

GB-KMV: An Augmented KMV Sketch for Approximate Containment Similarity Search

Yang Yang[†], Ying Zhang[§], Wenjie Zhang[†], Zengfeng Huang[†]

[†]University of New South Wales, [§]University of Technology Sydney
{yang.yang, zhangw}@cse.unsw.edu.au, ying.zhang@uts.edu.au

Abstract—In this paper, we study the problem of approximate containment similarity search. Given two records Q and X , the containment similarity between Q and X with respect to Q is $\frac{|Q \cap X|}{|Q|}$. Given a query record Q and a set of records S , the containment similarity search finds a set of records from S whose containment similarity regarding Q is not less than the given threshold. This problem has many important applications in commercial and scientific fields such as record matching and domain search. Existing solution relies on the *asymmetric LSH* method by transforming the containment similarity to well-studied Jaccard similarity. In this paper, we use an inherently different framework by transforming the containment similarity to set intersection. We propose a novel augmented *KMV* sketch technique, namely *GB-KMV*, which is data-dependent and can achieve a much better trade-off between the sketch size and the accuracy. We provide a set of theoretical analysis to underpin the proposed augmented *KMV* sketch technique, and show that it outperforms the state-of-the-art technique *LSH-E* in terms of estimation accuracy under practical assumption. Our comprehensive experiments on real-life datasets verify that *GB-KMV* is superior to *LSH-E* in terms of the space-accuracy trade-off, time-accuracy trade-off, and the sketch construction time. For instance, with similar estimation accuracy (F-1 score), *GB-KMV* is over 100 times faster than *LSH-E* on several real-life datasets.

I. INTRODUCTION

In many applications such as information retrieval, data cleaning, machine learning and user recommendation, an object (e.g., document, image, web page and user) is described by a set of elements (e.g., words, q -grams, and items). One of the most critical components in these applications is to define the set similarity between two objects and develop corresponding similarity query processing techniques. Given two records (objects) X and Y , a variety of similarity functions/metrics have been identified in the literature for different scenarios (e.g., [28], [15]). Many indexing techniques have been developed to support efficient *exact* and *approximate* lookups and joins based on these similarity functions.

Many of the set similarity functions studied are symmetric functions, i.e., $f(X, Y) = f(Y, X)$, including widely used Jaccard similarity and Cosine similarity. In recent years, much research attention has been given to the asymmetric set similarity functions, which are more appropriate in some applications. Containment similarity (a.k.a, Jaccard containment similarity) is one of the representative asymmetric set similarity functions, where the similarity between two records X and Y is defined as $f(X, Y) = \frac{|X \cap Y|}{|X|}$ in which $|X \cap Y|$ and $|X|$ are intersection size of X and Y and the size of X , respectively.

Compared with symmetric similarity such as Jaccard similarity, containment similarity gives special consideration on the query size, which makes it more suitable in some applications. As shown in [35], containment similarity is useful in record

matching application. Given two text descriptions of two restaurants X and Y which are represented by two “set of words” records: $\{five, guys, burgers, and, fries, downtown, brooklyn, new, york\}$ and $\{five, kitchen, berkeley\}$ respectively. Suppose query Q is $\{five, guys\}$, we have that the Jaccard similarity of Q and X (resp. Y) is $\frac{2}{9} = 0.22$ ($\frac{1}{4} = 0.25$). Note the Jaccard similarity is $f(Q, X) = \frac{|Q \cap X|}{|Q \cup X|}$. Based on the Jaccard similarity, record Y matches better to query Q , but intuitively X should be a better choice. This is because the Jaccard similarity unnecessarily favors the short records. On the other hand, the containment similarity will lead to the desired order with $f(Q, X) = \frac{2}{2} = 1.0$ and $f(Q, Y) = \frac{1}{2} = 0.5$. Containment similarity search can also support online error-tolerant search for matching user queries against addresses (map service) and products (product search). This is because the regular keyword search is usually based on the containment search, and containment similarity search provides a natural error-tolerant alternative [5]. In [44], Zhu et al. show that containment similarity search is essential in domain search which enables users to effectively search Open Data.

The containment similarity is also of interest to applications of computing the fraction of values of one column that are contained in another column. In a dataset, the discovery of all inclusion dependencies is a crucial part of data profiling efforts. It has many applications such as foreign-key detection and data integration(e.g., [22], [31], [8], [33], [30]).

Challenges. The problem of containment similarity search has been intensively studied in the literature in recent years (e.g., [5], [35], [44]). The key challenges of this problem come from the following three aspects: (i) The number of elements (i.e., vocabulary size) may be very large. For instance, the vocabulary will blow up quickly when the higher-order shingles are used [35]. Moreover, query and record may contain many elements. To deal with the sheer volume of the data, it is desirable to use sketch technique to provide effectively and efficiently approximate solutions. (ii) The data distribution (e.g., record size and element frequency) in real-life application may be highly skewed. This may lead to poor performance in practice for *data independent* sketch methods. (iii) A subtle difficulty of the approximate solution comes from the asymmetric property of the containment similarity. It is shown in [34] that there cannot exist any locality sensitive hashing (*LSH*) function family for containment similarity search.

To handle the large scale data and provide quick response, most existing solutions for containment similarity search seek to the *approximate* solutions. Although the use of *LSH* is restricted, the novel *asymmetric LSH* method has been designed in [34] to address the issue by padding techniques. Some enhancements of asymmetric *LSH* techniques are proposed in the following works by introducing different functions

(e.g., [35]). Observe that the performance of the existing solutions are sensitive to the skewness of the record size, Zhu et. al propose a partition-based method based on Minhash *LSH* function. By using optimal partition strategy based on the size distribution of the records, the new approach can achieve much better time-accuracy trade-off.

We notice that all existing approximate solutions rely on the *LSH* functions by transforming the containment similarity to well-studied *Jaccard similarity*. That is,

$$\frac{|Q \cap X|}{|Q|} = \frac{|Q \cap X|}{|Q \cup X|} \times |Q \cup X| \times \frac{1}{|Q|}$$

As the size of query is usually readily available, the estimation error come from the computation of Jaccard similarity and union size of Q and X . Note that although the union size can be derived from jaccard similarity [44], the large variance caused by the combination of two estimations remains. This motivates we to use a different framework by transforming the containment similarity to *set intersection size estimation*, and the error is only contributed by the estimation of $|Q \cap X|$. The well-known *KMV* sketch [11] has been widely used to estimate the set intersection size, which can be immediately applied to our problem. However, this method is *data-independent* and hence cannot well handle the skewed distributions of records size and element frequency, which is common in real-life applications. Intuitively, the record with larger size and the element with high-frequency should be allocated more resources. In this paper, we theoretically show that the existing *KMV*-sketch technique cannot consider these two perspectives by simple heuristics, e.g., explicitly allocating more resource to record with large size. Consequently, we develop an augmented *KMV* sketch to exploit both record size distribution and the element frequency distribution for better space-accuracy and time-accuracy trade-offs. Two technique are proposed: (i) we impose a global threshold to *KMV* sketch, namely *G-KMV* sketch, to achieve better estimate accuracy. As discussed in Section IV-A(2), this technique cannot be extended to the Minhash *LSH*. (ii) we introduce an extra buffer for each record to take advantage of the skewness of the element frequency. A cost model is proposed to carefully choose the buffer size to optimize the accuracy for the given total space budget and data distribution.

Contributions. Our principle contributions are summarized as follows.

- We propose a new augmented *KMV* sketch technique, namely *GB-KMV*, for the problem of approximate containment similarity search. By imposing a global threshold and an extra buffer for *KMV* sketches of the records, we significantly enhance the performance as the new method can better exploit the data distributions.
- We provide theoretical underpinnings to justify the design of *GB-KMV* method. We also theoretically show that *GB-KMV* outperforms the state-of-the-art technique *LSH-E* in terms of accuracy under realistic assumption on data distributions.
- Our comprehensive experiments on real-life set-valued data from various applications demonstrate the effectiveness and efficiency of our proposed method.

Road Map. The rest of the paper is organized as follows. Section II presents the preliminaries. Section III introduces the

Notation	Definition
\mathcal{S}	a collection of records
X, Q	record, query record
x, q	record size of X , query size of Q
$J(Q, X), s$	Jaccard similarity between query Q and set X
$C(Q, X), t$	Containment similarity of query Q in set X
s^*	Jaccard similarity threshold
\mathcal{L}_X	the <i>KMV</i> signature (i.e., hash values) of record X
$h(X)$	all hash values of the elements in record X
\mathcal{H}_X	the buffer of record X
t^*	containment similarity threshold
b	sketch space budget, measured by the number of signatures (i.e., hash values or elements)
τ	the global threshold for hash values
r	the buffer size(with <i>bit</i> unit) of <i>GB-KMV</i> sketch
m	number of records in dataset \mathcal{S}
n	number of distinct elements in dataset \mathcal{S}

TABLE I. THE SUMMARY OF NOTATIONS

existing solutions. Our approach, *GB-KMV* sketch, is devised in Section IV. Extensive experiments are reported in Section V, followed by the related work in Section VI. Section VII concludes the paper.

II. PRELIMINARIES

In this section, we first formally present the problem of containment similarity search, then introduce some preliminary knowledge. In Table I, we summarize the important mathematical notations appearing throughout this paper.

A. Problem Definition

In this paper, the element universe is $\mathcal{E} = \{e_1, e_2, \dots, e_n\}$. Let \mathcal{S} be a collection of records (sets) $\{X_1, X_2, \dots, X_m\}$, where X_i ($1 \leq i \leq m$) is a set of elements from \mathcal{E} .

Before giving the definition of containment similarity, we first introduce the Jaccard similarity.

Definition 1 (Jaccard Similarity). Given two records X and Y from \mathcal{S} , the Jaccard similarity between X and Y is defined as the size of the intersection divided by the size of the union, which is expressed as

$$J(X, Y) = \frac{|X \cap Y|}{|X \cup Y|} \quad (1)$$

Similar to the Jaccard similarity, the containment similarity (a.k.a Jaccard containment similarity) is defined as follows.

Definition 2 (Containment Similarity). Given two records X and Y from \mathcal{S} , the containment similarity of X in Y , denoted by $C(X, Y)$ is the size of the intersection divided by record size $|X|$, which is formally defined as

$$C(X, Y) = \frac{|X \cap Y|}{|X|} \quad (2)$$

Note that by replacing the union size $|X \cup Y|$ in Equation 1 with size $|X|$, we get the containment similarity. It is easy to see that Jaccard similarity is symmetric while containment similarity is asymmetric.

In this paper, we focus on the problem of containment similarity search which is to look up a set of records whose containment similarity towards a given query record is not smaller than a given threshold. The formal definition is as follows.

Definition 3 (Containment Similarity Search). Given a query Q , and a threshold $t^* \in [0, 1]$ on the containment

id	record	$C(Q, X_i)$
X_1	$\{e_1, e_2, e_3, e_4, e_7\}$	0.67
X_2	$\{e_2, e_3, e_5\}$	0.5
X_3	$\{e_2, e_4, e_5\}$	0.33
X_4	$\{e_1, e_2, e_6, e_{10}\}$	0.33
Q	$\{e_1, e_2, e_3, e_5, e_7, e_9\}$	

Fig. 1. A four-record dataset and a query Q ; $C(Q, X_i)$ is the containment similarity of Q in X_i

similarity, search for records $\{X : X \in \mathcal{S}\}$ from a dataset \mathcal{S} such that:

$$C(Q, X) \geq t^* \quad (3)$$

Next, we give an example to show the problem of containment similarity search.

Example 1. Fig. 1 shows a dataset with four records $\{X_1, X_2, X_3, X_4\}$, and the element universe is $\mathcal{E} = \{e_1, e_2, \dots, e_{10}\}$. Given a query $Q = \{e_1, e_2, e_3, e_5, e_7, e_9\}$ and a containment similarity threshold $t^* = 0.5$, the records satisfying $C(Q, X_i) \geq 0.5$ are X_1, X_2 .

Problem Statement. In this paper, we investigate the problem of approximate containment similarity search. For the dataset \mathcal{S} with a large number of records, we aim to build a synopsis of the dataset such that it (i) can efficiently support containment similarity search with high accuracy, (ii) can handle large size records, and (iii) has a compact index size.

B. Minwise Hashing

Minwise Hashing is proposed by Broder in [13], [14] for estimating the Jaccard similarity of two records X and Y . Let h be a hash function that maps the elements of X and Y to distinct integers, and define $h_{min}(X)$ and $h_{min}(Y)$ to be the minimum hash value of a record X and Y , respectively. Assuming no hash collision, Broder[13] showed that the Jaccard similarity of X and Y is the probability of two minimum hash values being equal: $Pr[h_{min}(X) = h_{min}(Y)] = J(X, Y)$. Applying such k different independent hash functions h_1, h_2, \dots, h_k to a record X (Y , resp.), the MinHash signature of X (Y , resp.) is to keep k values of $h_{min}^i(X)$ ($h_{min}^i(Y)$, resp.) for k functions. Let $\mathbf{n}_i, i = 1, 2, \dots, k$ be the indicator function such that

$$\mathbf{n}_i := \begin{cases} 1 & \text{if } h_{min}^i(X) = h_{min}^i(Y), \\ 0 & \text{otherwise.} \end{cases} \quad (4)$$

then the Jaccard similarity between record X and Y can be estimated as

$$\hat{s} = \hat{J}(X, Y) = \frac{1}{k} \sum_{i=1}^k \mathbf{n}_i \quad (5)$$

Let $s = J(X, Y)$ be the Jaccard similarity of set X and Y , then the expectation of \hat{J} is

$$E(\hat{s}) = s \quad (6)$$

and the variance of \hat{s} is

$$Var(\hat{s}) = \frac{s(1-s)}{k} \quad (7)$$

C. KMV Sketch

The k minimum values(KMV) technique introduced by Bayer et. al 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 synopsis of a record 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 k -th smallest hash value. By $h(X)$, we denote hash values of all elements in the record X .

In [11], Bayer et. al also methodically analyse the problem of distinct element estimation under multi-set operation. 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 [11], $\mathcal{L}_X \oplus \mathcal{L}_Y$ represents the set consisting of the k smallest hash values in $\mathcal{L}_X \cup \mathcal{L}_Y$ where

$$k = \min(k_X, k_Y) \quad (8)$$

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 $D_U = |X \cup Y|$ is as follows.

$$\hat{D}_U = \frac{k-1}{U_{(k)}} \quad (9)$$

For intersection operation, the KMV synopsis is $\mathcal{L} = \mathcal{L}_X \oplus \mathcal{L}_Y$ where $k = \min(k_X, k_Y)$. Let $K_\cap = |\{v \in \mathcal{L} : v \in \mathcal{L}_X \cap \mathcal{L}_Y\}|$, i.e., K_\cap is the number of common distinct hash values of \mathcal{L}_X and \mathcal{L}_Y within \mathcal{L} . Then the number of distinct elements in $X \cap Y$, denoted by D_\cap , can be estimated as follows.

$$\hat{D}_\cap = \frac{K_\cap}{k} \times \frac{k-1}{U_{(k)}} \quad (10)$$

The variance of \hat{D}_\cap , as shown in[11], is

$$Var[\hat{D}_\cap] = \frac{D_\cap(kD_U - k^2 - D_U + k + D_\cap)}{k(k-2)} \quad (11)$$

III. EXISTING SOLUTIONS

In this section, we present the state-of-the-art technique for the approximate containment similarity search, followed by theoretical analysis on the limits of the existing solution.

A. LSH Ensemble Method

LSH Ensemble technique, $LSH-E$ for short, is proposed by Zhu et. al in [44] to tackle the problem of approximate containment similarity search. The key idea is : (1) transform the containment similarity search to the well-studied Jaccard similarity search; and (2) partition the data by length and then apply the LSH forest [9] technique for each individual partition.

