On Gapped Set Intersection Size Estimation

Chen Chen Jianbin Qin Wei Wang
University of New South Wales, Australia
{cchen, jqin, weiw}@cse.unsw.edu.au

ABSTRACT

There exists considerable literature on estimating the cardinality of set intersection result. In this paper, we consider a generalized problem for integer sets where, given a gap parameter δ, two elements are deemed as matches if their numeric difference equals δ or is within δ. We call this problem the *gapped set intersection size estimation* (GSISE), and it can be used to model applications in database systems, data mining, and information retrieval. We first distinguish two subtypes of the estimation problem: the *point gap* estimation and *range gap* estimation. We propose optimized sketches to tackle the two problems efficiently and effectively with theoretical guarantees. We demonstrate the usage of our proposed techniques in mining top-K related keywords efficiently, by integrating with an inverted index. Finally, substantial experiments based on a large subset of the ClueWed09 dataset demonstrate the efficiency and effectiveness of the proposed methods.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval

Keywords

Indexing; Set Intersection Size Estimation; Top-k

1. INTRODUCTION

Set intersection is a fundamental operation in many fields in computer science. Given two sets S_A and S_B, set intersection $S_A \cap S_B$ is to find all the common elements from two sets. A common case is that all elements in the set are integers (e.g., document IDs or positions in an inverted index). Hence, the common element pair (a, b) satisfies $b - a = \delta$, where $a \in S_A$, $b \in S_B$ and $\delta = 0$.

In this paper, we generalize the set intersection on integer sets to allow for "gaps" (i.e., $\delta > 0$). We define two primitives: the point gap constraint corresponds to a fixed gap of δ, and the range gap constraint corresponds to a gap of size no larger than δ. We are interested in methods to estimate these gapped set intersection size efficiently and accurately. This problem has many applications. For example,

- *Information Retrieval*: In information retrieval, the search engine needs to intersect the positional inverted lists of query keywords to answer a multiple keyword phrase query. A state-of-the-art query processing method, *svs*, performs binary intersection using a heuristic order that is purely based on the length of the inverted lists [11]. This heuristic is not effective when search keywords are not very selective (e.g., "`to be or not to be`"). It is possible that a pair of query keywords with the positional constraint imposed by the phrase query will result in very small intermediate result size (e.g., "`be`" followed immediately by "`or`"). Hence, if we can estimate its cardinality accurately and efficiently, we may find a better inverted list intersection order to process such queries. We can model this as a gapped set intersection size estimation problem with a *point gap constraint* of 1.
- *Sentiment or Event Analyses*: In sentiment analysis, we extract different types of events, including a mention of a product, and an occurrence of sentiment (e.g., the word "`fantastic`"). We are usually interested in events that occur in a close vicinity. For instance, the *positions* of their occurrences in a document are within δ [17] or the timestamps of their occurrences in tweets are within δ [25]. We can model this as a gapped set intersection size estimation problem with a *range gap constraint* of δ.

Motivated by the examples, we define and study the problem of gapped set intersection size estimation (abbreviated as GSISE). For the estimation problem with point gap constraints, we propose a basic method that reduces the problem to the standard set intersection size estimation problem, which can be solved using the state-of-the-art sketch method. However, the index space is linear to the maximum query gap allowed. We improve it by judiciously selecting a subset of sketches to construct, and this reduces our sketch size from $O(N)$ to $O(\sqrt{N})$. The space cost of our approach is proved to be asymptotically optimal, while the time complexity remains the same. For the estimation problem with the range gap constraint, a baseline method is to reduce the problem into multiple estimation problems with different point gap constraints, and this requires estimation time linear in the gap size. We propose an extension of the bottom-k sketch to multiset and an unbiased estimator for inner product, we achieve an accurate and fast estimation method that is independent of the gap size. To demonstrate the use of our estimation methods, we consider the problem of finding highly correlated keywords from a large document collection. We design a new query processing method based

CIKM'15, October 19–23, 2015, Melbourne, Australia.

DOI: http://dx.doi.org/10.1145/2806416.2806438.

on indexing the hash values in our sketches. Finally, we perform large-scale experimental evaluation using 500 million documents from the ClueWeb09 data collection and demonstrate the accuracy and efficiency of our proposed methods.

Contributions. The contributions of this paper are summarized as follows.

- To the best of our knowledge, this is the first work to formally define the point and range gapped set intersection size estimation problems, which can be used as primitive operations in a wide spectrum of applications.
- We design space and time efficient sketch and estimation methods for both types of estimation tasks. Our estimates are unbiased and have theoretical guarantees.
- We demonstrate the application of our technique for approximately mining top-K related keywords from large document collections. Our technique is especially useful in this application scenario where an exact solution requires orders of magnitudes larger space and time.
- Comprehensive experiments on the ClueWed09 dataset demonstrate the efficiency and accuracy of the proposed methods.

Organization of the paper. The rest of the paper is organized as follows. Section 2 briefly surveys the related works. Section 3 defines the problem formally and introduces several useful techniques. Section 4 elaborates the estimation framework for point query and range query, together with their theoretical properties. Section 5 presents algorithm to top-K related keywords mining application. The experimental results are reported and analyzed in Section 6. We conclude the paper in Section 7.

2. RELATED WORKS

Exact set intersection. Set intersection has long been a fundamental problem. There are plenty of literature focusing on set intersection problem [12, 13]. The algorithms working on sorted set are mostly used [24]. By applying linear merge of two sorted set S_A and S_B, the set intersection can be finished in $O(|S_A| + |S_B|)$ time. However, when the set sizes are unbalanced, the method is inefficient. In [18], authors propose a set intersection approach requiring $O(|S_B| + \log \binom{|S_A|+|S_B|}{|S_A|})$ when $|S_A| \ll |S_B|$. There are also works further improving the performance by utilizing hashing, and adaptive methods [3, 2]. Unlike the above literature, we want to *estimate* the gapped set intersection size rather than calculating the exact result.

Set intersection size estimation. The problem of set intersection size estimation is to estimate the cardinality of intersection with a sketch method, which has been extensively studied in [6, 4, 7]. Nevertheless, all existing methods are designed for non-gapped set intersection size estimation. A naïve way to extend existing methods to support the GSISE problem is to build sketch for each possible query gap. However, it will result in the index space linear to the maximum gap. Furthermore, the computational complexity for range query will be linear to the query gap. In this paper, we try to reduce both space and computational complexity in processing GSISE. In addition, there are several works [26, 21] that calculate an upper bound of set intersection size, which is orthogonal to our problem.

Top-K query. One of the most important applications of GSISE is top-K related keyword mining. The problem is well studied in [16, 15]. Afterwards, there are varieties of follow-up works dealing with top-K ranking for different applications, e.g., keyword queries [20], top-K preference ranking queries [23] and semi-structure queries [27].

