Data Science and Engineering (2019) 4:254–268
https://doi.org/10.1007/s41019-019-00104-1

Selectivity Estimation on Set Containment Search

Yang Yang[1] · Wenjie Zhang[1] · Ying Zhang[2] · Xuemin Lin[1,3] · Liping Wang[3]

Received: 30 June 2019 / Revised: 20 August 2019 / Accepted: 27 August 2019 / Published online: 23 September 2019

Abstract
In this paper, we study the problem of selectivity estimation on set containment search. Given a query record Q and a record dataset $\mathcal{S}$, we aim to accurately and efficiently estimate the selectivity of set containment search of query Q over $\mathcal{S}$. We first extend existing distinct value estimating techniques to solve this problem and develop an inverted list and *G-KMV* sketch-based approach *IL-GKMV*. We analyze that the performance of *IL-GKMV* degrades with the increase in vocabulary size. Motivated by limitations of existing techniques and the inherent challenges of the problem, we resort to developing effective and efficient sampling approaches and propose an ordered trie structure-based sampling approach named *OT-Sampling*. *OT-Sampling* partitions records based on element frequency and occurrence patterns and is significantly more accurate compared with simple random sampling method and *IL-GKMV*. To further enhance the performance, a divide-and-conquer-based sampling approach, *DC-Sampling*, is presented with an inclusion/exclusion prefix to explore the pruning opportunities. Meanwhile, we consider weighted set containment selectivity estimation and devise stratified random sampling approach named *StrRS*. We theoretically analyze the proposed techniques regarding various accuracy estimators. Our comprehensive experiments on nine real datasets verify the effectiveness and efficiency of our proposed techniques.

Keywords Selectivity · Sampling · Trie

1 Introduction

Set-valued attributes are ubiquitous and play an important role in modeling database systems in many applications such as information retrieval, data cleaning, machine learning and user recommendation. For instance, such set-valued attributes may correspond to the profile of a person, the tags of a post, the domain information of a webpage and the tokens or q-grams of a document. In the literature, there has been a variety of interests in the computation of set-valued records including set containment search (e.g., [9, 21, 28, 36]), set similarity joins (e.g., [31, 33]) and set containment joins (e.g., [13, 24, 25, 34]).

In this paper, we focus on the problem of *selectivity estimation of set containment search*. Considering a query record Q and a collection of records $\mathcal{S}$ where a record consists of an identifier and a set of elements (i.e., terms), a set containment search retrieves records from $\mathcal{S}$ which are contained by Q, i.e., $\{X|X \in \mathcal{S} \wedge Q \supseteq X\}$, where Q *contains* X ($Q \supseteq X$) if all the elements in X are also in Q. Table 1 shows an example with eight records in a dataset and a query record Q where Q contains X_2, X_3 and X_5. Selectivity (cardinality) of a query refers to the size of the query result. For instance, the selectivity of Q in Table 1 is 3.

Selectivity estimation on set containment search aims at estimating the cardinality of the containment search. As an essential and fundamental tool on massive collections of set values, the problem has a wide spectrum of applications because it can provide users with fast and useful feedback. As a simple example, when introducing a new product to the market, its characteristics and features could be described as a set of keywords. Assume a preference dataset consists of such characteristics and features desired by users from online survey. Size estimation of the new

✉ Yang Yang
yang.yang@cse.unsw.edu.au

Wenjie Zhang
zhangw@cse.unsw.edu.au

Ying Zhang
ying.zhang@uts.edu.au

Xuemin Lin
lxue@cse.unsw.edu.au

Liping Wang
lipingwang@sei.ecnu.edu.cn

1 University of New South Wales, Sydney, Australia
2 University of Technology Sydney, Sydney, Australia
3 East China Normal University, Shanghai, China

Table 1 A record dataset with eight records and a query Q

Id	Record
X_1	$\{e_1, e_2, e_3, e_4, e_7\}$
X_2	$\{e_2, e_3, e_5\}$
X_3	$\{e_2, e_5, e_7\}$
X_4	$\{e_1, e_2, e_6, e_{10}\}$
X_5	$\{e_1, e_3, e_5, e_7\}$
X_6	$\{e_2, e_6, e_7, e_8\}$
X_7	$\{e_4, e_8\}$
X_8	$\{e_4, e_{10}\}$
Q	$\{e_1, e_2, e_3, e_5, e_7, e_9\}$

product descriptions on the preference dataset estimates the total number of users who may be interested in the product and could serve as a prediction of the product's market potential. In another example, companies may post positions in an online job market Web site where a position description contains a set of required skills. A job seeker may want to have a basic understanding of the job market by obtaining the total number of active job vacancies that he/she perfectly matches (i.e., the skill set of the job seeker contains the required skills of the job).

1.1 Challenges

The key challenges of selectivity estimation on set containment search come from the following three aspects. *First*, the dimensionality (i.e., the number of distinct elements) is high. Shingle (n-gram)-based representations for strings are common in practice [26]. Typical (first-order) shingle-based representations of a string of sentence are a collection of words each of which is separated by a space. High-order shingles are used to represent the strings with different combinations of words. As shown in our empirical studies, the vocabulary size in real-world dataset could reach more than 3 million when the high-order shingles are used. This makes the selectivity estimation techniques which are sensitive to dimensionality inapplicable to our problem. *Second*, the number of records in the dataset could be very large. Moreover, the length of query and data record may also be large. To deal with the sheer volume of the data, it is desirable to efficiently and effectively provide approximate solutions. *Third*, the distribution of element frequency may be highly skewed in real applications. It is desirable to devise sophisticated data-dependent techniques to properly handle the skewness of data distribution to boost the accuracy.

Even though selectivity estimation has been widely explored, most of the existing techniques cannot be trivially applied to handle the problem studied in this paper. We discuss two categories of techniques which can be extended to support the selectivity estimation problem, range counting estimating (e.g., [8, 15]) and distinct value estimating [12, 16].

Given the element universe (vocabulary) $\mathcal{E}$, a record X_i can be regarded as an $|\mathcal{E}|$-dimensional binary vector, where $X_{ij} = 1$ if element e_j appears in X_i ($e_j \in X_i$) and $X_{ij} = 0$ otherwise, for $1 \leq j \leq |\mathcal{E}|$. Let n denote the vocabulary size $|\mathcal{E}|$. Under this context, the dataset $\mathcal{S}$ can be modeled as a set of points in $\{0,1\}^n$ where each record corresponds to an n-dimensional point and the query is a hypercube in $\{0,1\}^n$. Thus, we can rewrite the selectivity estimation problem as the approximate range counting problem in computational geometry. However, the approximate range counting problem suffers from the curse of dimensionality where the computing cost is exponentially dependent on dimensionality n [16, 27]. As the vocabulary size is usually large, applying range counting estimating methods to our problem is not applicable.

Distinct value estimators (e.g., KMV [12], bottom-k, min-hash [16]) can effectively support size estimation for set operations (e.g., union and intersection) and are widely used for problems of size estimation under different contexts. In Sect. 3.2, we show how to extend the distinct value-based estimator to the problem studied in this paper combining with inverted list techniques. We also analyze that the performance of distinct value estimator-based approach degrades when the vocabulary size is large due to the inherent *superset* containment semantics of the problem studied in this paper. Wang et al. [32] study selectivity estimation on streaming spatio-textual data where the textual data are a set of keywords/terms (i.e., elements). However, the query semantic is different as it specifies a *subset containment search* on the textual data, i.e., the keywords (elements) in the query should be *contained by* the keywords from spatial objects. This is different from the *superset* query semantic in our problem which is more challenging to handle using distinct value estimators as discussed in Sect. 3.2.

1.2 Contributions

Motivated by the challenges and limitations of existing techniques, in the paper we aim to develop efficient and effective sampling-based approaches to tackle the problem. Naively applying random sampling over the dataset ignores the element frequency distribution and results in the compromised performance. Intuitively, combinations of high-frequency elements (i.e., frequent patterns) occur among data records with high frequency, and records with similar frequent patterns are more likely to be contained by the same query. Thus, we use the frequent patterns as labels and partition records by these labels to boost the efficiency and accuracy. Moreover, assume that the elements are ordered based on frequency, we use ordered trie structure to maintain partitions of the dataset and present *OT-Sampling* method. This ordered trie-based approach, though demonstrated to be highly efficient and accurate, does not consider element distribution of the query Q. Inspired by the observation that query Q must include a subset of record X in order to contain X, efficient pruning techniques are developed on the partitions of dataset. We further propose a divide-and-conquer-based sampling approach named *DC-Sampling* which only conducts sampling on the qualified partitions surviving from the pruning.

The principle contributions of this paper are summarized as follows.