Similarity Transformation. Given a record X with size $x = |X|$, a query Q with size $q = |Q|$, containment similarity $t = C(Q, X)$ and Jaccard similarity $s = J(Q, X)$. The transformation back and forth are as follows.

$$s = \frac{t}{\frac{x}{q} + 1 - t}, \quad t = \frac{(\frac{x}{q} + 1)s}{1 + s} \quad (12)$$

Given the containment similarity search threshold as t^* for the query q , we may come up with its corresponding Jaccard

similarity threshold s^* by Equation 12. A straightforward solution is to apply the existing approximate Jaccard similarity search technique for each individual record $X \in \mathcal{D}$ with the Jaccard similarity threshold s^* (e.g., compute Jaccard similarity between the query Q and a set X based on their MinHash signatures). In order to take advantages of the efficient indexing techniques (e.g., *LSH* forest [9]), *LSH-E* will partition the dataset \mathcal{S} .

Data Partition. By partitioning the dataset \mathcal{S} according to the record size, *LSH-E* can replace x in Equation 12 with its upper bound u (i.e., the largest record size in the partition) as an approximation. That is, for the given containment similarity t^* we have

$$s^* = \frac{t^*}{\frac{u}{q} + 1 - t^*} \quad (13)$$

The use of upper bound u will lead to false positives. In [44], an optimal partition method is designed to minimize the total number of false positives brought by the use of upper bound in each partition. By assuming that the record size distribution follows the power-law distribution and similarity values are uniformly distributed, it is shown that the optimal partition can be achieved by ensuring each partition has the equal number of records (i.e., equal-depth partition).

Containment Similarity Search. For each partition \mathcal{S}_i of the data, *LSH-E* applies the dynamic *LSH* technique (e.g., *LSH* forest [9]). Particularly, the records in \mathcal{S}_i are indexed by a MinHash *LSH* with parameter (b, r) where b is the number of bands used by the *LSH* index and r is the number of hash values in each band. For the given query Q , the b and r values are carefully chosen by considering their corresponding number of false positives and false negatives regarding the existing records. Then the candidate records in each partition can be retrieved from the MinHash index according to the corresponding Jaccard similarity thresholds obtained by Equation 13. The union of the candidate records from all partitions will be returned as the result of the containment similarity search.

B. Analysis

One of the *LSH-E*'s advantages is that it converts the containment similarity problem to Jaccard similarity search problem which can be solved by the mature and efficient MinHash *LSH* method. Also, *LSH-E* carefully considers the record size distribution and partitions the records by record size. In this sense, we say *LSH-E* is a data-dependent method and it is reported that *LSH-E* significantly outperforms existing asymmetric *LSH* based solutions [34], [35] (i.e., *data-independent* methods) as *LSH-E* can exploit the information of data distribution by partitioning the dataset. However, this benefit is offset by the fact that the the upper bound will bring extra false positives, in addition to the error from the MinHash technique.

Below we theoretically analyse the performance of *LSH-E* by studying the expectation and variance of its estimator.

Using the notations same as above, let $s = J(Q, X)$ be the Jaccard similarity between query Q and set X and $t = C(Q, X)$ be the containment similarity of Q in X . By Equation 5, given the MinHash signature of query Q and X respectively, an unbiased estimator \hat{s} of Jaccard similarity $s = J(Q, X)$ is the ratio of collisions in the signature, and the

variance of \hat{s} is $Var[\hat{s}] = \frac{s(1-s)}{k}$ where k is signature size of each record. Then by transformation Equation 12, the estimator \hat{t} of containment similarity $t = C(Q, X)$ by MinHash *LSH* is

$$\hat{t} = \frac{(\frac{x}{q} + 1)\hat{s}}{1 + \hat{s}} \quad (14)$$

where $q = |Q|$ and $x = |X|$. The estimator \hat{t}' of containment similarity $t = C(Q, X)$ by *LSH-E* is

$$\hat{t}' = \frac{(\frac{u}{q} + 1)\hat{s}}{1 + \hat{s}} \quad (15)$$

where $q = |Q|$ and u is the upper bound of $|X|$.

Next, we use Taylor expansions to approximate the expectation and variance of a function with one random variable [26]. We first give a lemma.

Lemma 1. *Given a random variable X with expectation $E[X]$ and variance $Var[X]$, the expectation of $f(X)$ can be approximated as*

$$E[f(X)] \approx f(E[X]) + \frac{f''(E[X])}{2} Var[X] \quad (16)$$

and the variance of $f(X)$ can be approximated as

$$Var[f(X)] \approx [f'(E[X])]^2 Var[X] - \frac{[f''(E[X])]^2}{4} Var^2[X] \quad (17)$$

According to Equation 14, let $\hat{t} = f(\hat{s}) = \alpha \frac{\hat{s}}{1 + \hat{s}}$ where $\alpha = \frac{x}{q} + 1$. We can see that the estimator \hat{t} is a function of \hat{s} , and $f'(\hat{s}) = \alpha \frac{1}{(1 + \hat{s})^2}$ and $f''(\hat{s}) = -2\alpha \frac{1}{(1 + \hat{s})^3}$. Then based on Lemma 1, the expectation and variance of \hat{t} are approximated as

$$E[\hat{t}] \approx t(1 - \frac{1-s}{k(1+s)^2}) \quad (18)$$

$$Var[\hat{t}] \approx \frac{D_{\hat{s}}^2(1-s)[k(1+s)^2 - s(1-s)]}{q^2 k^2 s(1+s)^4} \quad (19)$$

Similarly, the expectation and variance of *LSH-E* estimator \hat{t}' can be approximated as

$$E[\hat{t}'] \approx t(\frac{u+q}{x+q})(1 - \frac{1-s}{k(1+s)^2}) \quad (20)$$

$$Var[\hat{t}'] \approx (\frac{u+q}{x+q})^2 \frac{D_{\hat{s}}^2(1-s)[k(1+s)^2 - s(1-s)]}{q^2 k^2 s(1+s)^4} \quad (21)$$

The computation details are in technique report [41]. Since u is the upper bound of x , the variance of *LSH-E* estimator $Var[\hat{t}']$ is larger than that of MinHash *LSH* estimator. Also, by Equation 18 and Equation 20, we can see that both estimators are *biased* and *LSH-E* method is quite sensitive to the setting of the upper bound u by Equation 20. Because the presence of upper bound u will *enlarge* the estimator off true value, *LSH-E* method favours recall while the precision will be deteriorated. The larger the upper bound u is, the worse the precision will be. Our empirical study shows that *LSH-E* cannot achieve a good trade-off between accuracy and space, compared with our proposed method.

\mathcal{L}_{KMV}	k_i
\mathcal{L}_{X_1} $\{(e_2, 0.24), (e_7, 0.33), (e_4, 0.47)\}$	3
\mathcal{L}_{X_2} $\{(e_5, 0.10), (e_2, 0.24), (e_3, 0.85)\}$	3
\mathcal{L}_{X_3} $\{(e_5, 0.10), (e_2, 0.24)\}$	2
\mathcal{L}_{X_4} $\{(e_{10}, 0.18), (e_2, 0.24)\}$	2
\mathcal{L}_Q $\{(e_5, 0.10), (e_2, 0.24), (e_7, 0.33), (e_9, 0.56)\}$	4

Fig. 2. The *KMV* sketch of the dataset in Example 1, each signature consists of element-hash value pairs. k_i is the signature size of X_i

IV. OUR APPROACH

In this section, we introduce an augmented *KMV* sketch technique to achieve better space-accuracy trade-off for approximate containment similarity search. Section IV-A briefly introduces the motivation and main technique of our method, namely *GB-KMV*. The detailed implementation is presented in Section IV-B, followed by extensive theoretical analysis in Section IV-C.

A. Motivation and Techniques

The key idea of our method is to propose a *data-dependent* indexing technique such that we can exploit the distribution of the data (i.e., record size distribution and element frequency distribution) for better performance of containment similarity search. We augment the existing *KMV* technique by introducing a global threshold for sample size allocation and a buffer for frequent elements, namely *GB-KMV*, to achieve better trade-off between synopsis size and accuracy. Then we apply the existing set similarity join/search indexing technique to speed up the containment similarity search.

Below we outline the motivation of the key techniques used in this paper. Detailed algorithms and theoretical analysis will be introduced in Section IV-B and IV-C, respectively.

(1) Directly Apply *KMV* Sketch

Given a query Q and a threshold t^* on containment similarity, the goal is to find record X from dataset \mathcal{S} such that

$$\frac{|Q \cap X|}{|Q|} \geq t^*, \quad (22)$$

Applying some simple transformation to Equation 22, we get

$$|Q \cap X| \geq t^*|Q|, \quad (23)$$

Let $\theta = t^*|Q|$, then the containment similarity search problem is converted into finding record X whose intersection size with the query Q is not smaller than θ , i.e., $|Q \cap X| \geq \theta$.

Therefore, we can directly apply the *KMV* method introduced in Section II-C. Given *KMV* signatures of a record X and a query Q , we can estimate their intersection size ($|Q \cap X|$) according to Equation 10. Then the containment similarity of Q in X is immediately available given the query size $|Q|$. Below, we show an example on how to apply *KMV* method to containment similarity search.

Example 2. Fig. 2 shows the *KMV* sketch on dataset in Example 1. Given *KMV* signature of Q ($\mathcal{L}_Q = \{(e_5, 0.10), (e_2, 0.24), (e_7, 0.33), (e_9, 0.56)\}$) and X_1 ($\mathcal{L}_{X_1} = \{(e_2, 0.24), (e_7, 0.33), (e_4, 0.47)\}$), we have $k = \min\{k_Q, k_1\} = 3$, then the size- k *KMV* synopses of $Q \cup X_1$ is $\mathcal{L} = \mathcal{L}_Q \oplus \mathcal{L}_{X_1} = \{(e_5, 0.10), (e_2, 0.24), (e_7, 0.33)\}$, the k -th smallest hash value $U_{(k)}$ is 0.33 and the size of intersection of \mathcal{L}_Q and \mathcal{L}_{X_1} within \mathcal{L} is

$K_\cap = |\{v : v \in \mathcal{L}_Q \cap \mathcal{L}_{X_1}, v \in \mathcal{L}\}| = 2$. Then the intersection size of Q and X_1 is estimated as $\hat{D}_\cap = \frac{K_\cap}{k} \times \frac{k-1}{U_{(k)}} = \frac{2}{3} * \frac{2}{0.33} = 4.04$, and the containment similarity is $\hat{t} = \frac{\hat{D}_\cap}{|Q|} = 0.67$. Then X_1 is returned if the given containment similarity threshold t^* is 0.5.

Remark 1. In [44], the size of the query is approximated by *MinHash* signature of Q , where *KMV* sketch can also serve for the same purpose. But the exact query size is used their implementation for performance evaluation. In practice, the query size is readily available, we assume query size is given throughout the paper.

Optimization of *KMV* Sketch. Given a space budget b , we can keep size- k_i *KMV* signatures (i.e., k_i minimal hash values) for each record X_i with $\sum_{i=1}^n k_i = b$. A natural question is how to allocate the resource (e.g., setting of k_i values) to achieve the best overall estimation accuracy. Intuitively, more resources should be allocated to records with more frequent elements or larger record size, i.e., larger k_i for record with larger size. However, **Theorem 1** (Section IV-C2) suggests that, the optimal resource allocation strategy in terms of estimation variance is to use the same size of signature for each record. This is because the **minimal** of two k -values is used in Equation 8, and hence the best solution is to **evenly** allocate the resource. Thus, we have the *KMV* sketch based method for approximate containment similarity search. For the given budget b , we keep $k_i = \lfloor \frac{b}{m} \rfloor$ minimal hash values for each record X_i .

(2) Impose a Global Threshold to *KMV* Sketch (*G-KMV*)

The above analysis on optimal *KMV* sketch suggests an equal size allocation strategy, that is, each record is associated with the same size signature. Intuitively we should assign more resources (i.e., signature size) to the records with large size because they are more likely to appear in the results. However, the estimate accuracy of *KMV* for two sets size intersection is determined by the sketch with smaller size since we choose $k = \min(k_1, k_2)$ for *KMV* signatures of X_1 and X_2 for D_\cup and D_\cap in Equation 9, thus it is useless to give more resource to one of the records. We further explain the reason behind with the following example.

Before we introduce the global threshold to *KMV* sketch, consider the *KMV* sketch shown in the Fig. 2.

Example 3. Suppose we have $\mathcal{L}_Q = \{(e_5, 0.10), (e_2, 0.24), (e_7, 0.33), (e_9, 0.56)\}$ and $\mathcal{L}_{X_3} = \{(e_5, 0.10), (e_2, 0.24)\}$. Although there are four hash values in $\mathcal{L}_Q \cup \mathcal{L}_{X_3} = \{(e_5, 0.10), (e_2, 0.24), (e_7, 0.33), (e_9, 0.56)\}$, we can only consider $k = \min\{k_Q, k_{X_3}\} = 2$ smallest hash values of $\mathcal{L}_Q \cup \mathcal{L}_{X_3}$ by Equation 8, which is $\{(e_5, 0.10), (e_2, 0.24)\}$, and the k -th ($k = 2$) minimum hash value used in Equation 9 is 0.24. We cannot use $k = 4$ (i.e., $U_{(k)} = 0.56$) to estimate $|Q \cup X_3|$ because the 4-th smallest hash value in $\mathcal{L}_Q \cup \mathcal{L}_{X_3}$ **may not be** the 4-th smallest hash values in $h(Q \cup X_3)$, because the unseen 3-rd smallest hash value of X_3 might be $(e_4, 0.47)$ for example, which is smaller than 0.56. Recall that $h(Q \cup X_3)$ denote the hash values of all elements in $Q \cup X_3$.

Nevertheless, if we know that all the hash values smaller than a global threshold, say 0.6, are kept for every record, we can safely use the 4-th hash value of $\mathcal{L}_Q \cup \mathcal{L}_{X_3}$ (i.e., 0.56) for the estimation. This is because we can ensure the 4-th smallest

\mathcal{L}_{GKMV}	
\mathcal{L}_{X_1}	$\{(e_2, 0.24), (e_7, 0.33), (e_4, 0.47)\}$
\mathcal{L}_{X_2}	$\{(e_5, 0.10), (e_2, 0.24)\}$
\mathcal{L}_{X_3}	$\{(e_5, 0.10), (e_2, 0.24), (e_4, 0.47)\}$
\mathcal{L}_{X_4}	$\{(e_{10}, 0.18), (e_2, 0.24)\}$
\mathcal{L}_Q	$\{(e_5, 0.10), (e_2, 0.24), (e_7, 0.33)\}$

Fig. 3. The G - KMV sketch of the dataset in Example 1 with hash value threshold $\tau = 0.5$
hash value in $\mathcal{L}_Q \cup \mathcal{L}_{X_3}$ must be the 4-th smallest hash values in $h(Q \cup X_3)$.

Inspired by the above observation, we can carefully choose a global threshold τ (e.g., 0.6 in the above example) for a given space budget b , and ensure all hash values smaller than τ will be kept for KMV sketch of the records. By imposing a global threshold, we can identify a better (i.e., larger) k value used for estimation, compared with Equation 8.

