Bridging the Gap: Cardinality Estimation for Semantic Queries on Unstructured Data

Shihui Xu
Tsinghua University
China
xsh23@mails.tsinghua.edu.cn

Jiayi Wang
Tsinghua University
China
jiayi-wa20@mails.tsinghua.edu.cn

Guoliang Li
Tsinghua University
China
liguoliang@tsinghua.edu.cn

Abstract

Large language models (LLMs) have revolutionized the semantic processing of unstructured data, unlocking new avenues for data analytics beyond traditional relational databases. However, optimizing query execution over unstructured data requires accurate semantic cardinality estimation, a problem that remains unsolved due to the absence of schemas and the high cost of semantic judgments. In this paper, we present `SemStats`, a framework that efficiently and accurately estimates the number of results for semantic queries. `SemStats` first constructs a semantic catalog that identifies core semantic dimensions and their relationships within the corpus. Based on the catalog, `SemStats` then builds a semantic index that constructs the relationships between data points and semantic catalog. As it is expensive to assess the relationships between data points and catalog using LLMs, we propose specialized lightweight models to improve index-construction cost. For online estimation, `SemStats` employs stratified importance sampling guided by the semantic index and a lightweight judge model to refine query-specific evaluations. Experiments on multiple real-world datasets show that `SemStats` substantially reduces estimation error (up to 25×) and latency (up to 31×) compared to state-of-the-art baselines, enabling accurate and efficient cardinality estimation of semantic queries on unstructured data.

CCS Concepts

• **Information systems → Query optimization**.

Keywords

Cardinality Estimation, Semantic Query, Unstructured Data

1 Introduction

Large language models (LLMs) have significantly enhanced the semantic processing capability of unstructured data, unlocking new possibilities for data analytics beyond traditional relational databases [5, 8, 23–27, 29, 39, 40, 44–46, 51]. However, even though an appropriate query plan for unstructured data analytics can be constructed, optimizing its execution cost remains a significant challenge. Query plans typically consist of multiple semantic operators, many of which, such as semantic filters and joins, can be reordered into logically equivalent plans with significantly different execution costs. These cost differences mainly stem from the differences in intermediate result sizes, which determine the amount of data to be processed by subsequent operators. Therefore, effective query optimization directly relies on accurate estimations of intermediate result sizes, i.e., *semantic cardinality estimation*, to select an efficient plan for execution. However, semantic cardinality estimation over unstructured data is still a complex and unresolved problem [44].

Unlike relational databases, where fixed schemas facilitate precise cost and cardinality estimations, unstructured data presents challenges due to its lack of strict schemas and varied representation formats (see Table 1). Additionally, obtaining values for unstructured data typically involves complex semantic judgments, incurring high computational cost. An intuitive approach is to use uniform sampling to estimate the selectivity of a semantic query [26, 35]. This involves first sampling a subset of data, then executing the query on the sampled data, and finally estimating the selectivity based on the proportion of data that satisfies the query. However, this method has two limitations. First, performing random sampling can be time-consuming and costly when dealing with large-scale data. Second, if only a few samples satisfy the query, the results may be biased, potentially leading to catastrophic errors [37, 42, 43].

Central Problem. The primary focus of this paper is to address the following problem: *Given a collection of unstructured data, how can we accurately and efficiently estimate the number of data points that satisfy an online semantic query?*

For instance, in a corpus of sports news, consider the query: "How many articles discuss results of competitions in the USA?" Directly estimating the cardinality by semantically analyzing each data point can be accurate but incurs a high cost. In contrast, if relevant semantic dimensions (e.g., sports event, country) are pre-identified, the cardinality can be efficiently estimated by reasoning over their corpus-level data distributions, which significantly reduces cost.

Key idea. To solve this problem, we propose `SemStats` to efficiently and accurately estimate the cardinality of semantic queries. The key idea of `SemStats` is to construct semantic statistics that accurately reflects the data distributions of the key semantic dimensions within the data corpus. Query cardinality will then be effectively approximated based on the statistics.

Challenges. However, due to the high flexibility inherent in unstructured data, constructing semantic statistics for semantic cardinality estimation presents several challenges: what statistics to build, how to obtain their distribution information, and how to utilize the statistics for semantic cardinality estimation.

(C1) How to build accurate semantic statistics for the corpus? Unstructured data lacks a standardized schema and exhibits multiple semantic dimensions, each with varying levels of representativeness and significance. Building accurate semantic statistics therefore requires automatically identifying these dimensions from

Authors' Contact Information: Shihui Xu, Tsinghua University, China, xsh23@mails.tsinghua.edu.cn; Jiayi Wang, Tsinghua University, China, jiayi-wa20@mails.tsinghua.edu.cn; Guoliang Li, Tsinghua University, China, liguoliang@tsinghua.edu.cn.

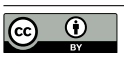

Table 1: Comparison of cardinality estimation: structured vs unstructured.

	Structured	Unstructured (SemStats)
Metadata	Schema about table, column, relationships and constraints	Semantic catalog representing semantic dimensions and their interrelationships
Query Type	SQL queries that strictly adhere to the schema	Natural language queries with flexible semantic filtering conditions
Distribution	Histograms and similar methods to capture the frequency of column values	Semantic statistics recording the distribution of data across semantic dimensions
Estimation Methods	Statistical synopsis, sampling or ML-based models	Sampling based on semantic statistics with semantic judgment by small models

the corpus and selecting the key ones that encapsulate the core information. This process should be unsupervised and data-driven, since users do not necessarily know the semantic dimensions of the data, which makes the statistics construction more challenging due to the diverse and flexible representations of the unstructured data. For instance, in a sports news corpus, it is nontrivial to decide which semantic dimensions (e.g., *sports event*, *country of the news*) should be precomputed for effective online estimation.

(C2) How to obtain data distribution of the semantic dimensions efficiently? Cardinality estimation involves estimating the number of data points that satisfy a query, which requires not only identifying the relevant semantic dimensions but also determining how many data points fall within these dimensions. Unlike structured data in relational databases, where distribution is relatively inexpensive to obtain due to straightforward attribute value extraction, deriving data distribution (e.g., the number of data points in specified semantic dimensions) from unstructured data is much more costly. This is because analyzing semantic dimensions requires complex semantic judgments between data points and semantic dimensions. Therefore, the second challenge lies in efficiently processing the corpus to obtain the data distribution. For instance, even after identifying the *sports event* dimension, it remains prohibitively costly to rely fully on LLMs to extract the specific event mentioned in each document across the large corpus.

(C3) How to conduct semantic cardinality estimation effectively with the semantic statistics? Even with structured semantic statistics and the distribution information, it remains challenging to estimate the result sizes for arbitrary natural language queries, since the query predicates typically do not exactly align with the identified semantic dimensions. It is challenging to reason over the semantic statistics for identifying usable query information and effectively handling the remaining query predicates for accurate semantic cardinality estimation. For example, estimating the number of documents about competitions in the USA cannot be answered directly from *sports event* dimension alone, since events such as Olympic Games may or may not be in the USA. Although strong correlations exist (e.g., documents exclusively about the European Cup are definitely not USA competitions), exploiting such correlations effectively for accurate estimation is non-trivial.

Our Approach. To address these challenges, we propose offline semantic catalog construction and semantic data distribution computation, along with online semantic cardinality estimation.

Autonomous Semantic Catalog Construction. We propose an autonomous approach to construct a *semantic catalog* for identifying the semantic dimensions of a corpus. This catalog is organized in a tree structure, where each node represents a semantic dimension, and the edge indicates the relationships between them. To improve construction efficiency, we first extract key phrases from the corpus, which provide a compact yet effective representation of the core semantics of the corpus. Based on these phrases, LLMs are leveraged to identify coherent, high-frequency semantic dimensions, which typically correspond to broader and more common semantics that represent key information in the data. More fine-grained and specific dimensions are then discovered hierarchically through iterative analysis of the associated phrases. Through this search process, both the semantic dimensions and their relationships are established. The resulting catalog accurately expresses the semantic statistics of the corpus (addressing **C1**).

Efficient Distribution Computation and Semantic Index Construction. We propose a cost-efficient data distribution computation method that links the data points to relevant semantic dimensions. To achieve this, we build a semantic index that maintains a list of data points for each dimension in the catalog, ensuring each data point is semantically relevant to its dimension. Instead of relying solely on LLMs to evaluate the relevance between a data point and a dimension, which can be computationally expensive, we use lightweight models for semantic relevance evaluation. This reduces costs while still providing approximate labeling of the corpus. Uncertain data points are then identified and re-evaluated using LLMs to ensure accuracy. This approach allows us to incorporate distribution of semantic dimensions to semantic statistics, which is critical for semantic cardinality estimation (addressing **C2**).

Accurate Semantic Cardinality Estimation. During online estimation, we transform query requirements into constraints over semantic dimensions by prompting LLMs to narrow the candidate space. Since semantic statistics may not fully cover the query, it still needs to estimate query selectivity in the candidate space. Unlike constrained sampling methods that only retrieve satisfying instances [10], selectivity estimation is to estimate the proportion of data points satisfying the query. To this end, we propose a Monte Carlo method that evaluates a sampled subset of data points against the query. For effective sampling, we propose a stratified importance sampling method, where relevant semantic dimensions define strata to cover diverse dimension values, and semantic similarity serves as the importance function focusing more on data likely to satisfy the query. Since online judgment by LLMs can be time-consuming, we employ a lightweight judge model trained offline to evaluate sampled points, invoking the LLMs only when necessary. This process ensures both accurate and efficient cardinality estimation for a wide range of semantic queries (addressing **C3**).

Contributions. In summary, we make the following contributions:
(1) We propose an autonomous method for semantic catalog construction that identifies the semantic dimensions of data without human effort to construct semantic statistics.
(2) We propose an efficient method that can accurately extract data distributions of various semantic dimensions with low LLM cost and index construction time (up to 10× reduction).
(3) We propose a stratified importance sampling-based estimation method for accurately and efficiently estimating the cardinality of semantic queries using the semantic statistics.

(4) Through comprehensive experiments on real-world datasets, we demonstrate that SemStats can significantly reduce error in cardinality estimation (up to 25× reduction) while maintaining low estimation latency (up to 31× reduction).

2 Preliminary

2.1 Problem Definition

Given an unstructured dataset consisting of N data points (e.g., documents, chunked segments), denoted as $\mathcal{D} = \{t_1, t_2, \ldots, t_N\}$, the task of *Semantic Cardinality Estimation* is to estimate the number of data points that satisfy a given semantic query without directly executing the query. Formally, let θ denote the semantic requirement of the query, which can be represented as a function that takes a document t as input and outputs $\theta(t) = 1$ if the document satisfies the requirement, and $\theta(t) = 0$ otherwise. Then, the cardinality of θ is the number of data points in $\mathcal{D}$ that satisfy θ, i.e., $card(\theta) = |\{t \in \mathcal{D} : \theta(t) = 1\}|$. The selectivity of θ is the fraction of data points that satisfy θ, i.e., $sel(\theta) = \frac{card(\theta)}{N}$.

