

# PLM4NDV: Minimizing Data Access for Number of Distinct Values Estimation with Pre-trained Language Models

XIANGHONG XU, ByteDance, China

XIAO HE, ByteDance, China

TIEYING ZHANG\*, ByteDance, USA

LEI ZHANG, ByteDance, USA

RUI SHI, ByteDance, China

JIANJUN CHEN, ByteDance, USA

Number of Distinct Values (NDV) estimation of a multiset/column is a basis for many data management tasks, especially within databases. Despite decades of research, most existing methods require either a significant amount of samples through uniform random sampling or access to the entire column to produce estimates, leading to substantial data access costs and potentially ineffective estimations in scenarios with limited data access. In this paper, we propose leveraging semantic information, i.e., schema, to address these challenges. The schema contains rich semantic information that can benefit the NDV estimation. To this end, we propose PLM4NDV, a learned method incorporating Pre-trained Language Models (PLMs) to extract semantic schema information for NDV estimation. Specifically, PLM4NDV leverages the semantics of the target column and the corresponding table to gain a comprehensive understanding of the column's meaning. By using the semantics, PLM4NDV reduces data access costs, provides accurate NDV estimation, and can even operate effectively without any data access. Extensive experiments on a large-scale real-world dataset demonstrate the superiority of PLM4NDV over baseline methods. Our code is available at <https://github.com/bytedance/plm4ndv>.

CCS Concepts: • **Information systems** → **Query optimization**.

Additional Key Words and Phrases: Number of Distinct Values; Minimize Data Access; Pre-trained Language Model

## ACM Reference Format:

Xianghong Xu, Xiao He, Tieying Zhang, Lei Zhang, Rui Shi, and Jianjun Chen. 2025. PLM4NDV: Minimizing Data Access for Number of Distinct Values Estimation with Pre-trained Language Models. *Proc. ACM Manag. Data* 3, 3 (SIGMOD), Article 199 (June 2025), 28 pages. <https://doi.org/10.1145/3725336>

## 1 Introduction

Estimating the number of distinct values (NDV) of a column is a fundamental problem in various domains, including databases [15, 41], networks [27, 60], biology [16, 88, 89] and statistics [19, 38]. In the field of databases, NDV estimation is the foundation of query optimization. For instance, the relative error in the join-selectivity formulas used in MySQL [5] is directly related to the relative

\*Tieying Zhang corresponds to this work.

Authors' Contact Information: Xianghong Xu, ByteDance, Beijing, China, [xuxianghong@bytedance.com](mailto:xuxianghong@bytedance.com); Xiao He, ByteDance, Hangzhou, China, [xiao.hx@bytedance.com](mailto:xiao.hx@bytedance.com); Tieying Zhang, ByteDance, San Jose, USA, [tieying.zhang@bytedance.com](mailto:tieying.zhang@bytedance.com); Lei Zhang, ByteDance, San Jose, USA, [zhanglei.michael@bytedance.com](mailto:zhanglei.michael@bytedance.com); Rui Shi, ByteDance, Beijing, China, [shirui@bytedance.com](mailto:shirui@bytedance.com); Jianjun Chen, ByteDance, San Jose, USA, [jianjun.chen@bytedance.com](mailto:jianjun.chen@bytedance.com).

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from [permissions@acm.org](mailto:permissions@acm.org).

© 2025 Copyright held by the owner/author(s). Publication rights licensed to ACM.

ACM 2836-6573/2025/6-ART199  
<https://doi.org/10.1145/3725336>

```

CREATE TABLE TableName(
ColumnName ColumnType ColumnConstraints, ColumnComment
EmployeeID int NOT NULL, COMMENT 'Identifier for each employee, unique in this
table',
EmployeeNation VARCHAR(30) NOT NULL, COMMENT 'Nationality for each employee',
-- Definitions of indexes below are omitted.
);

```

Listing 1. An illustration of a schema in MySQL.




error in the constituent NDV estimations [41]. Moreover, systems like Spark [7] and PostgreSQL [6] rely on NDV to compute cardinality, a crucial metric in query optimization. Additionally, recent studies also demonstrate that precise NDV estimation can lead to more efficient query plans that significantly reduce the execution latency for different database management systems (DBMSs) [43, 53]. Furthermore, the effectiveness of some external index advisors is strongly influenced by the accuracy of NDV estimations [67, 68, 78, 96].

Approaches for estimating NDV without an index can be broadly categorized into two classes: *sketch-based* and *sampling-based*. The key difference between the two categories lies in whether the full data is accessed. The sketch-based methods require scanning the entire table once and creating memory-efficient sketches [34, 77], which are then used to estimate NDV. Sampling-based methods, on the other hand, access a small randomly selected subset of the data and use the statistical features derived from these samples to estimate NDV [41, 52, 92]. Nevertheless, despite the significant progress achieved, existing methods still incur notable costs of data access when it comes to NDV estimation for in-production DBMSs.

Sketch-based methods are the most accurate NDV estimators; however, the full data access requirement leads to high I/O costs. For a production database with an OLTP workload running, the cost of accessing the full data is prohibitive. In such situations, sampling-based methods are the only alternatives, which seem to fit the situation perfectly at first glance. Nevertheless, they still incur significant data access costs when applied to production DBMSs, even though they only need a small subset of samples. The primary reason is that almost all sampling-based methods [19, 20, 38, 64, 76, 79] require the samples to be independent and identically distributed (IID), leading to sampling uniformly at random [62]. It is difficult to implement efficiently in DBMSs due to high random I/O costs on disk. Therefore, many applications rely on running SQL statements to obtain samples from the DBMS [43, 46, 67, 68] with a strict sampling cost budget. Specifically, sampling a small portion of data uniformly at random using standard SQL from a table with millions of rows can take more than ten seconds in practice. Besides, random sampling may lead to database failures when the database instance has a heavy workload. Though block-level sampling methods [15, 22] have been developed to obtain samples at the block (or page) level in some commercial DBMSs to improve efficiency, a considerable number of blocks or pages still need to be accessed.

**Motivation.** Motivated by the limitations of existing studies and the escalating practical requirements of data access cost constraints for NDV estimation in real-world applications, this paper endeavors to explore the feasibility of achieving precise NDV estimation with minimal sequential data access or even without any data access. Sequentially accessing the top hundred records violates the IID sample requirement of sampling-based methods but offers greater efficiency by reducing I/O costs. Moreover, operating without any data access eliminates data retrieval costs entirely, yet this approach has not been thoroughly investigated. To address these challenges, we propose leveraging database schemas to achieve precise NDV estimation with highly restricted data access. The semantics in schemas are closely related to the NDV of specific columns, making them a valuable resource for this task.

Table 1. Comparison of NDV estimation approaches, where the gray shadows indicate the accessed data and the shaded area roughly represents the relative size.

Categories	Data Access	Method Property
sketch		Memory-efficient sketches
sampling (learning)		Statistics of IID samples
semantic		Semantics with optional statistics

In the following, an example schema with a MySQL database is presented in Listing 1 to illustrate our proposed concept. Herein, `TableName` represents the name of the table. Meanwhile, the `ColumnName`, `ColumnType`, and `ColumnConstraints` define the name, data type, and data constraints of a column, respectively. Additionally, the `ColumnComment` is the comments usually provided by database administrators (DBA) to describe the structures, meanings, and usages of each column. To exemplify this, consider the columns `EmployeeID` and `EmployeeNation`, where the semantic information provides useful insights corresponding to their NDVs. From the column name and comments, it is evident that `EmployeeID` serves as a unique identifier for employees, implying that its NDV is likely to be comparable to the number of records within the table. Conversely, `EmployeeNation` would have an NDV that does not exceed the total number of recognized countries worldwide. Consequently, leveraging such semantic information can significantly enhance the accuracy of NDV estimations, thereby substantially reducing data access.

**Our approach.** Inspired by the aforementioned observations, this paper introduces PLM4NDV, a novel approach that integrates semantic information utilizing pre-trained language models (PLMs) for NDV estimation. PLMs such as BERT [50], GPT [71], and GPT2 [72] have demonstrated state-of-the-art (SOTA) performance across a wide range of natural language processing (NLP) tasks. Consequently, we propose leveraging a PLM to extract semantic features from the target column for NDV estimation. The design of PLM4NDV adheres to the following principles: (1) **Minimize data access.** PLM4NDV targets the practical NDV estimation problem with only limited sequential data access, e.g., accessing only the first hundred records. (2) **Semantics are critical.** The schema is indispensable for the storage of tabular data within a DBMS. Using semantics in the schema may be beneficial for NDV estimation, therefore, incorporating the output of PLM with a learned model can enhance NDV estimation. (3) **Auxiliary of table.** Given that PLM4NDV relies on semantics, we consider the auxiliary semantics of columns within the same table to estimate the NDV of the target column. We believe that the information in the table can improve the estimation accuracy of individual columns.

Existing methods are inapplicable without data access, whereas PLM4NDV can estimate NDV even in the absence of data, effectively addressing this gap. Scenarios where data access is unavailable do occur in practice, such as when users set access permissions due to privacy concerns or when the sampling cost budget is so limited that not even a single sample can be obtained. Although PLM4NDV may not always achieve very high accuracy using solely semantic information, it provides a unique option to address the issue. Therefore, PLM4NDV is distinctly different from existing methods and does not belong to sketch-based or sampling-based categories. Recent learning-based methods [52, 92] are categorized as sampling-based because they rely on IID samples as input. In contrast, our method represents a new category of NDV estimation: semantic-based. This distinction is illustrated in Table 1. We hope that our approach will inspire the community to explore and develop more semantic-based NDV estimation methods and other statistical estimation techniques that effectively leverage semantic information.

**Our contributions.** This paper makes the following contributions.

- We propose PLM4NDV, a practical NDV estimation method that minimizes the data access by incorporating PLM to extract the semantic features of columns. To the best of our knowledge, this is the first approach to leverage semantic information for NDV estimation.
- PLM4NDV provides the only alternative approach when the data access cost budget restricts sampling any data. We introduce a new paradigm in NDV estimation with minimal data access costs: a semantic-based approach that is distinct from both sketch-based and sampling-based methods.
- Our extensive experiments on a large-scale real-world dataset demonstrate the promising performance of PLM4NDV, particularly in scenarios where data access is limited. The experiments also reveal several insightful findings that inform future research directions.

## 2 Preliminaries

**Problem statement.** Consider a column  $C = (c_1, c_2, \dots, c_N)$  consisting of  $N$  items where each item  $c_i (1 \leq i \leq N, c_i \in \Omega)$  is a member of a universe of  $D (D \leq N)$  possible values. Different items within the column may share the same value. Let  $N$  denote the size of the column  $C$  and let  $D$  represent the NDV within  $C$ . A subset  $S$  of  $C$  is sampled either randomly or sequentially, with  $n$  denoting the size of  $S$ . The NDV of  $S$  is represented by  $d$  and the sampling rate is given by  $r = n/N$ . The column  $C$  has a name and data type defined within the database schema. Additionally, data constraints and comment descriptions of the column may be present in the schema. The objective is to estimate  $D$  utilizing the schema information and the sample  $S$ .

**Semantics.** Semantics refer to the meaning or interpretation of the data within the column, expressed in natural language. We assume that the semantics of the stored data align with the column name defined in the schema and a column with a meaningful column name tends to have certain NDV distributions, suggesting that leveraging semantics may be beneficial for NDV estimation. Misalignment can occur in real-world scenarios, for instance, a column storing dates may be arbitrarily named `columnA` by users. However, advanced techniques in the column annotation task [86] can effectively address these issues. Consequently, our assumptions are reasonable in practice.