Given a record X and a global threshold τ , the sketch of a record X is obtained as $\mathcal{L}_X = \{h(e) : h(e) \leq \tau, e \in X\}$ where h is the hash function. The sketch of Q (\mathcal{L}_Q) is defined in the same way. In this paper, we say a KMV sketch is a G - KMV sketch if we impose a global threshold to generate KMV sketch. Then we set k value of the KMV estimation as follows.

$$k = |\mathcal{L}_Q \cup \mathcal{L}_X| \quad (24)$$

Meanwhile, we have $K_\cap = |\mathcal{L}_Q \cap \mathcal{L}_X|$. Let $U_{(k)}$ be the k -th minimal hash value in $\mathcal{L}_Q \cup \mathcal{L}_X$, then the overlap size of Q and X can be estimated as

$$\hat{D}_\cap^{GKMV} = \frac{K_\cap k - 1}{k U_{(k)}} \quad (25)$$

Then the containment similarity of Q in X is

$$\hat{C} = \frac{\hat{D}_\cap^{GKMV}}{q} \quad (26)$$

where q is the query size. We remark that, as a by-product, the global threshold favours the record with large size because all elements with hash value smaller than τ are kept for each record.

Below is an example on how to compute the containment similarity based on G - KMV sketch.

Example 4. Fig. 3 shows the KMV sketch of dataset in Example 1 with a global threshold $\tau = 0.5$. Given the signature of Q ($\mathcal{L}_Q = \{(e_5, 0.10), (e_2, 0.24), (e_7, 0.33)\}$) and X_1 ($\mathcal{L}_{X_1} = \{(e_2, 0.24), (e_7, 0.33), (e_4, 0.47)\}$), the KMV sketch of $Q \cup X_1$ is $\mathcal{L} = \mathcal{L}_Q \cup \mathcal{L}_{X_1} = \{(e_5, 0.10), (e_2, 0.24), (e_7, 0.33), (e_4, 0.47)\}$, the k -th ($k = 4$) smallest hash value is $U_{(k)} = 0.47$, and the size of intersection of \mathcal{L}_Q and \mathcal{L}_{X_1} within \mathcal{L} is $K_\cap = |\{v : v \in \mathcal{L}_Q \cap \mathcal{L}_{X_1}, v \in \mathcal{L}\}| = 2$. Then the intersection size of Q and X_1 is estimated as $\hat{D}_\cap = \frac{K_\cap}{k} \times \frac{k-1}{U_{(k)}} = \frac{2}{4} * \frac{3}{0.47} = 3.19$, and the containment similarity is $\hat{t} = \frac{\hat{D}_\cap}{|Q|} = 0.53$. Then X_1 is returned if the given containment similarity threshold t^* is 0.5.

Correctness of G - KMV sketch. Theorem 2 in Section IV-C3 shows the correctness of the G - KMV sketch.

Comparison with KMV . In Theorem 3 (Section IV-C4), we theoretically show that G - KMV can achieve better accuracy compared with KMV .

	\mathcal{L}_H	\mathcal{L}_{GKMV}
X_1	$\{e_1, e_2\}$	$\{(e_7, 0.33), (e_4, 0.47)\}$
X_2	$\{e_2\}$	$\{(e_5, 0.10)\}$
X_3	$\{e_2\}$	$\{(e_5, 0.10)\}$
X_4	$\{e_1, e_2\}$	$\{e_{10}, 0.18\}$
Q	$\{e_1, e_2\}$	$\{(e_5, 0.10), (e_7, 0.33)\}$

Fig. 4. The GB - KMV sketch of dataset in Example 1

Remark 2. Note that the global threshold technique cannot be applied to MinHash based techniques. In minHash LSH, the k minimum hash values are corresponding to k different independent hash functions, while in KMV sketch, the k -value sketch is obtained under one hash function. Thus we can only impose this global threshold on the same hash function for the KMV sketch based method.

(3) Use Buffer for KMV Sketch (GB - KMV)

In addition to the skewness of the record size, it is also worthwhile to exploit the skewness of the element frequency. Intuitively, more resource should be assigned to high-frequency elements because they are more likely to appear in the records. However, due to the nature of the hash function used by KMV sketch, the hash value of an element is *independent* to its frequency; that is, all elements have the same opportunity contributing to the KMV sketch.

One possible solution is to divide the elements into multiple disjoint groups according to their frequency (e.g., low-frequency and high-frequency ones), and then apply KMV sketch for each individual group. The intersection size between two records Q and X can be computed within each group and then sum up together. However, our initial experiments suggest that this will lead to poor accuracy because of the summation of the intersection size estimations. In Theorem 4 (Section IV-C5), our theoretical analysis suggests that the combination of estimated results are very likely to make the overall accuracy worse.

To avoid combining multiple estimation results, we use a bitmap buffer with size r for each record to *exactly* keep track of the r most frequent elements, denoted by \mathcal{E}_H . Then we apply G - KMV technique to the remaining elements, resulting in a new augmented sketch, namely GB - KMV . Now we can estimate $|Q \cap X|$ by combining the intersection of their bitmap buffers (*exact solution*) and KMV sketches (*estimated solution*).

As shown in Fig. 4, suppose we have $\mathcal{E}_H = \{e_1, e_2\}$ and the global threshold for hash value is $\tau = 0.5$, then the sketch of each record consists of two parts \mathcal{L}_H and \mathcal{L}_{GKMV} ; that is, for each record we use bitmap to keep the elements corresponding to high-frequency elements $\mathcal{E}_H = \{e_1, e_2\}$, then we store the left elements with hash value less than $\tau = 0.5$.

Example 5. Given the signature of Q ($\mathcal{L}_Q = \{e_1, e_2\} \cup \{(e_5, 0.10), (e_7, 0.33)\}$) and X_1 ($\mathcal{L}_{X_1} = \{e_1, e_2\} \cup \{(e_7, 0.33), (e_4, 0.47)\}$), the intersection of High-frequency part is $\mathcal{L}_Q^H \cap \mathcal{L}_{X_1}^H = \{e_1, e_2\}$ with intersection size as 2; next we consider the G - KMV part. Similar to Example 4, we compute the intersection of \mathcal{L}_{GKMV} part. The KMV sketch is $\mathcal{L}' = \mathcal{L}'_Q \cup \mathcal{L}'_{X_1} = \{(e_5, 0.10), (e_7, 0.33), (e_4, 0.47)\}$. According to Equation 24, the k -th ($k = 3$) smallest hash value is $U_{(k)} = 0.47$, and the size of intersection of \mathcal{L}_Q and \mathcal{L}_{X_3} within \mathcal{L} is $K_\cap = |\{v : v \in \mathcal{L}_Q \cap \mathcal{L}_{X_3}, v \in \mathcal{L}\}| = 1$. Then the intersection size of Q and X_1 in \mathcal{L}_{GKMV} part is

estimated as $\hat{D}_\cap = \frac{K_\cap}{k} \times \frac{k-1}{U_{(k)}} = \frac{1}{3} * \frac{2}{0.47} = 1.4$; together with the High-frequency part, the intersection size of Q and X_1 is estimated as $2 + 1.4 = 3.4$ and the containment similarity is $\hat{t} = \frac{\hat{D}_\cap}{|Q|} = 0.53$. Then X_1 is returned if the given containment similarity threshold t^* is 0.5.

Optimal Buffer Size. The key challenge is how to set the size of bitmap buffer for the best expected performance of *GB-KMV* sketch. In Section IV-C6, we provide a theoretical analysis, which is verified in our performance evaluation.

Comparison with *G-KMV*. As the *G-KMV* is a special case of *GB-KMV* with buffer size 0 and we carefully choose the buffer size with our cost model, the accuracy of *GB-KMV* is not worse than *G-KMV*.

Comparison with *LSH-E*. Through theoretical analysis, we show that the performance (i.e., the variance of the estimator) of *GB-KMV* can always outperform that of *LSH-E* in **Theorem 5** (Section IV-C7).

B. Implementation of *GB-KMV*

In this section, we introduce the technique details of our proposed *GB-KMV* method. We first show how to build *GB-KMV* sketch on the dataset \mathcal{S} and then present the containment similarity search algorithm.

GB-KMV Sketch Construction. For each record $X \in \mathcal{S}$, its *GB-KMV* sketch consists of two components: (1) a buffer which exactly keeps high-frequency elements, denoted by \mathcal{H}_X ; and (2) a *G-KMV* sketch, which is a *KMV* sketch with a global threshold value, denoted by \mathcal{L}_X .

Algorithm 1: *GB-KMV* Index Construction

Input : \mathcal{S} : dataset; b : space budget;
 h : a hash function; r : buffer size
Output : $\mathcal{L}_\mathcal{S}$, the *GB-KMV* index of dataset \mathcal{S}

- 1 Compute buffer size r based on distribution statistics of \mathcal{S} and the space budget b ;
- 2 $\mathcal{E}_H \leftarrow$ Top r most frequent elements; $\mathcal{E}_K \leftarrow \mathcal{E} \setminus \mathcal{E}_H$;
- 3 $\tau \leftarrow$ compute the global threshold for hash values;
- 4 **for each** record $X \in \mathcal{S}$ **do**
- 5 $\mathcal{H}_X \leftarrow$ elements of X in \mathcal{E}_H ;
- 6 $\mathcal{L}_X \leftarrow$ hash values of elements $\{e\}$ of X with $h(e) \leq \tau$;

Algorithm 1 illustrates the construction of *GB-KMV* sketch. Let the element universe be $\mathcal{E} = \{e_1, e_2, \dots, e_n\}$ and each element is associated with its frequency in dataset \mathcal{S} . Line 1 calculates a buffer size r for all records based on the skewness of record size and elements as well as the space budget b in terms of elements. Details will be introduced in Section IV-C6. We use \mathcal{E}_H to denote the set of top- r most frequent elements (Line 1), and they will be kept in the buffer of each record. Let \mathcal{E}_K denote the remaining elements. Line 1 identifies maximal possible global threshold τ for elements in \mathcal{E}_K such that the total size of *GB-KMV* sketch meets the space budget b . For each record X , let n_X denote the number of elements in \mathcal{E}_K with hash values less than τ , we have $\sum_{X \in \mathcal{S}} (\frac{b}{32} + n_X) \leq b$. Then Lines 1-1 build the buffer \mathcal{H}_X and *G-KMV* sketch \mathcal{L}_X for every record $X \in \mathcal{S}$. In section 2, we will show the correctness of our sketch in Theorem 2.

Containment Similarity Search. Given the *GB-KMV* sketch of the query record Q and the dataset \mathcal{S} , we can conduct

approximate similarity search as illustrated in Algorithm 2. Given a query Q with size q and the similarity threshold t^* , let $\theta = t^* * q$ (Lines 1-2). With *GB-KMV* sketch $\{\mathcal{H}_Q, \mathcal{L}_Q\}$, we can calculate the containment similarity based on

$$|\widehat{Q \cap X}| = |\mathcal{H}_Q \cap \mathcal{H}_X| + \hat{D}_\cap^{GKMV} \quad (27)$$

where \hat{D}_\cap^{GKMV} is the estimation of overlap size of Q and X which is calculated by Equation 25 in Section IV-A.

Note that $|\mathcal{H}_Q \cap \mathcal{H}_X|$ is the number of common elements of Q and X in \mathcal{E}_H .

Algorithm 2: Containment Similarity Search

Input : Q , a query set
 t^* , containment similarity threshold
Output : R : records $\{X\}$ with $C(Q, X) \geq t^*$

- 1 $q \leftarrow |Q|$;
- 2 $\theta \leftarrow t^* * q$;
- 3 **for each** record $X \in \mathcal{S}$ **do**
- 4 $|\widehat{Q \cap X}| \leftarrow |\mathcal{H}_Q \cap \mathcal{H}_X| + \hat{D}_\cap^{GKMV}$;
- 5 **if** $|\widehat{Q \cap X}| \geq \theta$ **then**
- 6 $\mathcal{S}_{candidate} = \mathcal{S}_{candidate} \cup X$;
- 7 **return** $\mathcal{S}_{candidate}$

Implementation of Containment Similarity Search. In our implementation, we use a bitmap with size r to keep the elements in buffer where each bit is reserved for one frequent element. We can use *bitwise intersection* operator to efficiently compute $|\mathcal{H}_Q \cap \mathcal{H}_X|$ in Line 2 of Algorithm 2. Note that the estimator of overlap size by *G-KMV* method in Equation 25 is $\hat{D}_\cap^{GKMV} = \frac{K_\cap}{k} \frac{k-1}{U_{(k)}}$. As to the computation of $|\widehat{Q \cap X}|$, we

apply some transformation to $|\mathcal{L}_H^Q \cap \mathcal{L}_H^X| + \hat{D}_\cap^{GKMV} \geq \theta$. Then we get $K_\cap \geq o$ where $o = U_{(k)}(\theta - o_1)$ and $o_1 = |\mathcal{H}_Q \cap \mathcal{H}_X|$. Since K_\cap is the overlap size, then we make use of the PPjoin* [40] to speed up the search. Note that in order to make the PPjoin* which is designed for similarity join problem to be applicable to the similarity search problem, we partition the dataset \mathcal{S} by record size, and in each partition we search for the records which satisfy $K_\cap \geq o$, where overlap size is modified by the lower bound in corresponding partition.

Remark 3. Note that the size-aware overlap set similarity joins algorithm in [25] can not be applied to our *GB-KMV* method, because we need to online construct c -subset inverted list for each incoming query, which results in very inefficient performance.

Processing Dynamic Data. Note that our algorithm can be modified to process dynamic data. Particularly, when new records come, we compute the new global threshold τ under the fixed space budget by Line 1 of Algorithm 1, and with the new global threshold, we maintain the sketch of each record as shown in Line 1 of Algorithm 1.

C. Theoretical Analysis

In this section, we provide theoretical underpinnings of the claims and observations in this paper.

1) **Background:** We need some reasonable assumptions on the record size distribution, element frequency distribution and query work-load for a comprehensive analysis. Following are three popular assumptions widely used in the literature (e.g., [6], [29], [27], [18], [16], [44], [34]):

- The element frequency in the dataset follows the power-law distribution, with $p_1(x) = c_1 x^{-\alpha_1}$.
- The record size in the dataset follows the power-law distribution, with $p_2(x) = c_2 x^{-\alpha_2}$.
- The query Q is randomly chosen from the records.

Throughout the paper, we use the variance to evaluate the goodness of an estimator. Regarding the *KMV* based sketch techniques (*KMV*, *G-KMV* and *GB-KMV*), we have

Lemma 2. *In *KMV* sketch based methods, the larger the k value used in Equation 8 and Equation 24 is, the smaller the variance will be.*

It is easy to verify the above lemma by calculating the derivative of Equation 11 with respect to the variable k . Thus, in the following analysis of *KMV* based sketch techniques. We use the k value (i.e., the sketch size used for estimation) to evaluate the goodness of the estimation, the larger the better.

2) Optimal *KMV* Signature Scheme: In this part, we give an optimal resource allocation strategy for *KMV* sketch method in similarity search.

Theorem 1. *Given a space budget b , each set is associated with a size- k_i *KMV* signature and $\sum_{i=1}^m k_i = b$. For *KMV* sketch based containment similarity search, the optimal signature scheme is to keep the $\lfloor \frac{b}{m} \rfloor$ minimal hash values for each set X_i .*

Proof: Given a query Q and dataset $S = \{X_1, \dots, X_m\}$, an optimal signature scheme for containment similarity search is to minimize the average variance between Q and $X_i, i = 1, \dots, m$. Considering the query Q and set X_i with size- k_q *KMV* sketch \mathcal{L}_Q and size- k_i sketch \mathcal{L}_{X_i} , respectively, the sketch size is $k = \min\{k_q, k_i\}$ according to Equation 8. By Lemma 2, an optimal signature scheme is to maximize the total k value (say T), then we have the following optimization goal,

$$\begin{aligned} \max T &= \sum_{i=1}^m \min\{k_q, k_i\} \\ \text{s.t. } b &= \sum_{i=1}^m k_i, \quad k_i > 0, i = 1, 2, \dots, m \end{aligned}$$

Rank the k_i by increasing order, w.l.o.g., let k_1, k_2, \dots, k_m be the sketch size sequence after reorder. Let k_l be the first in the sequence such that $k_l = k_q$, then we have $T = k_1 + \dots + k_l + (m-l)k_q = b - \sum_{i=l+1}^m (k_i - k_q)$. In order to maximize T , we set $k_i = k_q, i = l+1, \dots, m$. Then by $b = \sum_{i=1}^m k_i$, we have $k_1 + \dots + k_l + k_q(m-l) = b$. Note that $k_i \leq k_q, i = 1, \dots, l$, we must have $k_i = k_q, i = 1, \dots, l$. Since Q is randomly selected from dataset S , we can get that all the $k_i, i = 1, \dots, m$ are equal and $k_i = \lfloor \frac{b}{m} \rfloor$. ■

3) Correctness of *GKMV* Sketch: In this section, we show that the *G-KMV* sketch is a valid *KMV* sketch.