Supported Query Types. SemStats supports natural language queries that apply semantic filtering over unstructured data collections and return a subset of data that satisfies the specified requirement. SemStats supports flexible filtering conditions, including both factual and sentiment-based queries. For example, SemStats supports queries that require semantic understanding of the data content, such as *"documents related to sports rules"*. Furthermore, similar to cardinality estimation methods for structured data [17, 42, 48, 49], SemStats focuses on query requirement to be essentially a conjunction of multiple semantic filtering conditions, since disjunctions can be solved through inclusion-exclusion principle [41, 42, 49]. For example, SemStats supports queries like *documents related to player transfer in the USA* that combines conditions *related to player transfer* and *in the USA*. Similarly, SemStats also supports negation and multi-hop semantics, since neither reasoning over the built semantic statistics nor the semantic-similarity-based sampling in SemStats impose any assumptions on the query predicates. For example, negation queries can be estimated by estimating the corresponding non-negated conditions, and multi-hop reasoning can be naturally solved during reasoning over the index.

Differences with Retrieval Queries. While semantic filtering and retrieval both leverage semantic similarity, their roles and requirements differ fundamentally. Consider a scenario of identifying "men's sports events"—retrieval can fail: NBA-related documents rarely contain the phrase "men's," yet the NBA is implicitly a men's league. Semantic filtering, guided by domain knowledge or calibrated models, can correctly classify such cases where surface-level similarity falls short. More importantly, semantic retrieval prioritizes ranking quality (e.g., "show top 10 men's sports"). Missing some relevant items is acceptable if the top results are strong. Semantic filtering, however, is a deterministic, all-or-nothing step. It must assign a definitive yes/no label to every record. There is no "top-k"—the output directly feeds into aggregations, joins, or cost models, where completeness and correctness are non-negotiable.

2.2 Related Work

Cardinality Estimation for Structured Data. Cardinality estimation is a core problem in query optimization for relational databases, which aims to predict the number of tuples satisfying an SQL query so that the optimizer can choose efficient execution plans. Traditional approaches rely on statistical synopses, such as histograms or sketches [30, 31, 34], or on sampling-based techniques that evaluate queries over subsets of the data [14, 22, 50]. However, statistical synopses are only approximations of data distributions and can incur large errors for rare values or when few or no samples satisfy the query. Sampling-based estimators are especially prone to such errors. To mitigate this issue, database systems often maintain separate statistics for the most frequent values [32, 36]. Learning-based methods [12, 16, 17, 21, 37, 48, 49] either train regression models on historical queries to directly predict query cardinalities [12, 21, 37], or model the joint data distribution with machine learning models [16, 17, 48, 49]. *While effective for structured data, these methods assume predefined schemas and cannot support semantic queries, making them unsuitable for semantic cardinality estimation.*

Data Analytics over Unstructured Data. The semantic reasoning capabilities of LLMs have extended data analytics beyond structured data to unstructured data [5, 8, 25–27, 29, 44]. Existing systems rely on execution plans that are either manually designed [5, 26, 29] or automatically generated [44, 47]. Similar to relational databases, their performance heavily depends on query optimization, which requires accurate estimation of cost and cardinality. Thus, effective semantic cardinality estimation is crucial for analytics efficiency. *However, existing systems rely solely on sampling-based techniques to estimate cardinality, which are neither accurate nor efficient.*

Indexing Unstructured Data. There are also some works aiming to preprocess the unstructured data for answering online queries. Graph-based methods such as Graph RAG [13] extract entities from document chunks, group them into communities via hierarchical structures, and generate summaries, yielding improved retrieval effectiveness compared to purely embedding-based methods. However, they overlook data distribution, rely solely on entity grouping, and incur high costs due to extensive LLM usage. Topic modeling approaches [6, 7, 19] uncover latent topics by extracting representative keywords or phrases, providing useful document summaries but failing to capture correlations across phrases that jointly reflect diverse semantic dimensions. Another line of works involves converting unstructured data into structured tables [8, 38], which simplifies analytics but suffers from schema misalignment, extensive missing values, and inability to represent hierarchical or nested relationships, potentially leading to substantial information loss.

3 The Framework of SemStats

This section presents the framework of SemStats. As shown in Figure 1, SemStats involves offline semantic statistics construction and online estimation. In the offline pre-processing, SemStats identifies semantic dimensions within the corpus and computes their data distributions. During online estimation, SemStats leverages the semantic statistics to perform efficient cardinality estimation.

Offline: Semantic Catalog Construction. Unlike structured data that adheres to predefined schemas, unstructured data lacks explicit metadata that describes the content. To perform semantic cardinality estimation, it is essential to understand what dimensions of semantic information the corpus contains and how these

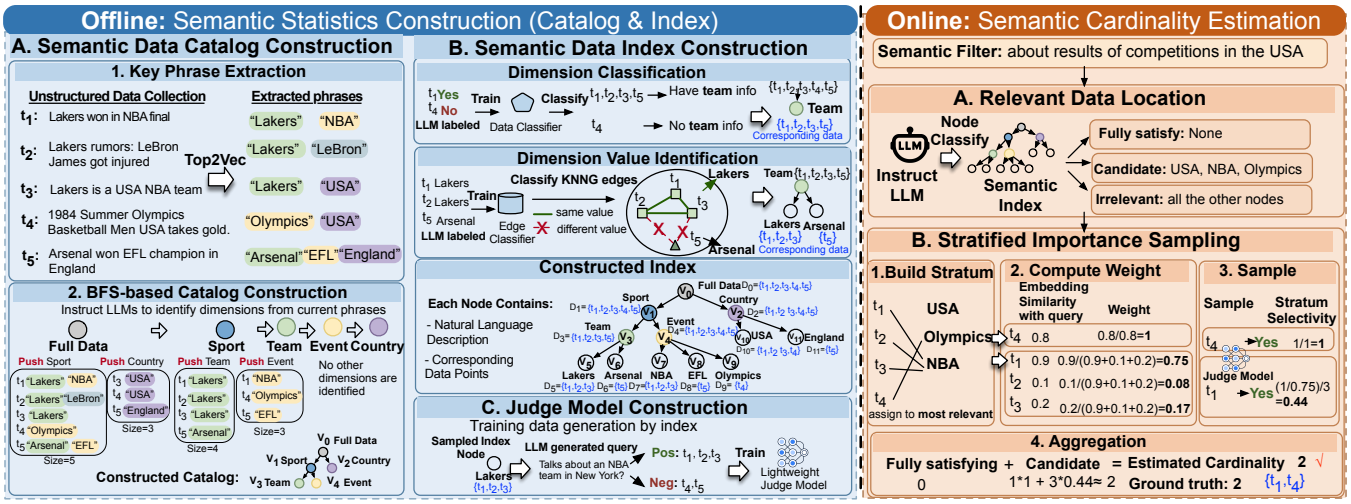

Figure 1: Framework of SemStats.

dimensions relate to each other. However, it is challenging to accurately capture the semantic dimensions, as the unstructured corpus typically contains multiple dimensions with complex relationships.

Semantic Catalog. We propose an autonomous *semantic catalog* construction method that automatically identifies and organizes the semantic dimensions in the corpus. The semantic catalog is a tree structure denoting semantic dimensions and their relationships. In the semantic catalog, each node represents a *semantic dimension*, which is a distinct dimension in the corpus (e.g., sports news collection may contain dimensions of sports team and sports event), and edges denote the hierarchical or nested relationships among these dimensions (e.g., sports-related data may contain sports event information, which in turn contains sports competitions). This catalog provides structured metadata about semantic dimensions within the corpus, which helps search for relevant data.

Construction Method. A straightforward catalog construction method is to prompt the LLM to identify and extract attributes for each individual data point and combine the results. However, this approach can be inaccurate because identifying common semantic dimensions requires simultaneously analyzing multiple data points, and the same semantic dimension may be inconsistently identified as different ones. Additionally, relationships between different dimensions cannot be captured, and the method is computationally inefficient due to the large number of LLM invocations required. An alternative is to exploit historical query workloads, which can reveal frequently queried attributes representing important semantic dimensions, e.g., via direct LLM prompting or combining entity recognition with clustering. However, relying solely on workload information may overlook important but previously unqueried dimensions, resulting in large estimation errors. Besides, historical workloads are often unavailable in practice. Therefore, SemStats constructs the catalog directly from the data corpus, while remaining compatible with dimensions provided by workload information.

To construct the catalog from the data collection efficiently, SemStats focuses on key phrases extracted from the corpus rather than processing the full text. Key phrases extracted through methods such as topic modeling [6, 7] can compactly represent the core semantics of documents while significantly reducing their size. For example, in Figure 1, phrases such as *Lakers* and *NBA* are extracted from the data point t_1 ("Lakers won in NBA final").

Starting from the complete set of extracted phrases, we employ LLMs to identify coherent semantic dimensions and their associated key-phrase groups. Each time, we calculate the frequency of the candidate dimensions by counting the number of data points that contain at least one associated phrase. If the total number of identified dimensions exceeds a predefined threshold, we retain only the dimensions with the highest frequencies. These selected dimensions are then refined in an iterative manner prioritizing the dimensions with higher frequencies: for each identified dimension, we further analyze its associated phrases to discover more fine-grained subdimensions. This iterative process helps identify a hierarchy of semantic dimensions, starting from broad ones and gradually refining them into more specific sub-dimensions. Finally, the constructed catalog serves as structured semantic statistics capturing the key semantic dimensions of the corpus. In Figure 1, SemStats first identifies two high-level dimensions, *Sport* and *Country*, with frequencies of 5 and 3. These dimensions are prioritized by frequency, so *Sport* is expanded first. From phrases associated with *Sport*, finer-grained dimensions including *Team* and *Event* are discovered with frequencies of 4 and 3 and added to the catalog. The remaining dimensions (*Team*, *Event*, and *Country*) are then examined in descending frequency order. Since no further high-frequency subdimensions are identified, the construction terminates.

Offline: Data Distribution Computation. Semantic cardinality estimation requires to estimate the number of data points that satisfy a query. Therefore, identifying key semantic dimensions is only the first step. It is equally crucial to determine which data points correspond to each dimension value in the semantic catalog. However, unlike relational databases, where attribute values can be directly retrieved, obtaining dimension values from unstructured data is costly, as it requires a semantic understanding of the content. A straightforward method is to prompt the LLM with descriptions of the semantic dimension and data-point content to extract the

corresponding relationship. However, this method relies on a large number of LLM invocations and is computationally prohibitive.

Efficient Distribution Computation. To address this problem, we propose an efficient method that replaces a large proportion of LLM calls with lightweight neural networks while maintaining accuracy. Specifically, for each semantic dimension, we first use the LLM to label a small subset of data about whether they contain information about the dimension. These labels are used to train a lightweight binary classifier, which is then applied to the remaining corpus. For example, in Figure 1, the LLM labels t_1 and t_4 for whether containing *Team*-relevant information. These labeled points are then used to train a binary classifier, which processes the remaining data points. This approach is effective for dimension classification but insufficient for identifying specific dimension values, as limited samples may not cover all labels. To mitigate this, we leverage semantic correlations among data points to guide labeling. Instead of directly analyzing the label for each individual data point, our key idea is to determine whether pairs of data points share the same value for a specific semantic dimension. We then partition the data points based on this prediction, label each partition, and assign the semantic dimension of the partition to all data points within it. However, it is expensive to check all pairs of data points since this incurs $O(n^2)$ complexity, which has low scalability.

