024. Commoncrawl. https://commoncrawl.org/

[6] 2024. GitHub. https://github.com/

[7] 2024. Pydistinct - Population Distinct Value Estimators. https://pydistinct.readthedocs.io/

[8] 2024. Source Code of MySQL. https://github.com/mysql/mysql-server/blob/824e2b4064053f7daf17d7f3f84b7a3ed92e5fb4/sql/join_optimizer/cost_model.cc

[9] 2024. Source Code of PostgreSQL. https://github.com/postgres/postgres/blob/master/src/backend/optimizer/plan/analyzejoins.c

[10] 2024. Source Code of Spark. https://github.com/apache/spark/blob/master/sql/catalyst/src/main/scala/org/apache/spark/sql/catalyst/plans/logical/statsEstimation/JoinEstimation.scala

[11] 2024. tablib-v1-sample dataset. https://huggingface.co/datasets/approximatelabs/tablib-v1-sample

[12] 2024. Violin plot. https://en.wikipedia.org/wiki/Violin_plot

[13] Jayadev Acharya, Hirakendu Das, Alon Orlitsky, and Ananda Theertha Suresh. 2017. A unified maximum likelihood approach for estimating symmetric properties of discrete distributions. In *International Conference on Machine Learning*. PMLR, 11–21.

[14] Sebastian Bruch, Masrour Zoghi, Michael Bendersky, and Marc Najork. 2019. Revisiting approximate metric optimization in the age of deep neural networks. In *Proceedings of the 42nd international ACM SIGIR conference on research and development in information retrieval*. 1241–1244.

[15] John Bunge and Michael Fitzpatrick. 1993. Estimating the number of species: a review. *Journal of the American statistical Association* 88, 421 (1993), 364–373.

[16] Kenneth P Burnham and Walter Scott Overton. 1978. Estimation of the size of a closed population when capture probabilities vary among animals. *Biometrika* 65, 3 (1978), 625–633.

[17] Kenneth P Burnham and W Scott Overton. 1979. Robust estimation of population size when capture probabilities vary among animals. *Ecology* 60, 5 (1979), 927–936.

[18] Anne Chao. 1984. Nonparametric estimation of the number of classes in a population. *Scandinavian Journal of statistics* (1984), 265–270.

[19] Anne Chao and Shen-Ming Lee. 1992. Estimating the number of classes via sample coverage. *Journal of the American statistical Association* 87, 417 (1992), 210–217.

[20] Moses Charikar, Surajit Chaudhuri, Rajeev Motwani, and Vivek Narasayya. 2000. Towards estimation error guarantees for distinct values. In *Proceedings of the nineteenth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*. 268–279.

[21] Surajit Chaudhuri, Rajeev Motwani, and Vivek Narasayya. 1998. Random sampling for histogram construction: How much is enough? *ACM SIGMOD Record* 27, 2 (1998), 436–447.

[22] Eli Chien, Olgica Milenkovic, and Angelia Nedich. 2021. Support estimation with sampling artifacts and errors. In *2021 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 244–249.

[23] Reuven Cohen and Yuval Nezri. 2019. Cardinality estimation in a virtualized network device using online machine learning. *IEEE/ACM Transactions on Networking* 27, 5 (2019), 2098–2110.

[24] Gus Eggert, Kevin Huo, Mike Biven, and Justin Waugh. 2023. TabLib: A Dataset of 627M Tables with Context. arXiv:2310.07875 [cs.CL]

[25] Otmar Ertl. 2024. UltraLogLog: A Practical and More Space-Efficient Alternative to HyperLogLog for Approximate Distinct Counting. *Proceedings of the VLDB Endowment* 17, 7 (2024), 1655–1668.

[26] Philippe Flajolet, Éric Fusy, Olivier Gandouet, and Frédéric Meunier. 2007. Hyperloglog: the analysis of a near-optimal cardinality estimation algorithm. *Discrete mathematics & theoretical computer science* Proceedings (2007).

[27] Leo A Goodman. 1949. On the estimation of the number of classes in a population. *The Annals of Mathematical Statistics* 20, 4 (1949), 572–579.