**Sample Statistics.** Two key sample statistics are widely used in NDV estimation, and they are defined as follows:

*Frequency.* The frequency of a value  $c_i$  in a column is the number of times it appears. Let  $N_{c_i}$  denote the frequency of  $c_i$  in  $C$  and  $n_{c_i}$  denotes the frequency of  $c_i$  in the sample  $S$ . By definition,  $n = \sum_{c_i} n_{c_i}$  and  $d = |\{c_i \in \Omega | n_{c_i} > 0\}|$ .

*Frequency profile.* The frequency profile represents the frequency of frequencies. Let  $F_j = |\{i \in \Omega | N_{c_i} = j\}|$  denote the NDV with frequency  $j$  in  $C$ , and  $f_j = |\{i \in \Omega | n_{c_i} = j\}|$  denote the NDV with frequency  $j$  in  $S$ . By this definition,  $n = \sum_j j \cdot f_j$  and  $d = \sum_j f_j$ .

**Illustration of NDV estimation methods.** We demonstrate how various methods establish NDV estimation, excluding sketch-based methods, which are not applicable under limited data access.

*Sampling-based methods.* Sampling-based methods primarily leverage the statistics of the IID samples to derive the estimation. In general, a formal description is given by  $\hat{D} = \mathcal{M}(N, f)$ , where  $\hat{D}$  denotes the estimated NDV and  $\mathcal{M}$  represents the general estimation function. For instance, a learned NDV estimation method [92] takes the cut-off frequency profiles and  $N$  as input and estimates NDV by a learnable multi-layer perception (MLP). The `Sichel` [80–82] method requires

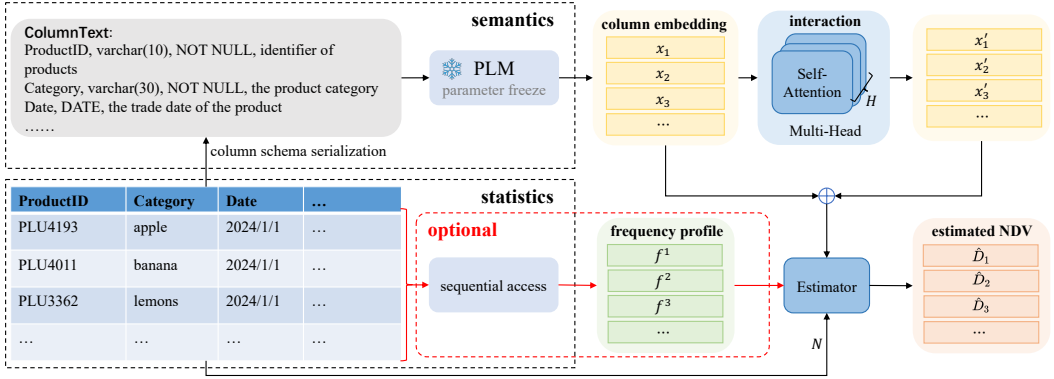


Fig. 1. The model architecture of PLM4NDV is illustrated using a hypothetical table. The statistic component is akin to existing sampling-based methods but operates more efficiently and with a reduced volume of data. In addition, accessing data in the table is optional for PLM4NDV.

solving non-linear equations:

$$(1 + g) \ln g - Ag + B = 0, \frac{f_1}{n} < g < 1, A = \frac{2n}{d} - \ln \frac{n}{f_1}, \quad (1)$$

$$B = \frac{2f_1}{d} + \ln \frac{n}{f_1}, \hat{b} = \frac{g \ln \frac{ng}{f_1}}{1 - g}, \hat{c} = \frac{1 - g^2}{ng^2}, \hat{D}_{\text{Sichel}} = \frac{2}{\hat{b}\hat{c}}.$$

Another representative method, Goodman [38], constructs a linear polynomial using  $N$  and  $f$  to establish the estimation  $\hat{D}_{\text{Goodman}}$ :

$$\hat{D}_{\text{Goodman}} = d + \sum_{i=1}^n (-1)^{i+1} \frac{(N - n + i - 1)!(n - i)!}{(N - n - 1)!n!} f_i. \quad (2)$$

*Semantic-based methods (ours).* Semantic-based methods utilize the semantics in the schema along with optional sample statistics. The formal description is  $\hat{D} = \mathcal{M}(\text{semantics}, N, \underline{f})$ , where  $\mathcal{M}$  is the general function and  $\underline{f}$  represents  $f$  is optional in the formula.

### 3 Methodology of PLM4NDV

#### 3.1 Model Architecture

The architecture of PLM4NDV is depicted in Figure 1, utilizing a hypothetical table that stores data related to various fruits to illustrate the model's estimation process. Our model relies on two main categories of features: semantic information and sample statistics. We consider accessing the first page of data as the limited data scenario because it incurs minimal costs when accessing data. In many database systems, a single page usually contains dozens to hundreds of rows, depending on the specific database and the width of the table. In this paper, we use the first 100 rows as an example to illustrate the limited data scenario.

**Data pipelines.** The semantic features encompass characteristics related to both individual columns and the corresponding table. Initially, the schema is serialized into natural language texts for each column. These textual representations are then processed through a PLM to derive the embeddings for each column. Subsequently, the column embeddings within the same table are fed into the column interaction component, thereby generating table-aware representations. Statistical features consist of the frequency profiles of each column derived from sequentially accessed samples, which

are optional inputs for the PLM4NDV estimation process. The learned estimator utilizes both semantic and statistical inputs to predict the NDVs for each column.

**Training.** During the training phase, we gather a large set of tables from the real world and record the following information for each column: the schema, the statistics of the top 100 rows, and the ground-truth NDV. By this means, our method can learn the relationship between the schema semantics and the corresponding NDV in real-world applications. The parameters of the PLM are kept frozen, therefore, the training of PLM4NDV focuses solely on the column interaction and NDV estimator modules.

**Inference.** Following training on existing datasets, PLM4NDV can perform estimations on unseen tables. First, the same PLM is used to extract semantic representations from the schema of the test table. Next, sample statistics are gathered from the first 100 rows of each column, which are sequentially accessed. Finally, the trained model is employed to estimate the NDV.

### 3.2 Column Embedding

As shown in Figure 1, the first step of PLM4NDV involves leveraging the structured schema by serializing it into natural language. Then, we extract semantic features from the serialized natural language text.

**Schema serialization.** Since PLMs require token sequences (i.e., natural language text) as input, the initial step involves converting the database schema into text sequences for each column. This transformation enables the structured schema to be processed by PLMs. Specifically, the serialized text sequence of a column is defined as follows:

ColumnText := `ColumnName` , `ColumnType` , `ColumnConstraints` , `ColumnComment`

where the column definitions and descriptions are concatenated using commas, employing a straightforward way to transform the schema of a column into a sequence of tokens.

In addition, a substantial number of column schemas may lack the components `ColumnConstraints` and `ColumnComment` in practical scenarios. We assume `ColumnName` and `ColumnType` are available for the model input, while `ColumnConstraints` and `ColumnComment` are optional. The example of the `Date` column in Figure 1 illustrates this situation. For columns that are missing these optional components in the schema, we exclude these elements and concatenate the serialized available components.

**Semantic embedding.** We leverage sentence transformers [61, 74] as our foundational models due to their proficiency in producing similar vector-space representations for semantically analogous sentences. This capability is particularly suitable for extracting semantic information for columns in databases. By leveraging these models, we can effectively capture the semantic relationships between different columns. For instance, the `ProductID`, `ProductCode`, and `ProductNumber` columns may possess different names (and thus different serialized text sequences) but convey similar semantic meanings. The PLMs enable us to process these variable-length text sequences as input and generate fixed-size vectors, thereby facilitating the utilization of schema information.

Since we require only the semantic embeddings of the serialized column text from the PLM, we can utilize the frozen parameters of the PLM to obtain the column embeddings without fine-tuning. Specifically, the column embedding can be formulated as:  $x = \text{PLM}(\text{ColumnText})$ ,  $x \in \mathbb{R}^l$ , where  $l$  is the embedding size of the PLMs.

### 3.3 Column Interaction

Columns with the same semantics may exhibit different selectivities in varying table contexts. Therefore, the semantics from other columns within the same table can provide valuable auxiliaries in NDV estimation for individual columns. For instance, consider a table `Product` containing two columns named `State` and `FIPSCode` (Federal Information Processing Standards Code used in the

USA), which store relevant information of products, respectively. Estimating the NDV of State alone using only semantic information may be challenging, as it is unclear whether the states refer to those in the USA or other countries. The `FIPSCode` column provides valuable auxiliaries, indicating that the state likely refers specifically to states within the USA.

Given this insight, we believe that the auxiliary semantic information from other columns within the same table can be beneficial. Therefore, we take the embeddings of all columns in a table as input to establish the interactions between them. Specifically, let the column embeddings in a table be denoted as:

$$X = [x_1, x_2, \dots, x_t]^\top, x_i \in \mathbb{R}^l, \quad (3)$$

where there are  $t$  columns in the table and  $x_i$  is the embedding of the  $i$ -th column from a frozen PLM. For each column embedding  $x_i$ , we consider the information from other columns, thereby enhancing the overall understanding of the semantics of individual columns within the table. To this end, we leverage Multi-Head Self-Attention (MHSA) [90] to achieve our goals as illustrated in Figure 1. The self-attention mechanism enables the model to focus on relevant aspects of the input data while disregarding irrelevant information, thereby facilitating the learning of relationships between the columns in the table.

Particularly, the attention mechanism involves three components: Query ( $Q$ ), Key ( $K$ ), and Value ( $V$ ), which are derived through three linear transformations:

$$Q = XW^Q + b^Q, K = XW^K + b^K, V = XW^V + b^V, \quad (4)$$

where  $W^Q, W^K, W^V \in \mathbb{R}^{l \times l}$  are learnable weight matrices and  $b^Q, b^K, b^V \in \mathbb{R}^l$  are learnable bias vectors. The Self-Attention (SA) mechanism is formulated as:

$$SA(Q, K, V) = \text{softmax} \left( \frac{QK^\top}{\sqrt{l}} \right) V, \quad (5)$$

where  $\text{softmax}$  [14] converts the dot-product correlations between the elements in  $Q$  and  $K$  into normalized attention coefficients (probability distribution). This allows for a weighted sum to be applied to  $V$ , thereby highlighting the relevant information from the table. The MHSA can be obtained by:

$$MHSA(Q, K, V) = [SA_1, SA_2, \dots, SA_H]W + b, \quad (6)$$

where  $H$  is the number of heads, and each SA component undergoes individual transformations as defined in Equation (4). The feature dimension for each SA head is  $l' = H \cdot l$ . The weight matrix  $W \in \mathbb{R}^{l' \times l'}$  and the bias vector  $b \in \mathbb{R}^{l'}$  are applied after concatenating the outputs from the multiple heads. Consequently, the output dimension of MHSA is denoted as  $X' \in \mathbb{R}^{t \times l'}$ , where  $x'_i \in \mathbb{R}^{l'}$  represents the column-interacted table semantics representation of the  $i$ -th column. For more details about MHSA not covered in this paper, please refer to [90].

### 3.4 Statistics Collection

Although our method can estimate NDV without accessing data, relying solely on semantic information does not yield high estimation accuracy. Due to the diversity in data distributions in real-world applications or the effects of database sharding, two tables with similar or identical definitions may exhibit substantial differences in NDV. Continuing from the previous example of table `Product`, another database may contain the same table but store product content from different regions, or multiple sharded databases may be organized according to specific categories by DBAs. Consequently, it is essential to access a portion of the data for accurate estimations.