To solve this problem, we first compute document embeddings and construct an approximate K-nearest neighbor (KNN) graph to represent semantic similarity among data points. By only focusing on neighbors with relatively high semantic similarity, we can avoid most irrelevant data pairs that are unlikely to have the same value for the semantic dimension. Using the LLM-labeled samples, we train a small model (e.g., GNN) to predict whether connected data points share the same dimension value. We then use the model predictions to update the edges of the KNN graph, which is subsequently partitioned into clusters that represent groups of data points likely to have the same dimension value. Each cluster's label is then inferred based on the labeled samples in it or by prompting the LLMs. To ensure labeling accuracy, we further identify uncertain data points by analyzing prediction confidence and graph consistency, and selectively re-evaluate them using the LLM. This selective refinement effectively improves accuracy with minimal additional LLM cost. For example, for the *Team* dimension in Figure 1, data points t_1, t_2, t_3, and t_5 are already identified as team-related. A KNN graph is built over these points, and a subset (t_1, t_2, t_5) is labeled for the dimension value using the LLM. An edge classifier then predicts whether data pairs in the KNN graph share the same team. As a result, t_1, t_2, and t_3 form a connected cluster labeled *Lakers*, while t_5 is isolated and labeled as *Arsenal*.

By iteratively applying this procedure for various semantic dimensions following the hierarchical structure of the semantic catalog, we identify dimension values and the subset of data points corresponding to each catalog node. The resulting semantic index provides distribution statistics for semantic cardinality estimation.

Judge Model Construction. Moreover, since no index can encode all information, online queries inevitably require checking whether certain data points satisfy ad hoc semantic conditions. To support such evaluations, we propose a *judge model*. Although LLMs can serve as effective judge models, frequently invoking LLMs for every check is costly. To address this, we train a lightweight semantic judge model using automatically generated training data derived from the index over the corpus. Each node in the index contains a natural language description and its associated data points. We sample nodes and combine their descriptions to construct training queries. LLMs can also be used to rewrite the query into equivalent ones. Data points within the nodes can serve as positive examples, while other data points in the corpus are negative ones. For instance, for the node representing the team *Lakers* in Figure 1, the LLM may generate the query "talks about an NBA team in New York," with data points in the node as positive data. In this way, we can construct a large, high-quality labeled dataset with limited LLM usage, enabling the semantic judge model to provide accurate semantic evaluations efficiently during online estimation.

In summary, through offline processing, we build semantic statistics that capture the semantic dimensions of the corpus along with their distributions, and develop a lightweight, accurate model for fast, on-demand semantic judgments.

Online: Semantic Cardinality Estimation. When processing online queries, `SemStats` leverages the pre-built semantic statistics to estimate cardinality by combining structured reasoning over the structured index with selective checks on uncertain data points.

`SemStats` first transforms the original query into structured constraints over the index to narrow down the data to be considered. The non-empty nodes are then identified for two categories: (i) *fully satisfying nodes*, which completely meet the query requirements, and (ii) *candidate nodes*, which are semantically relevant but require further verification. The total cardinality can be estimated by summing estimations from both categories. For fully satisfying nodes, `SemStats` directly retrieves subset sizes from the index. For candidate nodes, `SemStats` employs stratified importance sampling to estimate the proportion of satisfying data points. `SemStats` first utilizes the candidate nodes to build *strata*, i.e., subgroups in stratified sampling. Each data point is reassigned to the node with the highest semantic similarity, ensuring non-overlapping and distinguishable strata. Sampling budgets are then allocated proportionally to their sizes. Within each stratum, importance sampling based on semantic similarity is then applied to focus more on data points more likely to satisfy the query. Sampled data points are efficiently checked by the lightweight judge model, and estimations across strata are summed to compute the total cardinality. In Figure 1, for the query "about results of competitions in the USA," no node fully satisfies the query, while *USA*, *NBA*, and *Olympics* are identified as candidate nodes. These nodes induce two strata, *NBA* and *Olympics*. Importance sampling over both strata yields estimated count of 1, resulting in a total estimate of 2, which matches the ground truth.

By efficiently localizing relevant data points with semantic statistics, `SemStats` achieves efficient and accurate estimation.

4 Semantic Data Catalog

In this section, we first introduce the concept and design of the semantic data catalog (Section 4.1), and then introduce our autonomous catalog construction approach (Section 4.2).

4.1 Semantic Data Catalog in SemStats

Background. The absence of explicit schemas of unstructured data makes cardinality estimation for semantic queries challenging. To address this issue, we propose the concept of *semantic data catalog*, which provides a structured representation of the semantic information contained in the unstructured data corpus. Unlike traditional catalogs that rely solely on superficial metadata (e.g., file names or storage paths), a semantic catalog encodes multi-category semantic meanings such as concepts, entities, and their relationships.

Catalog Structure Design. The information in unstructured data holds several key properties that the catalog should capture. First, unstructured data often expresses multiple dimensions of semantic information, where we define each *semantic dimension* as a specific category of information that the unstructured data expresses. Therefore, an accurate catalog should be able to explicitly represent each dimension. Second, not all the semantic dimensions are independent. Instead, many of the dimensions have complex relationships, e.g., broader dimensions may cover several sub-dimensions that can provide finer-grained classification. An accurate catalog should be able to capture such relationships across dimensions.

To effectively capture these properties of unstructured data, the semantic catalog of SemStats is designed as a hierarchical tree that describes the multiple semantic dimensions (each dimension is a node in the tree, multiple dimensions under the same node form branches in the tree) of information with multi-level granularities between dimensions (layers in the tree). We define it formally.

Definition 1 (Semantic Data Catalog). *A semantic data catalog in* SemStats *is a tree-structured representation $C = (V, E)$, where V is the set of nodes and E is the set of edges. Each node $V_i \in V$ represents a semantic dimension of the corpus that encapsulates a specific informational facet with natural language descriptions. Each edge connecting parent node V_i and node V_j if and only if the semantic dimension denoted by V_i is the parent dimension of V_j.*

For the example in Figure 1, the original data is a collection of sports new. The built catalog effectively captures the key semantic dimensions of the corpus through a collection of nodes $\{V_1, V_2, V_3, V_4\}$, denoting sports, countries, sports teams, and events. Each node has a precise corresponding natural language description of the dimension. Meanwhile, the relationships among these dimensions are clearly denoted by the edges. For instance, *sports events* and *sports teams* are sub-dimensions of sports, which are precisely captured through the parent-child structure of the catalog.

From this example, we can observe that this tree-structured design is sufficient to explicitly record essential semantic dimensions and their nested relationships in a complex data corpus. The remaining problem that needs to be answered is how to autonomously construct such a catalog from a large, heterogeneous unstructured data collection, which we describe in the next section.

4.2 Semantic Data Catalog Construction

Basic Method. A straightforward approach to building a semantic catalog is to prompt the LLM to identify semantic dimensions for each individual data point and then aggregate the results. However, this strategy suffers from two main drawbacks. First, analyzing isolated data points (or small subsets limited by context length) fails to capture corpus-level common dimensions and their relationships. Second, it requires a large number of LLM invocations and tokens, making it prohibitively expensive.

Algorithm 1: Semantic Data Catalog Construction

Input: Unstructured data corpus $\mathcal{D}$, sampling ratio r, maximum number of dimensions per node K, maximum catalog depth H
Output: Tree-structured semantic data catalog C

```
// Step 1: Key Phrase Extraction
D_s ← sample r fraction of D
P ← ExtractKeyPhrases(D_s)
// Step 2: Hierarchical Catalog Construction
Initialize an empty catalog C with a root node
Initialize a priority queue Q that prioritizes high frequency
Insert (P, root node, 0) into Q with frequency |D_S|
while Q is not empty do
    (P, current_node, depth) ← pop the highest frequency element from Q
    if P = ∅ or depth ≥ H then
        return        // Stop if reach maximum depth
    (D, phrase_mapping) ← IdentifyDimensions(P)
    if D = ∅ then
        return        // Stop if no new dimensions
    Compute frequency of each candidate dimension in D
    Select top-K dimensions with the highest frequencies
    foreach d ∈ selected dimensions by frequency do
        Create node v representing dimension d and link it to current_node in C
        P_d ← phrases of data points that have phrases associated with dimension d
        Insert (P_d, v, depth + 1) into Q
```

Key Idea. To improve construction efficiency and effectiveness, SemStats constructs the catalog by extracting a set of *key phrases* from the corpus and analyzing the key phrases rather than the *full-text* content. These key phrases compactly represent the core semantics of documents, providing a smaller yet informative input for analyzing what content is within the corpus. We first distill the corpus into representative key phrases and then use LLMs to identify semantic dimensions hierarchically from coarse-grained to fine-grained, thereby building a tree-structured semantic catalog.

Key Phrases Extraction. We begin by sampling a subset of documents $\mathcal{D}_S$ from the corpus $\mathcal{D}$ (e.g., 10%). A semantic-aware phrase extraction method, such as contextualized Top2Vec [7], is employed to extract concise and informative phrases that capture each document's main content. While contextualized Top2Vec serves as the default extractor, this step is modular and can be replaced by any alternative methods (e.g., topic modeling or keyword extraction). After extraction, each document has a corresponding phrase set that compactly denotes its main content. The resulting phrase set, which is much smaller than the original corpus, is then used for catalog construction (see Figure 1).

Catalog Construction from Phrases. Starting with all the extracted phrases, SemStats progressively identifies semantic dimensions and their hierarchical relationships. At each step, the system selects broad dimensions for the catalog, explores their corresponding phrase subsets, and recursively discovers finer-grained dimensions. To prioritize broader dimensions with higher frequency, SemStats builds a priority queue initialized with the root node and frequency $|\mathcal{D}_S|$, and each time pops the element with the highest frequency from the queue to continue search for finer-grained dimensions. This top-down process produces a tree-structured catalog capturing both high-level and detailed dimensions.

Semantic Dimension Identification. Identifying semantic dimensions from a phrase collection requires deep semantic understanding capability. SemStats achieves this by prompting the LLM with both the current phrase set and the dimensions already included in the catalog. The LLM is instructed to identify other semantic dimensions and associate relevant phrases with each dimension, while avoiding previously identified dimensions. By iteratively prompting the LLM, we obtain a set of candidate dimensions, each associated with a phrase subset. Since these candidates may vary in granularity and representativeness, we select the broadest and most representative ones. Adopting fine-grained dimensions too early could hinder the inclusion of broader dimensions in the catalog.

Candidate Dimension Selection. For each candidate dimension, we compute its *frequency* as the number of data points having at least one phrase associated with it. Dimensions with higher frequencies are prioritized, as they tend to represent broader and more common semantics. If the number of candidate dimensions exceeds a predefined threshold K, only the top-K dimensions are retained.

Dimension Refinement. The selected dimensions are added as nodes in the catalog, each with natural language descriptions by prompting the LLM. To explore finer-grained semantics, SemStats inserts these nodes with corresponding frequency to the priority queue. In future search, these pushed dimensions will narrow the analysis to the key-phrase subsets corresponding to each selected dimension and apply the same dimension identification process recursively.