Theorem 2. *Given two records X and Y , let \mathcal{L}_X and \mathcal{L}_Y be the *G-KMV* sketch of X and Y , respectively. Let $k = |\mathcal{L}_X \cup \mathcal{L}_Y|$, then the size- k *KMV* synopsis of $X \cup Y$ is $\mathcal{L} = \mathcal{L}_X \cup \mathcal{L}_Y$.*

Proof: We show that the above $\mathcal{L} = \mathcal{L}_X \cup \mathcal{L}_Y$ is a valid *KMV* sketch of $X \cup Y$. Let $k = |\mathcal{L}_X \cup \mathcal{L}_Y|$ and v_k is the k -th smallest hash value in $\mathcal{L}_X \cup \mathcal{L}_Y$. In order to prove that

$\mathcal{L}_X \cup \mathcal{L}_Y$ is valid, we show that v_k corresponds the element with the k -th minimal hash value in $X \cup Y$. If not, there should exist an element e such that $h(e') < v_k, e' \in X \cup Y$ and $h(e') \notin \mathcal{L}_X \cup \mathcal{L}_Y$. Note that $v_k \leq \tau$, then $h(e') \leq \tau$, thus $h(e')$ is included in $\mathcal{L}_X \cup \mathcal{L}_Y$, which contradicts to the above statement. ■

4) *G-KMV*: A Better *KMV* Sketch: In this part, we show that by imposing a global threshold to *KMV* sketch, we can achieve better accuracy. Let \mathcal{L}_X^{KMV} and \mathcal{L}_Y^{KMV} be the *KMV* sketch of X and Y respectively. Let $k_1 = |\mathcal{L}_X^{KMV}|$ and $k_2 = |\mathcal{L}_Y^{KMV}|$, then the sketch size k value can be set by Equation 8. Similarly, let \mathcal{L}_X^{GKMV} and \mathcal{L}_Y^{GKMV} be the *G-KMV* sketch of X and Y respectively, and the sketch size k value can be set by Equation 24.

Theorem 3. *With the fixed index space budget, for containment similarity search the *G-KMV* sketch method is better than *KMV* method in terms of accuracy when the power-law exponent of element frequency $\alpha_1 \leq 3.4$.*

Proof: Let $x_j = |X_j|, j = 1, 2, \dots, m$ be the set size and k_j be the signature size of record X_j . The frequency of element e_i is set to be f_i . The index space budget is b .

For *KMV* sketch based method, by Theorem 1, the optimal signature scheme is $k = \min(k_j, k_l) = \lfloor \frac{b}{m} \rfloor$ given the index space budget b , then the average k value for all pairs of sets is

$$\bar{k}_{KMV} = \frac{1}{m^2} \sum_{j=1}^m \sum_{l=1}^m \min(k_j, k_l) = \lfloor \frac{b}{m} \rfloor \quad (28)$$

For *G-KMV* sketch based method, let τ be the hash value threshold. The probability that hash value $h(e_i)$ is included in signature $\mathcal{L}_{X_j}^{GKMV}$ is $Pr[h(e_i) \in \mathcal{L}_{X_j}^{GKMV}] = \tau \frac{f_i}{N} x_j$ where f_i is the frequency of element e_i , and $N = \sum_{i=1}^n f_i$ is the total number of elements. The size of $\mathcal{L}_{X_j}^{GKMV}$ can be computed by $l_j = \sum_{i=1}^n Pr[h(e_i) \in \mathcal{L}_{X_j}^{GKMV}] = \tau x_j$ then the total index space is $b = \sum_{j=1}^m l_j = \sum_{j=1}^m \tau x_j = \tau N$. and the hash value threshold $\tau = \frac{b}{N}$. Next we compute average sketch size k value of *G-KMV* method. The intersection size of \mathcal{L}_{X_j} and \mathcal{L}_{X_l}

$$|\mathcal{L}_{X_j} \cap \mathcal{L}_{X_l}| = \sum_{i=1}^n \tau \frac{f_i}{N} x_j * \tau \frac{f_i}{N} x_l = \tau^2 x_j x_l f_{n^2} \quad (29)$$

where $f_{n^2} = \frac{\sum_{i=1}^n f_i^2}{N^2}$. The k value of *G-KMV* method according to Equation 24 is

$$|\mathcal{L}_{X_j} \cup \mathcal{L}_{X_l}| = \tau x_j + \tau x_l - \tau^2 x_j x_l f_{n^2} \quad (30)$$

Then the average k value for all pairs of sets is

$$\bar{k}_{GKMV} = \frac{1}{m^2} \sum_{j=1}^m \sum_{l=1}^m |\mathcal{L}_{X_j} \cup \mathcal{L}_{X_l}| = \frac{2b}{m} - \frac{b^2}{m^2} f_{n^2} \quad (31)$$

Let $\bar{k}_{GKMV} \geq \bar{k}_{KMV}$, we get $\alpha_1 \in (0, 0.5] \cup [(1 + \frac{m}{b}) - \sqrt{(1 + \frac{m}{b}) \frac{m}{b}}, (1 + \frac{m}{b}) + \sqrt{(1 + \frac{m}{b}) \frac{m}{b}}]$. Note that for the common setting $\frac{m}{b} \leq 1$, we can get $\alpha_1 \leq 3.4$. The result makes sense since the power-law (Zipf's law) exponent of element frequency is usually less than 3.4 for real datasets. ■

5) **Partition of KMV Sketch Is Not Promising:** In this part, we show that it is difficult to improve the performance of *KMV* by dividing elements to multiple groups according to their frequency and apply *KMV* estimation individually. W.l.o.g., we consider dividing elements into two groups.

We divide the sorted element universe \mathcal{E} into two disjoint parts \mathcal{E}_{H_1} and \mathcal{E}_{H_2} . Let X and Y be two sets from dataset \mathcal{S} with *KMV* sketch \mathcal{L}_X and \mathcal{L}_Y respectively. Let $k_X = |\mathcal{L}_X|$ and $k_Y = |\mathcal{L}_Y|$. The estimator of containment similarity is $\hat{C} = \frac{\hat{D}_\cap}{q}$, where \hat{D}_\cap is the estimator of intersection size D_\cap and q is the query size(x or y).

Corresponding to \mathcal{E}_{H_1} and \mathcal{E}_{H_2} , we divide X (Y , resp.) to two parts X_1 and X_2 (Y_1 and Y_2 , resp.). We know that $X_1 \cap X_2 = \Phi$ and $Y_1 \cap Y_2 = \Phi$. Also, let $D_\cap = |X \cap Y|$, $D_U = |X \cup Y|$, we have $D_\cap = |X_1 \cap Y_1| + |X_2 \cap Y_2|$ and $D_U = |X_1 \cup Y_1| + |X_2 \cup Y_2|$ since \mathcal{E}_{H_1} and \mathcal{E}_{H_2} are disjoint. For simplicity, let $D_{\cap 1} = |X_1 \cap Y_1|$, $D_{U1} = |X_1 \cup Y_1|$, $D_{\cap 2} = |X_2 \cap Y_2|$ and $D_{U2} = |X_2 \cup Y_2|$. For X_1, X_2, Y_1 and Y_2 , the *KMV* sketches are $\mathcal{L}_{X_1}, \mathcal{L}_{X_2}, \mathcal{L}_{Y_1}$ and \mathcal{L}_{Y_2} with size $k_{X_1}, k_{X_2}, k_{Y_1}$ and k_{Y_2} , respectively. Based on this, we give another estimator as $\hat{C}' = \frac{\hat{D}_{\cap 1} + \hat{D}_{\cap 2}}{q}$, where $\hat{D}_{\cap 1}$ ($\hat{D}_{\cap 2}$, resp.) is the estimator of intersection size $D_{\cap 1}$ ($D_{\cap 2}$, resp.). Next, we compare the variance of \hat{C} and \hat{C}' .

Theorem 4. *After dividing the element universe into two groups and applying KMV sketch in each group, with the same index space budget, the variance of \hat{C}' is larger than that of \hat{C} .*

Proof: Recall the *KMV* sketch, we have $E(\hat{C}') = E(\hat{D}_{\cap 1}) + E(\hat{D}_{\cap 2}) = D_{\cap 1} + D_{\cap 2} = D_\cap$. Because of the two disjoint element groups, $\hat{D}_{\cap 1}$ and $\hat{D}_{\cap 2}$ are independent. Thus the variance $Var[\hat{C}'] = \frac{Var[\hat{D}_{\cap 1}] + Var[\hat{D}_{\cap 2}]}{q^2}$. Next, we will show

$$Var[\hat{D}_{\cap 1}] + Var[\hat{D}_{\cap 2}] \geq Var[\hat{C}].$$

Consider the *KMV* sketch for set X and Y , the sketch size according to Equation 8 is $k = \min\{k_X, k_Y\}$. Similarly, for X_1 and Y_1 , we have the sketch size $k_1 = \min\{k_{X_1}, k_{Y_1}\}$; for X_2 and Y_2 , we have the sketch size $k_2 = \min\{k_{X_2}, k_{Y_2}\}$. Since the index is fixed, we have $k_X = k_{X_1} + k_{X_2}$ and $k_Y = k_{Y_1} + k_{Y_2}$. Then, $k_1 + k_2 = \min\{k_{X_1}, k_{Y_1}\} + \min\{k_{X_2}, k_{Y_2}\} \leq \min\{k_X, k_Y\} = k$.

Let $\Delta = Var[\hat{D}_{\cap 1}] + Var[\hat{D}_{\cap 2}] - Var[\hat{C}]$, after some calculation, we have $\Delta = \frac{D_{\cap 1}^2}{k_1^2} + \frac{D_{\cap 2}^2}{k_2^2} - \frac{D_\cap^2}{k^2} + \frac{D_{\cap 1} D_{U1}}{k_1} + \frac{D_{\cap 2} D_{U2}}{k_2} - \frac{D_\cap D_U}{k}$. Next we show that $\frac{D_{\cap 1}^2}{k_1^2} + \frac{D_{\cap 2}^2}{k_2^2} - \frac{D_\cap^2}{k^2} \geq 0$. Let $k_1 = \frac{1}{\alpha}k$ and $k_2 = \frac{1}{\beta}k$ where $\frac{1}{\alpha} + \frac{1}{\beta} = 1$ and $\alpha, \beta > 1$. Then we have $\frac{D_{\cap 1}^2}{k_1^2} + \frac{D_{\cap 2}^2}{k_2^2} - \frac{D_\cap^2}{k^2} = \frac{\alpha^2 D_{\cap 1}^2 + \beta^2 D_{\cap 2}^2 - D_\cap^2}{k^2} = \frac{(\alpha^2 - 1)D_{\cap 1}^2 + (\beta^2 - 1)D_{\cap 2}^2 - 2D_{\cap 1} D_{\cap 2}}{k^2}$. As for the upper part in the above equation, by inequality of arithmetic and geometric means, we get $(\alpha^2 - 1)D_{\cap 1}^2 + (\beta^2 - 1)D_{\cap 2}^2 - 2D_{\cap 1} D_{\cap 2} \geq 2(\sqrt{(\alpha - 1)(\alpha + 1)(\beta - 1)(\beta + 1)} - 1)D_{\cap 1} D_{\cap 2}$. Since $(\alpha - 1)(\beta - 1) = 1$, we get $\sqrt{(\alpha - 1)(\alpha + 1)(\beta - 1)(\beta + 1)} - 1 = \sqrt{(\alpha + 1)(\beta + 1)} - 1 \geq 0$, thus $\frac{D_{\cap 1}^2}{k_1^2} + \frac{D_{\cap 2}^2}{k_2^2} - \frac{D_\cap^2}{k^2} \geq 0$.

Let $\Delta_1 = \frac{D_{\cap 1} D_{U1}}{k_1} + \frac{D_{\cap 2} D_{U2}}{k_2} - \frac{D_\cap D_U}{k}$, after some computation, we have $\Delta_1 = \frac{(k_1 D_{U2} - k_2 D_{U1})(k_1 D_{\cap 2} - k_2 D_{\cap 1})}{k k_1 k_2}$. As for the numerator(upper) of Δ_1 , consider the two parts after dividing

the element universe, if the union size in one part, say D_{U2} , is larger, meanwhile the corresponding intersection size $D_{\cap 2}$ is larger, we have $\Delta_1 \geq 0$. This case can be realized since one of the two groups divided from element universe is made of high-frequency elements, which will result in large intersection size and large union size under the proper choice of k, k_1, k_2 . \blacksquare

6) **Optimal Buffer Size r :** In this part, we show how to find optimal buffer size r by analysing the variance for *GB-KMV* method. Given the space budget b , we first show that the variance for *GB-KMV* sketch is a function of $f(r, \alpha_1, \alpha_2, b)$ and then we give a method to appropriately choose r . Below are some notations first.

Given two sets X and Y with *G-KMV* sketch \mathcal{L}_X and \mathcal{L}_Y respectively, the containment similarity of Q in X is computed by Equation 26 as $\hat{C}^{GKMV} = \frac{\hat{D}_\cap^{GKMV}}{q}$, where $\hat{D}_\cap^{GKMV} = \frac{K_\cap}{k} \times \frac{k-1}{U(k)}$ is the overlap size.

As for the *GB-KMV* method of set X and Y with sketch $\mathcal{H}_X \cup \mathcal{L}_X$ and $\mathcal{H}_Y \cup \mathcal{L}_Y$ respectively, the containment similarity of Q in X is computed by Equation 27 as $\hat{C}^{GBKMV} = \frac{|\mathcal{H}_Q \cap \mathcal{H}_X| + \hat{D}_\cap^{GKMV}}{q}$, where $|\mathcal{H}_Q \cap \mathcal{H}_X|$ is the number of common elements in \mathcal{E}_H part. It is easy to verify that \hat{C}^{GBKMV} is an unbiased estimator. Also, the variance of *GB-KMV* method estimator is $Var[\hat{C}^{GBKMV}] = \frac{Var[\hat{D}_\cap^{GKMV}]}{q^2}$, where $Var[\hat{D}_\cap^{GKMV}]$ corresponds to the variance of the *G-KMV* sketch in the *GB-KMV* sketch.

Next, with the same space budget b , we compute the average variance of *GB-KMV* method.

Consider the *GB-KMV* index construction which is introduced in Section IV-B by Algorithm 1. Let N be the total number of elements and b the space budget in terms of elements for index construction. Assume that we keep r high-frequency elements by bitmap in the buffer, which have $N_1 = \sum_{j=1}^m |\mathcal{H}_{X_j}| = \sum_{i=1}^r f_i$ elements and occupy $T_1 = m * r / 32$ index space. Then the total number of elements left for *G-KMV* sketch is $N_2 = N - N_1$ and the index space for *G-KMV* sketch is $T_2 = b - T_1$.