**Data access.** Random sampling is usually implemented by running SQL queries, which can hardly satisfy a limited sampling cost budget. Therefore, we utilize sequential access to get data samples.

We directly access the top  $n$  rows (e.g.,  $n=100$ ) for each column, which allows us to complete data access with minimal (i.e., once) I/O operation, significantly reducing the access costs.

**Frequency profile construction.** The accessed data sample is denoted as  $S$  and  $|S| = n$ . Each item  $s_i$  in  $S$  is a tuple containing  $t$  elements, where  $t$  is the number of columns. We construct the frequency profiles for each column (illustrated in Section 2) in  $S$ , and  $f^i \in \mathbb{R}^n$  represents the frequency profile of the  $i$ -th column.

**Number of rows.** Essentially, the total number of rows  $N$  is required for the estimation, and our method also leverages this information. The cost of accessing  $N$  is minimal in practice because the DBMS typically maintains these basic statistics. For instance, in MySQL, the number of rows for the tables can be assessed using the SQL query: `SHOW TABLE STATUS FROM DBName;`, where `DBName` is the database name containing the object tables.

### 3.5 Learned NDV Estimation

**Learned estimator.** The learned estimator takes both the semantic and statistical features as input to estimate NDV. We use an MLP to map the input features into the estimation:

$$\log \hat{D}_i = MLP([x_i + x'_i, \log N_i, \underline{f^i}]), \quad (7)$$

where  $\hat{D}_i$  is the estimated NDV of the  $i$ -th column in the dataset,  $x_i$  is the semantic feature vector of the column,  $x'_i$  is the feature vector of the column that interacted with other columns within the same table,  $f^i$  is the frequency profile vector of the data samples,  $\underline{f^i}$  represents  $f^i$  is optional in the formula,  $N_i$  is the size of the column, and  $[\cdot]$  indicates feature concatenation. We employ the logarithm operation on  $N$  and the estimation  $\hat{D}$  to reduce their magnitude and mitigate the impact of large values.

**Model learning.** Our objective is to minimize the difference between the estimation and the ground truth, the loss function is formulated as:

$$\mathcal{L} = \frac{1}{N} \sum_{i=1}^N (\log \hat{D}_i - \log D_i)^2, \quad (8)$$

where  $N$  is the number of training data points,  $\hat{D}_i$  and  $D_i$  are the estimation result and the ground truth of the  $i$ -th sample in the training set.

## 4 Experiments

### 4.1 Experimental Setup

**Dataset selection.** Due to the diversity of NDV estimation methods, different approaches focus on various aspects beyond estimation accuracy. Sketch-based methods emphasize memory usage, whereas sampling-based methods consider the sampling size. Therefore, most existing works are evaluated under various specific settings [45] and there is no publicly available, widely used evaluation dataset for NDV estimation. Furthermore, most sampling-based methods are evaluated on columns that follow standard distributions or on several manually crafted datasets that satisfy their assumptions [16, 21, 38, 41, 59, 80–82, 84]. Recent sampling-based works expand the evaluation to some open-source datasets [52, 92].

This situation may lead to two primary problems. On the one hand, a limited number of evaluation columns leads to incomplete evaluations. On the other hand, manually crafted columns differ from practical scenarios due to their lack of meaningful schema and monotonous data distribution. Therefore, in this paper, we evaluate our method on a large-scale dataset derived from real-world scenarios. In recent years, the database community has collected extensive relational tabular datasets [32, 49]. TabLib [32] is the largest dataset, containing 627M individual tables totaling 69



Table 2. Statistics of preprocessed datasets.

	train	test	validation
# Table	4003	1128	1267
# Column	34761	8725	10981

TiB. We select the TabLib sample version dataset [8], which comprises 0.1% of the full version (69 GB), to evaluate our method.

**Data preprocess.** The selected dataset contains 77 parquet files; however, we remove three of them (2d7d54b8, 8e1450ee, and dc0e820c) due to the storage and memory constraints. Each parquet file includes thousands of tables, and we split the remaining parquet files into training, testing, and validation sets. Previous works [52, 92] evaluate NDV estimation methods on large tables (exceeding one million rows) encompassing a total of 218 individual columns. In contrast, we broadened the range of table sizes from tens of thousands to several million rows to provide a more comprehensive evaluation. We filter out columns lacking useful semantics in their names due to misalignment issues, as column annotation [86] is beyond the scope of this paper. This includes columns with names consisting of one character or fewer, as well as those composed entirely of numbers, scientific notation, or timestamps. The statistics of the preprocessed datasets are shown in Table 2, with the largest table containing 696 columns. As shown in Table 2, the dataset used in this paper is substantial for evaluation purposes.

We conducted a statistical analysis of the data type of each column in the dataset. We observe that the majority of the data types are primitive, with very few composite types: 34.7% are big int, 33.4% are string, and 30.1% are double. The other data types present in very small quantities include bool, timestamp, unsigned int, time, list, and dict. The first three common data types account for over 98% of the dataset, so the experiment analysis in this paper focuses primarily on them.

**Evaluation metrics.** Ratio error, known as q-error [58], is widely used to evaluate the performance of NDV estimation:

$$\text{q-error} = \max\left(\frac{\hat{D}}{D}, \frac{D}{\hat{D}}\right), \quad (9)$$

where  $\hat{D}$  is the estimated NDV and  $D$  is the ground truth NDV. The error is always greater than or equal to 1, and a lower value indicates better performance.

**Implementation Details.** The MLP used in PLM4NDV estimator consists of three hidden layers with sizes 384, 128, and 64, respectively. The activation function employed is ReLU. The semantic embedding model is Sentence T5 [61]. We train the model by the Adam [51] optimizer with an initial learning rate of 0.001. The training batch size is 256. Model checkpoints are saved according to the 90% percentile of q-error on the validation set, and the performance is reported on the unseen testing set. All the experiments in this paper are conducted on an NVIDIA A100 80GB GPU.

**Baseline Methods.** For a fair comparison, we take sampling-based methods as our baseline methods while excluding sketch-based ones, as our method accesses only a portion of the column data rather than the full table. We present the baselines with brief descriptions below, and the heuristics or assumptions of these methods will be discussed in Section 6.

- Goodman [38] is the seminal NDV estimation method and it has the expression in Equation (2).
- GEE [21] estimates NDV by:  $\hat{D}_{\text{GEE}} = \sqrt{N/n}f_1 + \sum_{j=2}^n f_j$ .
- EB [23] improves GEE:  $\hat{D}_{\text{EB}} = \sqrt{N/n}f_1^+ + \sum_{j=2}^n f_j, f_1^+ = \max(1, f_1)$ .

- Chao [19, 64] has a nonlinear polynomial expression:  $\hat{D}_{\text{Chao}} = d + f_1^2/2f_2$ . If  $f_2$  is zero, we will use  $d$  as the estimation.
- Shlosser [79]:  $\hat{D}_{\text{Shlosser}} = d + \frac{f_i \sum_{i=1}^n (1-r)^i f_i}{\sum_{i=1}^n r^i (1-r)^{i-1} f_i}$ .
- Jackknife [17, 18]. Denote  $d_{n-1}(k)$ ,  $1 \leq k \leq n$ ,  $d_{n-1}(k) = d_n - 1$  if the attribute value for tuple  $k$  is unique; otherwise  $d_{n-1}(k) = d_n - 1$ . The first-order Jackknife method is:  $\hat{D}_{\text{Jackknife}} = d_n - (n-1)(d_{n-1} - d_n)$ .
- Sichel [80–82] method needs to solve non-linear equations and it is shown in Equation (1).
- Bootstrap [84]:  $\hat{D}_{\text{Boot}} = d + \sum_{j:n_j>0} (1 - n_j/n)^n$ . It may perform worse when  $D$  is large and  $n$  is small because  $\hat{D}_{\text{Boot}} \leq 2d$ .
- HT [76] defines  $h_n(x) = \frac{\Gamma(N-x+1)\Gamma(N-n+1)}{\Gamma(N-n-x+1)\Gamma(N+1)}$ , where  $\Gamma$  is the gamma function, and  $\hat{D}_{\text{HT}} = \sum_{j:n_j>0} \frac{1}{1-h_n(\hat{N}_j)}$ ,  
 $\hat{N}_j = N(n_j/n)$ .
- The family of MoM [16]. MoM1 needs to solve  $d = \hat{D}_{\text{MoM1}}(1 - e^{-n/\hat{D}_{\text{MoM1}}})$ . MoM2:  $d = \hat{D}_{\text{MoM2}}(1 - h_n(N/\hat{D}_{\text{MoM2}}))$  where  $h_n(\cdot)$  is defined identically to that in HT.
- LS [92] is the first method based on ML techniques. It is pre-trained on a manually crafted dataset with 0.72 million data points in which the frequency profile of the columns follows specific distributions. For a fair and comprehensive comparison, we fine-tune LS on our training set, and it is denoted as LS(FT).

Some baseline methods (ChaoLee [20], MoM3 [16], and SJ [41]), have extremely sophisticated expressions. Due to space limitations, we omit the equations in this paper and refer to [4, 85] for the details.

**Research Questions.** We conduct experiments to answer the following Research Questions (RQs).

- RQ1: What is the performance of baselines and the proposed PLM4NDV under sequential access and random sampling conditions when the sample sizes are the same? Can PLM4NDV significantly improve estimation accuracy under limited data scenarios?
- RQ2: We argue that PLM4NDV largely relies on semantic features from the schema. Does the semantic information contained in the schema significantly contribute to NDV estimation? Besides, how useful is the semantic information from other columns in the table?
- RQ3: What is the effect of different fine-tuned PLMs in column representation for NDV estimation? How sensitive is PLM4NDV to hyperparameters?
- RQ4: We claimed that the distinct advantage of PLM4NDV is its applicability even without data access. What is the performance in this scenario? How do PLM4NDV and the baselines perform with smaller sample sizes? Can PLM4NDV still outperform sampling-based baseline methods when randomly sampling a considerable data volume?
- RQ5: Efficiency significantly affects the practicality of NDV estimation methods. What are the efficiency and actual time consumption of PLM4NDV?
- RQ6: In the sequential access setting, the data layout of the original table may introduce bias. How does data layout impact NDV estimation under sequential access scenarios?

## 4.2 Main Results (RQ1)

For the same sample size, we evaluate the performance of PLM4NDV and baselines using both the first 100 rows accessed sequentially and 100 rows sampled randomly from each column, respectively. To comprehensively illustrate the performance of the methods, we present both the mean and various percentiles of q-error on the test set. The results are shown in Table 3 and Table 4, where  $\infty$  indicates that the number exceeds the representation limits of a 32-bit floating-point type. Based on the results, we can draw the following conclusions:

Table 3. Mean and quantiles of q-error performance on the test set under sequential access first 100 rows per column. The best-performing metrics are highlighted in boldface.