Most of the works, such as [28], are based on the strategy that gradually scans the inverted index and updates the upper and lower bounds of the ranking terms, in order to achieve early stopping. However, the above strategy is not well suited in our sketch based estimation framework. Since it is non-trivial to obtain a tight bound for the estimation. Instead, a hash table based method is proposed in Section 5 to accelerate the top-K query processing.

3. PRELIMINARIES

In this section, we define two estimation problems, introduce existing work on estimating the size of set intersection using sketch, and finally list important notations used in the paper.

3.1 Problem Definition

Given two sets S_A and S_B, their intersection is defined as $\{ (a,b) \mid \texttt{eq}(\texttt{a},\texttt{b}), a \in S_A, b \in S_B \}$, where the predicate $\texttt{eq}(a,b)$ checks if a equals b. We can generalize the set intersection by considering other meaningful predicates. Specifically, with a *gap parameter* $\delta \geq 0$, we consider the following two predicates:

- $\texttt{PointGap}_\delta(a,b)$, which returns **true** if and only if $b - a = \delta$.
- $\texttt{RangeGap}_\delta(a,b)$, which returns **true** if and only if $0 \leq b - a \leq \delta$.

We call set intersection with these two new predicates collectively *gapped set intersection*. They are denoted as $S_A \cap^{=\delta} S_B$ (named *point gapped set intersection*) and $S_A \cap^{\leq\delta} S_B$ (named *range gapped set intersection*), respectively. Obviously, the standard set intersection is a special case of both types of gapped set intersection where $\delta = 0$.

In this paper, we study space and time-efficient methods to estimate the results size of these two types of gapped set intersection. Motivated by the applications we aim at, we consider sets whose elements are integers.

DEFINITION 1 (GSISE). *Given two sets S_A and S_B, and a gap parameter $\delta \in [0, N]$ where N is a predefined maximum gap value. The* Point *and* Range *Gapped Set Intersection Size Estimation Problem is to estimate $|S_A \cap^{=\delta} S_B|$ and $|S_A \cap^{\leq\delta} S_B|$, respectively. They are abbreviated as* point estimation *and* range estimation *hereafter.*

EXAMPLE 1. *Let $S_A = \{1, 3, 4, 7\}$ and $S_B = \{2, 5, 6, 8\}$, and $\delta = 2$. Under the point gap constraint, $S_A \cap^{=2} S_B = \{(3,5), (4,6)\}$. Hence its size is* 2. *Under the range gap constraint, $S_A \cap^{\leq 2} S_B = \{(1,2), (3,5), (4,5), (4,6), (7,8)\}$. Hence its size is* 5.

Discussions. In the above definitions, we only need to consider $\delta \geq 0$. This is because $S_A \cap^{=\delta} S_B = S_B \cap^{=-\delta} S_A$ (this also holds for range gapped set intersection too).

Many other types of interesting gapped set intersections can be defined using our point and range gapped set intersection as primitives. For example, consider the predicate $\texttt{RangeWithin}_{\delta_1,\delta_2}(\texttt{a},\texttt{b})$ with two range parameters $0 \leq \delta_1 < \delta_2$, which checks if $\delta_1 \leq a - b \leq \delta_2$. It is easy to see that its query result is exactly the difference of two range gapped set intersection, i.e., $(S_A \cap^{\leq\delta_2} S_B) \setminus (S_A \cap^{\leq\delta_1} S_B)$.

In a similar fashion, we can derive point gapped set intersection based on range gapped set intersection, and

vice versa. Nevertheless, we still consider them as two separate primitives, as each of them can model different applications respectively, and we will propose related but different estimation methods and optimizations for them.

3.2 Bottom-k Sketch

The state-of-the-art method to estimate the size of set intersections is the bottom-k sketch [7, 8, 9]. It is a lightweight sketch that supports efficient update.

We use $sk(S_A)$ to denote the bottom-k sketch for a set S_A, which is a set of k hash values. To construct the sketch, we apply a random hash function h to every element $a \in S_A$, and keep the k minimum hash values. We also assume the codomain of h is sufficiently large such that we can safely assume that there is no collision.

An important property of the bottom-k sketch is that it is closed under set union operation. Specifically, we can directly compute the bottom-k sketch of the union of two sets from their respective bottom-k sketches. The resulting sketch is called *short combination sketch* [9]: $scs(sk(S_A), sk(S_B)) = \{ v \mid v \in sk(S_A) \cup sk(S_B), v < \min(sk(S_A)^{(k)}, sk(S_B)^{(k)}) \}$, where the notation $S^{(k)}$ denotes the k-th smallest value in the set S.

To estimate the intersection size of two sets S_A and S_B from their respective sketches, we compute

$$t = \frac{|sk(S_A) \cap sk(S_B) \cap scs(sk(S_A), sk(S_B))|}{\min\{ sk(S_A)^{(k)}, sk(S_B)^{(k)} \}}. \quad (1)$$

[9] proved that t is an unbiased estimator of $|S_A \cap S_B|$ that can be computed efficiently. Besides, it is shown to have smaller variance compared with the Minhash [5] method.

3.3 Notations

Table 1 lists notations frequently used in the paper.

Table 1: Summary of Notations

Symbol	Explanation
S_{id}	a set of integers
M_{id}	a multiset; each element has its multiplicity
S_{id}^{+d}	a shifted set by distance d
$\{d_1, d_2, \ldots\}$	the index of a multiset generated by merging the set shifted by d_1, d_2, ...
$S^{(i)}$ or $M^{(i)}$	i-th smallest value in the set S or the multiset M
N	the maximum gap
δ	the gap parameter; $\delta \in [0, N]$
k	number of hash values in a bottom-k sketch
$sk(S)$	bottom-k sketch of the set S
$msk(M)$	multiset bottom-k sketch of the multiset M
rus	range union sketch of $msk(M_A)$ and $msk(M_B)$

4. ESTIMATION METHODS FOR GAPPED SET INTERSECTION SIZE

In this section, we introduce solutions to point and range estimation problems. The naïve index structure and our intuition are explained in Section 4.1, followed by technique details of GSISE methods in Section 4.2 and 4.3.

4.1 A Baseline Method for Point Estimation

Point estimation is a challenging problem due to the following reasons:

- Almost all the set intersection size estimation methods are based on random hash functions. Therefore, given a gap value δ, and two *different* hash values $h(x+\delta)$ and $h(y)$, it is almost impossible to infer if $x+\delta$ is equal to y.
- While locality sensitive hash functions [19] do preserve locality probabilistically, the intersection size is highly sensitive to the point gap threshold. It is often the case that $|S_A \cap^{=\delta} S_B|$ and $|S_A \cap^{=\delta+1} S_B|$ differ substantially. For instance, in Example 1, $|S_A \cap^{=0} S_B|$ is 0 while $|S_A \cap^{=1} S_B|$ is 3. Therefore, even a small approximation in the gap may result in a large estimation error.