- This is the first work to systematically study the problem of selectivity estimation on set containment search, which is an essential tool for set-valued attributes analyses in a wide range of applications.
- Two baseline algorithms are devised. The first algorithm is based on random sampling. We also extend distinct value estimator *G-KMV* sketch and propose an inverted list-based approach *IL-GKMV*. Insights about the limitations of the two baseline approaches are theoretically analyzed and empirically studied.
- We develop two novel sampling-based techniques: *OT-Sampling* and *DC-Sampling*. *OT-Sampling* integrates ordered trie index structure to group the dataset and achieves higher accuracy by capturing the element frequency and frequent patterns. *DC-Sampling* employs divide-and-conquer philosophy and an exclusion/inclusion-set prefix to further improve the performance by exploring pruning opportunities and skipping sampling on pruned partitions of the dataset.
- We consider the selectivity estimation problem with respect to **weighted** set containment search, which is a generalization of the simple set containment search problem. A naive random sampling method and a stratified sampling approach are proposed to tackle this problem.
- Comprehensive experiments on a variety of real-life datasets demonstrate the superior performance of the proposed techniques compared with baseline algorithms.

2 Preliminary

In this section, we first formally present the problem of containment selectivity estimation and then give some preliminary knowledge. The notations used throughout this paper are summarized in Table 2.

2.1 Problem Definition

Suppose the element universe is $\mathcal{E} = \{e_1, e_2, \dots, e_n\}$. Each record X consists of a set of elements from domain $\mathcal{E}$. Let $\mathcal{S}$ be a collection of records $\{X_1, X_2, \dots, X_m\}$. Given two records X and Y, we say X contains Y, denoted as $X \supseteq Y$, if all elements of Y can be found in X. In the paper, we also say X is a superset of Y or Y is a subset of X. Given a query record Q and a dataset $\mathcal{S}$, a set containment search of Q over $\mathcal{S}$ returns all records from $\mathcal{S}$ which are contained by Q, i.e., $\{X | X \in \mathcal{S}, Q \supseteq X\}$. We use t to denote the selectivity (cardinality) of the set containment search. The selectivity of Q measures the number of records returned by the search; namely, $t = |\{X | X \in \mathcal{S}, Q \supseteq X\}|$.

Table 2 Summary of notations

Notation	Definition
$X, Q, \mathcal{S}$	A record, a query record, a set of records
$e, \mathcal{E}$	An element, element domain (vocabulary)
m	Number of records in $\mathcal{S}$
n	Number of distinct elements (vocabulary size)
t $(\hat{t})$	Containment selectivity (estimation of t)
$P_i, \mathcal{P}$	A partition of dataset, all partitions
m_i	Size of partition P_i
m'_i	Sampling size in partition P_i
p_i	Sampling probability in P_i
t_i	Containment selectivity of Q in P_i

Considering the containment relationship between a given query Q and a record $X_i \in \mathcal{S}$ $(1 \le i \le m)$, let $\mathbf{n}_i$ be the indicator function such that

$$\mathbf{n}_i := \begin{cases} 1 & \text{if} \quad Q \supseteq X_i, \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

and then, the *selectivity of the set containment search* on dataset $\mathcal{S}$ with respect to the query Q can also be calculated as $t = \sum_{X_i \in \mathcal{S}} \mathbf{n}_i$.

2.1.1 Problem Statement

In this paper, we investigate the problem of selectivity estimation on set containment search. Given a query record Q and a dataset $\mathcal{S}$, we aim to accurately and efficiently estimate the selectivity of the set containment search of Q on $\mathcal{S}$.

Hereafter, whenever there is no ambiguity, selectivity estimation on set containment search is abbreviated to containment selectivity estimation.

2.1.2 Weighted Set Containment Search

Weighted records are common in real world. For example, the product reviews in Amazon can be modeled as weighted dataset, where each user corresponds to one record and every entry in the record is a rating score of one product. The formal definition of weighted records is as follows.

Definition 1 (*Weighted records*) Given the element universe $\mathcal{E} = \{e_1, e_2, \dots, e_n\}$, an n-dimensional weighted record

X_i is a set of n elements w_{ij}'s (i.e., $X_i = \{w_{i1}, w_{i2}, \dots, w_{in}\}$), where w_{ij} is the weight of X_i with respect to e_j and w_{ij} belongs to the domain $D_j \in R$ (R is one-dimensional real space).

When setting $D_j = \{0, 1\}$, the (unweighted) records are obtained. Next, we present the definition of set containment search problem on weighted records.

Definition 2 (*Weighted records inclusion*) A weighted record $X_i = \{w_{i1}, \dots, w_{in}\}$ is said to be contained by $X_j = \{w_{j1}, w_{j2}, \dots, w_{jn}\}$, denoted by $X_i \subseteq_w X_j$, if the corresponding weights satisfy $w_{i1} \le w_{j1} \wedge w_{i2} \le w_{j2} \wedge \dots \wedge w_{in} \le w_{jn}$.

Given the definition of weighted records inclusion, we formally present the set containment search problem on weighted records.

Definition 3 (*Weighted set containment search*) Given a weighted query record $Q = \{q_1, q_2, \dots, q_n\}$, the weighted set containment search retrieves all the records X_i's from weighed dataset $\mathcal{S}$ that satisfy $X_i \subseteq_w Q$.

Our goal is to estimate the size of the query result given a query Q, i.e., the selectivity of weighted set containment search. Obviously, when the domains of weight are set as $\{0, 1\}$, the weighted set containment search problem degenerates to the simple set containment search problem.

2.1.3 Estimation Measure

In order to evaluate the accuracy of containment selectivity estimation, we apply the *mean square error* (MSE) to measure the expected difference between an estimator and the true value. The MSE formula is as follows,

$$E(\hat{t} - t)^2 = \mathrm{Var}(\hat{t}) + (E(\hat{t}) - t)^2 \qquad (2)$$

where $\hat{t}$ is an estimator for t. If $\hat{t}$ is an unbiased estimator, the MSE is simply the variance.

2.2 KMV Synopses

The k minimum value (*KMV*) technique first introduced in [11] is to estimate the number of distinct elements in a large dataset. Given a no-collision hash function h which maps elements to range [0, 1], a *KMV* synopses of a record (set) X, denoted by $\mathcal{L}_X$, is to keep k minimum hash values of X. Then, the number of distinct elements $|X|$ can be estimated by $\widehat{|X|} = \frac{k-1}{U_{(k)}}$ where $U_{(k)}$ is kth smallest hash value. Beyer et al. [12] also methodically analyze the problem of distinct element estimation under set operations. As for union operation, consider two records X and Y with corresponding *KMV* synopses $\mathcal{L}_X$ and $\mathcal{L}_Y$ of size k_X and k_Y, respectively. In [12], $\mathcal{L}_X \oplus \mathcal{L}_Y$ represents the set consisting of the k smallest hash values in $\mathcal{L}_X \cup \mathcal{L}_Y$ where $k = \min(k_X, k_Y)$. Then, the *KMV* synopses of $X \cup Y$ is $\mathcal{L} = \mathcal{L}_X \oplus \mathcal{L}_Y$. An unbiased estimator for the number of distinct elements in $X \cup Y$, denoted by $D_\cup = |X \cup Y|$, is as follows.

$$\hat{D}_\cup = \frac{k-1}{U_{(k)}} \qquad (3)$$

The variance of $\hat{D}_\cup$, as shown in [12], is

$$\mathrm{Var}[\hat{D}_\cup] = \frac{D_\cup(D_\cup - k + 1)}{k-2} \qquad (4)$$

As shown in [12], Eq. 3 can be modified to compound set operation where $\mathcal{L} = \mathcal{L}_{A_1} \oplus \cdots \oplus \mathcal{L}_{A_n}$ and $k = \min(k_{A_1}, \dots, k_{A_n})$.

An improved *KMV* sketch, named *G-KMV*, is proposed to estimate the multi-union size in [32]. *G-KMV* imposes a global threshold and ensures that all hash values smaller than the threshold will be kept. Considering a union operation $\bigcup X_i$ with the sketch as $\mathcal{L} = \mathcal{L}_{X_1} \cup \mathcal{L}_{X_2} \dots \cup \mathcal{L}_{X_n}$, the sketch size k for the union is $k = |\mathcal{L}_{X_1} \cup \mathcal{L}_{X_2} \dots \cup \mathcal{L}_{X_n}|$. The estimation variance by *G-KMV* method is smaller than that of simple *KMV* method under reasonable assumptions as analyzed in [35].

3 Baseline Solutions

In this section, we introduce two baseline solutions following simple random sampling and *G-KMV* sketching techniques, respectively.

3.1 Random Sampling Approach

A simple way to tackle the set containment estimation problem is to adopt the random sampling techniques and conduct set containment search over a sampled dataset $\mathcal{S}'$ which is usually much smaller compared with the original dataset $\mathcal{S}$. After getting the selectivity of Q on sampled dataset $\mathcal{S}'$, we scale it up to get an estimation of containment selectivity regarding $\mathcal{S}$.