Method	Mean	50%	75%	90%	95%	99%
Goodman	$\infty$	3.46	73.50	1.32e3	7.56e15	$\infty$
GEE	125.49	4.00	14.54	49.90	154.60	1.60e3
EB	17.70	7.38	15.14	30.07	51.09	186.41
Chao	225.09	14.20	150.00	474.21	964.15	3.24e3
Shlosser	129.23	2.04	10.17	61.00	191.00	1.71e3
Jackknife	205.39	7.07	76.53	265.84	603.82	3.02e3
Sichel	249.14	6.34	109.10	389.66	822.36	3.20e3
Bootstrap	86.63	27.77	70.34	143.71	300.08	1.09e3
HT	1.73e3	27.22	381.99	2.93e3	7.08e3	2.96e4
MoM1	3.01e5	6.20	104.00	1.67e5	4.39e5	3.38e6
MoM2	163.69	6.50	26.32	117.59	351.00	2.45e3
MoM3	169.98	7.93	37.27	148.53	360.23	2.45e3
ChaoLee	278.88	9.64	128.67	445.40	993.71	3.49e3
SJ	159.33	2.20	16.67	118.81	350.97	2.39e3
LS	43.05	2.90	6.87	31.08	93.35	492.48
LS(FT)	22.39	3.30	5.96	17.32	48.70	321.01
PLM4NDV	<b>13.33</b>	<b>1.86</b>	<b>3.81</b>	<b>10.81</b>	<b>25.18</b>	<b>148.76</b>

Table 4. Mean and quantiles of q-error performance on the test set under random sampling 100 rows per column. The best-performing metrics are highlighted in boldface.

Method	Mean	50%	75%	90%	95%	99%
Goodman	$\infty$	1.79	28.78	680.55	3.92e17	$\infty$
GEE	7.06	2.74	7.94	16.09	22.90	51.60
EB	9.54	6.15	12.15	20.59	29.28	52.78
Chao	210.76	3.27	123.22	449.78	962.46	3.27e3
Shlosser	22.35	2.00	10.41	43.07	87.30	335.26
Jackknife	72.57	2.25	41.16	143.26	303.14	1.16e3
Sichel	141.49	2.16	88.58	300.31	613.23	2.16e3
Bootstrap	83.52	25.00	63.94	137.93	290.58	1.09e3
HT	1.35e3	20.45	408.79	2.99e3	6.73e3	2.28e4
MoM1	4.90e4	2.00	12.30	1.49e5	3.73e5	6.78e5
MoM2	18.81	2.00	12.38	28.32	46.70	223.77
MoM3	26.50	2.54	18.02	57.10	107.62	242.63
ChaoLee	141.61	3.30	76.19	280.18	598.08	2.32e3
SJ	11.95	1.31	2.99	11.29	26.63	171.82
LS	3.74	2.62	4.09	6.48	9.27	16.77
LS(FT)	2.69	1.49	2.06	3.55	5.09	14.27
PLM4NDV	<b>2.17</b>	<b>1.30</b>	<b>1.76</b>	<b>2.84</b>	<b>4.26</b>	<b>11.64</b>

- PLM4NDV achieves the best performance across all metrics under both sequential access and random sampling conditions. In sequential access, the majority (90%) of test cases have a q-error

below 10.81, while in the random sampling setting, this metric is 2.84, which is significantly lower than that of other methods under both conditions. The consistently superior performance across all metrics and substantially low q-error in the majority of test cases demonstrate the effectiveness of our method.

- Under the sequential access condition, estimating NDV under limited data is challenging for sampling-based baseline methods. On the one hand, we observe that most of these methods have substantial NDV estimation q-errors. On the other hand, the growth between the quantiles of q-error is quite rapid, indicating unstable performance under limited data scenarios. Although a few methods (ShLosser, SJ, and LS) exhibit promising results in the 50% quantile of q-error that is less than 3, their poor performance in certain scenarios significantly diminishes their overall effectiveness.
- Under the random sampling condition, nearly all methods show a significant reduction in estimation error across each metric, indicating the importance of IID samples, as previously noted in prior approaches. This is because sequentially accessed samples may not be IID and can be influenced by the data layout, so we will investigate the impact of the data layout in Section 4.7. The overall performance of PLM4NDV is still the best among all methods, indicating that incorporating semantic information is also effective under the IID samples condition.
- The number of training data points for the learning-based baseline methods (LS and LS(FT)) is approximately twenty times greater than that of PLM4NDV. The fine-tuned version exhibits performance improvement over LS in most metrics (except for the 50% percentile of q-error under sequential access scenarios), indicating the difficulties of NDV estimation under sequential data access scenarios. LS(FT) exhibits inferior performance than PLM4NDV, demonstrating the significance of leveraging semantics in NDV estimation.
- Notably, PLM4NDV, utilizing sequentially accessed data, outperforms most baselines even under random sampling conditions across most metrics. This highlights that when the sampling cost budget is limited, PLM4NDV using sequential access is practical enough because its performance is comparable to most baselines using random sampling.
- In summary, our method PLM4NDV consistently outperforms existing approaches with the same data access costs. Additionally, PLM4NDV achieves higher estimation accuracy with lower data access costs. The superiority of PLM4NDV demonstrates the significant benefit of incorporating semantic information for NDV estimation.

### 4.3 Effect of Semantic Features (RQ2)

To verify the effectiveness of semantic features, we test the following variants of PLM4NDV. The variant “w/o col” indicates removing the column embedding  $x$ . Similarly, removing the interacted column information  $x'$  is denoted as “w/o tab”. Dropping all semantic features is represented as “w/o tab and col”. These three variants of PLM4NDV thoroughly explore the effect of semantic features from the column itself and the auxiliary of the table. In addition, we construct two variants to distort the semantics: “permute col” refers to the random permutation of the textual tokens in the serialized schema, intending to disrupt the semantics of each column. “permute tab” denotes the random permutation of column texts within the same table, which undermines the relationship between column semantics and column data. Their performance evaluated under the sequential access condition is depicted in Figure 2, from which we can draw the following conclusions:

- Semantic information significantly contributes to NDV estimation. The variant “w/o tab and col” solely leverages the frequency profile from the sampled data to establish the estimation. It is obvious that removing or distorting the semantic features of both the column and the table

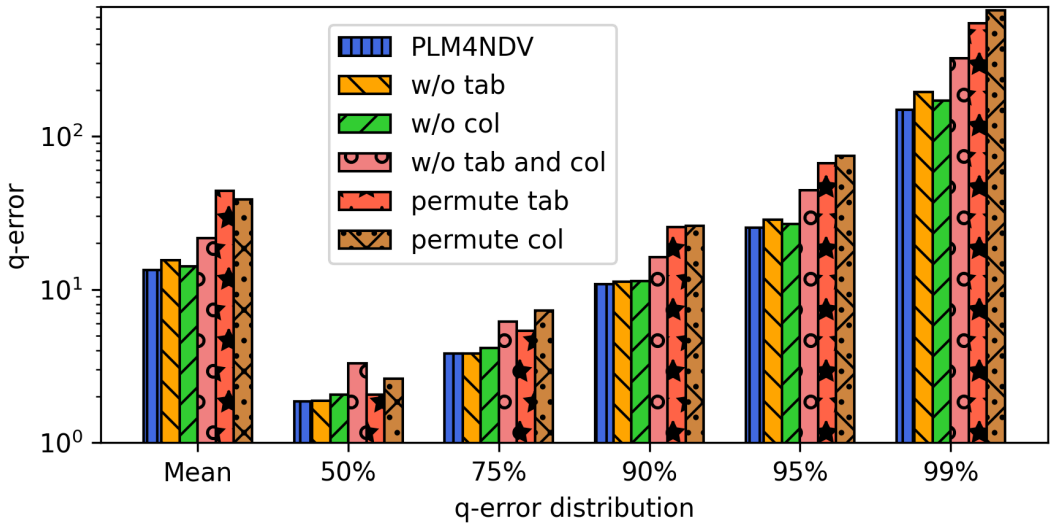


Fig. 2. Ablation study on semantic features evaluated under the sequential access condition, where “w/o” indicates removing a kind of semantic features from PLM4NDV.

leads to a significant increase in the mean and each quantile of q-error. This illustrates the effectiveness of using semantic features from the schema in NDV estimation.

- The auxiliary semantics of the columns within the same table are critical. Removing the auxiliary semantics from the table, the variant “w/o tab” relies on the column-independent semantic information in the estimation. We observe a consistent decline in the estimation performance, highlighting the significance of the table schema in NDV estimation and the effectiveness of the table semantics component in PLM4NDV.
- The semantics of the whole table are useful for estimations. The variant “w/o col” performs better than “w/o tab and col” but is comparable to “w/o tab”. On the one hand, this indicates that the semantics within the table, combined with the object column’s profile, can provide some useful context, as the semantics of the target column are inherently contained within the semantics of the table to some extent. On the other hand, relying solely on the table information does not fully convey the meaning of the object column, leading to a decline in performance compared to PLM4NDV. This further emphasizes the importance of understanding the semantic meanings of the object columns in NDV estimation.
- Using incorrect semantics (“permute col” and “permute tab”) significantly increases the mean, 90%, 95%, and 99% q-error compared to scenarios without semantics (“w/o tab and col”). This indicates that the accuracy of semantics is crucial for PLM4NDV, and its superior performance relies on effectively leveraging semantics. Among the two variants using incorrect semantics, “permute col” performs worse than “permute tab” in the percentiles of q-error. This difference can be explained as follows: a random permutation of the textual tokens in the serialized column schema may destroy the semantics of both columns and tables. For instance, a column with the serialized text “EmployeeID,int” could be transformed into “Il,etpeyDiomnE” which is semantically meaningless. On the contrary, the mean q-error of “permute tab” is greater than “permute col”, indicating that using the semantics of other columns may bring significant errors in extreme scenarios that increase the mean q-error. For instance, a column stores the data of unique identifiers but is named with the semantics of gender.

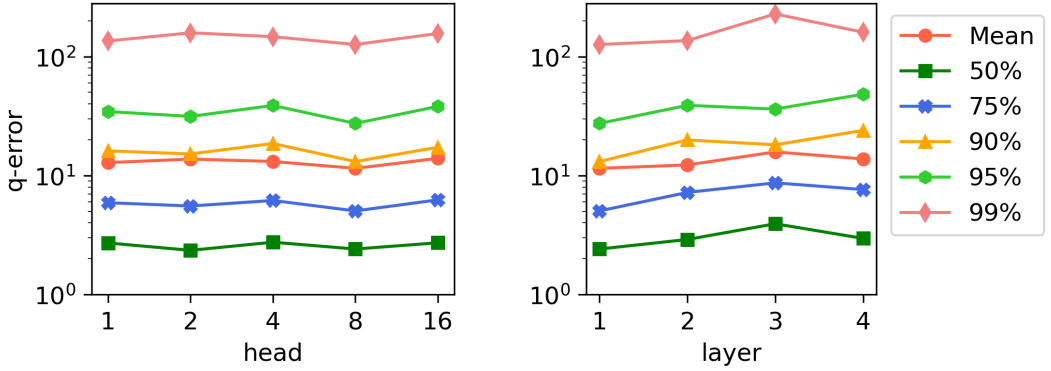


Fig. 3. Hyperparameter study of PLM4NDV.

- The above findings reveal that the effectiveness of our method stems from the combination of column and table semantic information. Relying on only one of them or using incorrect semantics does not yield optimal estimation results.

#### 4.4 Robustness Analysis (RQ3)

**Impact of hyperparameters.** There are two hyperparameters in PLM4NDV: the number of heads and layers in the MHSA column interaction component. We search the number of attention heads in  $\{1, 2, 4, 8, 16\}$  and the number of layers in  $\{1, 2, 3, 4\}$  to investigate the impact of them, the performance of PLM4NDV on the test set under different settings is depicted in Figure 3. Based on the results in the figure, we can derive the following findings.