Given two sets X_j and X_l , the variance of overlap size estimator in Equation 11 is as follows

$$Var[\hat{D}_\cap] = \frac{D_\cap(kD_U - k^2 - D_U + k + D_\cap)}{k(k-2)} \quad (32)$$

where $D_U = |X_j \cup X_l|$, $D_\cap = |X_j \cap X_l|$ and k is the sketch size. Since the variance is concerned with the union size D_U , the intersection size D_\cap and the signature size k , we first calculate these three formulas, then compute the variance.

Consider the two sets X_j, X_l from dataset \mathcal{S} with *GB-KMV* sketch $\mathcal{H}_{X_j} \cup \mathcal{L}_{X_j}$ and $\mathcal{H}_{X_l} \cup \mathcal{L}_{X_l}$ respectively. The element e_i is associated with frequency f_i , and the probability of element e_i appearing in record X_j is $Pr[h(e_i) \in \mathcal{L}_{X_j}] = \frac{f_i}{N} x_j$. Given a hash value threshold τ , the *G-KMV* signature size of set X_j is computed as $k_j = \tau(x_j - |\mathcal{H}_{X_j}|)$. The total index space in *G-KMV* sketch is $\sum_{j=1}^m k_j = T_2 = b - T_1 = b - \frac{\tau}{32} * m$, then we get $\tau = \frac{b - r/32 * m}{N - N_1}$.

Similar to Equation 29, 30, the sketch size k value for *GB-KMV* sketch is $k = \tau(x_j + x_l) - \tau^2 x_1 x_2 (f_{n^2} - f_{\tau^2})$ where

$f_{n^2} = \frac{\sum_{i=1}^n f_i^2}{N^2}$, $f_{r^2} = \frac{\sum_{i=1}^r f_i^2}{N^2}$. The intersection size and union size of X_j and X_l are $D_{\cap} = x_j x_l (f_{n^2} - f_{r^2})$ and $D_{\cup} = (x_j + x_l)(1 - f_r) - x_j x_l (f_{n^2} - f_{r^2})$ where $f_r = \frac{\sum_{i=1}^r f_i}{N}$, then the variance of *GB-KMV* method by Equation 32 is

$$\text{Var}[\hat{C}_{GBKMV}] = \frac{(x_j + x_l)x_l}{kx_j} F_1 + \frac{x_l^2}{k} F_2 + \frac{x_l}{x_j} F_3$$

where $F_1 = f_{n^2} - f_{r^2}$, $F_2 = -(f_{n^2} - f_{r^2})^2$ and $F_3 = -(f_{n^2} - f_{r^2})$, and the average variance of *GB-KMV* method $\text{Var}_{GBKMV} = \frac{1}{m^2} \sum_{j=1}^m \sum_{l=1}^m \text{Var}[\hat{C}_{GBKMV}]$ is

$$\text{Var}_{GBKMV} = L_1 F_1 + L_2 F_2 + L_3 F_3$$

where $L_1 = \frac{1}{m^2} \sum_{j=1}^m \sum_{l=1}^m \frac{(x_j+x_l)x_j x_l}{kx_j^2}$, $L_2 = \frac{1}{m^2} \sum_{j=1}^m \sum_{l=1}^m \frac{(x_j x_l)^2}{kx_j^2}$ and $L_3 = \frac{1}{m^2} \sum_{j=1}^m \sum_{l=1}^m \frac{x_l}{x_j}$.

Note that F_1, F_2, F_3 is concerned with the element frequency which can be computed by using the distribution $p_1(x) = c_1 x^{-\alpha_1}$; L_1, L_2, L_3 is related to the record size which can be computed by using $p_2(x) = c_2 x^{-\alpha_2}$ and k is related to the index budget size b and buffer size r , then Var_{GBKMV} can be restated as $\text{Var}_{GBKMV} = L_1 F_1 + L_2 F_2 + L_3 F_3 = \frac{1}{m^2} [A \frac{(d^{1-\alpha_1} - r^{1-\alpha_1})(d^{1-2\alpha_1} - r^{1-2\alpha_1})}{b - \frac{m}{32}r} - B \frac{(d^{1-\alpha_1} - r^{1-\alpha_1})(d^{1-2\alpha_1} - r^{1-2\alpha_1})^2}{b - \frac{m}{32}r} - C \frac{(d^{1-2\alpha_1} - r^{1-2\alpha_1})}{b - \frac{m}{32}r}]$ where

$$A = \frac{N(\alpha_1-1)^2}{(1-2\alpha_1)d^{1-\alpha_1}(d^{1-\alpha_1}-1)^2} \frac{(\alpha_2-1)^2}{-\alpha_2(2-\alpha_2)} \frac{(x_t^{2-\alpha_2} - x_1^{2-\alpha_2})(x_t^{-\alpha_2} - x_1^{-\alpha_2})}{(x_t^{-\alpha_2+1} - x_1^{-\alpha_2+1})^2},$$

$$B = \frac{N(\alpha_1-1)^4}{(1-2\alpha_1)^2 d^{1-\alpha_1} (d^{1-\alpha_1}-1)^4} \frac{(\alpha_2-1)^2}{-\alpha_2(3-\alpha_2)} \frac{(x_t^{3-\alpha_2} - x_1^{3-\alpha_2})(x_t^{-\alpha_2} - x_1^{-\alpha_2})}{(x_t^{-\alpha_2+1} - x_1^{-\alpha_2+1})^2}$$

$$\text{and } C = \frac{(\alpha_1-1)^2}{(1-2\alpha_1)(d^{1-\alpha_1}-1)^2} \frac{(\alpha_2-1)^2}{-\alpha_2(2-\alpha_2)} \frac{(x_t^{2-\alpha_2} - x_1^{2-\alpha_2})(x_t^{-\alpha_2} - x_1^{-\alpha_2})}{(x_t^{-\alpha_2+1} - x_1^{-\alpha_2+1})^2}$$

Moreover, we have $\text{Var}_{GBKMV} = \frac{a_1 r^{5\alpha_1+1} + a_2 r^{5\alpha_1+1} + a_3 r^{4\alpha_1+1} + a_4 r^{3\alpha_1+2} + a_5 r^{3\alpha_1+1} + a_6 r^{2\alpha_1+2} + a_7 r^{\alpha_1+2} + a_8 r^2}{(b - \frac{m}{32}r)^{5\alpha_1}}$ where $a_1 = C \frac{m}{32} d^{1-2\alpha_1}$, $a_2 = Ad^{1-\alpha_1} - Bd^{3-5\alpha_1} - bCd^{1-2\alpha_1}$, $a_3 = -Ad^{1-\alpha_1} + Bd^{2-4\alpha_1}$, $a_4 = -C \frac{m}{32}$, $a_5 = -Ad^{1-\alpha_1} + 2Bd^{2-2\alpha_1} + bC$, $a_6 = A - 2Bd^{1-2\alpha_1}$, $a_7 = -Bd^{1-\alpha_1}$ and $a_8 = B$.

We can see that the variance Var_{GBKMV} can be regarded as a function of $f(r, \alpha_1, \alpha_2, b)$, i.e.,

$$\text{Var}_{GBKMV} = f(r, \alpha_1, \alpha_2, b) \quad (33)$$

Similarly, for the *G-KMV* sketch based method, the variance can be calculated as

$$\text{Var}[\hat{C}_{GKMV}] = \frac{(x_j + x_l)x_j x_l}{kx_j^2} F'_1 + \frac{(x_j x_l)^2}{kx_j^2} F'_2 + \frac{x_j x_l}{x_j^2} F'_3$$

where $F'_1 = f_{n^2}$, $F'_2 = -f_{n^2}^2$, $F'_3 = -f_{n^2}$ and $k = \frac{b}{N}(x_j + x_l) - (\frac{b}{N})^2 x_j x_l f_{n^2}$. Let $\Delta \text{Var} = \text{Var}[\hat{C}_{GBKMV}] - \text{Var}[\hat{C}_{GKMV}]$, then for all pairs of X_j, X_l , the average of ΔVar is $V_{\Delta} = \frac{1}{m^2} \sum_{j=1}^m \sum_{l=1}^m \Delta \text{Var}$. Moreover, we can rewrite V_{Δ} as $V_{\Delta} = L_1(F'_1 - F_1) + L_2(F'_2 - F_2) + L_3(F'_3 - F_3)$.

Eventually, in order to find the optimal r , i.e., the number of high-frequency elements in *GB-KMV* method, we give the optimization goal as $\max_r V_{GBKMV} = f(r, \alpha_1, \alpha_2, b)$, s.t. $V_{\Delta} < 0$.

In order to compute the above optimization problem, we try to extract the roots of the first derivative function of Equation 33(i.e., $f(r, \alpha_1, \alpha_2, b)$) with respect to r . However, the derivative function is a polynomial function with degree

of r larger than four. According to Abel's impossibility theorem [39], there is no algebraic solution, thus we try to give the numerical solution.

Recall that we use bitmap to keep the r high-frequency elements, given the space budget b , the element frequency and record size distribution with power-law exponent α_1 and α_2 respectively, the optimization goal $\max_r V_{GBKMV}$ can be considered as a function $\max_r f(r, b, \alpha_1, \alpha_2)$. Given a dataset \mathcal{S} and the space budget b , we can get the power-law exponent α_1, α_2 . Then we assign 8, 16, 24, ... to r and calculate the $f(r, b, \alpha_1, \alpha_2)$. In this way, we can give a good guide to the choice of r .

7) *GB-KMV Sketch provides Better Accuracy than LSH-E Method*

In Section III-B, we have shown that the variance of *LSH-E* estimator(Equation 21) is larger than that of *MinHash LSH* estimator(Equation 19). Note that *G-KMV* sketch is a special case of *GB-KMV* sketch when the buffer size $r = 0$. By choosing an optimal buffer size r in IV-C6, it can guarantee that the performance of *GB-KMV* is not worse than *G-KMV*. Below, we show that *G-KMV* outperforms *MinHash LSH* in terms of estimate accuracy.

Theorem 5. *The variance of G-KMV method is smaller than that of minHash LSH method given the same sketch size.*

Proof: Suppose that the *minHash LSH* method uses k' hash functions to the dataset, then the total sketch size is $T = mk'$. Let τ be the global threshold of *G-KMV* method, we have $\tau = \frac{mk'}{N}$ where N is the total number of elements in dataset.

We first consider the *G-KMV* method. Similar to Equation 29, 30, the intersection size of X_j and X_l is $D_{\cap} = x_j x_l \sum_{i=1}^n \frac{f_i^2}{N^2}$, and the union size is $D_{\cup} = x_j + x_l - x_j x_l \sum_{i=1}^n \frac{f_i^2}{N^2}$. Then by Equation 11 the variance of the *G-KMV* method to estimate the containment similarity of X_j in X_l can be rewritten as

$$V_{G-KMV} = \frac{(x_j + x_l)x_j x_l}{kx_j^2} F_1 + \frac{(x_j x_l)^2}{kx_j^2} F_2 + \frac{x_j x_l}{x_j^2} F_3 \quad (34)$$

where $F_1 = f_{n^2}$, $F_2 = -(f_{n^2})^2$, $F_3 = -f_{n^2}$ and $f_{n^2} = \sum_{i=1}^n \frac{f_i^2}{N^2}$.

Next we compute the k value of the sketch. Note that τ is the global threshold of *G-KMV* method. The k value corresponding the intersection size of X_j and X_l by Equation 24 is $k = \tau(x_j + x_l) - \tau^2 x_j x_l f_{n^2}$. Then the average variance $V_1 = \frac{1}{m^2} \sum_{j=1}^m \sum_{l=1}^m V_{G-KMV}$ is

$$V_1 = \frac{1}{m^2} (L_1 F_1 + L_2 F_2 + L_3 F_3)$$

where $L_1 = \sum_{j=1}^m \sum_{l=1}^m \frac{(x_j+x_l)x_j x_l}{x_j^2 k}$, $L_2 = \sum_{j=1}^m \sum_{l=1}^m \frac{(x_j x_l)^2}{kx_j^2}$ and $L_3 = \sum_{j=1}^m \sum_{l=1}^m \frac{x_j x_l}{x_j^2}$. After some computation, $V_1 = \frac{1}{k'} [\frac{(\alpha_2-1)^3}{-\alpha_2(2-\alpha_2)} W_1 f_{n^2} + \frac{(\alpha_2-1)^3}{\alpha_2(2-\alpha_2)(3-\alpha_2)} W_2 (f_{n^2})^2 + k' \frac{(\alpha_2-1)^2}{\alpha_2(2-\alpha_2)} W_3 f_{n^2}]$ where $W_1 = \frac{(x_t^{2-\alpha_2} - x_1^{2-\alpha_2})^2 (x_t^{-\alpha_2} - x_1^{-\alpha_2})}{(x_t^{-\alpha_2+1} - x_1^{-\alpha_2+1})^3}$, $W_2 = \frac{(x_t^{2-\alpha_2} - x_1^{2-\alpha_2})(x_t^{-\alpha_2} - x_1^{-\alpha_2})(x_t^{3-\alpha_2} - x_1^{3-\alpha_2})}{(x_t^{-\alpha_2+1} - x_1^{-\alpha_2+1})^3}$, $W_3 = \frac{(x_t^{2-\alpha_2} - x_1^{2-\alpha_2})(x_t^{-\alpha_2} - x_1^{-\alpha_2})}{(x_t^{-\alpha_2+1} - x_1^{-\alpha_2+1})^2}$ and $f_{n^2} = \frac{(1-\alpha_1)^2 d^{1-2\alpha_1} - 1}{1-2\alpha_1 (d^{1-\alpha_1} - 1)^2}$.

Note that $x_t(x_1, \text{ resp.})$ is the largest(smallest, resp.) set size and d is the distinct number of elements.

Next we take into account the minHash *LSH* method.

Given two sets X_j and X_l , by Equation 19, the variance of minHash *LSH* method to estimate the containment similarity of X_j in X_l is $V_{\min H} = \frac{1}{k'}[a_1 f_{n^2} + a_2 (f_{n^2})^2 + a_3 (f_{n^2})^3 + a_4 (f_{n^2})^4]$ where $a_1 = x_l + \frac{x_j^2}{x_j}$, $a_2 = -4x_l^2$, $a_3 = 5 \frac{x_j x_l^3}{x_j + x_l}$ and $a_4 = -2 \frac{x_j^2 x_l^4}{(x_j + x_l)^2}$. Then the average variance $V_2 = \frac{1}{m^2} \sum_{j=1}^m \sum_{l=1}^m V_{\min H}$ is

$$V_2 = \frac{1}{k' m^2} [A_1 f_{n^2} + A_2 (f_{n^2})^2 + A_3 (f_{n^2})^3 + A_4 (f_{n^2})^4]$$

$$\begin{aligned} \text{where } A_1 &= \frac{\alpha_2 - 1}{2 - \alpha_2} \frac{x_t^{2-\alpha_2} - x_1^{2-\alpha_2}}{x_t^{-\alpha_2+1} - x_1^{-\alpha_2+1}} + \\ &\frac{(\alpha_2 - 1)^2}{-\alpha_2(3 - \alpha_2)} \frac{(x_t^{3-\alpha_2} - x_1^{3-\alpha_2})(x_t^{-\alpha_2} - x_1^{-\alpha_2})}{(x_t^{-\alpha_2+1} - x_1^{-\alpha_2+1})^2}, \quad A_2 = \\ &-4 \frac{\alpha_2 - 1}{3 - \alpha_2} \frac{x_t^{3-\alpha_2} - x_1^{3-\alpha_2}}{x_t^{-\alpha_2+1} - x_1^{-\alpha_2+1}}, \quad A_3 = 5 \frac{\alpha_2 - 1}{4 - \alpha_2} \frac{x_t^{4-\alpha_2} - x_1^{4-\alpha_2}}{x_t^{-\alpha_2+1} - x_1^{-\alpha_2+1}} \\ \text{and } A_4 &= -2 \left[\left(\frac{\alpha_2 - 1}{3 - \alpha_2} \frac{x_t^{3-\alpha_2} - x_1^{3-\alpha_2}}{x_t^{-\alpha_2+1} - x_1^{-\alpha_2+1}} \right)^2 - \right. \\ &2 \frac{(\alpha_2 - 1)^2}{(4 - \alpha_2)(2 - \alpha_2)} \frac{(x_t^{2-\alpha_2} - x_1^{2-\alpha_2})(x_t^{4-\alpha_2} - x_1^{4-\alpha_2})}{(x_t^{-\alpha_2+1} - x_1^{-\alpha_2+1})^2} + \\ &\left. 3 \frac{\alpha_2 - 1}{5 - \alpha_2} \frac{x_t^{5-\alpha_2} - x_1^{5-\alpha_2}}{x_t^{-\alpha_2+1} - x_1^{-\alpha_2+1}} \right] \end{aligned}$$

Note that f_{n^2} is computed by the distribution $p_1(x) = c_1 x^{-\alpha_1}$ and the sum over set size is computed by the set size distribution $p_2(x) = c_2 x^{-\alpha_2}$, and the variance V_1 and V_2 is dependent on α_1 and α_2 . Compare the variance V_1 and V_2 , we get that $V_1 < V_2$ for all $\alpha_1 > 0$ and $\alpha_2 > 0$.