Given sampling size budget b in terms of number of records, we describe the random sampling-based approach in the following two steps: (1) uniformly at random sample b ($b \ll m$) records $X_1, X_2, \dots, X_b$ from $\mathcal{S}$and (2) compare each sampled record X_i ($1 \le i \le b$) with the query Q and assign $\mathbf{n}_i$ accordingly. Recall that $\mathbf{n}_i$ is the containment indicator for a record X_i as shown in Eq. 1. Based on this, the containment selectivity estimator ($\hat{t_\mathcal{R}}$) of the random sampling approach is:

$$\hat{t}_\mathcal{R} = \frac{m}{b} \sum_{i=1}^{b} \mathbf{n}_i \qquad (5)$$

Note that $\mathbf{n}_i$ is a binary random variable because of the random sampling on records. Next, we show that the estimator for baseline solution $\hat{t_{\mathcal{R}}}$ is an unbiased estimator and then derive its variance. We first compute the probability of the event $\{\mathbf{n}_i = 1\}$. Let t denote the containment selectivity over dataset $\mathcal{S}$ with respect to query Q, i.e., $t = |\{X | X \in \mathcal{S}, Q \supseteq X\}|$, then $\Pr[\mathbf{n}_i = 1] = \frac{t}{m}$ where m is total number of records, and thus, the expectation of $\mathbf{n}_i$ is $E[\mathbf{n}_i] = \frac{t}{m}$. By the linearity of expectation, we get the expectation of the estimator for baseline solution in Eq. 5 is $E[\hat{t_{\mathcal{R}}}] = t$, and the variance is

$$\mathrm{Var}[\hat{t_{\mathcal{R}}}] = \frac{t(m-t)}{b}. \tag{6}$$

3.2 IL-GKMV: Inverted List and G-KMV Sketch-Based Approach

The random sampling method, which is very efficient, may result in poor accuracy because it ignores the data distribution information, e.g., the distribution of element frequency or record length. In this section, we develop containment selectivity estimation techniques which are data dependent by utilizing the inverted list and *G-KMV* sketch techniques.

In the first step, we build an inverted index $\mathcal{I}$ on the dataset $\mathcal{S}$ where an element (token) e_i is associated with a list of record identifiers such that the corresponding records contain the element e_i [10]. For instance, in Table 1, the inverted list of element e_3 is $\{X_1, X_2, X_5\}$. Let f_i denote the frequency of an element e_i, i.e., the size of the inverted list I_{e_i}; let $\Pr[e_i = 1]$ denote the probability that a record in a dataset contains the element e_i, then we have $\Pr[e_i = 1] = \frac{f_i}{m}$. Similarly, given a record $X = \{e_1, e_2, \dots, e_{|X|}\}$, the probability of X appearing in the dataset is

$$\Pr[X = 1] = \Pr\left[\bigcap_{e \in X}\{e = 1\}, \bigcap_{e \in \mathcal{E}\backslash X}\{e = 0\}\right].$$

Note that record X can be duplicated in the dataset $\mathcal{S}$; given a query Q, the containment selectivity t of Q is calculated as

$$\hat{t} = \sum_{X \in 2^Q} m * \Pr[X = 1] \tag{7}$$

where the sum is over all subsets of Q. The above equation enumerates every subset of the query Q to check if it appears in the dataset. In order to compute Eq. 7, we need to compute the joint probability $\Pr[X = 1]$ for each subset X of Q. Clearly, the complexity in Eq. 7 is exponentially dependent on the query size $|Q|$ which is not acceptable when $|Q|$ is large. Furthermore, the joint probability computation of $\Pr[X = 1]$ is complicated and expensive.

Given the difficulty of directly computing the containment selectivity, we consider the complement version of set containment search. It is easy to see that $X_i \subseteq Q$ if and only if $\mathcal{E}\backslash X_i \supset \mathcal{E}\backslash Q$; this implies that if an element $e \in \mathcal{E}\backslash Q$ and there exists a record X with $e \in X$, then record X is definitely not a subset of the query Q. Thus, if we exclude all the records that contain any element in $\mathcal{E}\backslash Q$, the remaining records in dataset $\mathcal{S}$ are all subsets of Q, namely, satisfying the set containment search. Given that, the containment selectivity t of query Q can be computed as

$$t = m - m * \Pr\left[\bigcup_{e \in \mathcal{E}\backslash Q} e = 1\right] \tag{8}$$

where $\Pr[e = 1]$ denotes the probability that some record in the dataset $\mathcal{S}$ contains the element e. Remind that the event $\{e = 1\}$ corresponds to all the records containing element e in dataset $\mathcal{S}$, i.e., the inverted list $I_e = \{X | e \in X\}$, we can rewrite Eq. 8 as

$$t = m - \left|\bigcup_{e \in \mathcal{E}\backslash Q} I_e\right| \tag{9}$$

The key point in the above equation is to calculate the union size of the inverted lists, which has the time complexity of $\sum_{e \in \mathcal{E}\backslash Q} |I_e|$ by merge join. Since the set of $\mathcal{E}\backslash Q$ and the inverted list I_e could both be very large, directly computing the multi-union operation could result in unaffordable time consumption. Based on this, we adopt approximate methods (e.g., *G-KMV* sketch) to estimate the union size of the inverted lists.

For each element $e \in \mathcal{E}$, $\mathcal{L}_e$ denotes the *G-KMV* synopsis of its inverted list with k ($=|\mathcal{L}_e|$) smallest hash values. Considering the union of inverted lists in Eq. 9, we have the sketch $\mathcal{L} = \bigcup_{e \in \mathcal{E}\backslash Q} \mathcal{L}_e$ and $k = |\mathcal{L}|$ as introduced in Sect. 2.2, and then, the size $D_\cup$ of the multi-union set $\bigcup_{e \in \mathcal{E}\backslash Q} I_e$ can be estimated as $\hat{D}_\cup = \frac{k-1}{U_{(k)}}$, where $U_{(k)}$ is the kth smallest hash value in the synopsis $\mathcal{L}$. Thus, the containment selectivity of *G-KMV* sketch-based method is computed as $\hat{t}_{\mathcal{G}} = m - \hat{D}_\cup$. Furthermore, the variance can be calculated as $\mathrm{Var}[\hat{t}_{\mathcal{G}}] = \frac{D_\cup(D_\cup - k + 1)}{k-2}$ by Eq. 4.

3.2.1 Analysis

Given the space budget b in terms of the number of records, the sketch size of *IL-GKMV* method is $|\mathcal{L}| \approx b * \bar{d}$ where $\bar{d}$ denotes the average record length. By *G-KMV* sketch, the budget size is proportionally assigned to each inverted list. Apparently, with the very large vocabulary size, the performance significantly deteriorates since each inverted list receives little sampling space. Remark that the *time complexity* for simple random sampling method is $O(b * C)$ where C is the time cost for set comparison. The time cost of *IL-GKMV* is $O(|\mathcal{L}|)$ which is comparable with $O(b * C)$ since $|\mathcal{L}| \approx b * \bar{d}$.

4 Our Approach

As analyzed in the previous section, the random sampling approach fails to capture the element frequency distribution. *IL-GKMV* approach, on the other hand, considers data distribution by utilizing the inverted lists (i.e., frequent elements are associated with longer inverted lists) and *G-KMV* sketch (i.e., inverted lists with larger size keep more hashing values) techniques. However, because the inherent superset query semantics studied in this paper, the number of inverted lists involved in *IL-GKMV* method linearly depends on the vocabulary size which leads to compromised accuracy. In this section, we aim to develop sophisticated sampling approaches which strike a balance between accuracy and efficiency.

4.1 Trie Structure-Based Stratified Sampling Approach

Trie is a widely used tree data structure for storing a set of records (i.e., dataset). Observing that combinations of high-frequency elements (i.e., frequent patterns) occur among records with high frequency, and records with similar frequent patterns are more likely to be included by the same query, we adopt the trie structure to partition the dataset using the combinations of high-frequency elements as *labels*. Assume that elements of the vocabulary $\mathcal{E}$ are ordered based on decreasing frequency in the underlying dataset. For example, the most frequent element in Table 1 is e_2 as it appears 5 times; e_7 appears 4 times and is ranked second place. Based on this ordering, we refer the top-k high-frequency elements as $\mathcal{E}_k$ and adopt the combination of high-frequency elements within $\mathcal{E}_k$ as label. The choice of k will be discussed later in Sect. 6.

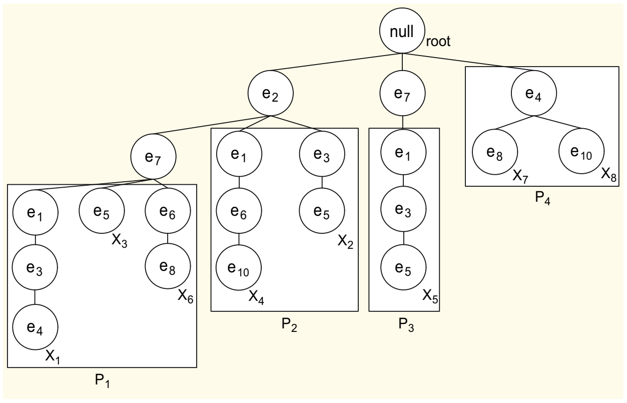

Fig. 1 Trie structure

Figure 1 illustrates an ordered trie T built on dataset in Table 1. It is easy to see that each record in the trie is stored in a top-to-down manner with a start node as *null*. Next, we give an example about the *labels*.