- The overall performance of PLM4NDV is not highly sensitive to the number of heads, with variations in the number of heads not resulting in significant performance differences. PLM4NDV demonstrates the optimal overall performance when the number of heads is 8.
- Increasing the number of layers leads to a decline in estimation accuracy, with the optimal estimation performance across all metrics obtained when the number of layers is 1. A possible reason is that the latent semantic information between columns can be captured effectively with just one interaction layer, and adding more layers may lead to overfitting.
- Although adjusting the hyperparameters results in variations in performance, we can conclude that PLM4NDV is not sensitive to the two hyperparameters because the overall performance of PLM4NDV under different settings outperforms most baseline methods. Therefore, the number of heads and layers is set to 8 and 1, respectively.

**Impact of foundation PLMs.** Due to the numerous foundational PLMs available, we selected two representative PLMs that are well fine-tuned for the task of semantic textual similarity: Sentence-RoBERTa (SR) [74] and Sentence-T5 (ST5) [61]. These are fine-tuned versions of RoBERTa [57] and T5 [73], respectively. We chose these two fine-tuned PLMs due to their effectiveness and generalization capabilities in encoding serialized schema sentences. We study the impact of PLMs with different sizes, and the number of parameters of the PLMs discussed in this section are shown in Table 5. The performance of using different PLMs as the column embedding component in PLM4NDV under sequential access scenario is depicted in Figure 4. Based on the sizes and performance of different PLMs, we can draw the following conclusions:

- The exploitation of the column embeddings obtained from different foundation PLMs yields varying estimation results in PLM4NDV, with significant differences observed in the log-scaled

Table 5. Number of parameters of the discussed foundation PLMs, where ‘M’ and ‘B’ represent million and billion, respectively.

SR-base	SR-large	ST5-base	ST5-large	ST5-xl	ST5-xxl
125M	355M	110M	335M	1.24B	4.86B

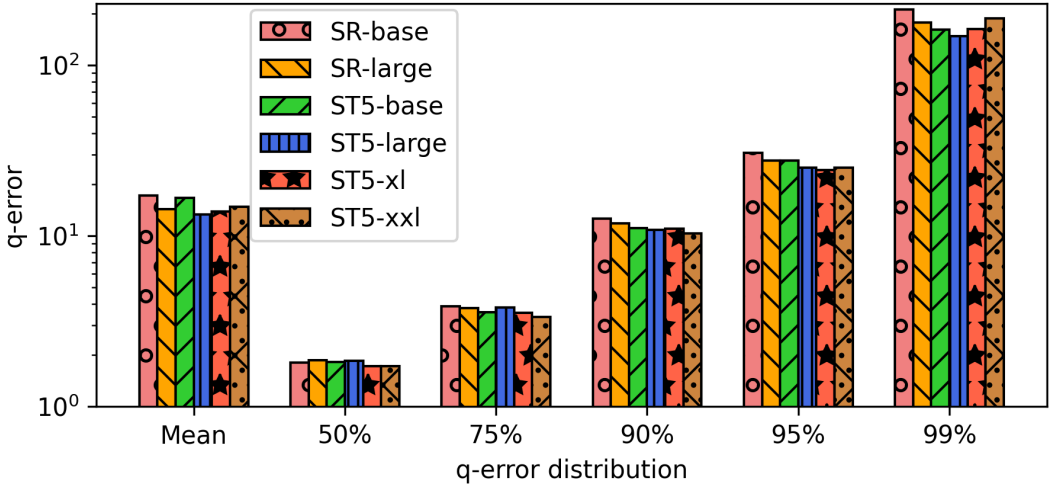


Fig. 4. Performance comparison of six PLMs as the foundation PLM in PLM4NDV.

q-errors depicted in the figure. This suggests that the semantic information extracted by different PLMs directly impacts NDV estimation and underscores the importance of selecting an appropriate PLM in our method for extracting semantic information from the schema.

- For the SR models, using SR-large leads to an improvement in estimation performance than that of SR-base in most metrics. The two variants of SR models have different sizes but similar architectures, SR-large exhibits better performance because it has a larger number of parameters, allowing for more effective semantic representation compared to SR-base, thereby improves the estimation accuracy.
- For the ST5 models, in the majority of test cases (the 90% percentile), increasing the model parameters improves estimation accuracy, except for ST5-xl. Using ST5-xxl as the semantic embedding model has the optimal performance in the 50%, 75%, 90%, and 95% percentile of q-error, while the worse performance in a few cases increases its mean q-error. In general, using ST5-xxl has the optimal performance in most metrics. This shares the same reason as SR-large that a larger model size leads to more effective semantic representations.
- Compared to the SR and ST5 models, we can observe that the ST5 models exhibit better performance than SR models where the number of parameters is similar. There are two possible reasons for the situation. On the one hand, the two PLMs are fine-tuned in different policies and datasets, resulting in different semantic embedding capabilities. On the other hand, T5 has an Encoder-Decoder architecture in the pre-training stage, which may make its encoder (ST5 models) have better semantic representation abilities compared to RoBERTa models, which are trained in the Encoder-only architecture.

Table 6. The 90% quantile of q-error of each method under different data volumes, where “N/A” represents not applicable, and ↓ indicates the performance is inferior to PLM4NDV without data access (underlined).

	n=0	n=10	n=20	n=50	n=0.01N
Goodman	N/A	1.32e4↓	5.53e4↓	3.20e4↓	101.15↓
GEE	N/A	49.90	88.14↓	63.00↓	10.00
EB	N/A	30.07	57.31	39.44	10.14
Chao	N/A	474.21↓	1.64e3↓	775.82↓	100.17↓
Shlosser	N/A	61.00↓	264.28↓	104.00↓	8.11
Jackknife	N/A	265.84↓	1.10e4↓	500.60↓	50.06
Sichel	N/A	389.66↓	1.62e3↓	720.02↓	100.65↓
Bootstrap	N/A	143.71↓	731.00↓	288.57↓	243.50↓
HT	N/A	2.93e4↓	3.24e4↓	2.92e4↓	6.33e3↓
MoM1	N/A	1.67e5↓	5.26e5↓	1.37e5↓	11.88
MoM2	N/A	117.59↓	252.94↓	184.46↓	13.72
MoM3	N/A	148.53↓	253.58↓	188.41↓	1.18e3↓
ChaoLee	N/A	445.40↓	1.92e3↓	848.43↓	98.78↓
SJ	N/A	118.81↓	366.0↓	191.37↓	32.41
LS	N/A	31.08	33.73	24.19	6.48
LS(FT)	N/A	43.49	33.09	22.77	2.04
PLM4NDV	<u>59.51</u>	<u>25.74</u>	<u>20.67</u>	<u>15.25</u>	<u>1.89</u>

- The above analysis demonstrates that the column embeddings obtained from different PLMs significantly impact NDV estimation. Nonetheless, even the lowest performance among the six variants still outperforms all baseline methods in the majority of test cases. Considering both estimation accuracy and model efficiency, we use ST5-large as the foundation PLM in PLM4NDV rather than the best-performing ST5-xxl. This analysis also highlights that effectively fine-tuning PLMs in the context of database schemas may be a promising area for future exploration.
- In addition, the T5-11B version is regarded as a pioneering Large Language Model (LLM) work in the NLP community [97], with ST5-xxl leveraging the encoder component of the T5-11B version. Since the above analysis reveals that the PLMs with more parameters may have better semantic representation abilities that bring better NDV estimation performance, we will investigate and discuss the effect of prompting LLMs with larger sizes for NDV estimation in Section 5.2.

#### 4.5 Access Data Volume Discussion (RQ4)

We compare the performance of each method under different data access volumes and discuss the 90th percentile of q-error to demonstrate their performance in the majority of the test cases, as shown in Table 6. Based on the results presented in the table, we proceed with the following discussions:

**Estimation without data access.** There are scenarios where estimation must be performed without sampling data, particularly in cases where accessing databases is prohibited or the data access cost budget is insufficient even for a single sample. Existing methods are not applicable when data is inaccessible, conversely, PLM4NDV is uniquely suited for this scenario as it can solely utilize the semantic features for estimation. The 90th percentile of the q-error for PLM4NDV when



estimating NDV without data is 59.51. While this indicates a relatively high estimation error under this scenario, the performance of PLM4NDV is significant from two perspectives. First, in the scenario of estimation without data access, our method is the only one that functions. Second, by comparing the results with those obtained from accessing 100 rows in Table 3, it is observed that PLM4NDV without accessing any data can even outperform most methods when accessing 100 rows, whether in sequential or random order. Therefore, the estimation performance of PLM4NDV in the no-data-access scenario is promising.

**Varying sequential access volumes.** We study the performance of baselines and our method for  $n = \{10, 20, 50\}$  to investigate the effect of different sequential access volumes under limited data scenarios. Referring to the results in Table 3 and Table 6, we can make the following observations. For all baseline models, increasing the data volume does not necessarily lead to performance improvements. Besides, LS(FT) shows a performance decline compared to LS when  $n$  is 10 and 20, demonstrating the challenges in estimating under limited data access. On the contrary, the estimation error of PLM4NDV consistently declines as the accessed volume increases, and the best performance in each scenario remains with PLM4NDV.

**Random sampling 1%.** To further examine whether the conclusions regarding PLM4NDV remain valid when random sampling a considerable amount of data, we explore the performance in the scenario of sampling 1% of the full data uniformly at random, which is typically required by sampling-based methods. Given that the frequency profile varies with different data lengths, we use a cut-off of the first 100 frequency profiles in PLM4NDV for this scenario. Referring to the results in Table 3 and Table 6, we can observe that most methods show substantial improvement compared to scenarios with limited data. However, despite randomly sampling a considerable amount of data, the performance of some methods remains inadequate, indicating the need for an even larger sample size. In contrast, PLM4NDV exhibits a significant performance improvement when accessing a considerable amount of data, with a 90% percentile of q-error of only 1.89, indicating that its estimation error can be substantially reduced in the majority of the test cases. This outcome highlights the practicality of incorporating semantic information in PLM4NDV. In addition, the performance of PLM4NDV without data access even beats many methods that randomly sample 1% of the full data, highlighting the distinct promising advantages of PLM4NDV.

The results in Table 3, Table 4, and Table 6 demonstrate that utilizing semantics can enhance NDV estimation in both sequential access and random sampling scenarios.

#### 4.6 Efficiency Analysis (RQ5)

The efficiency of PLM4NDV encompasses two phases: training and inference. Most baseline sampling-based methods are purely statistical and can be applied directly, with only LS involving a training stage. Since PLM4NDV is the only one that utilizes semantic information, there is no direct basis for comparison with other methods in the training phase. Therefore, we intuitively present the time consumption during training. For the inference stage, we demonstrate the average time consumption in each procedure of PLM4NDV, and the results are shown in Table 7. We record the time consumption on a common virtual machine with 8 cores and 16 GB of memory.

**Training.** The cost of constructing the frequency profiles is trivial, the majority of time consumption is using PLMs to obtain the column embeddings and training the learned NDV estimation model. Specifically, the column embedding time is approximately 30s and the training time is about 3100s. The time consumption for column embedding is minimal compared to model training, indicating that utilizing semantic information does not create an efficiency bottleneck. Furthermore, since PLM4NDV captures the semantic features of individual columns and their corresponding table using an attention model, the training process is relatively time-consuming. Nevertheless, the training is conducted offline and updated infrequently, making the cost acceptable in practice.

Table 7. Average time consumption for an estimation on a column in each procedure of PLM4NDV in the inference stage.

data access	statistics	semantics	estimation
~10ms	1ms<	~2ms(GPUs) ~1.23s (CPUs)	1ms<

**Inference.** As shown in Table 7, PLM4NDV significantly reduces data access costs, requiring only 10ms per access because it sequentially accesses a fixed number of samples rather than randomly sampling. Randomly sampling data from databases is extremely time-consuming [62], and this cost increases with the size of the table. In contrast, the data access time of PLM4NDV is agnostic to the table size, consistently requiring about 10ms for each access.