Next, we analyse the performance of the two methods with the dataset following uniform distribution(i.e., $\alpha_1 = 0$, $\alpha_2 = 0$).

For *G-KMV* method, the average variance is

$$V'_1 = \frac{1}{m^2} (L_1 F_1 + L_2 F_2 + L_3 F_3)$$

$$\text{where } L_1 = \sum_{j=1}^m \sum_{l=1}^m \frac{(x_j + x_l) x_j x_l}{x_j^2 k}, \quad L_2 = \sum_{j=1}^m \sum_{l=1}^m \frac{(x_j x_l)^2}{k x_j^2} \quad \text{and} \quad L_3 = \sum_{j=1}^m \sum_{l=1}^m \frac{x_j x_l}{x_j^2}.$$

$$\text{After some computation, } V'_1 = \frac{1}{k'} \left[\frac{(\alpha_2 - 1)^3}{2 - \alpha_2} W_1 f_{n^2} - \frac{(\alpha_2 - 1)^3}{(2 - \alpha_2)(3 - \alpha_2)} W_2 (f_{n^2})^2 - k' \frac{(\alpha_2 - 1)^2}{2 - \alpha_2} W_3 (f_{n^2})^3 \right] \quad \text{where}$$

$$W_1 = \frac{(x_t^{2-\alpha_2} - x_1^{2-\alpha_2})^2 (\ln x_t - \ln x_1)}{(x_t^{-\alpha_2+1} - x_1^{-\alpha_2+1})^3}, \quad W_2 = \frac{(x_t^{2-\alpha_2} - x_1^{2-\alpha_2})(\ln x_t - \ln x_1)(x_t^{3-\alpha_2} - x_1^{3-\alpha_2})}{(x_t^{-\alpha_2+1} - x_1^{-\alpha_2+1})^3}, \quad W_3 = \frac{(x_t^{2-\alpha_2} - x_1^{2-\alpha_2})(\ln x_t - \ln x_1)}{(x_t^{-\alpha_2+1} - x_1^{-\alpha_2+1})^2} \quad \text{and} \quad f_{n^2} = \frac{(1 - \alpha_1)^2 d^{1-2\alpha_1} - 1}{1 - 2\alpha_1 (d^{1-\alpha_1} - 1)^2}.$$

Note that $x_t(x_1, \text{ resp.})$ is the largest(smallest, resp.) set size and d is the distinct number of elements.

For *LSH-E* method, the average variance is

$$V'_2 = \frac{1}{k' m^2} [A_1 f_{n^2} + A_2 (f_{n^2})^2 + A_3 (f_{n^2})^3 + A_4 (f_{n^2})^4]$$

$$\begin{aligned} \text{where } A_1 &= \frac{\alpha_2 - 1}{2 - \alpha_2} \frac{x_t^{2-\alpha_2} - x_1^{2-\alpha_2}}{x_t^{-\alpha_2+1} - x_1^{-\alpha_2+1}} + \\ &\frac{(\alpha_2 - 1)^2}{3 - \alpha_2} \frac{(x_t^{3-\alpha_2} - x_1^{3-\alpha_2})(\ln x_t - \ln x_1)}{(x_t^{-\alpha_2+1} - x_1^{-\alpha_2+1})^2}, \quad A_2 = \\ &-4 \frac{\alpha_2 - 1}{3 - \alpha_2} \frac{x_t^{3-\alpha_2} - x_1^{3-\alpha_2}}{x_t^{-\alpha_2+1} - x_1^{-\alpha_2+1}}, \quad A_3 = 5 \frac{\alpha_2 - 1}{4 - \alpha_2} \frac{x_t^{4-\alpha_2} - x_1^{4-\alpha_2}}{x_t^{-\alpha_2+1} - x_1^{-\alpha_2+1}} \end{aligned}$$

$$\begin{aligned} \text{and } A_4 &= -2 \left[\left(\frac{\alpha_2 - 1}{3 - \alpha_2} \frac{x_t^{3-\alpha_2} - x_1^{3-\alpha_2}}{x_t^{-\alpha_2+1} - x_1^{-\alpha_2+1}} \right)^2 - \right. \\ &2 \frac{(\alpha_2 - 1)^2}{(4 - \alpha_2)(2 - \alpha_2)} \frac{(x_t^{2-\alpha_2} - x_1^{2-\alpha_2})(x_t^{4-\alpha_2} - x_1^{4-\alpha_2})}{(x_t^{-\alpha_2+1} - x_1^{-\alpha_2+1})^2} + \\ &\left. 3 \frac{\alpha_2 - 1}{5 - \alpha_2} \frac{x_t^{5-\alpha_2} - x_1^{5-\alpha_2}}{x_t^{-\alpha_2+1} - x_1^{-\alpha_2+1}} \right]. \end{aligned}$$

Similarly, we can get that $V'_1 < V'_2$. ■

Remark 4. We have illustrated that the variance of *GB-KMV* is smaller than that of *LSH-E*. Then by Chebyshev's inequality, i.e., $\Pr(|X - \mu| \geq \epsilon \sigma) \leq \frac{1}{\epsilon^2}$ where μ is the expectation, δ is the standard deviation and $\epsilon > 1$ is a constant, we consider the probability that values lie outside the interval $[\mu - \epsilon \delta, \mu + \epsilon \delta]$, that is, values deviating from the expectation. By Theorem 5, we get that the standard deviation δ_1 of *GB-KMV* is smaller than δ_2 of *LSH-E*, then with the same interval $[\mu - \epsilon \delta, \mu + \epsilon \delta]$, the constant ϵ_1 for *GB-KMV* is larger than ϵ_2 for *LSH-E*, thus the probability that values lie outside the interval for *GB-KMV* is smaller than that for *LSH-E*, which means that the result of *GB-KMV* is more concentrated around the expected value than that of *LSH-E*.

V. PERFORMANCE STUDIES

In this section, we empirically evaluate the performance of our proposed *GB-KMV* method and compare *LSH* Ensemble [44] as baseline. We also compare our approximate *GB-KMV* method with the exact containment similarity search method. All experiments are conducted on PCs with Intel Xeon 2 × 2.3GHz CPU and 128GB RAM running Debian Linux, and the source code of *GB-KMV* is made available [1].

A. Experimental Setup

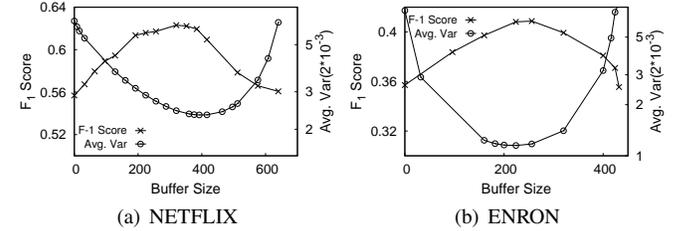


Fig. 5. Effect of Buffer Size

Approximate Algorithms. In the experiments, the approximate algorithms evaluated are as follows.

- ***GB-KMV***. Our approach proposed in Section IV-B.
- ***LSH-E***. The state-of-the-art approximate containment similarity search method proposed in [44].

The above two algorithms are implemented in Go programming language. We get the source code of *LSH-E* from [44]. For *LSH-E*, we follow the parameter setting from [44].

Exact Algorithms. To better evaluate the proposed methods, we also compare our approximate method *GB-KMV* with the following two exact containment similarity search methods.

- ***PPjoin****. We extend the prefix-filtering based method from [40] to tackle the containment similarity search problem.
- ***FrequentSet***. The state-of-the-art exact containment similarity search method proposed in [5].

Dataset	Abbrev	Type	Record	#Records	AvgLength	#DistinctEle	α_1 -eleFreq	α_2 -recSize
Netflix [12]	NETFLIX	Rating	Movie	480,189	209.25	17,770	1.14	4.95
Delicious [2]	DELIC	Folksonomy	User	833,081	98.42	4,512,099	1.14	3.05
CaOpenData [44]	COD	Folksonomy	User	65,553	6284	111,011,807	1.09	1.81
Enron [3]	ENRON	Text	Email	517,431	133.57	1,113,219	1.16	3.10
Reuters [4]	REUTERS	Folksonomy	User	833,081	77.6	283,906	1.32	6.61
Webspam [38]	WEBSPAM	Text	Text	350,000	3728	16,609,143	1.33	9.34
WDC Web Table [44]	WDC	Text	Text	262,893,406	29.2	111,562,175	1.08	2.4

TABLE II. CHARACTERISTICS OF DATASETS

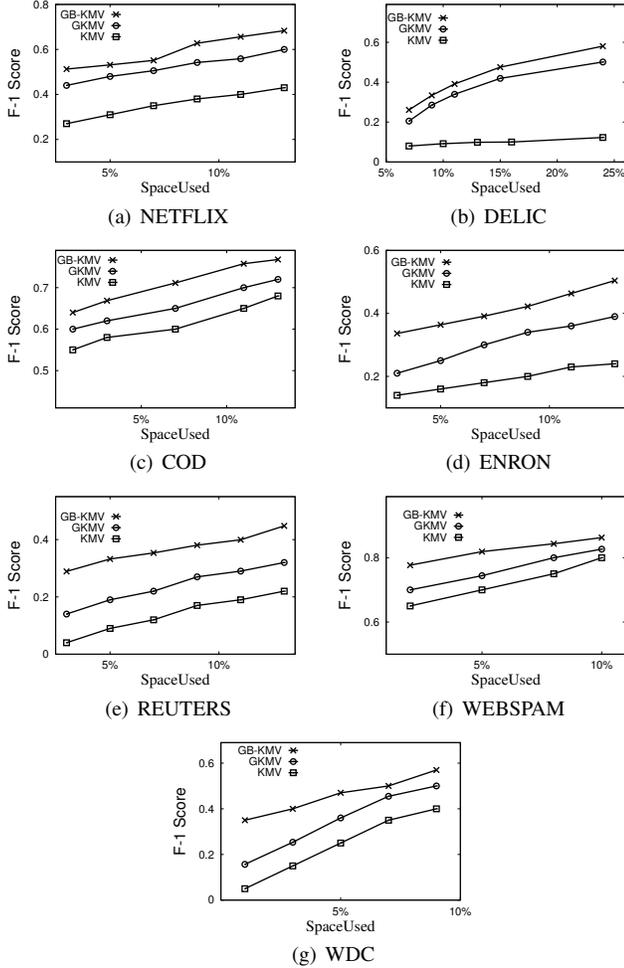


Fig. 6. GB-KMV, G-KMV, KMV comparison

Remark 5. A novel size-aware overlap set similarity join algorithm has been recently proposed in [25]. Although the containment similarity search relies on the set overlap, their technique cannot be trivially applied because we need to construct c -subset inverted lists for each possible query size. In particular, in the size-aware overlap set similarity join algorithm, it is required to build the c -subset inverted list for the given overlap threshold c . In our GB-KMV method, the threshold c corresponds to $|Q| * t^*$, where $|Q|$ is the query size and t^* is the similarity threshold, thus with different query size $|Q|$, we need to build different $|Q| * t^*$ -subset inverted lists, which is very inefficient.

Datasets. We deployed 7 real-life datasets with different data properties. Note that the records with size less than 10 are discarded from dataset. We also remove the stop words (e.g., "the") from the dataset. Table II shows the detailed characteristics of the 7 datasets. Each dataset is illustrated

with the dataset type, the representations of record, the number of records in the dataset, the average record length, and the number of distinct elements in the dataset. We also report the power-law exponent α_1 and α_2 (skewness) of the record size and element frequency of the dataset respectively. Note that we make use of the framework in [18] to quantify the power-law exponent. The dataset Canadian Open Data appears in the state-of-the-art algorithm *LSH-E* [44].

Settings. We borrow the idea from the evaluation of *LSH-E* in [44] to use F_α score ($\alpha=1, 0.5$) to evaluate the accuracy of the containment similarity search. Given a query Q randomly selected from the dataset \mathcal{S} and a containment similarity threshold t^* , we define $T = \{X : t(Q, X) \geq t^*, X \in \mathcal{S}\}$ as the ground truth set and A as the collection of records returned by some search algorithms. The precision and recall to evaluate the experiment accuracy are $Precision = \frac{|T \cap A|}{|A|}$ and $Recall = \frac{|T \cap A|}{|T|}$ respectively. The F_α score is defined as follows.

$$F_\alpha = \frac{(1 + \alpha^2) * Precision * Recall}{\alpha^2 * Precision + Recall} \quad (35)$$

Note that we use $F_{0.5}$ score because *LSH-E* favours recall in [44]. We use the datasets from Table II to evaluate the performance of our algorithm, and we randomly choose 200 queries from the dataset.

As to the default values, the similarity threshold is set as $t^* = 0.5$. In the experiments, we use the ratio of space budget to the total dataset size to measure the space used. For our GB-KMV method, it is set to 10%. For *LSH-E* method, we use the same default values in [44] where the signature size of each record is 256 and the number of partition is 32. By varying the number of hash functions, we change the space used in *LSH-E*.

B. Performance Tuning

As shown in Section IV-C6, we can use the variance estimation function to identify a good buffer size r for GB-KMV method based on the skewness of record size and element frequency, as well as the space budget. In Fig. 5, we use NETFLIX and ENRON to evaluate the goodness of the function by comparing the trend of the variance and the estimation accuracy. By varying the buffer size r , Fig. 5 reports the estimated variance (right side y axis) based on the variance function in Section IV-C6 as well as the F_1 score (left side y axis) of the corresponding GB-KMV sketch with buffer size r . Fig. 5(a) shows that the best buffer size for variance estimation (prefer small value) is around 400, while the GB-KMV method achieves the best F_1 score (prefer large value) with buffer size around 380. They respectively become 220 and 230 in Fig. 5(b). This suggests that our variance estimation function is quite reliable to identify a good buffer size. In the following experiments, GB-KMV method will use buffer size suggested by this system, instead of manually tuning.