Backtracking and Termination. Overall, the above construction process forms a best-first search (BFS) over the phrase set. Dimensions are processed in descending order of frequency. This ensures that the broad dimensions will be searched first during the catalog construction, which avoids mistakenly selecting fine-grained dimensions that hinder the identification of broader dimensions. As the process descends each branch, the phrase set becomes progressively smaller. To maintain efficiency, SemStats adopts two stopping conditions: (1) A maximum catalog depth is enforced to prevent over-specific dimensions, which ensures a balanced level of abstraction. (2) When no new dimensions can be identified from the current phrase set, it returns to search for other dimensions.

Complexity Analysis. Algorithm 1 presents the complete catalog construction process. During the whole process, key phrase extraction takes only a small fraction of time, with the majority of time spent on LLM invocations. Since LLM latency scales primarily with the number of output tokens [3], we analyze complexity in terms of the total number of output tokens. During semantic catalog construction, most LLM outputs are key phrases associated with identified dimensions (Line 10), while other formatting outputs incur only constant-time overhead. Let m denote the total number of phrases extracted during key phrase extraction (Line 2), and let c be the average token length of these phrases. Let τ denote the maximum frequency of any phrase in the LLM outputs throughout the process. The total number of output tokens is thus $O(\tau mc)$. In practice, τ is typically small, as each phrase is only associated with a limited number of semantic dimensions. For example, in Figure 1, the phrase *Lakers* appears only in dimensions related to sports teams. This limited repetition of phrases across dimensions ensures the total number of output tokens is small, making the catalog construction remain efficient even for large datasets. This indicates that by operating on concise phrases rather than full-text content, SemStats can achieve efficient catalog construction.

5 Index of Semantic Data Distribution

The semantic catalog constructed in Section 4 identifies *what* semantic information exists in the corpus. However, accurate cardinality estimation also requires knowledge of *how* such information is distributed. To this end, we augment the catalog with data distribution details for each semantic dimension, constructing a **semantic data index** (Section 5.1). Furthermore, as no static index can fully support all ad-hoc queries, we leverage this index to train a lightweight *judge model* that determines whether a given unstructured data satisfies a query's semantic constraint. This model enables efficient online estimation while reducing costly LLM calls (Section 5.2).

5.1 Efficient Index Construction

To capture the data distribution for each semantic dimension, we record which data points contain relevant information. Additionally, for each leaf node in the catalog, we further identify its dimension values and the corresponding data points. Such information enables efficient online estimation, as queries can be efficiently narrowed down to a subset of relevant data points, significantly improving both efficiency and accuracy. We formally define this structure as a *semantic data index*, which is aligned with the catalog structure to explicitly represent the semantic data distribution.

Definition 2 (Semantic Data Index). *A semantic data index in* SemStats *is an enhanced tree-structured representation $\mathcal{I} = (V, E)$ built upon the semantic catalog. The node set V consists of two subsets: V_C, representing the original catalog nodes (semantic dimensions), and V_S, representing dimension-value nodes corresponding to the leaf dimensions of the catalog. Each node V_i maintains both its natural-language description and the set of data points D_i that contain the associated semantic information. Edges E include both the hierarchical edges inherited from the catalog and the additional edges connecting each dimension node to its dimension-value nodes.*

However, constructing a semantic index is challenging, since extracting precise distribution information (e.g., frequencies or values for a specific dimension) from unstructured data typically requires semantic processing of large datasets. This process typically relies on LLMs, making it both inefficient and costly.

Problem. The key problem is thus *how to efficiently obtain accurate data distribution information for catalog dimensions to construct an accurate semantic index.*

Basic Method. A straightforward approach is to prompt an LLM to analyze and label every data point for each semantic dimension.

However, although effective, this method is prohibitively expensive due to the vast number of required LLM invocations.

Key Idea. To solve this problem, our key idea is to *replace most expensive LLM invocations with efficient inference by lightweight models*. Specifically, we sample a subset of data, utilize LLMs to label them, and train small models on these samples to infer labels for the remaining data points to improve efficiency.

We process two types of nodes in the index with this idea:

(1) **Dimension nodes** (V_C): These nodes need to determine whether a data point contains information about a specific dimension. For example, in Figure 1, to build V_1, it needs to check if the data points in D_0 contain information about *sports*.

(2) **Dimension-value nodes** (V_S): These nodes identify the specific value of a dimension and corresponding data points. For instance, in Figure 1, to build V_5 and V_6, D_3 is processed to identify various *sports teams* (e.g., *Lakers*), and the corresponding data.

Dimension Classification for V_C. For each dimension node, determining whether a data point contains the information of the semantic dimension is a binary classification task. Hence, we can directly use the LLM-labeled subset to train a lightweight binary classifier (e.g., logistic regression or XGBoost) that classifies based on document embeddings. After training, it enables efficient inference for the remaining corpus. For example, in Figure 1, for nodes V_1, a small classifier trained on sampled data from D_1 can determine which remaining data points in D_1 contain information about *sports teams*, yielding the corresponding data subset D_3.

Dimension Value Identification for V_S. Identifying dimension values and their corresponding data points is more challenging, since the full set of dimension values is not known and the samples may not cover all dimension values. For instance, for the V_3 in Figure 1 that denotes the dimension of *sports team*, it is possible that sampled data from D_3 do not cover all teams occurred in the corpus, e.g., containing only data points relevant to *Lakers*. Therefore, directly applying a classifier trained on limited samples risks incomplete or biased labeling. To overcome this, we exploit to infer shared dimension values through clustering. We identify clustering relationships within the data by checking whether pairs of data holds the same value for the dimension. Based on this pairwise relationship, we then cluster the data points and check the label for each cluster with LLMs to obtain labels for all data points.

Specifically, based on this strategy, to obtain the dimension value distribution for a certain dimension, we conduct the following steps:

Step 1. Pair Selection. Exhaustively checking all pairs of data in this dimension is infeasible due to the $O(n^2)$ complexity. We instead compute semantic embeddings for all data points (e.g., using Sentence-BERT [33]) and construct an approximate K-nearest-neighbor (KNN) graph [11] to retain only semantically related pairs, thus reducing the complexity of checking pairs to $O(nK)$. Since data points with low semantic similarity are unlikely to share the same dimension values, this method can effectively filter out unnecessary checks for most pairs unlikely to share the same value.

Step 2. Relationship Identification. Using the LLM-labeled samples, we can collect some training data, where data pairs with the same label are positive examples and vice versa. With these training data, we then train a classifier, e.g. GNN, to predict whether two connected data points in the KNN graph share the same dimension value. We then refine the graph by removing edges predicted as not holding the same value, yielding a dimension-specific graph capturing clustering relationships across data points.

Step 3. Cluster-Based Approximate Labeling. We partition the refined graph via spectral clustering into clusters whose members likely share the same dimension value. Each cluster is assigned an approximate label according to the dominant label (if containing sampled data points) within the cluster. If the cluster does not contain any labeled data points or the labels are inconsistent within this cluster, the label will be computed by instructing the LLM to check some samples from this cluster.

Step 4. Label Refinement. We assess labeling confidence by examining label consistency for each node with its connected data points. Data points with inconsistent labels with its neighbors or the cluster label are selectively re-evaluated by the LLM to improve the labeling accuracy.

As shown in Figure 1, after completing above steps for all index nodes, we can obtain a comprehensive semantic index that accurately reflects data distribution of the corpus. Our approach can achieve similar accuracy to fully labeling the dataset with LLMs while having much higher efficiency.

Handling Data Update. `SemStats` supports incremental index updates when new data arrives. Specifically, `SemStats` first applies the trained binary classifiers to check if the data matches any existing dimension nodes and updates the corresponding data collections. If a matched node is a leaf node in the catalog, `SemStats` selects the most relevant existing dimension value based on semantic similarity. If the similarity exceeds a predefined threshold, the data is directly added to the selected dimension-value node; otherwise, `SemStats` invokes the LLM to identify the appropriate dimension value and updates the existing node or creates a new one as needed. This incremental strategy efficiently updates the index when new data aligns with existing semantic dimensions. In the rare cases where a large amount of data with significantly different semantics occurs, rebuilding the index could be more suitable.

5.2 Train A Small Model for Online Judgment

While a semantic data index is effective when queries align with the indexed semantic dimensions, it may not capture all relevant information for ad-hoc queries. To enhance flexibility in semantic cardinality estimation for such cases, we introduce a lightweight *judge model* that effectively evaluates whether a given unstructured data point satisfies the conditions of an NL semantic query.

Judge Model. While LLMs can conduct this semantic judgment accurately, they are computationally prohibitive for large-scale cardinality estimation. To solve this problem, we employ a smaller and more efficient model that takes embeddings of both the data point and the query as input, and outputs a binary classification indicating whether the data meets the query criteria. However, training an accurate model capable of handling flexible query conditions across the corpus requires a substantial amount of high-quality, corpus-specific training data. Therefore, the key challenge lies in acquiring sufficient high-quality training data.

Training Data Generation. To generate this training data, we leverage the built semantic index with its semantic dimensions, corresponding NL descriptions, and associated data. By utilizing

the distribution of these index dimensions, we can automatically generate labeled training examples. Specifically, we sample several nodes from the index and combine their descriptions through templates, e.g., concatenation by *AND*. Furthermore, we can also prompt an LLM to generate diverse equivalent NL queries that express the same requirement. The index structure allows for automatic labeling: Data points corresponding to the sampled nodes are labeled as **positive** examples, as they meet the NL description, and the remaining data points can serve as **negative** examples. This process can be repeated multiple times to generate a large set of labeled query-data pairs, requiring only a small number of LLM calls. For example, in Figure 1, nodes V_5 (*Lakers*) and V_7 (*NBA*) can generate the query: "Is the data related to the Lakers and NBA?" The positive training data can be obtained by $D_5 \cap D_7$. Other data points, i.e., $\{1, 2, \ldots, N\} - D_5 \cap D_7$, serve as negative examples. This process demonstrates that for any query derived from the NL descriptions of index nodes, we obtain a corresponding set of positive and negative examples. The resulting dataset reflects corpus content and supports the training of the lightweight judge model.

6 Semantic Cardinality Estimation

This section presents our method for efficiently estimating query cardinality using the semantic catalog and semantic index. The process involves two main steps: (i) reasoning over the index to distinguish data that satisfies, potentially satisfies, or does not satisfy the query (Section 6.1), and (ii) applying an effective sampling strategy over the relevant data that requires further checks for cardinality estimation (Section 6.2).

6.1 Relevant Data Location

With our constructed semantic index, we estimate cardinality for online queries by reasoning about high-level semantic dimensions rather than raw document content, thereby improving efficiency.

Transforming Queries into Structured Constraints. The query is first mapped into constraints over semantic index for those query requirements that can be accurately aligned with the index. This is achieved by prompting the LLMs with the query and index node descriptions. For instance, for the index in Figure 1, consider a query requiring "documents about results of competitions in the USA". The requirement *"in the USA"* can be aligned with the *Country* dimension of the index and translated into a condition Country = USA. Such structured constraints enable `SemStats` to narrow down the candidate data from full data to a smaller subset. In this example, candidate data is reduced into the data subset corresponding to the *USA* node. By transforming part of semantic query requirements into structured constraints, we filter out data points not satisfying the requirement, reducing unnecessary downstream computations.