Therefore, we first propose a baseline method, which reduces the point estimation problem to a standard set intersection size estimation problem, which in turn can be solved using the state-of-the-art method, such as the method based on bottom-k sketches.

Our reduction is based on the notion of *shifted sets*. Given a set $S_A = \{ a_1, a_2, \ldots, a_n \}$ and integer parameter d, we define the shifted set with shift d as follows: $S_A^{+d} \coloneqq \{ a_i + d \mid a_i \in S_A \}$.

Given the maximum gap size N, we compute $N+1$ bottom-k sketches by shifting S_A: the i-th bottom-k sketch is built for the shifted set S_A^{+i} $(0 \le i \le N)$. To perform the estimation for $S_A \cap^{=\delta} S_B$, we retrieve the sketches of $S_A^{+\delta}$ and S_B^{+0}, and perform the estimation using the bottom-k estimation procedure (i.e., Eq. (1)).

Table 2: The Random Hash Function h

x	0	1	2	3	4	5	6	7	8
$h(x)$	0.47	0.26	0.32	0.84	0.74	0.79	0.22	0.42	0.95
x	9	10	11	12	13	14	15	16	17
$h(x)$	0.48	0.68	0.89	0.16	0.63	0.37	0.53	0.15	0.21

EXAMPLE 2. *Table 3 shows the sketches built by the baseline method for two sets S_A and S_B, where the bottom-k sketch size is 3 and maximum gap N is 9.*

To perform the point estimate for $\delta = 5$, we load $sk(S_A^{+5})$ and $sk(S_B^{+0})$, the estimate according to Eq. (1) is $1/\min(0.48, 0.79) = 2.08$. The actual point gapped intersection size is 2.

Table 3: $N+1$ bottom-k Sketches Built for $S_A = \{1,3,4,7\}$ and $S_B = \{2,5,6,8\}$ using the Hash Function in Table 2. $N = 9$.

shift	Sketches for S_A	Sketches for S_B
0	$sk(S_A^{+0}) = \{0.26, 0.42, 0.74\}$	$sk(S_B^{+0}) = \{0.22, 0.32, 0.79\}$
⋮	⋮	⋮
5	$sk(S_A^{+5}) = \{0.16, 0.22, 0.48\}$	$sk(S_B^{+5}) = \{0.42, 0.63, 0.68\}$
⋮	⋮	⋮
9	$sk(S_A^{+9}) = \{0.15, 0.16, 0.63\}$	$sk(S_B^{+9}) = \{0.21, 0.37, 0.53\}$

Obviously, this baseline method achieves the same accuracy guarantee as the standard bottom-k sketch [4, 7, 22], and the estimation time is $O(k)$. The main problem is its space complexity of $O(N \cdot k)$ for each set, which is not optimal.

4.2 Improved Point Query Estimation

In this subsection, we seek to improve the space complexity per set from $N \cdot k$ to $\sqrt{N} \cdot k$, while still maintaining the same $O(k)$ estimation time. We also show that this space

complexity is asymptotically optimal with an approximation ratio of $\sqrt{2}$.

We observe that we can generate sketches for a *judiciously chosen subset* of all possible shifted sets, to ensure that can still find two appropriate sketches for two sets to perform the point estimation for any $\delta \in [0, N]$.

Let $\lambda \coloneqq \lceil \sqrt{N} \rceil$. For a set S_A, we generate 2λ sketches for S_A shifted by $i \in I$, where

$$I = \{ 0, 1, 2, \cdots, \lambda - 1 \} \cup \{ i \cdot \lambda \mid i \in [1, \lambda] \} \tag{2}$$

Algorithm 1 describes the estimation procedure. Lines 2–5 compute the correct offsets (also called indices) of the sketches of S_A and S_B. Then Line 6 performs the estimation with the corresponding bottom-k sketches.

Algorithm 1: PointQueryOnlineEstimation(S_A, S_B, δ)

Input : δ: query gap. N: maximum gap.
Output: An estimate of $|S_A \cap^{=\delta} S_B|$

1 $\lambda \leftarrow \lceil \sqrt{N} \rceil$;
2 **if** $\delta \pmod{\lambda} = 0$ **and** $\delta \neq N$ **then**
3 $\quad i_A \leftarrow \lambda + \delta$; $\quad i_B \leftarrow \lambda$;
4 **else**
5 $\quad i_A \leftarrow \lceil \delta / \lambda \rceil \cdot \lambda$; $\quad i_B \leftarrow \lceil \delta / \lambda \rceil \cdot \lambda - \delta$;
6 **return** Bottom-k-Estimate $(sk(S_A^{+i_A}), sk(S_B^{+i_B}))$;

EXAMPLE 3. *Continue the previous example.* $\lambda = \lceil \sqrt{N} \rceil = 3$. *In our improved method, the following sketches are generated for every set* S

$$sk(S_i^{+0}), sk(S_i^{+1}), sk(S_i^{+2}), sk(S_i^{+3}), sk(S_i^{+6}), sk(S_i^{+9}).$$

For point estimation with $\delta = 5$, *we compute* $i_A = 6$ *and* $i_B = 1$ *according to Algorithm 1. Then we load* $sk(S_A^{+6})$ *and* $sk(S_B^{+1})$ *and perform the estimation.*

The correctness of Algorithm 1 depends on two facts:

- Gapped set intersection size is shift-invariant as shown in Lemma 1, and
- $\forall \delta \in [0, N]$, we can always locate the i_A and i_B from the index set (Equation (2)) such that $i_A - i_B = \delta$.

LEMMA 1. $\forall d$, $|S_A \cap^{=\delta} S_B| = |S_A^{+d} \cap^{=\delta} S_B^{+d}|$.

The improved method has a $O(\sqrt{N} \cdot k)$ space complexity per set, $O(k)$ estimation time, and the same estimation quality guarantees as the bottom-k sketch.

4.2.1 Asymptotic Space Optimality and Approximation Ratio

We show that our improved method is asymptotically space optimal in this reduction framework. To facilitate the analysis, a computational model is formalized below in Definition 2.

DEFINITION 2. *Given set* $G = \{ 0, 1, 2, \cdots, N \}$, *where* N *is the maximum gap predefined. We want to find an integer set* P *with minimum cardinality, such that* $\forall g \in G$, *there exist* $i, j \in P$ *satisfying* $i - j = g$.

PROPERTY 1. *The lower bound of* $|P|$ *is* $\Omega(\sqrt{N})$.