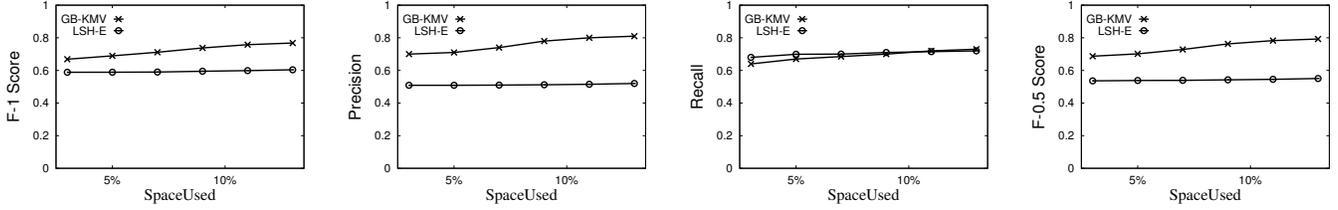


Fig. 7. Accuracy versus Space on COD

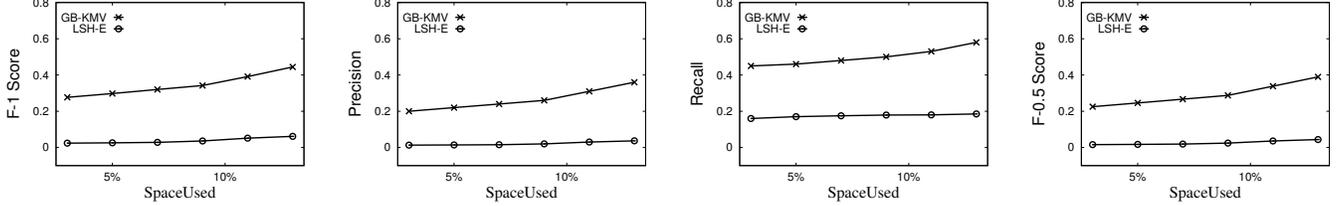


Fig. 8. Accuracy versus Space on DELIC

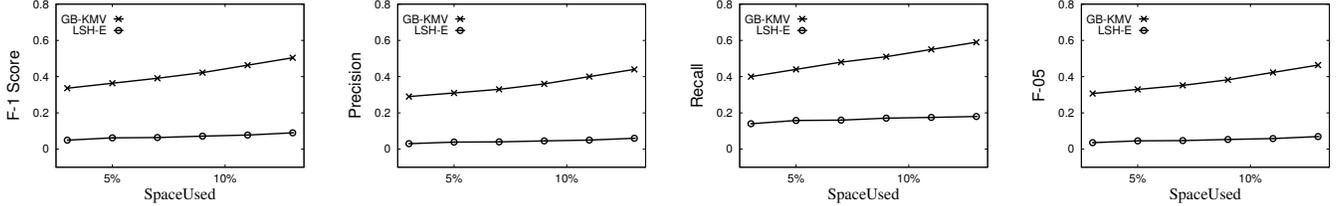


Fig. 9. Accuracy versus Space on ENRON

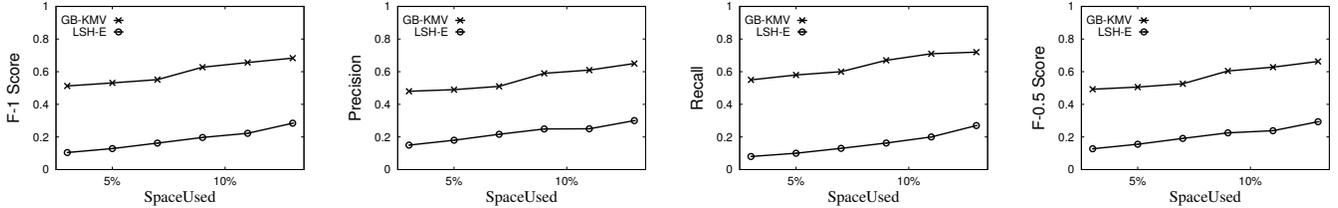


Fig. 10. Accuracy versus Space on NETFLIX

We also compare the performance of *KMV*, *G-KMV*, and *GB-KMV* methods in Fig. 6 to evaluate the effectiveness of using global threshold and the buffer on 7 datasets. It is shown that the use of new *KMV* estimator with global threshold (i.e., Equation 26) can significantly improve the search accuracy. By using a buffer whose size is suggested by the system, we can further enhance the performance under the same space budget. In the following experiments, we use *GB-KMV* for the performance comparison with the state-of-the-art technique *LSH-E*.

C. Space v.s. Accuracy

An important measurement for sketch technique is the trade-off between the space and accuracy. We evaluate the space-accuracy trade-offs of *GB-KMV* method and *LSH-E* method in Figs. 7-13 by varying the space usage on five datasets NETFLIX, DELIC, COD, ENRON, REUTERS, WEBSPPAM and WDC. We use F_1 score, $F_{0.5}$ score, precision and recall to measure the accuracy. By changing the number of hash functions, we tune the space used in *LSH-E*. It is reported that our *GB-KMV* can beat the *LSH-E* in terms of space-accuracy trade-off with a big margin under all settings.

We also plot the distribution of accuracy (i.e., min, max and average value) to compare our *GB-KMV* method with *LSH-E* in Fig. 14.

Meanwhile, by changing the similarity threshold, F_1 score is reported in Fig. 15 on dataset NETFLIX and COD. We can see that with various similarity thresholds, our *GB-KMV* always outperforms *LSH-E*.

We also evaluate the space-accuracy trade-offs on synthetic datasets with 100K records in Fig. 16 where the record size and the element frequency follow the zipf distribution. We can see that on datasets with different record size and element frequency skewness, *GB-KMV* consistently outperforms *LSH-E* in terms of space-accuracy trade-off.

D. Time v.s. Accuracy

Another important measurement for the sketch technique is the trade-off between time and accuracy. Hopefully, the sketch should be able to quickly complete the search with a good accuracy. We tune the index size of *GB-KMV* to show the trade-off. As to the *LSH-E*, we tune the number of hash functions. The time is reported as the average search time per query. In Fig. 17, we evaluate the time-accuracy trade-offs for *GB-KMV* and *LSH-E* on four datasets COD, NETFLIX, DELIC and ENRON where the accuracy is measured by F_1 score. It is shown that with the similar accuracy (F_1 score), *GB-KMV* is significantly faster than *LSH-E*. For datasets COD, DELIC and ENRON, *GB-KMV* can be 100 times faster than *LSH-E* with the same F_1 score. It is observed that the accuracy (F_1 score) improvement of *LSH-E* algorithm is very slow compared with *GB-KMV* method. This is because the *LSH-E* method favours recall and the precision performance is quite poor even for a large number of hash functions, resulting in a poor F_1 score which considers both precision and recall.

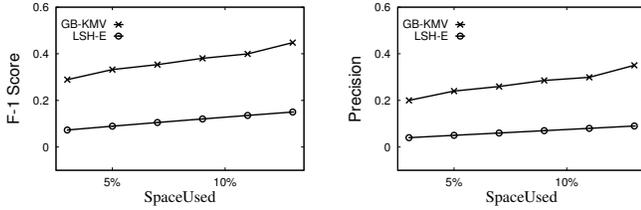


Fig. 11. Accuracy versus Space on REUTERS

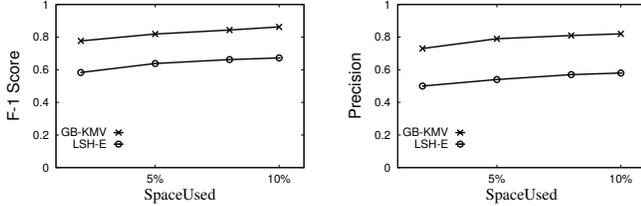


Fig. 12. Accuracy versus Space on WEBSpAM

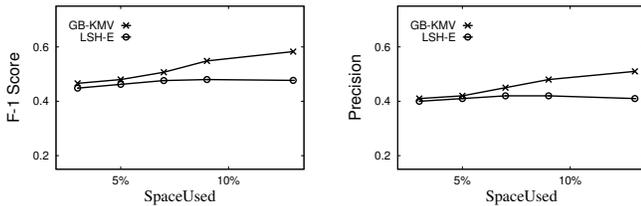


Fig. 13. Accuracy versus Space on WDC

Dataset	GB-KMV	LSH-E
NETFLIX	10	118
DELIC	10	211
COD	10	4
ENRON	10	185
REUTERS	10	329
WEBSpAM	10	7
WDC	10	109

TABLE III. THE SPACE USAGE(%)

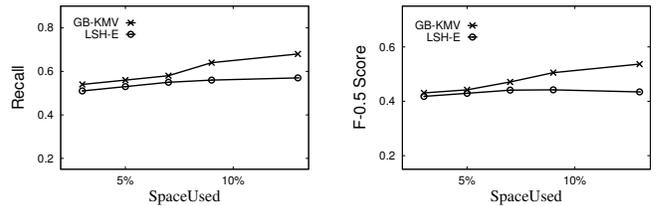
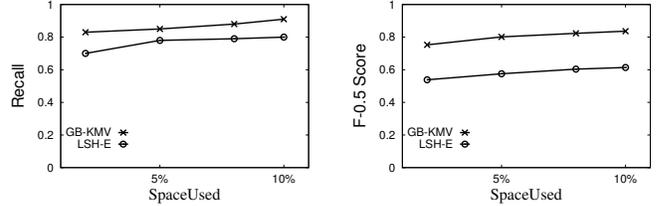
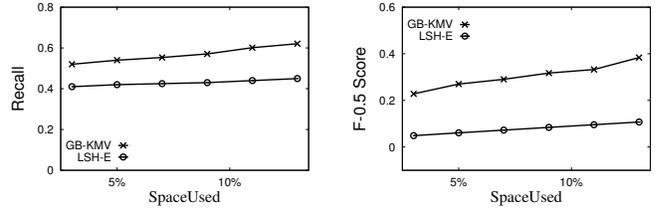
E. Sketch Construction Time

In this part, we compare the sketch construction time of *GB-KMV* and *LSH-E* on different datasets under default settings. As expected, *GB-KMV* uses much less sketch construction time than that of *LSH-E* since *GB-KMV* sketch need only one hash function, while *LSH-E* needs multiple for a decent accuracy. Note that, for the internet scale dataset WDC, the index construction time for *GB-KMV* is around 10 minutes, while for *LSH-E* it is above 60 minutes. We also give the space usage of the two methods on each dataset in Table III. The space usage of *GB-KMV* is 10% as mentioned in Settings. For *LSH-E* in some dataset, the space is over 100% because there are many records with size less than the number of hash functions 256.

F. Supplementary Experiment

Evaluation on Uniform Distribution. In Theorem 5, we have theoretically shown that when the dataset follows uniform distribution (i.e., $\alpha_1 = 0$ and $\alpha_2 = 0$), our *GB-KMV* method can outperform the *LSH-E* method. In this part, we experimentally illustrate the performance on dataset with uniform distribution. We generate 100k records where the record size is uniformly distributed between 10 and 5000, and each element is randomly chosen from 100,000 distinct elements. Fig. 19(a) illustrates the time-accuracy trade-off of *GB-KMV* and *LSH-E* on the synthetic dataset with 100K records. It is reported that, to achieve the same accuracy (F_1 score), *GB-KMV* consumes much less time than *LSH-E*.

Comparison with Exact Algorithms. We also compare the



running time of our proposed method *GB-KMV* with two exact containment similarity search methods PPjoin* [40] and FreqSet [5]. Experiments are conducted on the dataset WebSpam, which consists of 350,000 records and has the average length around 3,700. We partition the data into 5 groups based on their record size with boundaries increasing from 1,000 to 5,000. As expected, Fig. 19(b) shows that the running time of our approximate algorithm is not sensitive to the growth of the record size because a fixed number of samples are used for a given budget. *GB-KMV* outperforms two exact algorithm by a big margin, especially when the record size is large, with a decent accuracy (i.e., with F_1 score and recall always larger than 0.8 and 0.9 under all settings).

G. Discussion Summary

In the accuracy comparison between *GB-KMV* and *LSH-E*, it is remarkable to see that the accuracy (i.e., F_1 score) is very low on some datasets. We give some discussions as follows.

First we should point out that in [44], the accuracy of *LSH-E* is only evaluated on **only** one dataset COD, in which both our *GB-KMV* method and *LSH-E* can achieve decent accuracy performance with F_1 score above 0.5.

As mentioned in III-A, the *LSH-E* method first transforms the containment similarity to Jaccard similarity, then in order to make use of the efficient index techniques, *LSH-E* partitions the dataset and uses the upper bound to approximate the record size in each partition. which can favour recall but result in extra false positives as analysed in section III-B. However, the *LSH-E* method does not provide a partition scheme associated with different data distribution, and the algorithm setting (e.g., 256 hash functions and 32 partitions) can not perform well in some dataset.

VI. RELATED WORK

In this Section, we review two closely related categories of work on set containment similarity search.

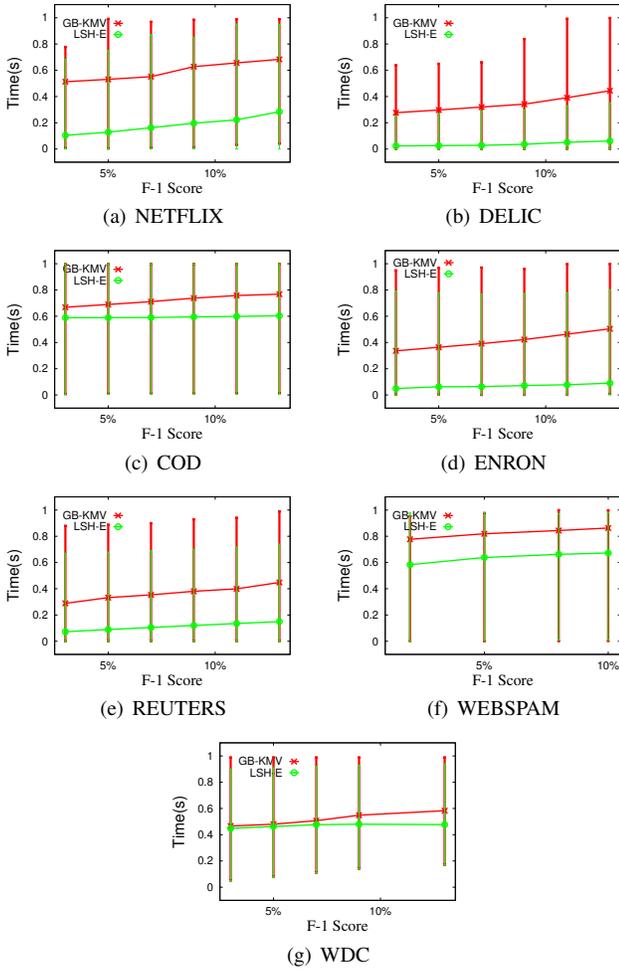


Fig. 14. The distribution of Accuracy

Exact Set Similarity Queries. Exact set similarity query has been widely studied in the literature. Existing solutions are mainly based on the filtering-verification framework which can be divided into two categories, prefix-filter based method and partition-filter based method. Prefix-filter based method is first introduced by Bayardo *et al.* in [10]. Xiao *et al.* [40] further improve the prefix-filter method by exploring positional filter and suffix filter techniques. In [32], Mann *et al.* introduce an efficient candidate verification algorithm which significantly improves the efficiency compared with the other prefix filter algorithms. Wang *et al.* [36] consider the relations among records in query processing to improve the performance. Deng *et al.* in [23] present an efficient similarity search method where each object is a collection of sets. For partition-based method, in [7], Arasu *et al.* devise a two-level algorithm which uses partition and enumeration techniques to search for exact similar records. Deng *et al.* in [24] develop a partition-based method which can effectively prune the candidate size at the cost higher filtering cost. In [43], Zhang *et al.* propose an efficient framework for exact set similarity search based on tree index structure. In [25], Deng *et al.* present a size-aware algorithm which divides all the sets into small and large ones by size and processes them separately. Regarding exact containment similarity search, Agrawal *et al.* in [5] build the inverted lists on the token-sets and considered the string transformation.