Node Classification. We next conduct a more in-depth reasoning of the non-empty nodes. We take the NL descriptions of all such nodes to prompt the LLMs to identify nodes of the following types: (1) *Fully satisfying nodes*: their contents meet all query constraints. (2) *Candidate nodes*: semantically relevant to the query and may satisfy the query, but require further verification. When the identified nodes of these types cannot cover all candidate data, leaf nodes containing uncovered data of the index will also be marked as candidate nodes.

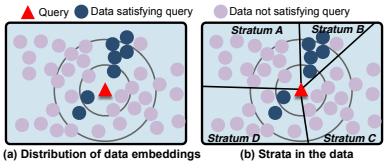

Figure 2: Illustration of stratified importance sampling.

Data can thus be partitioned into three categories: (i) satisfies the query based on index information: the union of all the data points of the *fully satisfying nodes*. (ii) requires additional checking: all the data that are not filtered by the structured constraints while also have not found to satisfy constraints based on index information (data points of the *candidate nodes*). (iii) does not satisfy the query: all the other data points.

Note that conflicts may arise due to misclassification, particularly with parent-child relationships between nodes. For instance, a child node may be classified as a candidate node while its parent node is classified as fully satisfying. To resolve such conflicts, we enforce a rule that a parent node cannot have a higher satisfaction level than any of its child nodes. If a conflict is detected, the parent node's satisfaction level is downgraded to match that of the child node.

Estimation Strategy. The final estimated cardinality is computed as: $\hat{C} = C_{\text{satisfy}} + \hat{C}_{\text{candidate}}$, where C_{satisfy} is the cardinality of data points already identified to satisfy the query based on the index information and can be obtained directly from the index, and $\hat{C}_{\text{candidate}}$ is the final cardinality of the data that requires additional checking. Since the query requirement cannot be fully checked based on index information, estimating the second term requires further checking samples. This strategy ensures that even when no index nodes fully align with the query (i.e., $C_{\text{satisfy}} = 0$), `SemStats` can still provide an effective estimation by computing $\hat{C}_{\text{candidate}}$ over the candidate subset through index-guided sampling.

By shifting reasoning from raw documents to semantic attributes in the index, `SemStats` significantly improves the estimation efficiency for online queries. However, since in reality, many queries have a large set of candidate data points that need further verification, the sampling method is crucial for overall estimation accuracy.

6.2 Stratified Importance Sampling

We estimate the query cardinality over the *candidate* set through a stratified importance sampling strategy. The key idea for effective sampling is to (i) *stratification* guided by the semantic index to ensure coverage across diverse semantic dimension values, and (ii) *importance sampling* within each stratum to prioritize data points more semantically relevant to the query.

Stratum Construction. In stratified sampling, a *stratum* is a non-overlapping subgroup of the dataset that shares a common characteristic, and the collection of all *strata* constitutes the complete dataset. In our context, these strata correspond naturally to groups defined by semantic dimension values in the index. We construct strata by descending each candidate node down to its leaf nodes in the semantic index. For example, in Figure 1, the candidate node *sports event* leads to its leaf nodes of *NBA* and *Olympics*. To prevent overly small strata, any stratum with a size below a predefined threshold is merged upward with its sibling strata until a sufficient

sample size is achieved. The resulting strata thus form coherent, well-balanced partitions of the candidate set.

Deduplication. Data points already evaluated as satisfying or not satisfying the query are removed. Since a data point may appear in multiple nodes in the index, only the occurrence in the node whose description has the highest semantic similarity (measured by cosine embedding distance) to the data point is retained, ensuring strata is formed by non-overlapping subgroups.

Stratified Sampling. Stratified sampling is then applied.

Sampling Budget Allocation. We first allocate the sampling budget across strata proportionally to their sizes, ensuring that all semantic dimensions are represented in the samples. Formally, suppose that the i-th stratum has a size of N_i, it will be assigned $\frac{N_i}{\sum_i N_i}$ proportion of the total sample budget.

Why Stratification Helps. Direct importance sampling over the entire candidate set based on semantic similarity with the query can be ineffective in high-dimensional embedding spaces. This is because the scalar similarity score compresses hundreds of embedding dimensions into a single digital value, which neglects semantic dimension diversity. Typically, points with similar semantic similarity may differ greatly in their true probability of satisfying the query, because similar similarity values may come from similarity in different semantic dimensions. Therefore, basic importance sampling may overlook important areas that have a high probability of satisfying the query, resulting in huge errors. On the contrary, stratification using the index preserves coverage of various dimension values utilizing semantic dimensions, thus obtaining high accuracy.

For example, in Figure 2, satisfying points cluster along two distinct semantic directions (dimension values). Global importance sampling may not sample any data in these clusters because they occupy only a small fraction within their similarity range. On the contrary, as shown in Figures 2(b), taking the semantic index partitions (e.g., Stratum B and Stratum D) as strata guarantees that both clusters are bound to have samples, significantly improving coverage and effectiveness.

Importance Sampling Within Strata. For each stratum i, we compute importance weights for data point j by $w_{ij} = \frac{s_{ij}}{\sum_{k=1}^{N_i} s_{ik}}$, where s_{ij} is the similarity between query and data embeddings, and N_i is the stratum size. Sampling according to w_{ij} focuses more on semantically relevant points that are more likely to satisfy the query while still retaining less relevant ones, improving estimation accuracy with limited samples.

Cardinality Estimation. Let $y_{ij} \in \{0, 1\}$ indicate if sampled point j satisfies the query (verified by the judge model in Section 5.2). The selectivity in stratum i is estimated as: $\hat{p}_i = \frac{1}{n_i} \sum_{j=1}^{n_i} \frac{y_{ij}}{w_{ij}} \cdot \frac{1}{N_i}$, where n_i is the number of samples for stratum i. Therefore, the estimated query cardinality for stratum i is $\hat{C}_i = N_i \cdot \hat{p}_i$. Finally, the overall cardinality for the whole candidate set is: $\hat{C}_{\text{candidate}} = \sum_i \hat{C}_i$, and the total query cardinality is $\hat{C} = C_{\text{satisfy}} + \hat{C}_{\text{candidate}}$.

Handling Sample Bias. Similar to sampling over structured tables, it is also possible that few or no sampled data points satisfy the query, leading to large errors. In `SemStats`, such cases are rare because the sampling process has been optimized by restricting the sampling space to relevant semantic dimensions and prioritizing data points that are more likely to satisfy the query. Nevertheless, to mitigate such errors, we propose to record query frequencies of index nodes. For frequently queried nodes, we apply finer-grained partitioning or re-identify dimension values with LLMs to improve the index quality. In this way, samples based on the index are more likely to cover satisfying data points, thus reducing estimation errors.

In summary, stratified importance sampling leverages both the *semantic structure* of the index and the *relevance-aware* weighting of embeddings, enabling efficient and accurate cardinality estimation.

6.3 Supporting Semantic Joins

`SemStats` also supports cardinality estimation for joins, where semantic filtering is applied to the Cartesian product of two unstructured data collections. The key challenge for joins is estimating, for each data point in one collection, how many matching points are expected in the other collection. To address this, `SemStats` estimates by sampling from one collection and estimating the number of matching points in the other for each sample using the estimation method for single-collection filtering. Specifically, this includes three steps: (1) `SemStats` first randomly samples several data points from one collection. By default uniform sampling is used. (2) For each sampled data point, `SemStats` prompts the LLM to rewrite its join condition as a filtering query over the other collection. This transforms the join cardinality estimation for the sample into a basic filtering estimation, which `SemStats` already supports. (3) Finally, `SemStats` estimates the overall join cardinality by multiplying the size of the collection being sampled by the average estimated number of matches across all samples. This simple strategy supports joins effectively, as `SemStats` well supports filtering estimations.

7 Experiments

7.1 Experimental Settings

Datasets. We conduct extensive experiments on five real-world, publicly available unstructured text datasets of varying scales.
(1)`Sports` [4]: 3,898 web pages sampled from the Sports Stack Exchange websites.
(2)`Wiki` [28]: 14,290 Wikipedia articles that meet the platform's Good or Featured article standards, as determined by editors. The articles are relatively long, with an average length of 3,406 tokens.
(3)`Bills` [18]: 32,661 bill summaries from the U.S. Congress.
(4)`Contracts` [15]: 604 real-world commercial legal contracts.
(5)`Rights` [9]: 1,032 terms of service from online platforms.

Baselines. We compare `SemStats` with the following baselines:
(1) `Uniform`: Uniformly samples a fixed proportion (default: 1%) of the dataset, then prompts the LLM to check the sampled points and estimate selectivity, which is widely adopted in prior works [26, 35].
(2) `Importance`: Performs importance sampling based on semantic similarity between query embeddings and data embeddings [44, 47].
(3) `Cluster`: Clusters data via embeddings using KMeans. The number of clusters is set to 30 for `Sports`, and increases to 100 for `Wiki` and `Bills` with larger data sizes. It then prompts the LLM to summarize each cluster. During online estimation, it identifies relevant clusters by prompting the LLMs to match query content to summaries. Selectivity is estimated by conducting importance sampling on data points in these relevant clusters.

Table 2: Average latency and q-errors (both the lower the better) of different methods on queries with different number of predicates. `Manual` **results are in gray since they are impractical in real-world applications and only serve as a reference.**

Dataset	Method	Latency (s)	Single-predicate Queries					Multi-predicate Queries				
			50th	90th	95th	99th	Max	50th	90th	95th	99th	Max
Sports	Uniform	4.72	2.65	15.30	16.65	30.14	35.0	4.17	12.40	14.98	26.88	29.59
	Importance	4.51	2.12	10.54	19.22	54.73	55.0	3.65	7.22	13.65	19.29	19.84
	Cluster	6.49	2.27	7.67	12.20	19.02	20.80	2.18	3.93	5.66	11.97	13.97
	HierCluster	6.22	1.90	9.43	42.38	55.46	56.0	3.09	7.47	8.50	9.63	11.36
	Structure	1.32	43.66	75.35	89.64	96.81	97.45	15.11	32.65	36.43	73.92	84.91
	RAG-CE	4.32	8.51	29.94	37.36	78.54	92.38	7.34	21.62	27.75	28.34	29.68
	Keyword	0.01	3.26	9.86	14.21	23.97	27.37	6.76	18.56	22.91	32.62	35.03
	SemStats	0.76	**1.42**	**4.35**	**5.27**	**9.86**	**11.40**	**1.49**	**3.86**	**5.42**	**6.52**	**7.09**
	Manual	5.08	1.19	3.36	6.16	10.71	12.24	2.85	3.73	4.18	5.21	5.55
Wiki	Uniform	73.31	2.29	19.33	30.22	89.76	117.41	3.34	12.61	33.54	68.06	164.0
	Importance	81.01	2.21	13.23	15.21	31.13	37.88	2.94	14.34	21.02	71.03	85.07
	Cluster	70.19	1.88	13.21	21.74	56.83	68.98	1.66	5.13	8.03	33.57	106.0
	HierCluster	79.95	2.09	13.10	19.34	32.83	45.80	2.25	9.61	13.24	39.32	328.0
	Structure	1.56	26.08	123.84	265.27	276.70	381.67	37.12	123.40	238.17	367.43	528.98
	RAG-CE	91.47	4.35	23.79	26.93	104.70	141.17	5.53	19.42	30.36	61.17	72.14
	Keyword	0.08	2.67	6.08	7.81	91.14	127.2	2.57	10.71	43.61	78.37	82.15
	SemStats	2.56	**1.24**	**2.05**	**3.55**	**9.09**	**10.83**	**1.34**	**3.43**	**6.21**	**7.35**	**7.64**
	Manual	74.87	1.11	1.96	8.71	16.54	17.18	1.23	2.74	22.89	38.74	45.56
Bills	Uniform	35.67	4.62	39.43	68.69	163.64	197.87	4.73	21.82	61.60	85.42	91.24
	Importance	37.40	4.39	30.72	65.71	164.24	199.20	4.69	24.55	46.29	83.10	86.95
	Cluster	34.72	2.01	11.57	17.40	87.51	114.0	2.08	4.90	8.77	13.45	16.06
	HierCluster	38.82	2.94	14.76	27.99	75.03	91.75	2.51	13.30	18.85	28.40	28.94
	Structure	1.74	43.37	182.06	235.96	278.29	286.5	54.25	92.00	97.21	103.49	105.70
	RAG-CE	40.17	7.83	30.51	34.60	38.98	40.48	6.43	23.16	26.22	32.65	34.45
	Keyword	0.15	3.02	8.22	10.49	44.07	59.5	2.69	8.48	10.72	18.89	55.16
	SemStats	1.82	**1.34**	**3.78**	**5.84**	**7.74**	**7.97**	**1.66**	**3.85**	**4.12**	**5.43**	**6.28**
	Manual	37.97	1.73	4.74	6.50	10.88	12.13	1.69	5.42	9.20	21.31	28.03