[28] Peter J Haas, Jeffrey F Naughton, S Seshadri, and Lynne Stokes. 1995. Sampling-based estimation of the number of distinct values of an attribute. In *VLDB*, Vol. 95. 311–322.

[29] Yuxing Han, Haoyu Wang, Lixiang Chen, Yifeng Dong, Xing Chen, Benquan Yu, Chengcheng Yang, and Weining Qian. 2024. ByteCard: Enhancing Data Warehousing with Learned Cardinality Estimation. *arXiv preprint arXiv:2403.16110* (2024).

[30] Yi Hao and Alon Orlitsky. 2019. Unified sample-optimal property estimation in near-linear time. *Advances in Neural Information Processing Systems* 32 (2019).

[31] Hazar Harmouch and Felix Naumann. 2017. Cardinality estimation: An experimental survey. *Proceedings of the VLDB Endowment* 11, 4 (2017), 499–512.

[32] Madelon Hulsebos, Çagatay Demiralp, and Paul Groth. 2023. Gittables: A large-scale corpus of relational tables. *Proceedings of the ACM on Management of Data* 1, 1 (2023), 1–17.

[33] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

[34] Jiajun Li, Runlin Lei, Sibo Wang, Zhewei Wei, and Bolin Ding. 2024. Learning-based Property Estimation with Polynomials. *Proceedings of the ACM on Management of Data* 2, 3 (2024), 1–27.

[35] Jiajun Li, Zhewei Wei, Bolin Ding, Xiening Dai, Lu Lu, and Jingren Zhou. 2022. Sampling-based estimation of the number of distinct values in distributed environment. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 893–903.

[36] Pengfei Li, Wenqing Wei, Rong Zhu, Bolin Ding, Jingren Zhou, and Hua Lu. 2023. ALECE: An Attention-based Learned Cardinality Estimator for SPJ Queries on Dynamic Workloads. *Proceedings of the VLDB Endowment* 17, 2 (2023), 197–210.

[37] Tie-Yan Liu et al. 2009. Learning to rank for information retrieval. *Foundations and Trends® in Information Retrieval* 3, 3 (2009), 225–331.

[38] Guido Moerkotte, Thomas Neumann, and Gabriele Steidl. 2009. Preventing bad plans by bounding the impact of cardinality estimation errors. *Proceedings of the VLDB Endowment* 2, 1 (2009), 982–993.

[39] Rajeev Motwani and Sergei Vassilvitskii. 2006. Distinct values estimators for power law distributions. In *2006 Proceedings of the Third Workshop on Analytic Algorithmics and Combinatorics (ANALCO)*. SIAM, 230–237.

[40] Suman Nath, Phillip B Gibbons, Srinivasan Seshan, and Zachary Anderson. 2008. Synopsis diffusion for robust aggregation in sensor networks. *ACM Transactions on Sensor Networks (TOSN)* 4, 2 (2008), 1–40.

[41] Alon Orlitsky, Narayana P Santhanam, Krishnamurthy Viswanathan, and Junan Zhang. 2004. On modeling profiles instead of values. In *Proceedings of the 20th conference on Uncertainty in artificial intelligence*. 426–435.

[42] Gultekin Ozsoyoglu, Kaizheng Du, A Tjahjana, W-C Hou, and DY Rowland. 1991. On estimating COUNT, SUM, and AVERAGE relational algebra queries. In *Database and Expert Systems Applications: Proceedings of the International Conference in Berlin, Federal Republic of Germany, 1991*. Springer, 406–412.

[43] Patrick O'Neil, Elizabeth O'Neil, Xuedong Chen, and Stephen Revilak. 2009. The star schema benchmark and augmented fact table indexing. In *Performance Evaluation and Benchmarking: First TPC Technology Conference, TPCTC 2009, Lyon, France, August 24-28, 2009, Revised Selected Papers 1*. Springer, 237–252.

[44] Rama Kumar Pasumarthi, Sebastian Bruch, Xuanhui Wang, Cheng Li, Michael Bendersky, Marc Najork, Jan Pfeifer, Nadav Golbandi, Rohan Anil, and Stephan Wolf. 2019. TF-Ranking: Scalable TensorFlow Library for Learning-to-Rank. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (Anchorage, AK). 2970–2978.