Approximate Set Similarity Queries. The approximate set similarity queries mostly adopt the Locality Sensitive

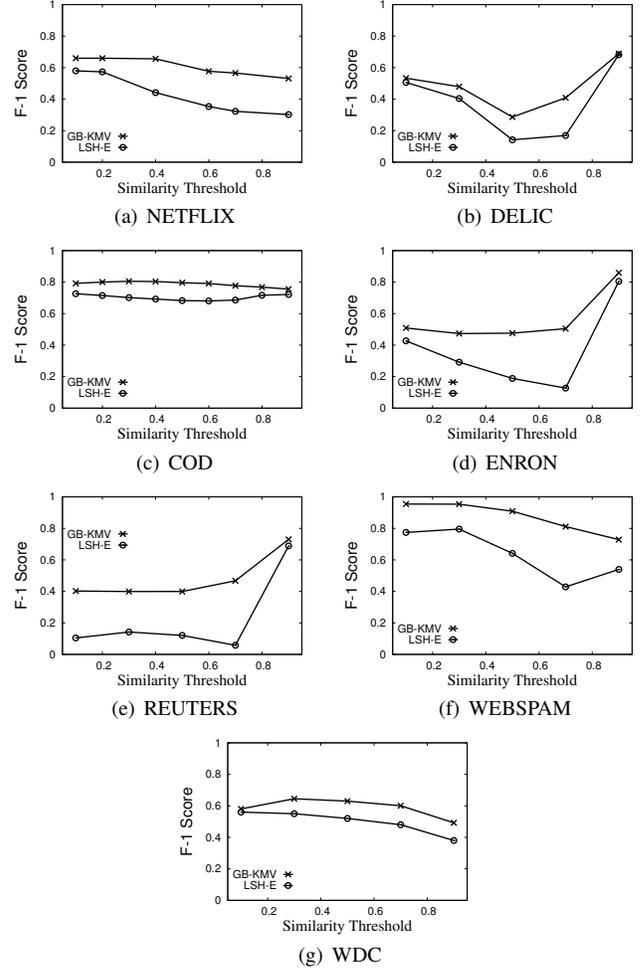


Fig. 15. Accuracy versus Similarity threshold

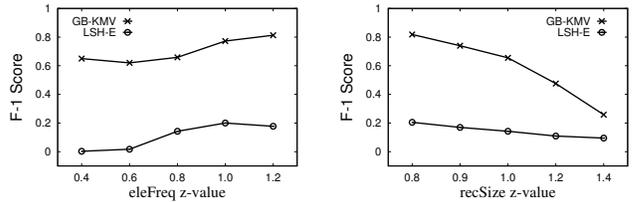


Fig. 16. EleFreq z -value varying from 0.4 to 1.2 with recSize z -value 1.0; recSize z -value varying from 0.8 to 1.4 with eleFreq z -value 0.8 Hashing(LSH) [28] techniques. For Jaccard similarity, Min-Hash [14] is used for approximate similarity search. Asymmetric minwise hashing is a technique for approximate containment similarity search [35]. This method makes use of vector transformation by padding some values into sets, which makes all sets in the index have same cardinality as the largest set. After the transformation, the near neighbours with respect to Jaccard similarity of the transformed sets are the same as near neighbours in containment similarity of the original sets. Thus, MinHash LSH can be used to index the transformed sets, such that the sets with larger containment similarity scores can be returned with higher probability. In [35], they show that asymmetric minwise hashing is advantageous in containment similarity search over datasets such as news articles and emails, while Zhu *et al.* in [44] finds that for datasets which are very skewed in set size distribution, asymmetric minwise hashing will reduce the recall.

The *KMV* sketch technique has been widely used to esti-

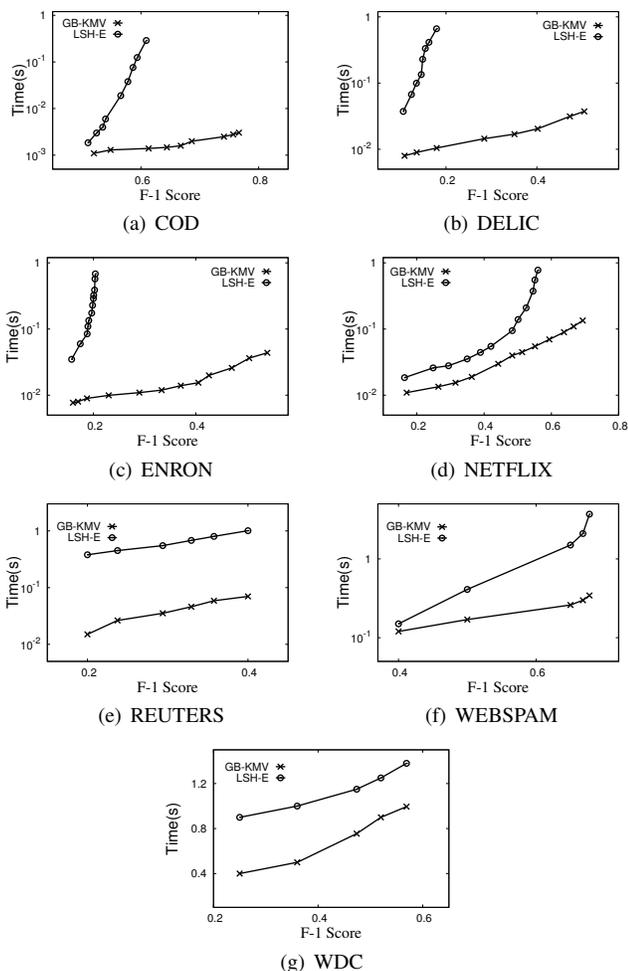


Fig. 17. Time versus Accuracy

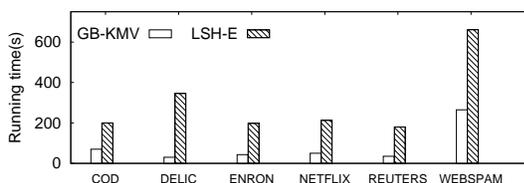


Fig. 18. Sketch Construction Time

mate the cardinality of record size [42], [20], [37]. The idea of imposing a global threshold on *KMV* sketch is first proposed in [37] in the context of term pattern size estimation. However, there is no theoretical analysis for the estimation performance. In [17], Christiani *et al.* give a data structure for approximate similarity search under Braun-Blanquet similarity which has a 1-1 mapping to Jaccard similarity if all the sizes of records are fixed. In [19], Cohen *et al.* introduce a new estimator for set intersection size, but it is still based on the MinHash technique.

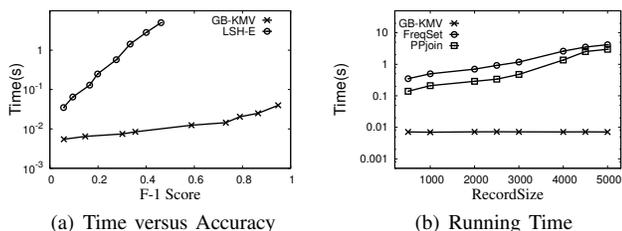


Fig. 19. Supplementary experiments

In [21], Dahlgaard *et al.* develop a new sketch method which has the alignment property and same concentration bounds as MinHash.

VII. CONCLUSION

In this paper, we study the problem of approximate containment similarity search. The existing solutions to this problem are based on the MinHash *LSH* technique. We develop an augmented *KMV* sketch technique, namely *GB-KMV*, which is data-dependent and can effectively exploit the distributions of record size and element frequency. We provide thorough theoretical analysis to justify the design of *GB-KMV*, and show that the proposed method can outperform the state-of-the-art technique in terms of space-accuracy trade-off. Extensive experiments on real-life set-valued datasets from a variety of applications demonstrate the superior performance of *GB-KMV* method compared with the state-of-the-art technique.

REFERENCES

- [1] <https://www.dropbox.com/s/wu9342zoaejs1x/ContainmentSimilaritySearch.zip?dl=0>
- [2] <http://dai-labor.de/IRML/datasets>.
- [3] <http://www.cs.cmu.edu/~enron>.
- [4] <http://trec.nist.gov/data/reuters/reuters.html>.
- [5] P. Agrawal, A. Arasu, and R. Kaushik. On indexing error-tolerant set containment. In *SIGMOD*, pages 927–938. ACM, 2010.
- [6] R. Albert. R. albert, h. jeong, and a.-l. barabási, nature (london) 401, 130 (1999). *Nature (London)*, 401:130, 1999.
- [7] A. Arasu, V. Ganti, and R. Kaushik. Efficient exact set-similarity joins. In *Proceedings of the VLDB Endowment*, pages 918–929. VLDB Endowment, 2006.
- [8] J. Bauckmann, U. Leser, and F. Naumann. Efficiently computing inclusion dependencies for schema discovery. In *Data Engineering Workshops, 2006. Proceedings. 22nd International Conference on*, pages 2–2. IEEE, 2006.
- [9] M. Bawa, T. Condie, and P. Ganesan. Lsh forest: self-tuning indexes for similarity search. In *WWW*, pages 651–660. ACM, 2005.
- [10] R. J. Bayardo, Y. Ma, and R. Srikant. Scaling up all pairs similarity search. In *WWW*, pages 131–140. ACM, 2007.
- [11] K. Beyer, P. J. Haas, B. Reinwald, Y. Sismanis, and R. Gemulla. On synopses for distinct-value estimation under multiset operations. In *SIGMOD*, pages 199–210, 2007.
- [12] P. Bouros, N. Mamoulis, S. Ge, and M. Terrovitis. Set containment join revisited. *Knowledge and Information Systems*, 49(1):375–402, 2016.
- [13] A. Z. Broder. On the resemblance and containment of documents. In *Compression and Complexity of Sequences 1997. Proceedings*, pages 21–29. IEEE, 1997.
- [14] A. Z. Broder, M. Charikar, A. M. Frieze, and M. Mitzenmacher. Min-wise independent permutations. In *STOC*, pages 327–336. ACM, 1998.
- [15] M. S. Charikar. Similarity estimation techniques from rounding algorithms. In *STOC*, 2002.
- [16] E. Cho, S. A. Myers, and J. Leskovec. Friendship and mobility: user movement in location-based social networks. In *SIGKDD*, pages 1082–1090. ACM, 2011.
- [17] T. Christiani and R. Pagh. Set similarity search beyond minhash. *arXiv preprint arXiv:1612.07710*, 2016.
- [18] A. Clauset, C. R. Shalizi, and M. E. Newman. Power-law distributions in empirical data. *SIAM review*, 51(4):661–703, 2009.
- [19] R. Cohen, L. Katzir, and A. Yehezkel. A minimal variance estimator for the cardinality of big data set intersection. In *SIGKDD*, 2017.
- [20] G. Cormode, M. Garofalakis, P. J. Haas, and C. Jermaine. Synopses for massive data: Samples, histograms, wavelets, sketches. *Foundations and Trends in Databases*, 4(1–3):1–294, 2012.
- [21] S. Dahlgaard, M. B. T. Knudsen, and M. Thorup. Fast similarity sketching. *arXiv preprint arXiv:1704.04370*, 2017.
- [22] F. De Marchi and J.-M. Petit. Zigzag: a new algorithm for mining large inclusion dependencies in databases. In *Data Mining, 2003. ICDM 2003. Third IEEE International Conference on*, pages 27–34. IEEE, 2003.

- [23] D. Deng, A. Kim, S. Madden, and M. Stonebraker. Silkmoth: An efficient method for finding related sets with maximum matching constraints. *arXiv preprint arXiv:1704.04738*, 2017.
- [24] D. Deng, G. Li, H. Wen, and J. Feng. An efficient partition based method for exact set similarity joins. *Proceedings of the VLDB Endowment*, 9(4):360–371, 2015.
- [25] D. Deng, Y. Tao, and G. Li. Overlap set similarity joins with theoretical guarantees. 2018.
- [26] A. Esmaili. Probability models in engineering and science, 2006.
- [27] M. L. Goldstein, S. A. Morris, and G. G. Yen. Problems with fitting to the power-law distribution. *The European Physical Journal B-Condensed Matter and Complex Systems*, 41(2):255–258, 2004.
- [28] P. Indyk and R. Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *STOC*, 1998.
- [29] H. Jeong, B. Tombor, R. Albert, Z. N. Oltvai, and A.-L. Barabási. The large-scale organization of metabolic networks. *Nature*, 407(6804):651, 2000.
- [30] S. Kruse, T. Papenbrock, C. Dullweber, M. Finke, M. Hegner, M. Zabel, C. Zoßlner, and F. Naumann. Fast approximate discovery of inclusion dependencies. *Datenbanksysteme für Business, Technologie und Web (BTW 2017)*, 2017.
- [31] S. Lopes, J.-M. Petit, and F. Toumani. Discovering interesting inclusion dependencies: application to logical database tuning. *Information Systems*, 27(1):1–19, 2002.
- [32] W. Mann, N. Augsten, and P. Bouros. An empirical evaluation of set similarity join techniques. *Proceedings of the VLDB Endowment*, 9(9):636–647, 2016.
- [33] T. Papenbrock, S. Kruse, J.-A. Quiané-Ruiz, and F. Naumann. Divide & conquer-based inclusion dependency discovery. *Proceedings of the VLDB Endowment*, 8(7):774–785, 2015.
- [34] A. Shrivastava and P. Li. Asymmetric lsh (alsh) for sublinear time maximum inner product search (mips). In *NIPS*, pages 2321–2329, 2014.
- [35] A. Shrivastava and P. Li. Asymmetric minwise hashing for indexing binary inner products and set containment. In *WWW*, 2015.
- [36] X. Wang, L. Qin, X. Lin, Y. Zhang, and L. Chang. Leveraging set relations in exact set similarity join. *Proceedings of the VLDB Endowment*, 10(9):925–936, 2017.
- [37] X. Wang, Y. Zhang, W. Zhang, X. Lin, and W. Wang. Selectivity estimation on streaming spatio-textual data using local correlations. *Proceedings of the VLDB Endowment*, 8(2):101–112, 2014.
- [38] S. Webb, J. Caverlee, and C. Pu. Introducing the webb spam corpus: Using email spam to identify web spam automatically. In *CEAS*, 2006.
- [39] E. W. Weisstein. Abels impossibility theorem. *From MathWorld—A Wolfram Web Resource*. <http://mathworld.wolfram.com/AbelsImpossibilityTheorem.html>.
- [40] C. Xiao, W. Wang, X. Lin, J. X. Yu, and G. Wang. Efficient similarity joins for near-duplicate detection. *TODS*, 36(3):15, 2011.
- [41] Y. Yang, Y. Zhang, W. Zhang, and Z. Huang. Gb-kmv: An augmented kmv sketch for approximate containment similarity search. <http://www.cse.unsw.edu.au/~yingz/GBKMV.pdf>, 2018.
- [42] M. Zhang, M. Hadjieleftheriou, B. C. Ooi, C. M. Procopiuc, and D. Srivastava. On multi-column foreign key discovery. *Proceedings of the VLDB Endowment*, 3(1-2):805–814, 2010.
- [43] Y. Zhang, X. Li, J. Wang, Y. Zhang, C. Xing, and X. Yuan. An efficient framework for exact set similarity search using tree structure indexes. In *ICDE*, pages 759–770. IEEE, 2017.
- [44] E. Zhu, F. Nargesian, K. Q. Pu, and R. J. Miller. Lsh ensemble: internet-scale domain search. *Proceedings of the VLDB Endowment*, 9(12):1185–1196, 2016.