The length of the frequency profiles of PLM4NDV is unaffected by the table size, allowing for high efficiency in statistics construction. It takes less than 1ms for a column.

PLM4NDV model involves two parts: semantic embedding and the learnable component estimation. It takes about 1.75s to load the PLM from disk for the first time. The average PLM embedding time is approximately 2ms on GPUs and about 1.23s on CPUs for a column. The average inference time for a column of the learnable model is less than 1ms on both GPUs and CPUs.

**Efficiency of baselines.** The time consumption of baselines in the estimation stage is minimal, as each method takes only a few milliseconds to estimate the NDV of a column using the frequency profiles as input. Although the baselines appear to have higher efficiency in the estimation phase compared to PLM4NDV, their efficiency in random sampling is extremely low, leading to overall efficiency that is inferior to our approach.

**Practicality analysis.** In practice, the model is trained offline, meaning that inference efficiency is the key factor determining its practicality. Additionally, the column semantic embeddings can be computed independently before estimating NDV, as the number of schemas within a database is typically limited, and the storage requirements of column embeddings are finite. Consequently, the computational cost is acceptable (even when using CPUs), as each column needs to be computed only once for storage and can be pre-computed during idle time, allowing for direct querying of column embeddings from the storage during NDV estimation. Ultimately, the time consumption of PLM4NDV in online NDV estimation is minimal, with the average end-to-end estimation for a column taking only tens of milliseconds on CPUs. Therefore, even with the use of PLMs to capture semantic information in the schema, PLM4NDV maintains promising practicality.

**Limitations of PLM4NDV.** PLM4NDV may be impractical for extremely wide tables due to memory limitations, as the space complexity of the column interaction component is  $O(t^2)$ , where  $t$  is the number of columns of a table. However, the occurrence of such scenarios is infrequent in practical applications. Besides, PLM4NDV may struggle to meet the real-time NDV estimation requirements when it needs to compute semantic embeddings without GPUs.

#### 4.7 Study of Data Layout (RQ6)

We construct a synthesis table with two columns: `EmployeeID` and `Office Address`, where the data types are `int` and `string`, respectively. We investigate the impact of the data layout of `Office Address` column on NDV estimation. Denote  $p = \frac{D}{N}$  is the selectivity of the column. Assuming that  $N = 10,000$  and we set a relatively high selectivity  $p = 0.7$  such that sequentially accessing the first 100 rows can introduce bias in NDV estimation. In this scenario, we consider that the first 3,001 rows share the same office address.

To explore the impact of data layout on the sequential access scenario, we gradually replace the first 100 rows one by one by randomly selecting a row from rows 101 to 10,000. PLM4NDV is trained

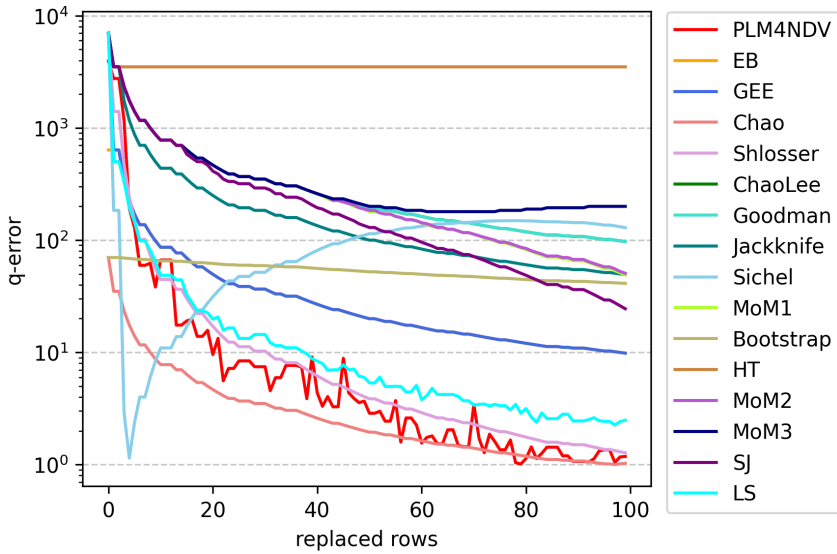


Fig. 5. Estimation error variation with the number of replaced rows in the first 100 rows of the synthesis table.

on the training set as described in Section 4.1. The performance of baselines and PLM4NDV is shown in Figure 5 and we can derive the following conclusions:

- The data layout of the table significantly affects the performance of most NDV estimation methods. For the synthesis column with a selectivity of 0.7, when the office addresses of the first 100 rows are identical (i.e., the number of replaced rows is 0), it can lead to significant estimation errors in most methods.
- When increasing the randomness of the layout in the first 100 rows (i.e, increasing the number of replaced rows), some methods show an estimation error decrease, while a few methods remain unchanged. Besides, some methods may initially decrease and then increase, indicating that some baselines are more sensitive to data layout. The proposed method PLM4NDV tends to achieve a substantial reduction in estimation error and gradually outperforms most baselines.
- Estimating NDV under sequential access the top 100 rows is significantly influenced by data layout. However, based on the experimental results presented in Table 3, when the sampling budget allows accessing limited data, PLM4NDV emerges as the most effective alternative among existing methods.

### 5 Preliminary Exploration of LLMs

Large language models (LLMs) such as GPT4 [10] and Llama [87] have achieved notable success in NLP, demonstrating impressive zero-shot reasoning abilities across various domains and scenarios owing to their robust generalization and adaptability. These models possess significantly larger parameters than earlier PLMs like RoBERTa [57] and T5 [73], which were studied in this paper. Given these advancements, it is natural to inquire whether LLMs can be leveraged to provide better semantic understandings and improve NDV estimation accuracy.

Fully leveraging LLMs in databases involves costly prompt engineering or even fine-tuning [36, 39, 40, 48, 69, 83, 98–100], which is not the focus of this paper. In this section, we conduct a preliminary exploration of LLMs to validate the significance of utilizing semantics in NDV estimation and show some insightful findings for future works.

```

You are a data expert, your task is to estimate the number of distinct values of
a column by the following rules:
(1) The number of distinct values of the column must be smaller than or equal to
the total rows.
...

Here are the target column information:
Column name: [ColumnName]. Column type: [ColumnType].
Constraints: [ColumnConstraints]. Comment: [ColumnComment]. Total rows:
[ColumnSize].
Data samples: [Samples]
Information of the table: [TableInfo]
<Output Format>
{
  "Reason": "XXX",
  "NDV": "XXX"
}
<\Output Format>
Try your best to guess the number of distinct values of the column. In any case,
give an exact number of NDV. Let's think step by step in order to produce the
answer. The number of distinct values of the column is:

```

Listing 2. Template of prompting LLMs for NDV estimation. Some rules are omitted for space limitations.

## 5.1 Prompting LLMs for NDV Estimation

One of the most fascinating aspects of LLMs is their ability to move beyond the fine-tuning paradigm [29] that was often required by PLMs to adapt to downstream tasks. Instead, LLMs have shifted towards the prompt-tuning paradigm [56]. There are multiple categories of methods in the prompt-tuning paradigm, we leverage the widely used *tuning-free prompting* paradigm (**prompting** for brevity) to adapt LLMs in the task of NDV estimation. Given that NDV estimation is not a typical NLP task, we formulate NDV estimation as a natural language question by constructing a simple prompt template, and the LLMs are requested to answer the question based on the provided schema and sampled data records.

**Prompt template.** The prompt template construct in this paper is illustrated in Listing 2, where the slots are enclosed in square brackets, and each slot is designed to be filled with specific variables. The column information corresponds to the schema and `ColumnSize` represents the size of the column. The `Samples` and `TableInfo` represent the accessed samples in the column and the information of other columns within the same table, respectively. Due to space limitations, some textual rules have been omitted. The complete prompt template is provided in the supplementary material.

**Prompting and results.** The prompt can be generated by filling in the prompt template slots with information from the target column, after which the prompt is input into the LLMs to obtain the results. The basic principle of LLM inference involves autoregressively generating the next token with the highest probability based on the input prompt, continuing until an end-of-sentence marker is produced. We can derive the final estimation and the reason from the sentence generated by the LLM.

## 5.2 Experiments and Discussion

**LLM setup.** We employ several SOTA open-source and commercial LLMs to investigate their performance in the task of NDV estimation. Specifically, we use the instruct version of Llama 3.1 [1] with 8B and 70B parameters, denoted as Llama31-8B and Llama31-70B; the instruct version of Qwen2 [95] 7B and 72B parameters, denoted as Qwen2-7B and Qwen2-72B; as well as OpenAI

Table 8. Performance of prompting LLMs under sequential access 100 rows condition.

	Mean	50%	75%	90%	95%	99%
Llama31-8B	3.5e6	6.15	82.81	602.12	1.87e3	1.29e4
Qwen2-7B	2.5e5	9.14	102.55	847.39	4.00e3	4.45e4
Llama31-70B	213.8	2.00	9.11	59.63	191.85	1.73e3
Qwen2-72B	230.6	2.16	10.85	74.0	232.19	2.64e3
GPT3.5	5e14	4.17	31.04	170.28	515.26	1.02e4
GPT4o	79.25	2.00	6.00	28.53	90.73	1.09e3
PLM4NDV	13.33	1.86	3.81	10.81	25.18	148.76

GPT3.5 (gpt-35-turbo-0125) [2] and GPT4o (gpt-4o-2024-05-13) [3]. The prompt of each LLM is using the template in Listing 2 filled with information of each column. Due to the cost issues associated with invoking LLMs, we conduct experiments solely in the limited sequential data access scenarios. **Overall results.** The overall performance of the LLMs is presented in Table 8. Based on the results in Table 8 and the results from sequential access conditions reported in Table 3 and Table 4, we can derive the following conclusions:

- LLMs are applicable to NDV estimation. By using prompting methods, the NDV estimation task can be transformed into a natural language question, allowing for the estimation to be accomplished by leveraging the capabilities of LLMs.
- Among the six LLMs, GPT4o demonstrates the best overall performance. Additionally, the larger-sized LLMs (Llama31-70B and Qwen2-72B) perform significantly better than their smaller counterparts (Llama31-8B and Qwen2-7B) for the same architecture. The overall performance of NDV estimation using LLMs is closely related to their semantic understanding and reasoning capabilities.
- The performance of prompting GPT4o surpasses most baselines in most metrics. Particularly, the 75% percentile of q-error is 6.00, demonstrating its superior accuracy compared to the baselines in most test cases. In addition, GPT4o and Llama31-70B outperform all the baseline methods at the 50% percentile of q-error. This suggests that prompting LLMs may derive outstanding results in considerable test cases, however, poor estimation outcomes in a few cases lead to overall suboptimal performance.
- For the LLMs having poor overall performance (Llama31-8B, Qwen2-7B, and GPT3.5), the mean error is significantly higher than the 99% percentile. In addition, the mean q-error is greater than the 90% percentile for all LLMs, indicating that their estimation errors are substantial in extreme scenarios, contributing to this phenomenon.
- PLM4NDV outperforms all the LLMs across all metrics. This can be attributed to the fact that PLM4NDV is trained for the NDV estimation task using the training datasets, whereas the LLMs are not. Fine-tuning LLMs for the NDV estimation task using existing data with schema, sample records, and ground truth NDV is a non-trivial task, as there are no intermediate results to map them together. The underlining rules can be very complex as shown in Equation (1) and Equation (2). PLM4NDV learns this mapping from the training data, incorporating semantic information, which contributes to its superior performance.