Example 1 Consider the top-2 elements $\mathcal{E}_2$ in Fig. 1; $\{e_2, e_7\}$ is the label for records X_1, X_3, X_6, $\{e_2\}$ is for records X_4, X_2, and $\{e_7\}$ is for X_5.

```
Algorithm 1: Ordered Trie Structure Based Estimation
Input  : $Q$, a query set; $b$, sample size budget
         $\mathcal{S}$, a dataset; $k$, top-$k$ high-frequency elements
Output : $\hat{t}$: estimation of containment selectivity under query $Q$
1 $\mathcal{E}_k \leftarrow$ the top-$k$ high-frequency elements;
2 construct a trie $T$ on dataset $\mathcal{S}$;
3 $L \leftarrow$ all labels in trie $T$ w.r.t $\mathcal{E}_k$;
4 for each label $L_i \in L$ do
5     $P_i \leftarrow$ records with $L_i$ as the prefix in trie $T$;
6     $P'_i \leftarrow$ sample $m'_i$ records from $P_i$ based on sample size budget $b$;
7     conduct containment search regarding $Q$ over sampled records $P'_i$;
8 $\hat{t} \leftarrow$ estimator based on each partition $\mathcal{P} = \{P_1, ..., P_{|\mathcal{P}|}\}$;
9 return $\hat{t}$
```

It is interesting to notice that the left and upper part of the trie encompasses most of the datasets, since this part is made up of high-frequency elements in the dataset. Based on this, there is a natural partition strategy generated by the trie T. Namely, from the root node along the high-frequency part (left and upper of trie), each path (label for records) comprises a partition of the dataset since records in the corresponding partition are all made up of this path as prefix. Note that all the remaining records that do not share any high-frequency element are accumulated as a partition by themselves, and we set the label of this partition as ϕ. Here is an example about the partition on trie.

Example 2 In Fig. 1, there are four partitions as $\{X_1, X_3, X_6\}$, $\{X_2, X_4\}$, $\{X_5\}$ and $\{X_7, X_8\}$ with labels $\{e_2, e_7\}$, $\{e_2\}$, $\{e_7\}$ and ϕ, respectively.

Next, we propose an approximate method to compute the containment selectivity based on the partition $\mathcal{P} = \{P_1, \dots, P_{|\mathcal{P}|}\}$. Given a query record Q and sample size budget b (number of sampled records), we allocate the sample size budget proportionally to the size $m_i = |P_i|$ of each partition in $\mathcal{P}$ (i.e., stratified sampling). Namely, for partition P_i, there are $m'_i = \frac{|P_i|}{m} * b$ records uniformly at random sampled from P_i. Let P'_i denote these sampled records, i.e., $P'_i = \{X_{i1}, \dots, X_{im'_i}\}$, then in each partition, the query Q is compared with each sampled records X_{ij}; let $\mathbf{n}_{ij}$ be the indicator such that

$$\mathbf{n}_{ij} := \begin{cases} 1 & \text{if} \quad X_{ij} \subseteq Q, \\ 0 & \text{otherwise,} \end{cases} \tag{10}$$

then an estimator of the containment selectivity is

$$\hat{t}_{\mathcal{P}} = \sum_{P_i \in \mathcal{P}} \frac{m_i}{m'_i} \sum_{j=1}^{m'_i} \mathbf{n}_{ij} \tag{11}$$

Algorithm 1 illustrates the ordered trie-based sampling approach (*OT-Sampling*). Line 1 collects the k most frequent elements $\mathcal{E}_k$, and Line 2 constructs the ordered trie structure based on the dataset $\mathcal{S}$, followed by obtaining the labels according to $\mathcal{E}_k$ (Line 3). Lines 4–7 group the dataset based on the labels, and conduct the set containment search over each sampled P'_i from individual partitions regarding Q. Line 8 retrieves the final selectivity estimation.

4.1.1 Analysis

Next, we show that the estimator $\hat{t}_{\mathcal{P}}$ in Eq. 11 is unbiased, followed by an analysis of the variance $\text{Var}[\hat{t}_{\mathcal{P}}]$. Recall that the containment selectivity is $t = |\{X|X \subseteq Q \text{ and } X \in \mathcal{S}\}|$; for each partition P_i, let t_i be the size of subsets of Q in partition P_i, i.e., $t_i = |\{X|X \subseteq Q \text{ and } X \in P_i\}|$, and $t = \sum_{P_i \in \mathcal{P}} t_i$, then we have $\Pr[\mathbf{n}_{ij} = 1] = \frac{t_i}{m_i}$ which means that the probability of a sampled record X_{ij} in partition P_i being the subset of Q is $\frac{t_i}{m_i}$; the expectation of $\mathbf{n}_{ij}$ is $E[\mathbf{n}_{ij}] = \frac{t_i}{m_i}$, and variance is $\text{Var}[\mathbf{n}_{ij}] = \frac{t_i(m_i - t_i)}{m_i^2}$. Let $\hat{t}_i = \frac{m_i}{m'_i} \sum_{j=1}^{m'_i} \mathbf{n}_{ij}$, then $E[\hat{t}_i] = t_i$ and $\text{Var}[\hat{t}_i] = \frac{t_i(m_i - t_i)}{m'_i}$ by linearity of expectation, and thus, the expectation of Eq. 11 is

$$E[\hat{t}_{\mathcal{P}}] = \sum_{P_i \in \mathcal{P}} E[\hat{t}_i] = t$$

which proves that $\hat{t}_{\mathcal{P}}$ is an unbiased estimator of containment selectivity. Similarly, the variance of $\hat{t}_{\mathcal{P}}$ is

$$\text{Var}[\hat{t}_{\mathcal{P}}] = \sum_{P_i \in \mathcal{P}} \text{Var}[\hat{t}_i] = \sum_{P_i \in \mathcal{P}} \frac{t_i(m_i - t_i)}{m'_i} \tag{12}$$

4.1.2 Compare with Random Sampling (RS) Approach

Comparing the variance of *OT-Sampling* in Eq. 12 with that of *RS-Sampling* in Eq. 6, we show that $\text{Var}[\hat{t}_{\mathcal{P}}] \leq \text{Var}[\hat{t}_B]$ as follows. Let p_i denote the sampling probability in partition P_i, and there is $p_i = \frac{m'_i}{m_i} = \frac{b}{m}$ by the stratified sampling strategy. Suppose that the number of partitions is $q = |\mathcal{P}|$, then we have $\text{Var}[\hat{t}_{\mathcal{P}}] - \text{Var}[\hat{t}_B] = -\sum_{(i,j) \in \binom{q}{2}} \frac{\prod_{k=1}^{q} m_k}{m_i m_j} (t_i m_j - t_j m_i)^2 \leq 0$.

4.1.3 Time Complexity

The time complexity of the *OT-Sampling* method is $O(b * C) + O(P)$ where C is the containment check cost and $O(P)$ is the preprocess time on trie partition. As demonstrated in our empirical studies, $O(b * C)$ is the dominating cost and $O(P)$ is negligible since we only consider top-k (small k).

4.2 Divide-and-Conquer-Based Sampling Approach

In *OT-Sampling*, the sampling strategy is independent of query workload; that is, we do not distinguish the data information (e.g., labels) of each partition with respect to the query. In this section, we propose a query-oriented sampling approach to improve the estimation accuracy.

Consider the records X's in a dataset as binary vectors with respect to the element universe $\mathcal{E} = \{e_1, \dots, e_n\}$, i.e., each record is regarded as a size-n vector with ith position as 1 if $e_i \in X$ and 0 otherwise; *divide* the element universe $\mathcal{E}$ into two disjoint parts as $\mathcal{E}_1$ and $\mathcal{E}_2$, and then, each record X can be written as two parts X_1 and X_2 corresponding $\mathcal{E}_1$ and $\mathcal{E}_2$, respectively, and we have $X = \{X_1; X_2\}$ where X_1 is concatenated with X_2. We give a lemma based on the division.

Lemma 1 (Subset inclusion) *Given a query record Q and a record X from the dataset $\mathcal{S}$, Q and X are under the same division strategy described above and let $Q = \{Q_1; Q_2\}$ and $X = \{X_1; X_2\}$. We have $X \subseteq Q$ if and only if $X_1 \subseteq Q_1$ and $X_2 \subseteq Q_2$.*

The proof of the lemma is straightforward. From this lemma, a simple pruning technique can be derived such that if $X_1 \nsubseteq Q_1$ then $X \nsubseteq Q$.

Recall the tire-based partition method, we partition the dataset into several groups by the labels of records, where the label can be regarded as the *representative* for each partition. Before drawing samples from a partition with label X_1, we can calculate if X_1 is a subset of query Q. If not, we can skip sampling from that collection of records with X_1 as a label. In order to specify the grouping of records, we give a definition as follows.

Definition 4 (*(E_1, E_2)-prefix collection*) Given E_1 and E_2 as the subsets of element universe $\mathcal{E}$, the *(E_1, E_2)-prefix collection* of records denoted as $\mathcal{S}(E_1, E_2)$ consists of all records X's from dataset $\mathcal{S}$ such that all elements of E_1 are contained in X while no element of E_2 appears in X, that is, $\mathcal{S}(E_1, E_2) = \{X \in \mathcal{S} | E_1 \subseteq X \text{ and } E_2 \cap X = \Phi\}$.

Note that E_1 and E_2 are, respectively, named as inclusion element set and exclusion element set.

Example 3 An $(\{e_2\}, \{e_7\})$-prefix collection in Table 1 is $\{X_2, X_4\}$.

$$t_{\mathcal{S}(E_1,E_2)} = \sum_{X \in \mathcal{S}(E_1,E_2)} \mathbf{n}_X * \frac{\Pr[X]}{\Pr[\mathcal{S}(E_1, E_2)]} \tag{14}$$

Now, we can present the lemma which lay the foundation of the divide-and-conquer algorithm.