PROOF. First there are no duplicate elements in P, otherwise $|P|$ cannot reach the minimum since it is allowed to choose two identical elements from P to get $0 \in G$. Now that all elements in P are different, we can construct a mapping f from $i - j$ to $G \setminus \{ 0 \}$. It is easy to see the lower bound will be achieved when the mapping is bijective. Hence, the number of all possible choices of $i - j$ is $\binom{|P|}{2}$. Since $\binom{|P|}{2} \geq N$, we have $|P| = \Omega(\sqrt{N})$. □

COROLLARY 1. *The number of sketches constructed in our improved method is within a factor of* $\sqrt{2}$ *of the optimal solution.*

4.3 Range Estimation

4.3.1 Basic Method for Range Estimation

Our basic method for range estimation is to reduce a range estimation to multiple point estimations, due to the following equivalence.

THEOREM 1. $S_A \cap^{\leq \delta} S_B = \bigcup_{i=0}^{\delta} \left(S_A \cap^{=i} S_B \right)$. *In addition,* $\forall i \neq j, (\{ S_A \cap^{=i} S_B \}) \cap (\{ S_A \cap^{=j} S_B \}) = \emptyset$.

Theorem 1 reveals that the range gapped set intersection results can be partitioned into disjoint subsets, each is the result of a point gapped set intersection. Taking the cardinality on both sides of the equation and we have the following Corollary.

COROLLARY 2. $|S_A \cap^{\leq \delta} S_B| = \sum_{i=0}^{\delta} |S_A \cap^{=i} S_B|$.

Corollary 2 enables us to sum up $\delta + 1$ point estimation results to answer a range estimation. While this does not increase the space complexity, the estimation time is linear in the range δ. When δ is large, the estimation time grows quickly, which is not desirable.

4.3.2 Merge of Shifted Sets

We introduce several essential concepts, which enable us to present an observation using an example. This relates the range estimation to the problem of estimating the inner product of multisets, each obtained by merging shifted sets in a particular pattern.

We define a multiset M as a set of elements, each associated with its multiplicity, i.e., $M = \{ a_1 : m_1, a_2 : m_2, \ldots, a_n : m_n \}$. $\mathsf{elems}(M)$ returns all the elements in the multiset M, i.e., $\{ a_1, a_2, \ldots, a_n \}$. The multiplicity of an element e in M is denoted as $\mathsf{cnt}_M(e)$. Note that a set is just a special case of a multiset where all the multiplicities equal 1.

Let U be the universe of all elements. Each multiset can be implicitly cast into a $|U|$-dimensional vector where the dimension values are the multiplicities of the corresponding elements (default to 0). Hence, we can define the *inner product* of two multisets as

$$\langle M_A, M_B \rangle = \sum_{e \in \mathsf{elems}(M_A) \cap \mathsf{elems}(M_B)} \mathsf{cnt}_{M_A}(e) \cdot \mathsf{cnt}_{M_B}(e).$$

Now, we can illustrate an important observation that leads to our improved range estimation in Example 4.

EXAMPLE 4. *Consider the same instance in Example 3 and we want to estimate* $S_A \cap^{\leq \delta} S_B$. *Table 4(a) enumerates for all possible* $\delta \in [0, 8]$ *the shifted* S_A *and* S_B *that will be used for set intersection with point gap* δ. *For example, the*

Table 4: Illustration of Shift Sets Used for Point/Range Estimation for $\delta \in [0, 8]$

(a) Shifted Sets (Illustrating $\delta = 5$)

S_A^{+9}	8	7	6
S_A^{+6}	5	4	3
S_A^{+3}	2	1	0
	S_B^{+1}	S_B^{+2}	S_B^{+3}

(b) Merged Shifted Sets (Illustrating $\delta = 7$)

$\uplus_{i\in\{3,6,9\}} S_A^{+i}$	8	7	6	S_A^{+9}
$\uplus_{i\in\{3,6\}} S_A^{+i}$	5	4	3	S_A^{+6}
$\uplus_{i\in\{3\}} S_A^{+i}$	2	1	0	S_A^{+3}
	$\uplus_{i\in\{1,2,3\}} S_B^{+i}$	$\uplus_{i\in\{2,3\}} S_B^{+i}$	$\uplus_{i\in\{3\}} S_B^{+i}$	

cell with green background is for $\delta = 5$; it is associated with S_A^{+6} and S_B^{+1}. This means $|S_A \cap^{=5} S_B| = |S_A^{+6} \cap S_B^{+1}|$.

Now consider the range gapped set intersection with $\delta = 5$. Corollary 2 shows that the result size $|S_A \cap^{\leq 5} S_B|$ equals the sum of the size of 6 point gapped set intersections, with the gap constraint between 0 and 5. By looking at Table 4(a), we can see the latter is equivalent to $\langle M_A, M_B \rangle$, where

$$M_A = (S_A^{+3} \uplus S_A^{+6}), \qquad M_B = (S_B^{+1} \uplus S_B^{+2} \uplus S_B^{+3})$$

*Note that we need to use multiset union (also called **merge** in this paper, denoted as $\uplus$) and the inner product, as there may be potential duplicate elements. Obviously, if we can estimate the inner product of multisets, then we can perform one estimation rather than six point estimations.*

By considering all possible δ values in Example 4, we can observe the pattern where multiple shifted sets are merged. To simplify the notation, we use the set of shift values as an identifier (or index) for the merged multiset, i.e., if $M = S_A^{+i_1} \uplus S_A^{+i_2} \ldots \uplus S_A^{+i_j}$, then we say M's **index** is $\{ i_1, i_2, \ldots, i_j \}$ and it uniquely identifies M.

Let $\lambda = \lceil \sqrt{N} \rceil$, and $i \geq 1$.

- For shift values within $[1, \lambda]$, we need to merge its suffixes, i.e., generating indices $\{ i, i+1, \ldots, \lambda \}$, for $1 \leq i < \lambda$.
- For shift values within $[\lambda, \lambda^2]$, we need to merge its prefixes, i.e., generating indices $\{ \lambda, 2\lambda, \ldots, i \cdot \lambda \}$, for $1 < i \leq \lambda$.

4.3.3 Estimating the Inner Product of Two Multisets

As motivated in Example 4, we need to estimate the inner product of two multisets. While there are many alternative methods (such as Tug-of-War [1] and Count-Min [10] sketches), we observe that the input multisets are always those shifted sets generated for the point estimation task, which means each of them has already its bottom-k sketch built or maintained. Therefore, we develop an estimator by *extending the bottom-k sketch* as follows.

Firstly, we define our **multiset bottom-k sketch** for a multiset obtained by merging multiple shifted sets, each with its own bottom-k sketch. Let $\mathcal{S} = \{ S_1, S_2, \ldots, S_n \}$ denote a set of shifted sets, each with its bottom-k sketch $sk(S_i)$. Let $M = \uplus_{i=1}^{n} S_i$. M's multiset bottom-k sketch, denoted by $msk(M)$, is obtained by a *truncated merge* of the sketches, i.e.,

1. first merging $sk(S_i)$ into a multiset M', and
2. then keeping only the k smallest elements and their multiplicities in M'.