**Case study.** To intuitively demonstrate how we derive the estimation  $\hat{D}$  from the LLM output and how the LLM performs reasoning based on the prompts, we present an output case in Listing 3. From the JSON format output in the listing, we can obtain the estimation  $\hat{D} = 1000$  along with the reasoning process behind the estimation.

```

{
  "Reason": "The column name 'batting-position' suggests that it represents the
batting position of a player in a sports game, likely cricket or baseball. In
these sports, batting positions typically range from 1 to 11. However,
considering the total rows (37963), it is unlikely that there are 37963 unique
batting positions. A reasonable estimate would be around 1000 distinct batting
positions, assuming there are multiple players and games represented in the
dataset.",
  "NDV": "1000"
}

```

Listing 3. A case study of LLM for NDV estimation

By examining the reasons in the LLM output, we find that, on the one hand, LLM can infer hidden semantic information and specific meanings based solely on the input column names, even identifying the ground-truth NDV range (from 1 to 11). On the other hand, providing the total row count is necessary for the LLM to address the NDV estimation problem by answering the natural language question. However, due to potential limitations in the reasoning ability of the LLM itself or the suboptimal engineering of the prompt template, the LLM may understand the semantics of the column but still fail to present an accurate estimation.

**Lessons.** From the above observations, we can derive the following key insights.

- LLMs show great potential in solving NDV estimation tasks, as evidenced by the performance of three LLMs (Llama31-70B, Qwen2-72B, and GPT4o), which achieve a q-error of at most 2.16 in the 50th percentile. The performance of the three LLMs beat most baselines in Table 3 in the 50th percentile.
- The case study directly demonstrates the benefits of semantic information. The reasons provided in the LLM output indicate that the LLM can extract semantic information from the schema, which aids in NDV estimation. It can infer the meaning and the value domain based on the column name.
- However, prompting LLMs for NDV estimation may be limited by their reasoning abilities, as their estimations are very poor in a few scenarios. LLMs may understand the meaning of the column but is difficult to estimate NDV, as shown in the case study.
- Fine-tuning LLMs for NDV estimation is challenging due to the lack of intermediate results linking the schema, sample records, and ground truth NDV. This makes it difficult to train the model to understand these relationships, requiring careful structuring of the training data.
- Therefore, incorporating LLMs with existing NDV estimation techniques may be a challenging and interesting topic for future research.

## 6 Related Work

Our work is positioned at the intersection of NDV estimation and the application of PLMs/LLMs.

### 6.1 NDV Estimation

**Sketch-based methods.** Sketch-based methods need to access the entire data column to maintain a memory-efficient sketch and then estimate the NDV from the sketch [31, 34, 37, 91]. Sketch-based methods can be classified into two subcategories. *Logarithmic hashing* methods use a bitmap and a hash function to track the most uncommon element observed so far, and the hash function maps each element with a probability varying by the bit significance. The representative methods in this subcategory are FM [35], PCSA [35], AMS [11], LogLog [31], SuperLogLog [31], HyperLogLog [34], HyperLogLog++ [47], and UltraLogLog [33]. *Uniform hashing* methods use a uniform hash function to hash the entire dataset into an *interval* or a set of *buckets*. The former is based on how

packed the interval is, and the representative methods are Cohen [26], BJKST1 [12], AKMV [13], and MinCount [37]. The latter is based on the probability that a bucket is (non)empty, and the typical methods are BJKST2 [12], Linear Counting [91], and Bloom Filter [65].

**Sampling-based methods.** Since sampling-based methods utilize a portion of IID samples rather than the full data for estimation, most of these methods are based on various heuristics formulated from specific assumptions [38]. For instance, Chao [19, 64] and ChaoLee [20] assume the size of the column is infinite, Shlosser [79] supposes there are certain conditions of data skewness, Horvitz-Thompson (HT) [76] and Bootstrap [84] hypothesize the distinct elements in the sample are distinct in the full dataset, Method of Movement (MoM1) [16] postulates the column size is infinite and the frequency of each distinct element is constant, the variant of MoM1 (MoM2) [16] removes the infinite size assumption, and another variant of MoM1 (MoM3) [16] assumes the frequency of each distinct element are unequal. GEE [21] is constructed to match the theoretical lower bound of estimation error. The Error Bound (EB) [23] estimator takes the estimation in the sampling process and claims its estimation error can be bounded. Sichel [80–82] aims to fit a zero-truncated generalized inverse Gaussian-Poisson distribution to estimate NDV. Jackknife [17, 18] and Smoothed Jackknife (SJ) [41] assume the existence of a non-zero constant that can satisfy an expression to correct the estimation error iteratively. Such harsh assumptions can hardly be satisfied in practice, resulting in large estimation errors. In recent years, Machine Learning (ML) techniques [9, 44, 63, 66, 93] have been introduced in NDV estimation [52, 92, 94]. Besides, numerous studies have made much effort to improve the efficiency of sampling in databases [22–25, 30, 75].

## 6.2 Pre-trained Language Models

PLMs [50, 71, 72] have achieved SOTA results in various NLP tasks. In recent years, large-scale PLMs, known as Large Language Models (LLMs), have emerged, characterized by their extensive parameters and remarkable advancements. Refer to [42, 97] for a comprehensive review, in this paper, we focus on their applications in databases.

**PLMs in Databases.** The most representative application of PLMs in databases is Text-to-SQL, for which a comprehensive literature review is provided in [70]. Additionally, the database community has integrated PLMs into various data management and database tasks. For instance, Li et al. [54] use PLMs as a base model for entity matching, Deng et al. [28] utilize PLMs as pre-training frameworks for table understanding, and Suhara et al. [86] leverage PLMs to annotate columns in databases.

**LLMs in Databases.** Numerous recent works in Text-to-SQL based on LLM have emerged [36, 39, 40, 69]. Leveraging the advantages of LLMs in few-shot and zero-shot performance, these works have led to innovative approaches in database management. For instance, FormaT5 [83] utilizes LLM for table formatting, GENREWRITE [55] uses LLM to rewrite SQL for performance improvement, and LLMTune [48] accelerates the knob tuning process using LLMs. Furthermore, recent efforts have attempted to build holistic LLM-centric database optimization and diagnosis systems [98–100].

## 7 Conclusion

In this paper, we aim to minimize data access in NDV estimation and present several distinct contributions, marking a significant advancement in the field. We propose PLM4NDV, the pioneer NDV estimation method that leverages schema information using Pre-trained Language Models. We are the first to highlight the benefits of the historically ignored schema information in NDV estimation. By integrating the schema semantics, our method substantially augments the accuracy of NDV estimation under limited sampling cost budgets. Besides, we show that PLM4NDV can establish the estimation even without data access and achieve promising performance. Consequently, we

provide a third NDV estimation paradigm that differs from existing sketch-based and sampling-based approaches. Our extensive empirical studies on a large-scale real-world dataset demonstrate that PLM4NDV significantly reduces the data access and improves NDV estimation accuracy compared to baselines.

**Future work.** Our work showcases the promising new opportunities in NDV estimation and raises several open research questions. For instance, how to serialize column schemas to obtain more effective semantic embeddings, generate the missing optional components in the schema and leverage them, discern the impact of data types, fine-tune PLMs in the context of database schema, and effectively leverage LLMs for NDV estimation. We look forward to exploring these questions with the community in the future.

## References

- [1] 2024. Llama 3.1 model. <https://llama.meta.com/>
- [2] 2024. OpenAI ChatGPT. <https://openai.com/chatgpt/>
- [3] 2024. OpenAI GPT-4o. <https://openai.com/index/hello-gpt-4o/>
- [4] 2024. Pydistinct - Population Distinct Value Estimators. <https://pydistinct.readthedocs.io/>
- [5] 2024. Source Code of MySQL. [https://github.com/mysql/mysql-server/blob/824e2b4064053f7daf17d7f3f84b7a3ed92e5fb4/sql/join\\_optimizer/cost\\_model.cc](https://github.com/mysql/mysql-server/blob/824e2b4064053f7daf17d7f3f84b7a3ed92e5fb4/sql/join_optimizer/cost_model.cc)
- [6] 2024. Source Code of PostgreSQL. [https://github.com/postgres/postgres/blob/master/src/backend/optimizer/plan\\_analyzejoins.c](https://github.com/postgres/postgres/blob/master/src/backend/optimizer/plan_analyzejoins.c)
- [7] 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>
- [8] 2024. tablib-v1-sample dataset. <https://huggingface.co/datasets/approximatelabs/tablib-v1-sample>
- [9] 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.
- [10] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774* (2023).
- [11] Noga Alon, Yossi Matias, and Mario Szegedy. 1996. The space complexity of approximating the frequency moments. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*. 20–29.
- [12] Ziv Bar-Yossef, TS Jayram, Ravi Kumar, D Sivakumar, and Luca Trevisan. 2002. Counting distinct elements in a data stream. In *Randomization and Approximation Techniques in Computer Science: 6th International Workshop, RANDOM 2002 Cambridge, MA, USA, September 13–15, 2002 Proceedings 5*. Springer, 1–10.
- [13] Kevin Beyer, Peter J Haas, Berthold Reinwald, Yann Sismanis, and Rainer Gemulla. 2007. On synopses for distinct-value estimation under multiset operations. In *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*. 199–210.
- [14] John S Bridle. 1990. Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition. In *Neurocomputing: Algorithms, architectures and applications*. Springer, 227–236.
- [15] Jake D Brutlag and Thomas S Richardson. 2002. A block sampling approach to distinct value estimation. *Journal of Computational and Graphical Statistics* 11, 2 (2002), 389–404.
- [16] John Bunge and Michael Fitzpatrick. 1993. Estimating the number of species: a review. *Journal of the American statistical Association* 88, 421 (1993), 364–373.
- [17] 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.
- [18] 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.
- [19] Anne Chao. 1984. Nonparametric estimation of the number of classes in a population. *Scandinavian Journal of statistics* (1984), 265–270.
- [20] 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.
- [21] 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.