Lemma 2 *Considering a prefix collection $\mathcal{S}(E_1, E_2)$ and an element e which does not belong to $E_1 \cup E_2$, the containment selectivity of a given query Q within $\mathcal{S}(E_1, E_2)$ can be calculated as*

$$t_{\mathcal{S}(E_1,E_2)} = \Pr[e = 1 | \mathcal{S}(E_1, E_2)] * t_{\mathcal{S}(E_1 \cup \{e\}, E_2)} + \Pr[e = 0 | \mathcal{S}(E_1, E_2)] * t_{\mathcal{S}(E_1, E_2 \cup \{e\})}.$$

The key point in the proof of Lemma 2 is to consider the conditional probability. We omit the detailed proof here due to space limitation.

Algorithm 2: Divide-And-Conquer Exact Algorithm

```
Input  : $\mathcal{S}$, a collection of records as dataset; $Q$, a query set
         $E_1$ ($E_2$), elements included (excluded) in the prefix collection
Output : $\hat{t}$: containment selectivity of query $Q$ within $\mathcal{S}(E_1, E_2)$
1 procedure $\mathbf{T}(\mathcal{S}, E_1, E_2, Q)$
2 if $E_1 \nsubseteq Q$ then
3     return 0
4 choose an element $e \notin E_1 \cup E_2$;
5 return $Pr[e = 1|\mathcal{S}(E_1, E_2)] * \mathbf{T}(\mathcal{S}, E_1 \cup \{e\}, E_2, Q) + Pr[e =$
  $0|\mathcal{S}(E_1, E_2)] * \mathbf{T}(\mathcal{S}, E_1, E_2 \cup \{e\}, Q)$
```

Recall that in Sect. 3.2, we model the record X as a random variable and give the probability that X appears in dataset $\mathcal{S}$. Similarly, we compute the *generating probability* of the prefix collection $\mathcal{S}(E_1, E_2)$ as follows:

$$\Pr[\mathcal{S}(E_1, E_2)] = \Pr\left[\bigcap_{e \in E_1} \{e = 1\}, \bigcap_{e \in E_2} \{e = 0\}\right]. \tag{13}$$

Next, we compute the number of subsets of a given query Q within the prefix collection $\mathcal{S}(E_1, E_2)$, i.e., the containment selectivity in regard to $\mathcal{S}(E_1, E_2)$. Let $\mathbf{n}_X$ denote the indicator function such that

$$\mathbf{n}_X := \begin{cases} 1 & \text{if} \quad Q \supseteq X, \\ 0 & \text{otherwise} \end{cases}$$

then the containment selectivity of Q with respect to $\mathcal{S}(E_1, E_2)$ is

Based on Lemma 2, we propose the divide-and-conquer algorithm illustrated in Algorithm 2. We can calculate the containment selectivity of Q within dataset $\mathcal{S}$ by invoking procedure $\mathbf{T}(\mathcal{S}, \phi, \phi, Q)$; by lemma 2, the dataset is partitioned into two groups of records by choosing an element $e \in \mathcal{E}$ and we have

$$t_{\mathcal{S}(\phi,\phi)} = \Pr[e = 1 | \mathcal{S}(\phi, \phi)] * t_{\mathcal{S}(\{e\},\phi)} + \Pr[e = 0 | \mathcal{S}(\phi, \phi)] * t_{\mathcal{S}(\phi,\{e\})}$$

and then compute the containment selectivity in each of the two groups recursively as shown in Line 4–5. When there is $E_1 \nsubseteq Q$, we can prune this collection of records $\mathcal{S}(E_1, E_2)$ by Lemma 1. Obviously, the time complexity of the exact divide-and-conquer algorithm is $O(C * 2^n)$ where n is the size of the element universe $\mathcal{E}$ and C is the cost of set comparison. Recall that the element frequency distribution is usually skew in real dataset, and we can arrange the elements by decreasing frequency order when choosing the element e in Line 4 of Algorithm 2, which can accelerate the computation by pruning more records corresponding to the high-frequency elements.

4.2.1 Approximate Divide-and-Conquer Algorithm

Next, we propose an approximate method based on the exact divide-and-conquer algorithm. In Algorithm 2, the dataset $\mathcal{S}$ is recursively partitioned into two collections of records by choosing an element $e \notin E_1 \cup E_2$. In addition, we can order the elements by decreasing element frequency to boost the computation efficiency. However, the complexity is still $O(C * 2^n)$. In this section, we only consider the top-k high-frequency elements $\mathcal{E}_k$, from which the element is selected to partition the dataset. After finishing all the elements in $\mathcal{E}_k$, we end up with 2^k prefix collections of records $\mathcal{S}_i(E_1, E_2)$, $i = 1, 2, \dots, 2^k$, which is much smaller than 2^n. Note that (E_1, E_2) can be regarded as the label for each prefix collection.

Recall Lemma 1, all the records X's can be described as the binary vector with $X = \{X_1;X_2\}$ where X_1 corresponds to the top-k high-frequency elements part $\mathcal{E}_k$ and X_2 is the rest part concatenated with X_1. Similarly, when a query record Q arrives, let Q be $Q = \{Q_1;Q_2\}$ following the same manner; then, by Lemma 1, we can exclude all the prefix collections $\mathcal{S}(E_1, E_2)$ with $E_1 \nsubseteq Q_1$. For the remaining prefix collections, we sample some records from each group and conduct containment search of Q over sampled records. Let $X = \{X_1;X_2\}$ be a sampled record, and it is only required to test if $X_2 \subseteq Q_2$ since $X_1 \subseteq Q_1$. In the following part, we formally demonstrate how to estimate the containment selectivity of Q by the divide-and-conquer method.

Let $\mathbf{I}_i$ denote the indicator function for prefix collection $\mathcal{S}_i(E_1, E_2)$ ($\mathcal{S}_i$ for short) such that $\mathbf{I}_i = 1$ when $E_1 \subseteq Q_1$ otherwise 0. The size of prefix collection $\mathcal{S}_i(E_1, E_2)$ can be computed as $m_i = |\mathcal{S}_i(E_1, E_2)| = m * \Pr[\mathcal{S}_i(E_1, E_2)]$ by Eq. 13. Let p_i be the sampling probability in $\mathcal{S}_i$, then the sample size is $m'_i = m_i * p_i$. For any sampled record, $X_j = \{X_1;X_2\}$ in this prefix collection $\mathcal{S}_i$, and let $\mathbf{n}_{ij}$ be the indicator for which $\mathbf{n}_{ij} = 1$ if $X_2 \subseteq Q_2$ otherwise 0. Then, an estimator for the containment selectivity of Q by divide-and-conquer algorithm can be expressed as

$$\hat{t}_{\mathcal{D}} = \sum_{\mathcal{S}_i} \mathbf{I}_i \sum_{j=1}^{m'_i} \frac{\mathbf{n}_{ij}}{p_i} \tag{15}$$

It can be verified that $\hat{t}_{\mathcal{D}}$ is an unbiased estimator and the variance of $\hat{t}_{\mathcal{D}}$ is

$$\mathrm{Var}[\hat{t}_{\mathcal{D}}] = \sum_{\mathcal{S}_i} \mathbf{I}_i * \frac{t_i(m_i - t_i)}{p_i m_i} \tag{16}$$

where t_i is the number of records satisfying $X_2 \subseteq Q_2$ in $\mathcal{S}_i$. Let $\mathcal{S}_i$, $i = 1, 2, \dots, l$ be all the prefix collections with $E_1 \subseteq Q_1$ for a given query Q, then the variance can be written as $\mathrm{Var}[\hat{t}_{\mathcal{D}}] = \sum_{i=1}^{l} \frac{t_i(m_i - t_i)}{p_i m_i}$.

4.2.2 Compare with OT-Sampling

Obviously, in *DC-Sampling* method, we avoid allocating the space budget to unqualified partitions compared with *OT-Sampling*. In formal, assume there are q partitions (corresponding to prefix collections) in total with $\{P_1, \dots, P_q\}$; after pruning, there remains l partitions, w.l.o.g, $\{P_1, \dots, P_l\}$. Then, for *DC-Sampling*, the sampling probability is $p_i = \frac{b}{\sum_{i=1}^{l} m_i}$ where $m_i = |P_i|$ and b is space budget, and the sampling probability of *OT-Sampling* is $p'_i = \frac{b}{\sum_{i=1}^{q} m_i}$. Thus, we have $\mathrm{Var}[\hat{t}_{\mathcal{P}}] - \mathrm{Var}[\hat{t}_{\mathcal{D}}] = \sum_{i=1}^{l} (\frac{1}{p'_i m_i} - \frac{1}{p_i m_i}) t_i(m_i - t_i) + \sum_{i=l+1}^{q} \frac{1}{p'_i m_i} t_i(m_i - t_i) \geq 0$ since $p'_i \leq p_i$.