Figure 1 illustrates the relationship between a multiset bottom-k sketch and its constituent bottom-k sketches.

Given two multiset bottom-k sketches $msk(M_A)$ and $msk(M_B)$, their *range union sketch*, denoted by $rus(msk(M_A), msk(M_B))$, is a multiset that contains all the hash values in $\mathsf{elems}(msk(M_A)) \uplus \mathsf{elems}(msk(M_B))$ that are smaller than $\min(msk(M_A)^{(k)}, msk(M_B)^{(k)})$, as well as the sum of their multiplicities in $msk(M_A)$ and $msk(M_B)$. Given

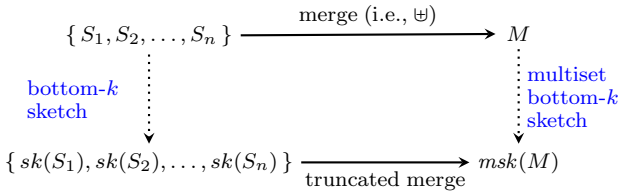


Figure 1: From bottom-k Sketches to a Multiset bottom-k Sketch

$msk(M_A)$ and $msk(M_B)$, we propose an unbiased estimator of $\langle M_A, M_B \rangle$, as shown in Theorem 2. Furthermore, it can be shown that, by using the range union sketch, our method takes advantage of all the information available in $msk(M_A)$ and $msk(M_B)$ to arrive at the best possible estimation.

THEOREM 2. *Given two multiset bottom-k sketches $msk(M_A)$ and $msk(M_B)$, let $rus_\cap = \mathsf{elems}(msk(M_A)) \cap \mathsf{elems}(msk(M_B)) \cap \mathsf{elems}(rus(msk(M_A), msk(M_B)))$, then*

$$\hat{t}_r = \frac{\sum_{e \in rus_\cap} \left(\mathsf{cnt}_{msk(M_A)}(e) \cdot \mathsf{cnt}_{msk(M_B)}(e) \right)}{\min\{ msk(M_A)^{(k)}, msk(M_B)^{(k)} \}}. \quad (3)$$

is an unbiased estimator of $\langle M_A, M_B \rangle$.

PROOF. Define *adjusted multiplicity* for each $e \in \mathsf{elems}(M_A) \cap \mathsf{elems}(M_B)$ to be:

$$a_e = \begin{cases} \frac{\mathsf{cnt}_{msk(M_A)}(e) \cdot \mathsf{cnt}_{msk(M_B)}(e)}{rus^{(l)}} & \text{, if } e \text{ is sampled in } rus_\cap \\ 0 & \text{, otherwise.} \end{cases} \quad (4)$$

where $rus^{(l)} = \min\{ msk(M_A)^{(k)}, msk(M_B)^{(k)} \}$, i.e., it is the l-th smallest hash values in rus. Then we can write $\hat{t}_r$ as $\sum_{e \in rus_\cap} \frac{\mathsf{cnt}_{msk(M_A)}(e) \cdot \mathsf{cnt}_{msk(M_B)}(e)}{rus^{(l)}}$. The expectation is

$$\begin{aligned} \mathbf{E}\left[\hat{t}_r\right] &= \mathbf{E}\left[\sum_{e \in rus_\cap} \frac{\mathsf{cnt}_{msk(M_A)}(e) \cdot \mathsf{cnt}_{msk(M_B)}(e)}{rus^{(l)}} \right] \\ &= \sum_{e \in \mathsf{elems}(M_A) \cap \mathsf{elems}(M_B)} \mathbf{E}\left[a_e\right] \\ &= \sum_{e \in \mathsf{elems}(M_A) \cap \mathsf{elems}(M_B)} \mathsf{cnt}_{msk(M_A)}(e) \cdot \mathsf{cnt}_{msk(M_B)}(e) \\ &= \sum_{e \in \mathsf{elems}(M_A) \cap \mathsf{elems}(M_B)} \mathsf{cnt}_{M_A}(e) \cdot \mathsf{cnt}_{M_B}(e). \end{aligned}$$

The last step is because rus keeps *complete* multiplicity information for each underlying set element included in rus. *Complete* means if element e is sampled in $rus_\cap$, it holds that $\mathsf{cnt}_{msk(M)}(e) = \mathsf{cnt}_M(e)$. Since we are doing consistent uniform sampling in $\mathsf{elems}(M_A) \cup \mathsf{elems}(M_B)$. Therefore, $\mathbf{E}\left[\hat{t}_r\right] = \langle M_A, M_B \rangle$. □

Furthermore, by setting k to an appropriate value, we can achieve a probabilistic guarantee for $\hat{t}_r$, as shown in Theorem 3.

THEOREM 3. *Let $\mu = \langle M_A, M_B \rangle$. For any given ϵ and ρ, by setting $k = \min\{\max\{k_1, k_2\}, \max\{|\mathsf{elems}(M_A)|, |\mathsf{elems}(M_B)|\}\}$, where k_1 satisfies Eq. (5) and $k_2 = \frac{(|\mathsf{elems}(M_A)|+2)N}{\mu} \cdot \frac{2+\epsilon}{\epsilon^2} \ln \frac{4}{\rho}$, we can guarantee that* $\mathbf{Pr}\left[|\hat{t}_r - \mu| \leq \epsilon\mu\right] \geq 1 - \rho$.

PROOF. We first define two propositions (a) and (b). Let $X_i = a_i \cdot \frac{k}{(|\mathsf{elems}(M_A)|+2)N}$ and e_i be the corresponding element, a_i as defined in Eq. (4), and k_1 and k_2 as defined in the Theorem.

- Proposition (a): When $k \geq k_1$, $\mathbf{Pr}\left[X_i > 1\right] \leq \frac{\rho}{2}$.
- Proposition (b): When $k \geq k_2$, $\mathbf{Pr}\left[|\hat{t}_r - \mu| \geq \epsilon\mu\right] \leq \frac{\rho}{2}$.

If both of them are proved, then when $k \geq \max\{k_1, k_2\}$, we know that $\mathbf{Pr}\left[|\hat{t}_r - \mu| \geq \epsilon\mu\right] \leq \rho$ based on the Union Bound.