(4) `HierCluster`: Similar to `Cluster`, but applies hierarchical clustering (using Ward's method) to the data embeddings. The default hierarchical setting is a three-level structure, with 10 nodes at the second level and 100 nodes at the third level.
(5) `Structure`: Extracts structured information from unstructured data [8]. Since the method targets semi-structured inputs, only limited attributes can be obtained for most datasets. The query is transformed into SQL via NL2SQL (prompting the LLMs), and selectivity is estimated using histograms over the extracted tables.
(6) `RAG-CE`: Retrieve and evaluate the top-1% data points with the highest semantic similarity to the query. The selectivity is estimated as the fraction of evaluated data points that satisfy the query, divided by a constant factor (default 2) to compensate for the higher likelihood that top-K retrieved results satisfy the query.
(7) `Keyword`: Compute TF-IDF [20] scores for all documents offline and represent each document as a weighted bag of words. Given a query, sum the TF-IDF weights of query words appearing in each document. Documents whose weight sum exceeds a predefined threshold (tuned on a small set of queries) are considered as satisfying the query.
(8) `Manual`: Constructs the index through manual analysis of the data, identifying and grouping data points that share common information. The estimation process is identical to that of `SemStats`, except that an LLM is employed to verify the sampled data.

Test Workloads. For the first three datasets, we construct 100 natural language queries, each containing up to three filtering conditions. Queries are generated by filling manually crafted query templates with specific values sampled from the corpus and rewritten into equivalent formats by LLMs. For `Contracts` and `Rights`, 50 and 30 expert-curated real-world queries from the datasets are used. The ground-truth results for all queries are obtained through manually evaluating the documents by experts. The queries are grouped based on the number of predicates into single-predicate queries and queries with multiple predicates.

Evaluation Metrics. We evaluate both accuracy and efficiency. Accuracy is measured by q-error [42, 43], defined as $\frac{\max(card, \hat{card})}{\min(card, \hat{card})}$, where $card$ and $\hat{card}$ denote the true and estimated cardinalities respectively. Lower q-error values indicate higher accuracy. Efficiency is evaluated by the average estimation latency per query.

Hyper-parameter Setting. We use Meta-Llama-3.1-70B-Instruct as the base LLM, and Sentence-Transformers [33] for computing semantic embeddings of queries and data. For key phrase extraction, we apply contextualized Top2Vec [7], retaining 5 key phrases per document with length not exceeding 5. For the catalog, maximum number of dimensions per node and maximum catalog depth are set as 5. In semantic index construction, 10% of data labeled by LLMs are used for training small models to handle the remaining data. For dimension classification, XGBoost with 200 trees is used. For dimension value identification, we build a 100-NN graph, check its edges with a two-layer Graph Attention Network followed by a two-layer MLP classifier, and partition the KNN graph by semi-supervised harmonic label propagation [52]. The judge model embeds a concatenated query-data pair and applies a linear layer on top of the pooled representation to predict result. During online estimation, all methods are evaluated with the same sample ratio of 1%.

Environment. Experiments run on an Ubuntu server with an Intel Xeon Platinum 8558 CPU, 8 Nvidia H200 GPUs, and 2TB RAM.

7.2 Overall Evaluation

7.2.1 Comparison of Accuracy. Table 2 reports the q-errors of different methods across the three datasets. Overall, SemStats significantly outperforms all baselines for every dataset and query type across the entire q-error distribution. For instance, on the Wiki dataset, SemStats achieves a median q-error of 1.24 for single-predicate queries, compared with 2.29 (Uniform), 2.21 (Importance), 1.88 (Cluster), 2.09 (HierCluster), and 26.08 (Structure). Besides, SemStats reduces the maximum q-error by up to 25×. Beyond single-predicate queries, SemStats also achieves significantly higher accuracy than the baselines on compound queries involving multiple predicates. These results demonstrate the effectiveness of the semantic index constructed by SemStats and the accurate estimation achieved by our stratified importance sampling method.

Uniform has lower accuracy compared with SemStats, particularly for q-errors at the tail, because its random sampling over the entire corpus overlooks the relationship between queries and data points that satisfy the query. When few or no samples satisfy the query, its estimations can result in huge errors. Importance improves Uniform in most cases through sampling by semantic similarity between query and data, which focuses more on potentially relevant samples. However, it remains significantly less accurate than SemStats because it still samples from the entire dataset that contains a large number of irrelevant data. In addition, its full reliance on embedding-based similarity as the importance function is insufficient for queries with complex semantics.

Both Cluster and HierCluster also perform poorly due to the limited quality of their constructed indices. They build indices based on embedding similarity without semantic processing of the raw data. This makes them unable to accurately identify underlying relationships among data points in the corpus. Although HierCluster exploits potential hierarchical structures, it still fails to accurately capture multiple semantic dimensions of data.

Structure suffers from low accuracy because its structured data extraction requires common templates within data, making it suitable only for semi-structured data. This limitation prevents it from handling complex queries over unstructured datasets effectively.

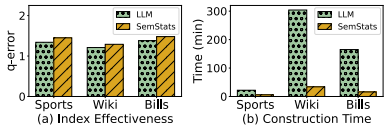

Figure 3: Evaluation of index construction.

RAG-CE also exhibits low accuracy, since the fraction of data points satisfying the query within semantically similar regions cannot reflect the global query selectivity. Keyword incurs high errors, as relying on word overlap fails to capture semantic relationships.

SemStats achieves accuracy similar to Manual due to similar index quality, e.g., indexes constructed by SemStats are comprehensive, consisting of 3 (18), 4 (31), 4 (20) layers (nodes) for Sports, Wiki, and Bills datasets, respectively. However, for large datasets, manually constructed indexes may overlook important informational dimensions, leading to substantial errors for certain queries. In contrast, SemStats identifies semantic dimensions in the catalog through iterative LLM-based verification, enabling broader coverage of semantic dimensions. Moreover, the significant human effort required by Manual makes it impractical for real-world applications.

7.2.2 Comparison of Latency. Table 2 also reports the average latency of different methods in semantic cardinality estimation. The results show that SemStats is up to 8×, 30× and 20× faster than other methods on Sports, Wiki and Bills respectively, demonstrating its high efficiency. For instance, on the Wiki dataset, SemStats estimates with an average latency of 2.56s per query, while that of other methods are 73.31s (Uniform), 81.01s (Importance), 70.19s (Cluster), 79.95s (HierCluster), 91.47s (RAG-CE) and 74.87s (Manual). SemStats achieves significantly lower latency by employing a lightweight judge model to verify sampled data points. This allows SemStats to maintain high efficiency under the same number of samples. Although Structure achieves low latency (1.56s) by estimating over extracted tables, its accuracy is much lower than SemStats due to the information loss during extraction, making it unsuitable for complex semantic queries over unstructured data. Although Keyword is efficient because it only involves simple word-overlap computations, it suffers from high errors due to its inability to capture underlying semantic relationships.

7.3 Evaluation of Index Construction

We evaluate our semantic index from three aspects: (i) the effectiveness of the constructed index, (ii) the efficiency of the index construction process and (iii) construction cost analysis. Specifically, SemStats employs lightweight models trained on a small subset of data to process the majority of data points, whereas the baseline LLM constructs the index by using the LLM to label every data point directly. For a fair comparison, both methods use the same semantic catalog built by SemStats.

7.3.1 Effectiveness of Constructed Index. We evaluate the effectiveness of our index construction method by comparing the median q-errors of indices built by SemStats and LLM across three datasets (Figure 3(a)). The results show that SemStats achieves accuracy comparable to LLM, indicating that the indices are of similar quality. This effectiveness comes from SemStats' ability to effectively

Table 3: Evaluation of Estimation on Real-world Queries.

Dataset	Method	Latency (s)	50th	90th	95th	99th	Max
Contracts	Uniform	1.08	2.01	63.0	129.59	142.5	252.0
	Importance	1.15	1.94	4.69	11.79	79.69	89.0
	Cluster	3.24	4.54	45.3	80.6	176.5	183.5
	HierCluster	3.09	3.24	31.1	53.8	79.7	86.0
	Structure	1.02	6.23	16.2	23.1	58.9	71.4
	RAG-CE	1.13	2.89	7.83	12.09	51.31	77.0
	Keyword	0.01	3.09	14.59	18.35	75.13	94.24
	SemStats	0.71	**1.81**	**4.27**	**6.78**	**12.16**	**13.54**
	Manual	2.29	1.42	3.91	5.41	7.60	9.27
Rights	Uniform	0.84	2.64	79.39	117.6	168.4	188.0
	Importance	0.86	2.09	19.59	56.39	103.27	117.0
	Cluster	4.56	4.33	13.47	21.32	23.38	24.0
	HierCluster	3.32	3.44	18.09	19.93	27.23	30.0
	Structure	1.23	23.45	82.55	103.2	216.7	258.0
	RAG-CE	0.87	5.71	28.61	32.23	50.09	56.75
	Keyword	0.01	6.88	17.09	25.79	32.88	47.2
	SemStats	0.65	**1.86**	**9.82**	**13.61**	**15.27**	**16.32**
	Manual	1.64	1.33	5.04	6.17	10.54	12.0

leverage a small set of LLM-labeled samples to train lightweight models that accurately label the remaining data.