4.2.3 Time Complexity

The time complexity of DC-Sampling method is $O(b * \hat{C}) + O(P)$ where$\hat{C}$ is the cost for two-record containment check. Here, we use merge join to check if one record is included by a given query record Q, and $O(P)$ is the preprocess time on partition the records by prefix. After pruning the unqualified partitions, we can skip comparing the prefix part of a record with the query by our algorithm, and thus, the time cost of $\hat{C}$ is smaller than that of OT-sampling, which leads to better efficiency than DC-Sampling.

5 Selectivity Estimation on Weighted Set Containment Search

In this section, we consider the set containment search on weighted records. We first present a simple random sampling method to address the problem, followed by the stratified sampling method to boost the estimation accuracy.

5.1 Random Sampling Approach

Similar to the selectivity estimation of simple set containment search problem, we can apply the naive random sampling method to selectivity estimation with regard to weighted datasets. Namely, given a (weighted) query record Q and a space budget b, we first uniformly and at random sample b weighted records $(X_1, X_2, \dots, X_b)$ from the dataset $\mathcal{S}$, and then, we compare query Q with each sampled record X_i to verify if X_i is weighted included by Q (according to Definition 2) and count the number of sampled records satisfying $X_i \subseteq_w Q$; we finally scale up the counting result to get a selectivity estimation on weighted set containment search. Similar to Eq. 5, the estimator $\hat{t}^w_{\mathcal{R}}$ is denoted as:

$$\hat{t}^w_{\mathcal{R}} = \frac{m}{b}\sum_{i=1}^{b} \mathbf{n}^w_i \tag{17}$$

where $\mathbf{n}^w_i$ is the indicator function based on weighted records inclusion, i.e., $\mathbf{n}^w_i = 1$ if $X_i \subseteq_w Q$; otherwise, $\mathbf{n}^w_i = 0$. Obviously, the estimator $\hat{t}^w_{\mathcal{R}}$ is an unbiased estimator; the variance can be computed as

$$\mathrm{Var}[\hat{t}^w_{\mathcal{R}}] = \frac{t(m-t)}{b}. \tag{18}$$

5.2 Stratified Random Sampling

In order to boost the estimation accuracy, we can also utilize the partition-based sampling method (e.g., stratified sampling). Unfortunately, it is not applicable to group the records by building a trie structure on the weighted dataset, because the weights of each dimension among the records can be quite distinct and the number of partitions based on trie structure could be the same order of the number of records in the dataset. In the following part, we take into account the distribution of weights in each dimension and partition the dataset recursively on each dimension by dichotomizing the corresponding domain. Based on the partition, we present a divide-and-conquer algorithm given a query record to estimate the selectivity of weighted set containment search.

Consider the domain D_j of jth dimension corresponding to e_j; let $w_j = \{w_{1j}, w_{2j}, \ldots, w_{mj}\}$ denote all the weights of dataset in jth dimension. Assume w_{ij} $(i = 1, 2, \ldots, m)$ follows norm distribution with $w_{ij} \sim \mathcal{N}(\mu_j, \sigma_j^2)$, and then, we can use the mean value μ_j as a boundary to partition the records; that is, records X_is with ith weight $w_{ij} < \mu_j$ are grouped together and the remaining records are collected in another group. Also, we choose the top-k high-frequency weights to partition the dataset. Remark that the weights of each record are sorted by the decreasing order of weight frequency in the dataset, and the weight frequency of jth dimension (e_j) is the number of nonzero weights in this dimension. Similar to the estimation in Sect. 4.1, Algorithm 3 illustrates the estimation strategy of stratified sampling.

Note that when the weights are binary values, the above partition strategy is the same as Algorithm 2 for the simple (unweighted) case. Based on the partition $\mathcal{P}^w = \{P_1, \ldots, P_{|\mathcal{P}|}\}$, we can estimate the weighted set containment selectivity similar to Eq. 11 as

$$\hat{t}^w_{\mathcal{P}^w} = \sum_{P_i \in \mathcal{P}^w} \frac{m_i}{m'_i} \sum_{j=1}^{m'_i} \mathbf{n}_{ij} \tag{19}$$

which is an unbiased estimator with variance computed as

$$\mathrm{Var}[\hat{t}^w_{\mathcal{P}^w}] = \sum_{P_i \in \mathcal{P}^w} \mathrm{Var}[\hat{t}^w_i] = \sum_{P_i \in \mathcal{P}^w} \frac{t_i(m_i - t_i)}{m'_i} \tag{20}$$

Similarly, it can be proved that the variance of stratified sampling approach is smaller than that of naive random sampling method, i.e., $\mathrm{Var}[\hat{t}^w_{\mathcal{P}^w}] \le \mathrm{Var}[\hat{t}^w_{\mathcal{R}}]$.

5.3 Query-Oriented Sampling

Furthermore, the estimation accuracy can be improved by utilizing the query record information. Given a weighted query record Q with top-k high-frequency weighted elements $\mathcal{E}_Q$, we first compare $\mathcal{E}_Q$ with the labels of each partition in $\mathcal{P}^w = \{P_1, \ldots, P_{|\mathcal{P}^w|}\}$ to prune some partitions with records that cannot be included by the query Q. With the pruned partition $\mathcal{P}' = \{P'_1, \ldots, P'_{|\mathcal{P}'|}\}$, we can get an estimator as follows:

$$\hat{t}' = \sum_{P_i \in \mathcal{P}'} \frac{m_i}{m'_i} \sum_{j=1}^{m'_i} \mathbf{n}_{ij} \tag{21}$$

It can also be shown that the variance of $\hat{t}'$ is smaller than that of $\hat{t}^w_{\mathcal{P}^w}$, $\mathrm{Var}[\hat{t}'] \le \mathrm{Var}[\hat{t}^w_{\mathcal{P}^w}]$.

6 Experimental Evaluation

In this section, we evaluate the estimation accuracy and computation efficiency of different strategies on a variety of real-life datasets. All experiments are conducted on PCs with Intel Xeon 2 × 2.3 GHz CPU and 128 GB RAM running Debian Linux.

Algorithm 3: **Stratified Random Sampling Approach**

```
Input  : $Q$, a query set; $b$, sample size budget
         $\mathcal{S}$, a weighted dataset with weights sorted by decreasing frequency
         $k$, top-$k$ high-frequency weighted elements
Output : $\hat{t}$: estimation of containment selectivity under query $Q$
1 $\mathcal{E}^w_k \leftarrow$ the top-$k$ high-frequency elements;
2 partition the weighted dataset by $\mathcal{E}^w_k$ into $\mathcal{P}^w = \{P_1, ..., P_{|\mathcal{P}^w|}\}$ ;
3 $\hat{t} \leftarrow$ estimator based on each partition $\mathcal{P}^w = \{P_1, ..., P_{|\mathcal{P}^w|}\}$;
4 return $\hat{t}$
```

6.1 Experimental Setting

Algorithms Since there exists no previous work for tackling the problem of set containment selectivity estimation, we evaluate the following estimation methods introduced in this paper.

- *RS* Direct random sampling method in Sect. 3.1.
- *IL-GKMV* Inverted lists and G-KMV sketch-based method in Sect. 3.2.
- *OT-Sampling* Ordered trie structure-based sampling method in Sect. 4.1.
- *DC-Sampling* The divide-and-conquer-based sampling method in Sect. 4.2.

We also evaluate the methods for weighted set containment selectivity estimation as follows.

- *RS* Direct random sampling method in Sect. 5.
- *StrRS* Stratified random sampling method in Sect. 5.
- *StrQRS* Stratified query-oriented random sampling method in Sect. 5.

The above algorithms are implemented in C++. In verifying the inclusion relationship between the query and records, we apply the merge join method. For records with large size, we utilize the prefix tree structure to boost the computation efficiency.

Datasets We deploy nine real-life datasets which are chosen from various domains with different data properties. In Table 3, we illustrate the characteristics of these nine datasets in details. Note that the last three are weighted datasets. For each dataset, we show the representations of record and element, the number of records, the average record length and the number of distinct elements in dataset.

Workload The workload for the selectivity estimation of set containment search is made up of 10,000 queries, each of which is uniformly and at random selected from the dataset. Note that we exclude the queries with size smaller than 10 in order to evaluate the accuracy properly.

Measurement In the following part, we use relative error to measure the accuracy. Let t be the exact result, and $\hat{t}$ be the estimation one, then the relative error denoted by ϵ is calculated as $\epsilon = \frac{|t-\hat{t}|}{t}$. The sampling size is in terms of the number of records. For *IL-GKMV* approach, the space budget is allocated as discussed at the end of Sect. 3.

Tuning k In order to evaluate the impact of the high-frequency elements in *OT-Sampling* and *DC-Sampling*, we first tune the number of the highest frequency elements, i.e., top-k. By experimental study, we set the k value as 12 which can well balance the trade-off between accuracy and efficiency.