Proof of Proposition (a). Without loss of generality, we assume $|\mathsf{elems}(M_A)| > |\mathsf{elems}(M_B)|$. Given ρ, let k_1 be solution to Eq. (5):

$$\int_0^{\frac{k}{|\mathsf{elems}(M_A)|+2}} \frac{t^{k-1}(1-t)^{|\mathsf{elems}(M_A) \cup \mathsf{elems}(M_B)|-k}}{B(k, |\mathsf{elems}(M_A) \cup \mathsf{elems}(M_B)| - k + 1)} dt \leq \frac{\rho}{2}, \tag{5}$$

where $B(k, |\mathsf{elems}(M_A) \cup \mathsf{elems}(M_B)| - k + 1)$ is the Beta function. It also holds:

$$\begin{aligned} &\mathbf{Pr}\left[X_i > 1\right] \\ &= \mathbf{Pr}\left[\frac{\mathsf{cnt}_{msk(M_A)}(e_i) \cdot \mathsf{cnt}_{msk(M_B)}(e_i)}{msk(M_A)^{(k)}} > \frac{(|\mathsf{elems}(M_A)|+2)N}{k}\right] \\ &\leq \mathbf{Pr}\left[msk(M_A) < \frac{k}{|\mathsf{elems}(M_A)|+2}\right]. \end{aligned} \tag{6}$$

The last step is because $\mathsf{cnt}_{msk(M_A)}(e_i) \cdot \mathsf{cnt}_{msk(M_B)}(e_i) \leq N$. Besides, when $k > k_1$, Eq. (5) is equivalent to:

$$\mathbf{Pr}\left[msk(M_A)^{(k)} < \frac{k}{|\mathsf{elems}(M_A)|+2}\right] < \frac{\rho}{2}. \tag{7}$$

Since we are doing uniform random sample in the space of $\mathsf{elems}(M_A) \cup \mathsf{elems}(M_B)$, where $msk(M_A)^{(k)}$ is the k-th order statistics therein.

Thus from Eq. (6) and (7), we know when $k > k_1$, $\mathbf{Pr}\left[X_i > 1\right] \leq \frac{\rho}{2}$.

Proof of Proposition (b). Now that we can guarantee $X_i \in [0, 1]$ almost surely by proving Proposition (a), we can define $X = \sum X_i$ and apply the Chernoff bound to X.[1]

Let $k_2 = \frac{(|\mathsf{elems}(M_A)|+2)N}{\mu} \cdot \frac{2+\epsilon}{\epsilon^2} \ln \frac{4}{\rho}$. When $k > k_2$, we have

$$2\exp\left(-\frac{\epsilon^2}{2+\epsilon} \frac{\mu k}{(|\mathsf{elems}(M_A)|+2)N}\right) \leq \frac{\rho}{2}.$$

According to the Chernoff bound, we have

$$\begin{aligned} &\mathbf{Pr}\left[|\hat{t}_r - \mu| \geq \epsilon\mu\right] \\ &\leq 2\exp\left(-\frac{\epsilon^2}{2+\epsilon} \cdot \frac{\mu k}{(|\mathsf{elems}(M_A)|+2)N}\right) \leq \frac{\rho}{2}. \end{aligned}$$

Then from Union Bound, it guarantees that when $k \geq \max(k_1, k_2)$, $\mathbf{Pr}\left[|\hat{t}_r - \mu| \geq \epsilon\mu\right] \leq \delta$. □

[1] Since bottom-k belongs to sampling without replacement strategy, it will cause negative associations between each sample value [14]. Nevertheless, according to [14], Chernoff bounds are still applicable to sums of random variables with negative associations.

4.3.4 Improved Range Estimator

In our improved methods, given a maximum gap N, let $\lambda = \lceil\sqrt{N}\rceil$. We generate three categories of sketches:

- Category I: $\lambda + 1$ *bottom-k sketches* for shifts $i \in \{\, j\lambda \mid 0 \leq j \leq \lambda \,\}$
- Category II: $\lambda - 1$ *multiset bottom-k sketches* with indices $\{\, \lambda, 2\lambda \,\}, \{\, \lambda, 2\lambda, 3\lambda \,\}, \cdots, \{\, \lambda, 2\lambda, \ldots, \lambda^2 \,\}$.
- Category III: $\lambda - 1$ *multiset bottom-k sketches* with indices $\{\, \lambda, \lambda - 1 \,\}, \{\, \lambda, \lambda - 1, \lambda - 2 \,\}, \cdots, \{\, \lambda, \lambda - 1, \ldots, 1 \,\}$.

Table 4(b) illustrates Category I sketches on the right-hand side, Category II sketches on the left-hand side, and Category III sketches on the bottom side, for the running example.

For any range $[0, \delta]$, it can always be decomposed to at most two sub-ranges. For example, Table 4(b) illustrates that $[0, 7]$ is decomposed into

- $[0, 5]$. This corresponds to $S_A \cap^{\leq 5} S_B$, and can be answered by estimating the size of $\langle \biguplus_{i \in \{3,6\}} S_A^{+i}, \biguplus_{i \in \{1,2,3\}} S_B^{+i} \rangle$.
- $[6, 7]$. This corresponds to $S_A \cap^{[6,7]} S_B$, and can be answered by estimating the size of $\langle S_A^{+9}, \biguplus_{i \in \{2,3\}} S_B^{+i} \rangle$.[2]

Algorithm 2: DecomposeRange (S_A, S_B, δ)

Input : δ: query gap. N: maximum gap.
Output: An array of indices pairs.

```
1  λ ← ⌈√N⌉;
2  if δ < λ or (δ + 1) mod x = 0 then
3      i_A ← { i · λ | i ∈ [1, max(⌈δ/λ⌉, 1)] };
4      i_B ← { λ − i | i ∈ [0, δ − ⌊δ/λ⌋ · λ] };
5      return { (i_A, i_B) };
6  else
7      i_A^l ← { i · λ | i ∈ [1, ⌊(δ + 1)/λ⌋] };
8      i_B^l ← { i | i ∈ [1, λ] };
9      if δ = N then
10         i_A^r ← N;
11         i_B^r ← 0;
12     else
13         i_A^r ← λ · ⌈(δ + 1)/λ⌉;
14         i_B^r ← { λ − i | i ∈ [0, δ mod λ] };
15     return { (i_A^l, i_B^l), (i_A^r, i_B^r) };
```

Algorithm 3: RangeQueryOnlineEstimation (S_A, S_B, δ)

Input : δ: query gap. N: maximum gap.
Output: $\hat{t}_r$: an estimate for $S_A \cap^{\leq \delta} S_B$.

```
1  arr ← DecomposeRange(δ) ;
2  t̂_r ← 0;
3  for each (i_A, i_B) ∈ arr do
4      Retrieve
       S_A's sketch based on the index of i_A into msk(M_A)
       and retrieve S_B's sketch for i_B into msk(M_B);
5      t̂_r ← t̂_r + Estimate(msk(M_A), msk(M_B));
       /* Perform estimation using two multiset
       bottom-k sketches based on Theorem 2 */;
6  return t̂_r;
```

We use Algorithm 3 to perform a range estimation with the gap constraint δ. In Line 1, it calls the DecomposeRange

[2] We deem a bottom-k sketch as a special multiset bottom-k sketch where the multiplicities are all set to 1.