7.3.2 Construction Efficiency. We measure index construction efficiency by construction time. As the results shown in Figure 3(b), the efficient index construction method of SemStats in Section 5.1 can construct accurate indices significantly faster than LLM (4× for Sports, 9× for Wiki and 10× for Bills). Although LLM produces slightly more accurate indices, it requires a much longer time and LLM invocation cost due to extensive LLM calls. Overall, SemStats can build semantic indices with accuracy similar to LLM but with significantly shorter time.

7.3.3 Cost Analysis. Since our experiments are conducted on a local server, we estimate the construction cost by mapping our setup to comparable cloud configurations. Specifically, we adopt Amazon Bedrock's LLM pricing [1] for Llama-3.1-70B ($0.72 per million input/output tokens), AWS i7i.12xlarge instances ($4.53/hour), and H200 GPUs ($5/hour) [2]. Under this approximation, the construction costs of SemStats on Sports, Wiki, and Bills are approximately $1.18, $10.16, and $2.74, respectively. The overall cost is dominated by LLM usages, whereas non-LLM components, such as KNN graph construction, account for only a small fraction (about 6%) of the total time and cost. This is because these steps involve much fewer computations and rely on much smaller models. Overall, the results demonstrate that SemStats incurs low construction cost across all datasets and proves its efficiency and practicality for real-world deployment.

7.4 Evaluation of Real-World Queries

We evaluate the estimation accuracy of SemStats on Contracts and Rights under realistic workloads, i.e., expert-curated real-world queries from these datasets. As shown in Table 3, all methods experience performance degradation due to the increased complexity of real-world queries. For example, queries over Contracts often involve complex legal judgments, such as determining whether a contract clause imposes obligations after termination or expiration, which require deeper semantic reasoning and remain challenging even for LLM-based approaches. Despite this, SemStats consistently achieves the highest accuracy among all methods. This is because its semantic index effectively narrows the candidate space by locating relevant data, while its sampling strategy enables accurate selectivity estimation within the reduced candidate set.

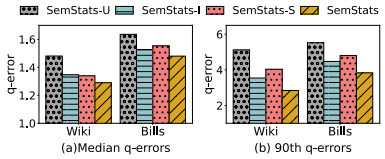

Figure 4: Evaluation of sampling methods.

Table 4: Evaluation of Data Update.

1/3 Training	+1/3			+1/3		
	time (min)	50th	90th	time (min)	50th	90th
No-Update	-	2.21	8.51	-	3.44	21.26
Inc-Update	12	1.34	3.37	10	1.32	3.35
Rebuild	21	1.30	3.15	34	1.28	2.84

7.5 Evaluation of Sampling Method

We evaluate our stratified importance sampling method using the index constructed by SemStats on the Wiki and Bills datasets. After identifying relevant nodes from the index, we compare SemStats with three alternative sampling strategies for estimating cardinality of candidate set: (1) SemStats − U: uniform sampling over all data points in the relevant nodes, (2) SemStats − I: importance sampling based on semantic similarity in embedding space, and (3) SemStats − S: stratified sampling by node sizes in the index, using uniform sampling within each node. As shown in Figure 4, SemStats consistently achieves lower q-errors than all baselines. The uniform sampling in SemStats − U, though restricted to relevant nodes, overlooks semantic relations between the query and data, resulting in low estimation accuracy. The importance sampling in SemStats − I simply relies on embedding similarity, thus may not be able to cover various dimensions like SemStats. Meanwhile, SemStats − S suffers from large estimation errors when some relevant nodes are large in size, as uniform sampling within large nodes has low sampling effectiveness. In contrast, SemStats integrates stratification with semantic importance, enabling query-aware sampling that captures both structural and semantic relationships, thereby yielding more accurate estimations.

7.6 Evaluation of Data Update

As introduced in Section 5.1, SemStats supports incremental index updates to handle data update. To evaluate its effectiveness, we randomly shuffle and partition the Wiki dataset into three disjoint splits that are added sequentially. We first build the semantic index using the initial $\frac{1}{3}$ split. At each subsequent step, we add additional $\frac{1}{3}$ split into the corpus and evaluate the same query workload on the updated corpus. As shown in Table 4, we compare the incremental update strategy of SemStats (Inc-Update) with using the stale index (No-Update) and rebuilding the index from scratch over the updated corpus (Rebuild). The results show that if still using the stale index, the estimation errors increase as the underlying data changes. In contrast, Inc-Update obtains accuracy comparable to Rebuild, while requiring significantly less update time, since Inc-Update only needs to update the index nodes affected by data changes. These results demonstrate that SemStats can efficiently and accurately adapt to data updates by incremental update.

Table 5: Evaluation of Scalability.

Method	Wiki (sent. chunk)			Bills (sent. chunk)		
	Latency	50th	90th	95th	Latency	50th	90th	95th
SemStats	**5.87**	**1.31**	**3.27**	**4.63**	**3.92**	**1.49**	**3.86**	**5.07**
Importance	160.6	2.38	6.99	9.12	94.2	2.54	9.24	13.7
Cluster	106.5	1.69	6.27	12.3	70.4	1.82	6.53	10.2

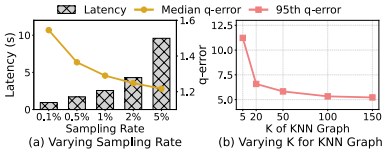

Figure 5: Evaluation of hyperparameters.

7.7 Scalability Evaluation

To evaluate the scalability of SemStats, we conduct an experiment in which each text chunk is treated as a tuple. We split all documents in Wiki and Bills into sentence-level chunks, resulting in a total of 134,552 and 109,072 data points, respectively. We compare with Importance (representing sampling-based methods) and Cluster (representing clustering-based methods) with the same experiment settings as in Section 7.1. As shown in Table 5, SemStats maintains high accuracy and efficiency as data scales. This efficiency stems from effective index-based reasoning that quickly identifies relevant data and replacing LLMs with lightweight judge models, enabling SemStats to achieve both scalability and better performance.

7.8 Evaluation of Hyperparameters

We further analyze the effect of some key hyperparameters.

7.8.1 Varying Sampling Rate. The sampling rate in online cardinality estimation influences both estimation accuracy and query latency. We evaluate various sampling rates on the Wiki dataset and report the median q-error and average per-query latency in Figure 5(a). Overall, increasing the sampling rate improves accuracy (lower q-error) but incurs higher latency. However, owing to the lightweight yet highly accurate judge model in SemStats (with average accuracy of 95% on various queries across three datasets), SemStats has high accuracy and maintains highly efficient even at relatively high sampling rates. From the results, we can also observe that a sampling rate of approximately 1% can provide a balance between accuracy and efficiency.

7.8.2 Varying k for KNN Graph. During index construction, the hyperparameter k in the KNN graph controls the number of data pairs examined, which affects index quality. To evaluate its impact, we vary k across $\{5, 20, 50, 100, 150\}$ and report the 95th-percentile q-errors on Sports. As shown in Figure 5(b), when k is small (e.g., $k = 5$), accuracy is low as checking limited number of data pairs fails to identify data points with the same dimension values. With larger k, more data pairs are checked and leads to more accurate index. As shown in Figure 5(b), setting $k \geq 50$ provides a good balance, yielding high accuracy while further increasing k has limited improvement in accuracy.

8 Conclusion

In this paper, we propose SemStats, a novel framework for semantic cardinality estimation over unstructured data. By constructing a semantic catalog, building a cost-efficient semantic index, and applying stratified importance sampling with a lightweight judge model, SemStats achieves both high accuracy and efficiency in estimating semantic cardinality. Experimental results on multiple real-world datasets demonstrate that SemStats significantly outperforms existing baseline methods.

Acknowledgments

This paper was supported by National Key R&D Program of China (2023YFB4503600), NSF of China (62525202, 62232009), Shenzhen Project (CJGJZD20230724093403007), China Railway Science Research Institute Group Co., Ltd, Zhongguancun Lab, Huawei, and Beijing National Research Center for Information Science and Technology (BNRist). Jiayi Wang and Guoliang Li are corresponding authors.

References

[1] [n. d.]. Amazon Bedrock Pricing. https://aws.amazon.com/cn/bedrock/pricing/?refid=5eef8ffa-5155-4b81-9303-69672b9516c1.
[2] [n. d.]. Amazon EC2 Pricing. https://aws.amazon.com/cn/ec2/.
[3] [n. d.]. OpenAI Latency optimization. https://platform.openai.com/docs/guides/latency-optimization.
[4] [n. d.]. Sports stackexchange data. https://archive.org/download/stackexchange/sports.stackexchange.com.7z.
[5] Eric Anderson, Jonathan Fritz, Austin Lee, Bohou Li, Mark Lindblad, Henry Lindeman, Alex Meyer, Parth Parmar, Tanvi Ranade, Mehul A Shah, et al. 2024. The Design of an LLM-powered Unstructured Analytics System. *arXiv preprint arXiv:2409.00847* (2024).
[6] Dimo Angelov. 2020. Top2Vec: Distributed Representations of Topics. *CoRR* abs/2008.09470 (2020). arXiv:2008.09470 https://arxiv.org/abs/2008.09470
[7] Dimo Angelov and Diana Inkpen. 2024. Topic Modeling: Contextual Token Embeddings Are All You Need. In *Findings of the Association for Computational Linguistics: EMNLP 2024, Miami, Florida, USA, November 12-16, 2024*, Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (Eds.). Association for Computational Linguistics, 13528–13539. https://doi.org/10.18653/V1/2024.FINDINGS-EMNLP.790
[8] Simran Arora, Brandon Yang, Sabri Eyuboglu, Avanika Narayan, Andrew Hojel, Immanuel Trummer, and Christopher Ré. 2023. Language Models Enable Simple Systems for Generating Structured Views of Heterogeneous Data Lakes. *Proc. VLDB Endow.* 17, 2 (2023), 92–105. https://doi.org/10.14778/3626292.3626294
[9] Ilias Chalkidis, Abhik Jana, Dirk Hartung, Michael J. Bommarito II, Ion Androutsopoulos, Daniel Martin Katz, and Nikolaos Aletras. 2022. LexGLUE: A Benchmark Dataset for Legal Language Understanding in English. In *ACL 2022, Dublin, Ireland, May 22-27, 2022*. 4310–4330. https://doi.org/10.18653/V1/2022.ACL-LONG.297
[10] Luc Devroye. 2006. Chapter 4 Nonuniform Random Variate Generation. In *Simulation*, Shane G. Henderson and Barry L. Nelson (Eds.). Handbooks in Operations Research and Management Science, Vol. 13. North-Holland, 83–121. https://doi.org/10.1016/S0927-0507(06)13004-2
[11] Wei Dong, Charikar Moses, and Kai Li. 2011. Efficient k-nearest neighbor graph construction for generic similarity measures. In *Proceedings of the 20th international conference on World wide web*. 577–586.
[12] Anshuman Dutt, Chi Wang, Azade Nazi, Srikanth Kandula, Vivek Narasayya, and Surajit Chaudhuri. 2019. Selectivity estimation for range predicates using lightweight models. *Proceedings of the VLDB Endowment* 12, 9 (2019), 1044–1057.
[13] Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, Dasha Metropolitansky, Robert Osazuwa Ness, and Jonathan Larson. 2024. From local to global: A graph rag approach to query-focused summarization. *arXiv preprint arXiv:2404.16130* (2024).
[14] Cristian Estan and Jeffrey F Naughton. 2006. End-biased samples for join cardinality estimation. In *22nd International Conference on Data Engineering (ICDE'06)*. IEEE, 20–20.
[15] Neel Guha, Julian Nyarko, Daniel E. Ho, Christopher Ré, Adam Chilton, K. Aditya, Alex Chohlas-Wood, et al. 2023. LegalBench: A Collaboratively Built Benchmark for Measuring Legal Reasoning in Large Language Models. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16,*

2023, Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (Eds.).