- [22] Surajit Chaudhuri, Gautam Das, and Utkarsh Srivastava. 2004. Effective use of block-level sampling in statistics estimation. In *Proceedings of the 2004 ACM SIGMOD international conference on Management of data*. 287–298.
- [23] 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.
- [24] Xingguang Chen, Fangyuan Zhang, and Sibow Wang. 2022. Efficient Approximate Algorithms for Empirical Variance with Hashed Block Sampling. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (Washington DC, USA) (KDD '22)*. Association for Computing Machinery, New York, NY, USA, 157–167. doi:10.1145/3534678.3539377
- [25] Xiang Ci and Xiaofeng Meng. 2015. An efficient block sampling strategy for online aggregation in the cloud. In *Web-Age Information Management: 16th International Conference, WAIM 2015, Qingdao, China, June 8-10, 2015. Proceedings 16*. Springer, 362–373.
- [26] Edith Cohen. 1997. Size-estimation framework with applications to transitive closure and reachability. *J. Comput. System Sci.* 55, 3 (1997), 441–453.
- [27] Reuven Cohen and Yuval Nezi. 2019. Cardinality estimation in a virtualized network device using online machine learning. *IEEE/ACM Transactions on Networking* 27, 5 (2019), 2098–2110.
- [28] Xiang Deng, Huan Sun, Alyssa Lees, You Wu, and Cong Yu. 2022. Turl: Table understanding through representation learning. *ACM SIGMOD Record* 51, 1 (2022), 33–40.
- [29] Ning Ding, Yujia Qin, Guang Yang, Fuchao Wei, Zonghan Yang, Yusheng Su, Shengding Hu, Yulin Chen, Chi-Min Chan, Weize Chen, et al. 2022. Delta tuning: A comprehensive study of parameter efficient methods for pre-trained language models. *arXiv preprint arXiv:2203.06904* (2022).
- [30] Arnaud Doucet, Mark Briers, and Stéphane Sénécal. 2006. Efficient block sampling strategies for sequential Monte Carlo methods. *Journal of Computational and Graphical Statistics* 15, 3 (2006), 693–711.
- [31] Marianne Durand and Philippe Flajolet. 2003. Loglog counting of large cardinalities. In *Algorithms-ESA 2003: 11th Annual European Symposium, Budapest, Hungary, September 16-19, 2003. Proceedings 11*. Springer, 605–617.
- [32] Gus Eggert, Kevin Huo, Mike Biven, and Justin Waugh. 2023. TabLib: A Dataset of 627M Tables with Context. arXiv:2310.07875 [cs.CL]
- [33] 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.
- [34] 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).
- [35] Philippe Flajolet and G Nigel Martin. 1985. Probabilistic counting algorithms for data base applications. *Journal of computer and system sciences* 31, 2 (1985), 182–209.
- [36] Dawei Gao, Haibin Wang, Yaliang Li, Xiuyu Sun, Yichen Qian, Bolin Ding, and Jingren Zhou. 2023. Text-to-sql empowered by large language models: A benchmark evaluation. *arXiv preprint arXiv:2308.15363* (2023).
- [37] Frédéric Giroire. 2009. Order statistics and estimating cardinalities of massive data sets. *Discrete Applied Mathematics* 157, 2 (2009), 406–427.
- [38] 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.
- [39] Zihui Gu, Ju Fan, Nan Tang, Lei Cao, Bowen Jia, Sam Madden, and Xiaoyong Du. 2023. Few-shot text-to-sql translation using structure and content prompt learning. *Proceedings of the ACM on Management of Data* 1, 2 (2023), 1–28.
- [40] Zihui Gu, Ju Fan, Nan Tang, Songyue Zhang, Yuxin Zhang, Zui Chen, Lei Cao, Guoliang Li, Sam Madden, and Xiaoyong Du. 2023. Interleaving pre-trained language models and large language models for zero-shot nl2sql generation. *arXiv preprint arXiv:2306.08891* (2023).
- [41] 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.
- [42] Xu Han, Zhengyan Zhang, Ning Ding, Yuxian Gu, Xiao Liu, Yuqi Huo, Jiezhong Qiu, Yuan Yao, Ao Zhang, Liang Zhang, et al. 2021. Pre-trained models: Past, present and future. *AI Open* 2 (2021), 225–250.
- [43] 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).
- [44] Yi Hao and Alon Orlitsky. 2019. Unified sample-optimal property estimation in near-linear time. *Advances in Neural Information Processing Systems* 32 (2019).
- [45] Hazard Harmouch and Felix Naumann. 2017. Cardinality estimation: An experimental survey. *Proceedings of the VLDB Endowment* 11, 4 (2017), 499–512.
- [46] Xiao He, Jian Tan, Bin Wu, Feifei Li, Xinping Zhang, Gaozhong Liang, and Jinfeng Xu. 2023. Active Sampling for Sparse Table by Bayesian Optimization with Adaptive Resolution. In *39th IEEE International Conference on Data Engineering, ICDE 2023, Anaheim, CA, USA, April 3-7, 2023*. IEEE, 816–828. doi:10.1109/ICDE55515.2023.00068

- [47] Stefan Heule, Marc Nunkesser, and Alexander Hall. 2013. Hyperloglog in practice: Algorithmic engineering of a state of the art cardinality estimation algorithm. In *Proceedings of the 16th International Conference on Extending Database Technology*. 683–692.
- [48] Xinmei Huang, Haoyang Li, Jing Zhang, Xinxin Zhao, Zhiming Yao, Yiyang Li, Zhuohao Yu, Tieying Zhang, Hong Chen, and Cuiping Li. 2024. LLMtune: Accelerate Database Knob Tuning with Large Language Models. *arXiv preprint arXiv:2404.11581* (2024).
- [49] Madelon Hulsebos, Çağatay 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.
- [50] Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of NAACL-HLT*. 4171–4186.
- [51] 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*.
- [52] Jiajun Li, Runlin Lei, Sibowang, 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.
- [53] 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.
- [54] Yuliang Li, Jinfeng Li, Yoshihiko Suhara, Anh Hai Doan, and Wang-Chiew Tan. 2020. Deep entity matching with pre-trained language models. *Proceedings of the VLDB Endowment* 14, 1 (2020), 50–60.
- [55] Jie Liu and Barzan Mozafari. 2024. Query Rewriting via Large Language Models. *arXiv preprint arXiv:2403.09060* (2024).
- [56] Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2023. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *Comput. Surveys* 55, 9 (2023), 1–35.
- [57] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *CoRR* abs/1907.11692 (2019). arXiv:1907.11692 <http://arxiv.org/abs/1907.11692>
- [58] 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.
- [59] 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.
- [60] 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.
- [61] Jianmo Ni, Gustavo Hernandez Abrego, Noah Constant, Ji Ma, Keith B Hall, Daniel Cer, and Yinfei Yang. 2021. Sentence-t5: Scalable sentence encoders from pre-trained text-to-text models. *arXiv preprint arXiv:2108.08877* (2021).
- [62] Frank Olken. 1993. *Random sampling from databases*. Ph.D. Dissertation. Citeseer.
- [63] 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.
- [64] 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.
- [65] Odysseas Papapetrou, Wolf Siberski, and Wolfgang Nejdl. 2010. Cardinality estimation and dynamic length adaptation for bloom filters. *Distributed and Parallel Databases* 28 (2010), 119–156.
- [66] Dmitri S Pavlichin, Jiantao Jiao, and Tsachy Weissman. 2019. Approximate profile maximum likelihood. *Journal of Machine Learning Research* 20, 122 (2019), 1–55.
- [67] Gan Peng, Peng Cai, Kaikai Ye, Kai Li, Jinlong Cai, Yufeng Shen, and Han Su. 2023. A Data-Driven Index Recommendation System for Slow Queries. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*. 5086–5090.
- [68] Gan Peng, Peng Cai, Kaikai Ye, Kai Li, Jinlong Cai, Yufeng Shen, Han Su, and Weiyuan Xu. 2024. Online Index Recommendation for Slow Queries. In *2024 IEEE 40th International Conference on Data Engineering (ICDE)*. IEEE, 5294–5306.
- [69] Mohammadreza Pourreza and Davood Rafiei. 2024. Din-sql: Decomposed in-context learning of text-to-sql with self-correction. *Advances in Neural Information Processing Systems* 36 (2024).
- [70] Bowen Qin, Binyuan Hui, Lihan Wang, Min Yang, Jinyang Li, Binhua Li, Ruiying Geng, Rongyu Cao, Jian Sun, Luo Si, et al. 2022. A survey on text-to-sql parsing: Concepts, methods, and future directions. *arXiv preprint arXiv:2208.13629* (2022).

- [71] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. 2018. Improving language understanding by generative pre-training. (2018).
- [72] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog* 1, 8 (2019), 9.
- [73] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research* 21, 140 (2020), 1–67.
- [74] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*.
- [75] Viktor Sanca, Periklis Chrysogelos, and Anastasia Ailamaki. 2023. LAQy: Efficient and Reusable Query Approximations via Lazy Sampling. *Proceedings of the ACM on Management of Data* 1, 2 (2023), 1–26.
- [76] Carl-Erik Särndal, Bengt Swensson, and Jan Wretman. 1992. Model Assisted Survey Sampling. *Springer Series in Statistics* (1992).
- [77] P Griffiths Selinger, Morton M Astrahan, Donald D Chamberlin, Raymond A Lorie, and Thomas G Price. 1979. Access path selection in a relational database management system. In *Proceedings of the 1979 ACM SIGMOD international conference on Management of data*. 23–34.
- [78] Jiachen Shi, Gao Cong, and Xiao-Li Li. 2022. Learned index benefits: Machine learning based index performance estimation. *Proceedings of the VLDB Endowment* 15, 13 (2022), 3950–3962.
- [79] 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.
- [80] 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.
- [81] Herbert S Sichel. 1986. Word frequency distributions and type-token characteristics. *Math. Scientist* 11 (1986), 45–72.
- [82] 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.
- [83] Mukul Singh, José Cambronero, Sumit Gulwani, Vu Le, Carina Negreanu, Elnaz Nouri, Mohammad Raza, and Gust Verbruggen. 2023. FormaT5: Abstention and Examples for Conditional Table Formatting with Natural Language. *Proceedings of the VLDB Endowment* 17, 3 (2023), 497–510.
- [84] Eric P Smith and Gerald van Belle. 1984. Nonparametric estimation of species richness. *Biometrics* (1984), 119–129.
- [85] Bruce Spang and Nick McKeown. 2019. On estimating the number of flows. In *Stanford Workshop on Buffer Sizing*.
- [86] Yoshihiko Suhara, Jinfeng Li, Yuliang Li, Dan Zhang, Çağatay Demiralp, Chen Chen, and Wang-Chiew Tan. 2022. Annotating columns with pre-trained language models. In *Proceedings of the 2022 International Conference on Management of Data*. 1493–1503.
- [87] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971* (2023).
- [88] 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.
- [89] Paul Valiant and Gregory Valiant. 2013. Estimating the Unseen: Improved Estimators for Entropy and other Properties. *Advances in Neural Information Processing Systems* 26 (2013).
- [90] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).
- [91] Kyu-Young Whang, Brad T Vander-Zanden, and Howard M Taylor. 1990. A linear-time probabilistic counting algorithm for database applications. *ACM Transactions on Database Systems (TODS)* 15, 2 (1990), 208–229.
- [92] 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. doi:10.14778/3489496.3489508
- [93] 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.
- [94] Xianghong Xu, Tieying Zhang, Xiao He, Haoyang Li, Rong Kang, Shuai Wang, Linhui Xu, Zhimin Liang, Shangyu Luo, Lei Zhang, et al. 2025. AdaNDV: Adaptive Number of Distinct Value Estimation via Learning to Select and Fuse Estimators. *arXiv preprint arXiv:2502.16190* (2025).
- [95] An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, et al. 2024. Qwen2 technical report. *arXiv preprint arXiv:2407.10671* (2024).
- [96] Tao Yu, Zhaonian Zou, Weihua Sun, and Yu Yan. 2024. Refactoring Index Tuning Process with Benefit Estimation. *Proceedings of the VLDB Endowment* 17, 7 (2024), 1528–1541.

- [97] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. 2023. A survey of large language models. *arXiv preprint arXiv:2303.18223* (2023).
- [98] Xuanhe Zhou, Guoliang Li, and Zhiyuan Liu. 2023. Llm as dba. *arXiv preprint arXiv:2308.05481* (2023).
- [99] Xuanhe Zhou, Guoliang Li, Zhaoyan Sun, Zhiyuan Liu, Weize Chen, Jianming Wu, Jiesi Liu, Ruohang Feng, and Guoyang Zeng. 2024. D-Bot: Database Diagnosis System using Large Language Models. *Proc. VLDB Endow.* 17, 10 (aug 2024), 2514–2527. doi:10.14778/3675034.3675043
- [100] Xuanhe Zhou, Xinyang Zhao, and Guoliang Li. 2024. LLM-Enhanced Data Management. *arXiv preprint arXiv:2402.02643* (2024).

Received October 2024; revised January 2025; accepted February 2025