(δ) to decompose the range $[0, \delta]$ into one or two parts, utilizing the multiset bottom-k sketches. The returned results are one or two pairs of indices to these sketches. Lines 3–5 retrieve the corresponding sketches and perform the estimation based on Eq. (3) in Theorem 2.

The space complexity of the sketch for the range estimation is $O(\sqrt{N} \cdot k)$. And the time complexity for the estimate is $O(k)$. Thanks to Theorem 3, by setting the proper k, our estimation has at most ϵ relative error with probability at least $1 - \rho$.

Table 5: Multiset bottom-3 Sketch

indices	$msk(M_A)$	$msk(M_B)$
$\{9\}$	$\{0.15:1, 0.16:1, 0.63:1\}$	$\{0.21:1, 0.37:1, 0.53:1\}$
$\{2,3\}$	$\{0.22:2, 0.42:1, 0.48:1\}$	$\{0.42:1, 0.48:1, 0.68:1\}$
$\vdots$	$\vdots$	$\vdots$
$\{3,6\}$	$\{0.22:1, 0.42:2, 0.48:1\}$	$\{0.16:1, 0.37:1, 0.48:1\}$
$\{1,2,3\}$	$\{0.22:2, 0.32:1, 0.42:1\}$	$\{0.22:1, 0.42:2, 0.48:2\}$

EXAMPLE 5. *We run Algorithm 3 for the same running example given before. Given $\delta = 7$, we want to estimate $|S_A \cap^{\leq 7} S_B|$. Line 1 decomposes $[0, 7]$ into two parts of indices, which is $arr = \{(\{3,6\}, \{1,2,3\}), (\{9\}, \{2,3\})\}$. For $(\{3,6\}, \{1,2,3\})$, Line 4 retrieves $msk(S_A^{+3,6}) = \{0.22 : 1, 0.42 : 2, 0.48 : 1\}$ and $msk(S_B^{+1,2,3}) = \{0.22 : 1, 0.42 : 2, 0.48 : 2\}$ from Table 5. Then Line 5 calculates $\hat{t}_r$ according to Theorem 2. It returns $\hat{t}_r$ to be $(1 \cdot 1 + 2 \cdot 2)/_{0.48} = 10.42$. Similar steps go for the second part indexed by $(\{9\}, \{2,3\})$, while there is no matching for this part. Finally, the algorithm returns $\hat{t}_r$ to be 10.42. The actual $|S_A \cap^{\leq 7} S_B|$ is 11.*

5. APPLICATIONS

The proposed methods have many important applications, e.g. top-K related keywords mining, query optimization for search engine, and system troubleshooting in log analysis. In this section, we present the details of efficient mining top-K related keywords from a document collection.

Let $\mathcal{V}$ be the vocabulary of the document collection. For ease of illustration, we concatenate all documents into one single document D with suitable padding of out-of-vocabulary keywords. For each keyword $v \in \mathcal{V}$, we create a set S_v, which consists of all the positions in D where it occurs. Let query S_Q be a set of positions. For any given keyword v, we can measure its correlation with the query by counting the number of occurrences of v in a δ-vicinity of any position in S_Q. The top-K related keywords problem is to find the K keywords in $\mathcal{V}$ that has the highest correlation.

Solving the problem exactly requires either intersecting all keywords in $\mathcal{V}$ or retrieving the δ-vicinity centered at positions in S_Q. Neither method scales well with large document collections.

To solve the problem approximately, we can apply range estimation to estimate the correlation, then return the K keywords with largest estimated size. Nevertheless, this method is still time consuming, as it is linear to vocabulary size $|\mathcal{V}|$.

We observe that most of the keywords set do not have a significant gapped intersection size with the query, hence it is highly likely that their sketches share *no* common hash value with the sketch of the query. It is thus desirable to consider only those keywords that share at least one hash value in their appropriate multiset bottom-k sketches with the bottom-k sketch of the query.

Therefore, we propose to build an inverted index, which maps $(hashValue, index)$ to a keyword v. Intuitively, by probing the inverted index with every hash value in S_Q's sketch, we can obtain a list of candidate keywords. Due to the range decomposition, we also have the additional constraint that the indices of these shared hash values must agree with those calculated for the current δ (i.e., returned by DecomposeRange).

Algorithm 4: Top$K$RangeEstimation (S_Q, δ, K, I)

Input : S_Q: sets of query positions; δ: query gap; K: top-K; I: the inverted index that maps hash values (and sketch's indices) to a keyword.
Output: Top-K keywords based on their estimated intersection size with S_Q

1 $R \leftarrow \emptyset$; $Cand \leftarrow \emptyset$;
2 $arr \leftarrow$ DecomposeRange(δ) ;
3 **for each** $(i_A, i_B) \in arr$ **do**
4 Retrieve S_Q's multiset bottom-k sketch whose index value is i_A into M_Q;
5 **for each** *element* $e \in$ elems($msk(M_q)$) **do**
6 $Cand \leftarrow Cand \cup I[(e, i_B)]$;
 `/* A list of keywords will be returned by looking up the index` I `*/`;
7 **for each** *keyword* $v \in Cand$ **do**
8 $R \leftarrow R \cup (v,$ RangeQueryOnlineEstimation$(S_Q, S_v, \delta))$;
9 **return** the K largest entries in R in terms of the estimated intersection size;

Algorithm 4 gives the pseudocode for the algorithm. Initially the candidate set $Cand$ is empty (Line 1). In Line 2, we obtain one or two pairs of indices, indexing into S_Q's and a potential candidate set's multiset bottom-k sketches. We iterate over all the pairs. For each pair of indices, we use i_A to retrieve the sketch of S_Q, and use Line 5–6 to retrieve all keywords such that their sketches with index value i_B share the same hash value (i.e., e in the code). This step is aided by the precomputed inverted index as a simple index lookup. Finally, we perform range estimate between S_Q and the set of each candidate keywords and return the largest K keywords.

The algorithm issues at most $2k - 2$ index lookups, and performs $|Cand|$ number of range estimations, where $|Cand|$ is the size of the candidate set. Finding the top-K entries from R and sorting them take $O(K \log K)$ time. Therefore, the total estimation time is $O(|Cand| \cdot k + K \log K)$.

In our implementation, we also perform the following optimizations. For every keyword returned from the index lookup (Line 6), we can accumulate its partial inner product with the query's sketch (i.e., the numerator of Eq. (3)). It can be shown that the numerator's value will be correctly calculated after the loop of Lines 3–6 ends. Therefore, the RangeQueryOnlineEstimation function only needs to do $O(1)$ computation to get the range estimation for each candidate.