[16] Shohedul Hasan, Saravanan Thirumuruganathan, Jees Augustine, Nick Koudas, and Gautam Das. 2020. Deep Learning Models for Selectivity Estimation of Multi-Attribute Queries. In *Proceedings of the 2020 International Conference on Management of Data, SIGMOD Conference 2020, online conference [Portland, OR, USA], June 14-19, 2020*, David Maier, Rachel Pottinger, AnHai Doan, Wang-Chiew Tan, Abdussalam Alawini, and Hung Q. Ngo (Eds.). ACM, 1035–1050. https://doi.org/10.1145/3318464.3389741

[17] Benjamin Hilprecht, Andreas Schmidt, Moritz Kulessa, Alejandro Molina, Kristian Kersting, and Carsten Binnig. 2020. DeepDB: Learn from Data, not from Queries! *Proc. VLDB Endow.* 13, 7 (2020), 992–1005. https://doi.org/10.14778/3384345.3384349

[18] Alexander Miserlis Hoyle, Rupak Sarkar, Pranav Goel, and Philip Resnik. 2022. Are Neural Topic Models Broken?. In *Findings of the Association for Computational Linguistics: EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang (Eds.). Association for Computational Linguistics, 5321–5344. https://doi.org/10.18653/V1/2022.FINDINGS-EMNLP.390

[19] Hamed Jelodar, Yongli Wang, Chi Yuan, Xia Feng, Xiahui Jiang, Yanchao Li, and Liang Zhao. 2019. Latent Dirichlet allocation (LDA) and topic modeling: models, applications, a survey. *Multim. Tools Appl.* 78, 11 (2019), 15169–15211. https://doi.org/10.1007/S11042-018-6894-4

[20] Karen Spärck Jones. 2004. A statistical interpretation of term specificity and its application in retrieval. *J. Documentation* 60, 5 (2004), 493–502. https://doi.org/10.1108/00220410410560573

[21] Andreas Kipf, Thomas Kipf, Bernhard Radke, Viktor Leis, Peter A. Boncz, and Alfons Kemper. 2019. Learned Cardinalities: Estimating Correlated Joins with Deep Learning. In *9th Biennial Conference on Innovative Data Systems Research, CIDR 2019, Asilomar, CA, USA, January 13-16, 2019, Online Proceedings*. www.cidrdb.org. http://cidrdb.org/cidr2019/papers/p101-kipf-cidr19.pdf

[22] Viktor Leis, Bernhard Radke, Andrey Gubichev, Alfons Kemper, and Thomas Neumann. 2017. Cardinality Estimation Done Right: Index-Based Join Sampling.. In *Cidr*.

[23] Guoliang Li, Ji Sun, James Pan, Jiang Wang, Yongqing Xie, Ruicheng Liu, and Wen Nie. 2025. GaussDB-Vector: A Large-Scale Persistent Real-Time Vector Database for LLM Applications. *Proc. VLDB Endow.* 18, 12 (2025), 4951–4963. https://doi.org/10.14778/3750601.3750619

[24] Guoliang Li, Jiayi Wang, Chenyang Zhang, and Jiannan Wang. 2025. Data+AI: LLM4Data and Data4LLM. In *Companion of the 2025 International Conference on Management of Data, SIGMOD/PODS 2025, Berlin, Germany, June 22-27, 2025*, Volker Markl, Joseph M. Hellerstein, and Azza Abouzied (Eds.). ACM, 837–843. https://doi.org/10.1145/3722212.3725641

[25] Yiming Lin, Madelon Hulsebos, Ruiying Ma, Shreya Shankar, Sepanta Zeighami, Aditya G. Parameswaran, and Eugene Wu. 2024. Towards Accurate and Efficient Document Analytics with Large Language Models. *CoRR* abs/2405.04674 (2024). https://doi.org/10.48550/ARXIV.2405.04674 arXiv:2405.04674

[26] Chunwei Liu, Matthew Russo, Michael Cafarella, Lei Cao, Peter Baile Chen, Zui Chen, Michael Franklin, Tim Kraska, Samuel Madden, Rana Shahout, et al. 2025. Palimpzest: Optimizing ai-powered analytics with declarative query processing. In *Proceedings of the Conference on Innovative Database Research (CIDR)*.

[27] Samuel Madden, Michael J. Cafarella, Michael J. Franklin, and Tim Kraska. 2024. Databases Unbound: Querying All of the World's Bytes with AI. *Proc. VLDB Endow.* 17, 12 (2024), 4546–4554. https://doi.org/10.14778/3685800.3685916

[28] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2017. Pointer Sentinel Mixture Models. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net. https://openreview.net/forum?id=Byj72udxe

[29] Liana Patel, Siddharth Jha, Carlos Guestrin, and Matei Zaharia. 2024. LOTUS: Enabling Semantic Queries with LLMs Over Tables of Unstructured and Structured Data. *CoRR* abs/2407.11418 (2024). https://doi.org/10.48550/ARXIV.2407.11418 arXiv:2407.11418

[30] Viswanath Poosala, Peter J Haas, Yannis E Ioannidis, and Eugene J Shekita. 1996. Improved histograms for selectivity estimation of range predicates. *ACM Sigmod Record* 25, 2 (1996), 294–305.

[31] Viswanath Poosala and Yannis E Ioannidis. 1997. Selectivity estimation without the attribute value independence assumption. In *VLDB*, Vol. 97. 486–495.

[32] PostgreSQL. 2026. PG STATISTICS. https://www.postgresql.org/docs/current/view-pg-stats.html. PostgreSQL Documentation.

[33] 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 and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*. Association for Computational Linguistics, 3980–3990. https://doi.org/10.18653/V1/D19-1410

[34] 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.

[35] Shreya Shankar, Tristan Chambers, Tarak Shah, Aditya G. Parameswaran, and Eugene Wu. 2025. DocETL: Agentic Query Rewriting and Evaluation for Complex Document Processing. *Proc. VLDB Endow.* 18, 9 (2025), 3035–3048. https://www.vldb.org/pvldb/vol18/p3035-shankar.pdf

[36] SQL Server. 2026. SQL Server STATISTICS. https://learn.microsoft.com/en-us/sql/relational-databases/statistics/statistics?view=sql-server-ver17. SQL Server Documentation.

[37] Ji Sun and Guoliang Li. 2019. An end-to-end learning-based cost estimator. *Proc. VLDB Endow.* 13, 3 (Nov. 2019), 307–319. https://doi.org/10.14778/3368289.3368296

[38] Zhaoze Sun, Deng Qiyan, Chengliang Chai, Kaisen Jin, Xinyu Guo, Han Han, Ye Yuan, Guoren Wang, and Lei Cao. 2025. QUEST: Query Optimization in Unstructured Document Analysis. *CoRR* abs/2507.06515 (2025). https://doi.org/10.48550/ARXIV.2507.06515 arXiv:2507.06515

[39] Zhaoyan Sun, Jiayi Wang, Xinyang Zhao, Jiachi Wang, and Guoliang Li. 2025. Data agent: A holistic architecture for orchestrating data+ ai ecosystems. *arXiv preprint arXiv:2507.01599* (2025).

[40] Zhaoyan Sun, Xuanhe Zhou, Guoliang Li, Xiang Yu, Jianhua Feng, and Yong Zhang. 2025. R-Bot: An LLM-based Query Rewrite System. *Proc. VLDB Endow.* 18, 12 (2025), 5031–5044. https://doi.org/10.14778/3750601.3750625

[41] Jiayi Wang, Lei Cao, Chengliang Chai, and Guoliang Li. 2025. Federated Data Analytics with Differentially Private Density Estimation Model. In *2025 IEEE 41st International Conference on Data Engineering (ICDE)*. IEEE, 2768–2781.

[42] Jiayi Wang, Chengliang Chai, Jiabin Liu, and Guoliang Li. 2021. FACE: A Normalizing Flow based Cardinality Estimator. *Proc. VLDB Endow.* 15, 1 (2021), 72–84. https://doi.org/10.14778/3485450.3485458

[43] Jiayi Wang, Chengliang Chai, Jiabin Liu, and Guoliang Li. 2024. Cardinality estimation using normalizing flow. *VLDB J.* 33, 2 (2024), 323–348. https://doi.org/10.1007/S00778-023-00808-X

[44] Jiayi Wang and Jianhua Feng. 2025. Unify: An Unstructured Data Analytics System. In *2025 IEEE 41st International Conference on Data Engineering (ICDE)*. IEEE Computer Society, 4662–4674.

[45] Jiayi Wang and Guoliang Li. 2025. AOP: Automated and Interactive LLM Pipeline Orchestration for Answering Complex Queries. In *CIDR 2025*.

[46] Jiayi Wang, Guoliang Li, and Jianhua Feng. [n. d.]. iDataLake: An LLM-Powered Analytics System on Data Lakes. *Data Engineering* ([n. d.]), 57.

[47] Jiayi Wang, Yuan Li, Jianming Wu, Shihui Xu, and Guoliang Li. 2025. Unify: A System For Unstructured Data Analytics. *Proc. VLDB Endow.* 18, 12 (2025), 5287–5290. https://www.vldb.org/pvldb/vol18/p5287-wang.pdf

[48] Zongheng Yang, Amog Kamsetty, Sifei Luan, Eric Liang, Yan Duan, Xi Chen, and Ion Stoica. 2020. NeuroCard: One Cardinality Estimator for All Tables. *Proc. VLDB Endow.* 14, 1 (2020), 61–73. https://doi.org/10.14778/3421424.3421432

[49] Zongheng Yang, Eric Liang, Amog Kamsetty, Chenggang Wu, Yan Duan, Xi Chen, Pieter Abbeel, Joseph M. Hellerstein, Sanjay Krishnan, and Ion Stoica. 2019. Deep Unsupervised Cardinality Estimation. *Proc. VLDB Endow.* 13, 3 (2019), 279–292. https://doi.org/10.14778/3368289.3368294

[50] Zhuoyue Zhao, Robert Christensen, Feifei Li, Xiao Hu, and Ke Yi. 2018. Random sampling over joins revisited. In *Proceedings of the 2018 International Conference on Management of Data*. 1525–1539.

[51] Xuanhe Zhou, Zhaoyan Sun, and Guoliang Li. 2024. DB-GPT: Large Language Model Meets Database. *Data Sci. Eng.* 9, 1 (2024), 102–111. https://doi.org/10.1007/S41019-023-00235-6

[52] Xiaojin Zhu, Zoubin Ghahramani, and John D. Lafferty. 2003. Semi-Supervised Learning Using Gaussian Fields and Harmonic Functions. In *(ICML 2003), August 21-24, 2003, Washington, DC, USA*. 912–919.

Received October 2025; revised January 2026; accepted February 2026