6.2 Overall Performance

Figure 2a compares the estimation accuracy and time cost of the four algorithms on six datasets. The sample size is

Table 3 Characteristics of datasets

Dataset	Abbreviation	Record	Elements	#Records	AvgLength	#Elements
Bookcrossing [1]	BOOKC	Book	User	340,523	3.38	105,278
Delicious [2]	DELIC	User	Tag	833,081	98.42	4,512,099
Livejournal [3]	LIVEJ	User	Group	3,201,203	35.08	7,489,073
Netflix [13]	NETFLIX	Movie	Rating	480,189	209.25	17,770
Sualize [4]	SUALZ	Picture	Tag	495,402	3.63	82,035
Twitter [22]	TWITTER	Partition	User	371,586	65.96	1318
MovieLens [5]	MLENS	Review	Rating	138,000	110.25	27,000
Amazon [6]	AMAZON	Product	Rating	7,781,990	4.67	548,552
Dating [7]	DATING	Review	Rating	17,359,346	40.5	168,791

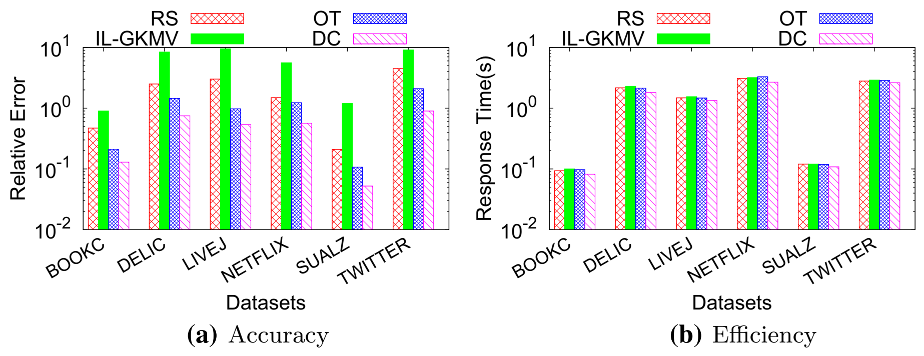

Fig. 2 Overall performance

set as 1000 in terms of the number of records; for trie structure-based approach and divide-and-conquer algorithm, the k value is 12 as mentioned above. Overall, we can see that the divide-and-conquer (*DC-Sampling*) algorithm achieves the best performance in the accuracy on all datasets, which can reduce the relative error of the random sampling (*RS*) method by around 60% and cut the relative error of *IL-GKMV* method by more than 80%. Also, the ordered trie structure-based approach (*OT-Sampling*) can diminish the relative error of *RS* by around 40% for most datasets and narrow the relative error of *IL-GKMV* by about 70%. Moreover, divide-and-conquer (*DC-Sampling*) algorithm outperforms the ordered tire structure-based approach (*OT-Sampling*) by decreasing the relative error about half.

Figure 2b reports the query response time on six datasets with 10,000 queries, where *DC-Sampling* method consumes

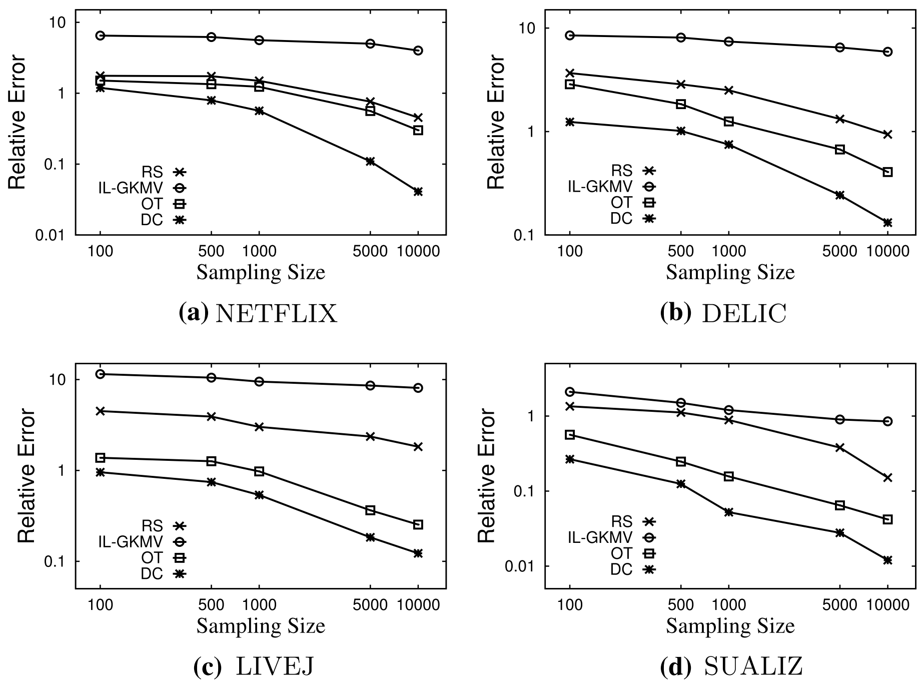

Fig. 3 Accuracy versus space

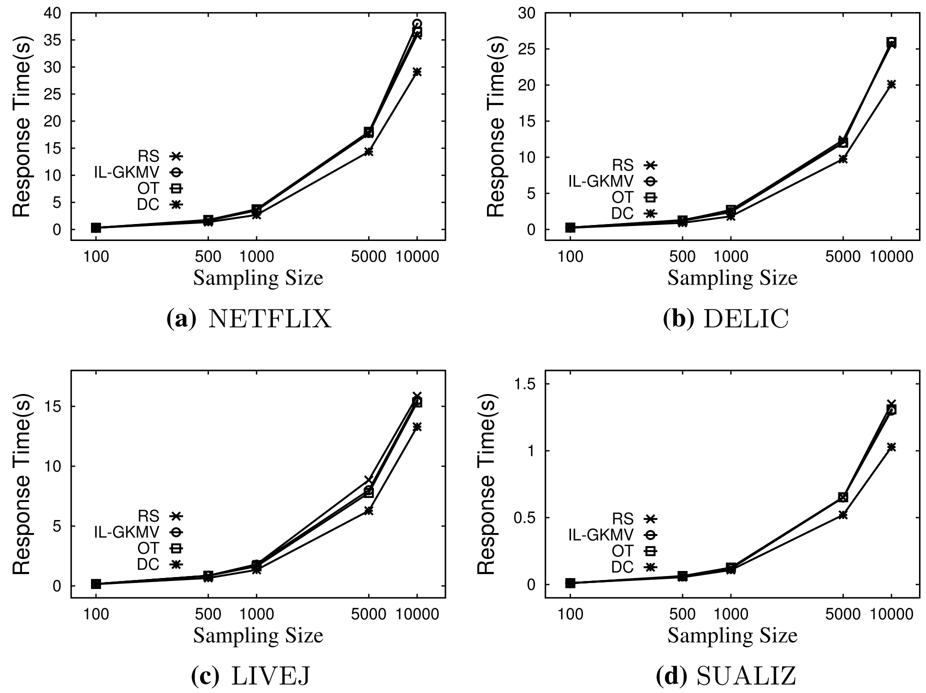

Fig. 4 Efficiency versus space

less time than the other three because of the pruning techniques. It is remarkable that for each dataset, the time costs of the four algorithms are comparable since we keep the same sample size in every algorithm. Meanwhile, the response time varies among different datasets because of the diverse average record lengths, and datasets with larger average length, e.g., NETFLIX with AvgLength 209.25, consume more query time.

6.3 Estimation Accuracy Evaluation

In this section, we assess the effectiveness of the four methods in terms of relative error. We consider the effect of space budget on the estimation accuracy by changing the sampling size. Figure 3 illustrates superior accuracy achievement of *DC-Sampling* against the other three by varying the space budget. As anticipated, the accuracy performance of all algorithms is ameliorated when more sampling size is provided.

6.4 Computation Efficiency Evaluation

In this section, we evaluate the efficiency of the four algorithms in terms of query response time with 10,000 queries. Figure 4 demonstrates the response time of four algorithms with different space budgets. Obviously, the query response time increases as the sampling size grows. The *DC-Sampling* method outperforms the other three algorithms because of the pruning techniques.

6.5 Weighted Set Containment Search

In the last part of experiment, we assess the estimation accuracy and efficiency of selectivity estimation of weighted set containment search. As for the experiment setting, we choose the value *k* as 12 for top-*k* high-frequency weighted elements. Figure 5 illustrates the overall accuracy and efficiency performance of WRS, StrRS and StrQRS method. We can see that the StrQRS method outperforms the other two algorithms under the same space budget, and StrRS method can achieve better accuracy than RS method. The time cost of the three algorithms (with 10,000 query records) is similar since the sample budget (1000 records) is same. Figure 6 compares the accuracy of the three algorithms. Obviously, with the space budget growing, the relative error gets smaller, i.e., the accuracy of the three methods keeps increasing. We can also find that the StrQRS method always beats the others with different sampling sizes.

7 Related Work

To the best our knowledge, there is no existing work on selectivity estimation of set containment search. In this section, we review two important directions closely related to the problem studied in this paper.