[45] Dmitri S Pavlichin, Jiantao Jiao, and Tsachy Weissman. 2019. Approximate profile maximum likelihood. *Journal of Machine Learning Research* 20, 122 (2019), 1–55.

[46] Jinglin Peng, Bolin Ding, Jiannan Wang, Kai Zeng, and Jingren Zhou. 2022. One Size Does Not Fit All: A Bandit-Based Sampler Combination Framework with Theoretical Guarantees. In *Proceedings of the 2022 International Conference on Management of Data*. ACM, Philadelphia PA USA, 531–544. https://doi.org/10.1145/3514221.3517900

[47] Carl-Erik Särndal, Bengt Swensson, and Jan Wretman. 1992. Model Assisted Survey Sampling. *Springer Series in Statistics* (1992).

[48] A Shlosser. 1981. On estimation of the size of the dictionary of a long text on the basis of a sample. *Engineering Cybernetics* 19, 1 (1981), 97–102.

[49] HS Sichel. 1986. Parameter estimation for a word frequency distribution based on occupancy theory. *Communications in Statistics-Theory and Methods* 15, 3 (1986), 935–949.

[50] Herbert S Sichel. 1986. Word frequency distributions and type-token characteristics. *Math. Scientist* 11 (1986), 45–72.

[51] HERBERT S Sichel. 1992. Anatomy of the generalized inverse Gaussian-Poisson distribution with special applications to bibliometric studies. *Information Processing & Management* 28, 1 (1992), 5–17.

[52] Eric P Smith and Gerald van Belle. 1984. Nonparametric estimation of species richness. *Biometrics* (1984), 119–129.

[53] Gregory Valiant and Paul Valiant. 2017. Estimating the unseen: improved estimators for entropy and other properties. *Journal of the ACM (JACM)* 64, 6 (2017), 1–41.

[54] Paul Valiant and Gregory Valiant. 2013. Estimating the Unseen: Improved Estimators for Entropy and other Properties. *Advances in Neural Information Processing Systems* 26 (2013).

[55] Xuanhui Wang, Cheng Li, Nadav Golbandi, Michael Bendersky, and Marc Najork. 2018. The lambdaloss framework for ranking metric optimization. In *Proceedings of the 27th ACM international conference on information and knowledge management*. 1313–1322.

[56] Yining Wang, Liwei Wang, Yuanzhi Li, Di He, and Tie-Yan Liu. 2013. A theoretical analysis of NDCG type ranking measures. In *Conference on learning theory*. PMLR, 25–54.

[57] Renzhi Wu, Bolin Ding, Xu Chu, Zhewei Wei, Xiening Dai, Tao Guan, and Jingren Zhou. 2021. Learning to Be a Statistician: Learned Estimator for Number of Distinct Values. *Proc. VLDB Endow.* 15, 2 (oct 2021), 272–284. https://doi.org/10.

14778/3489496.3489508

[58] Yihong Wu and Pengkun Yang. 2019. Chebyshev polynomials, moment matching, and optimal estimation of the unseen. *The Annals of Statistics* 47, 2 (2019), 857–883.

[59] Fen Xia, Tie-Yan Liu, Jue Wang, Wensheng Zhang, and Hang Li. 2008. Listwise approach to learning to rank: theory and algorithm. In *Proceedings of the 25th international conference on Machine learning*. 1192–1199.

[60] Jintao Zhang, Chao Zhang, Guoliang Li, and Chengliang Chai. 2023. AutoCE: An Accurate and Efficient Model Advisor for Learned Cardinality Estimation. In *2023 IEEE 39th International Conference on Data Engineering (ICDE)*. IEEE, Anaheim, CA, USA, 2621–2633. https://doi.org/10.1109/ICDE55515.2023.00201

[61] Xinyi Zhang, Hong Wu, Yang Li, Zhengju Tang, Jian Tan, Feifei Li, and Bin Cui. 2023. An Efficient Transfer Learning Based Configuration Adviser for Database Tuning. *Proceedings of the VLDB Endowment* 17, 3 (2023), 539–552.