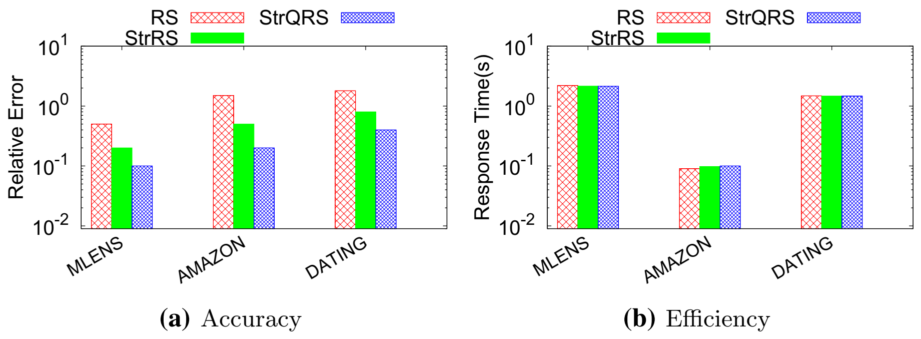

Fig. 5 Overall performance

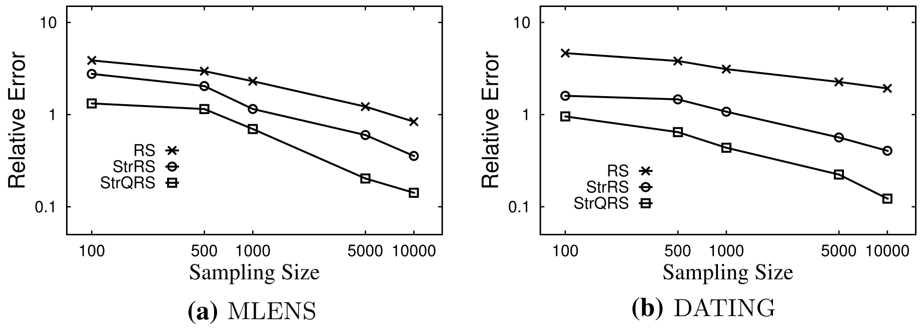

Fig. 6 Accuracy versus space

7.1 Searching Set-Valued Data

The study of set-valued data has attracted great attention from research communities and industrial organizations due to an ever increasing prevalence of set-valued data in a wide range of applications. The research in this area focuses on set containment search [18, 19, 28], set similarity and set containment joins [20, 22, 23, 30]. In one of the representative works on set containment search [28], Terrovitis et al. introduce a OIF index combined the inverted index with B-tree to tackle three kinds of set containment queries: subset queries, equality queries and superset queries. In a recent work [34], Yang et al. propose a TT-join method for the set containment join problem, which is based on prefix tree structure and utilize the element frequency information; they also present a detailed summary of the existing set containment join methods. The containment queries can also be modeled as range searching problem in computational geometry [8]; nevertheless, the performance is exponentially dependent on dimension n which is unsuitable in practice for our problem.

7.2 Selectivity Estimation

The problem of selectivity estimation has been studied for a large variety of queries and over a diverse range of data types such as range queries (e.g., [16]), Boolean queries (e.g., [14]), relational joins (e.g., [29]), spatial join (e.g., [17]) and set intersection (e.g., [16]). Nevertheless, many of the techniques developed above are sensitive to the dimension of data and not applicable to the problem studied in this paper. Moreover, the superset containment semantics brings in extra challenges in adopting existing techniques. Although the set containment search query can be naturally modeled as range counting problem as discussed in Sect. 1, existing range counting techniques are exponentially dependent on the dimensionality (i.e., the number of distinct elements in our problem) and not applicable to solving the containment selectivity estimation problem in our problem [16, 27]. Distinct value estimators (e.g., *KMV* [12], bottom-k, minhash [16]) are adopted in [32] to solve subset containment search (i.e., query record is a subset of data record). We also extend the distinct value estimator *KMV* and develop the *IL-GKMV* approach in Sect. 3 and demonstrate theoretically and through extensive experiments that distinct value estimators cannot efficiently and accurately support the superset containment semantics studied in this paper.

8 Conclusion

The prevalence of set-valued data generates a wide variety of applications that call for sophisticated processing techniques. In this paper, we investigate the problem of selectivity estimation on set containment search and develop novel and efficient sampling-based techniques, *OT-Sampling* and *DC-Sampling*, to address the inherent challenges of set containment search and the limitations of existing techniques. Simple random sampling techniques and a *G-KMV* sketch-based estimating approach *IL-GKMV* are also devised as baseline solutions. Meanwhile, we consider the selectivity estimation of weighted set containment search and propose stratified sampling method to tackle this problem. We theoretically analyze the accuracy of the proposed techniques by means of expectation and variance. Our comprehensive experiments on six real-life datasets empirically verify the effectiveness and efficiency of the sampling-based approaches.

References

1. http://www.informatik.uni-freiburg.de/~cziegler/BX/
2. http://dai-labor.de/IRML/datasets
3. http://socialnetworks.mpi-sws.org/data-imc2007.html
4. http://vi.sualize.us/
5. https://grouplens.org/datasets/movielens/
6. https://snap.stanford.edu/data/amazon-meta.html
7. http://www.occamslab.com/petricek/data/
8. Agarwal PK (1996) Range searching. Technical report, Duke Univ Durham NC Department of Computer Science
9. Agrawal P, Arasu A, Kaushik R (2010) On indexing error-tolerant set containment. In: SIGMOD, pp 927–938
10. Baeza-Yates R, Ribeiro-Neto B et al (1999) Modern information retrieval, vol 463. ACM press, New York
11. Bar-Yossef Z, Jayram T, Kumar R, Sivakumar D, Trevisan L (2002) Counting distinct elements in a data stream. In: International workshop on randomization and approximation techniques in computer science. Springer, pp 1–10
12. Beyer P, Haas PJ, Reinwald B, Sismanis Y, Gemulla R (2007) On synopses for distinct-value estimation under multiset operations. In: SIGMOD, pp 199–210
13. Bouros P, Mamoulis N, Ge S, Terrovitis M (2016) Set containment join revisited. Knowl Inf Syst 49(1):375–402
14. Chen Z, Korn F, Koudas N, Muthukrishnan S (2000) Selectivity estimation for boolean queries. In: PODS, pp 216–225
15. Cohen E, Cormode G, Duffield NG (2011) Structure-aware sampling on data streams. In: SIGMETRICS, pp 197–208
16. Cohen E, Cormode G, Duffield NG (2014) Is min-wise hashing optimal for summarizing set intersction? In: PODS, pp 109–120
17. Das A, Gehrke J, Riedewald M (2004) Approximation techniques for spatial data. In: SIGMOD, pp 695–706
18. Goldman R, Widom J (2000) Wsq/dsq: a practical approach for combined querying of databases and the web. In: ACM SIGMOD record, vol 29. ACM, pp 285–296
19. Helmer S, Moerkotte G (2003) A performance study of four index structures for set-valued attributes of low cardinality. VLDB J 12(3):244–261

20. Jampani R, Pudi V (2005) Using prefix-trees for efficiently computing set joins. In: International conference on database systems for advanced applications. Springer, pp 761–772
21. Li C, Lu J, Lu Y (2008) Efficient merging and filtering algorithms for approximate string searches. In: ICDE, pp 257–266
22. Luo Y, Fletcher GH, Hidders J, De Bra P (2015) Efficient and scalable trie-based algorithms for computing set containment relations. In: ICDE. IEEE, pp 303–314
23. Mamoulis N (2003) Efficient processing of joins on set-valued attributes. In: SIGMOD. ACM, pp 157–168
24. Melnik S, Garcia-Molina H (2003) Adaptive algorithms for set containment joins. TODS 28(1):56–99
25. Ramasamy K, Patel JM, Naughton JF, Kaushik R (2000) Set containment joins: the good, the bad and the ugly. In: VLDB, pp 351–362
26. Shrivastava A, Li P (2015) Asymmetric minwise hashing for indexing binary inner products and set containment. In: Proceedings of the 24th international conference on world wide web. International World Wide Web Conferences Steering Committee, pp 981–991
27. Suri S, Toth C, Zhou Y (2006) Range counting over multidimensional data streams. Discrete Comput Geom 36(4):633–655
28. Terrovitis M, Bouros P, Vassiliadis P, Sellis T, Mamoulis N (2011) Efficient answering of set containment queries for skewed item distributions. In: Proceedings of the 14th international conference on extending database technology. ACM, pp 225–236
29. Tzoumas K, Deshpande A, Jensen CS (2013) Efficiently adapting graphical models for selectivity estimation. PVLDB 22(1):3–27
30. Vernica R, Carey MJ, Li C (2010) Efficient parallel set-similarity joins using mapreduce. In: SIGMOD, pp 495–506. ACM
31. Wang J, Li G, Feng J (2012) Can we beat the prefix filtering?: an adaptive framework for similarity join and search. In: SIGMOD, pp 85–96
32. Wang X, Zhang Y, Zhang W, Lin X, Wang W (2014) Selectivity estimation on streaming spatio-textual data using local correlations. PVLDB 8(2):101–112
33. Xiao C, Wang W, Lin X, Yu JX (2008) Efficient similarity joins for near duplicate detection. In: WWW, pp 131–140
34. Yang J, Zhang W, Yang S, Zhang Y, Lin X (2017) Tt-join: efficient set containment join. In: ICDE, pp 509–520
35. Yang Y, Zhang Y, Zhang W, Huang Z (2018) Gb-kmv: an augmented kmv sketch for approximate containment similarity search. arXiv preprint arXiv:1809.00458
36. Zhu E, Nargesian F, Pu KQ, Miller RJ (2016) Lsh ensemble: internet scale domain search. In: VLDB, pp 1185–1196