EXAMPLE 6. *Using the same example, we can build the inverted index as shown in Table 6.*

Consider the top-1 related keyword query for A with $\delta = 5$. DecomposeRange gives us one pair of indices $(\{3,6\}, \{1,2,3\})$. This means we are concerned with the query's sketch indexed by $\{3,6\}$ and the candidate's sketch indexed by $\{1,2,3\}$. We load the query sketch $msk(S_A^{+3,6}) =$

Table 6: Inverted Index for top-K Related Keyword Mining

Key	List of Keywords
$(0.15, \{9\})$	S_A
$(0.16, \{3, 6\})$	S_B
$(0.16, \{9\})$	S_A
$(0.21, \{9\})$	S_B
$(0.22, \{1, 2, 3\})$	S_A, S_B
$(0.22, \{2, 3\})$	S_A
$(0.22, \{3, 6\})$	S_A
...	...

$\{\, 0.22 : 1, 0.42 : 2, 0.48 : 1 \,\}$. *We issue three lookups with keys:* $(0.22, \{1, 2, 3\})$, $(0.42, \{1, 2, 3\})$, *and* $(0.48, \{1, 2, 3\})$. *These lookups find matches* S_B, *which is added to the candidate set. Finally we perform range estimation between the query and the candidates then return the top-1 keyword.*

6. EXPERIMENTAL EVALUATION

In this section, we present the results of a comprehensive performance study to evaluate the efficiency and effectiveness of the proposed techniques.

6.1 Experiment Setup

We use the following algorithms for point and range estimation.

- GSISE-k is our proposed point (range) estimation method and k determines the size of a single bottom-k sketch.
- PointSum-k is our basic range estimation methods by summing up multiple point estimation results.
- Exact calculates the exact answer for point and range estimation, respectively.

We use the following algorithms for the top-K related keyword query.

- TopK is the hash table based top-K range estimation method proposed in Section 5.
- $\text{Exact}_{\text{topk}}$ is the exact algorithm for top-K range estimation. The algorithm is conducted by scanning the whole dataset, finding each occurrence of the query keyword, and bookkeeping the count of every other keyword that occurs within the δ neighborhood. Finally, it returns the K keywords with highest number of occurrences.

The dataset we use is a subset of the **ClueWeb** containing 500 million English web pages from the ClueWeb09 collection.[3] We remove infrequent keywords and keep the 100k most frequent keywords. We build the positional inverted index for these keywords, and treat each inverted list as a set. We then build three sets of sketches on these sets, i.e., point sketch, range sketch, and top-K sketch for the respective estimation problems.

The complete original positional index requires more than 2 days to construct using Hadoop on a cluster of 20 PCs, and the final index is stored on the HDFS of the Hadoop cluster. Without compression, the overall index consumes around 1TB space, which is impossible to load into a single commodity PC's memory. The overall size of the sketch is small enough to fit into our testing environment with 96GB RAM. Therefore, the experiments conducted on the sketches are memory-based, while exact algorithms that processes inverted lists or scanning documents have to use disk I/Os.

[3] `http://lemurproject.org/clueweb09.php`

Estimation Workload. We randomly select 100 pairs of keywords to perform point and range estimation with different parameter settings on their corresponding sets. We set maximum gap $N = 100$. k varies between 1,000 to 100,000 (default). The query gap δ varies between 20 and 100.

We measure the **running time** and **relative error** for point and range estimation methods. We measure **recall** and **extended recall** for TopK methods. Recall is defined as $^{A_K}/_K$, where A_K is the number of exact top-K results returned by an algorithm that outputs K results. The extended recall is defined by $^{A_{L\times K}}/_K$, where $A_{L\times K}$ is the number of exact top-K results returned by an algorithm that outputs $L \times K$ results, L is the extension rate. All measurements shown are averaged over 100 queries.

The experiment parameter settings of the evaluated algorithms are listed in Table 7, where the default parameters are highlighted in bold.

Table 7: Parameter Settings

Parameters	Setting (Defaults are in bold)
maximum gap	**100**
top-K	**1**, 10, 100
bottom-k	1000, 5000, 10000, 15000, **20000**
query gaps	20, 40, 60, 80, **100**

6.2 Point Estimation

In Figure 2(a), we show the point estimation time by varying query gaps. Compared with the exact algorithm, the sketch based method is more than 2 order of magnitudes faster. Query gaps do not affect the estimation time and exact time. As both exact and estimation methods rely on intersection algorithm of complexity $O(k)$ to find common elements. GSISE-1k, GSISE-5k, GSISE-10k, GSISE-15k, and GSISE-20k spend 0.38, 1.66, 3.47, 5.03 and 6.92 milliseconds to perform one estimation. The trend is linear. Average length of inverted list size is around 2.5 million, which is 125 times of GSISE-20k's sketch size. Meanwhile, Exact algorithm uses $892ms$ to calculate exact answer, which is $892/6.92 = 128$ times of GSISE-20k's estimation time. When dataset size increases, exact results will need longer time, while the sketch estimation time is immune to the increasing dataset size.

In Figure 2(b) and figure 2(c), we show how the relative error decreases while the bottom-k sketch size is increasing. When gap is 80, the relative error drops from 1 to 0.4 with k changing from 1000 to $20,000$. By further increasing k up to $100,000$, the relative error reduces to less than 0.19. Based on the analysis in section 3.2, the relative error is inverse proportional to the square root of bottom-k size. Despite the fluctuation, relative error is not affected by the different query gaps, the fluctuation is caused by different intersection sizes at different gaps. In our experiments, gap 40 has slightly more intersections than gap 100, which reflects that gap 40 has slightly less relative error than gap 100.

6.3 Range Estimation

From Figure 2(d) to 2(f), we present range query estimation results.

- **Time.** Figure 2(d) shows that our GSISE range estimation method has the best performance compared to other methods. For example, GSISE-20k is 128 times faster than Exact and 108 times faster than PointSum-20k when gap is 100. Similar to point estimation, GSISE is stable with different gaps. According to Algorithm 2, any range query

can be decomposed into at most two estimation queries, and the efficiency of estimation only relies on k. PointSum performs individual point estimations for each gap and uses the summation as the estimation result. Thus the processing time of PointSum is linear to the query gap. When gap is 20, 40, 60, 80 and 100, the processing time of PointSum-20k is 109.0, 213.1, 302.7, 410.6, and 508.0 milliseconds, which is linearly increasing with respect to gap.

- **Relative Error.** As shown in Figure 2(e), when query gap increases, the relative errors of all methods decrease. This is because larger range gap tends to result in larger intersection size. According to Theorem 3, it is equivalent to using a larger k.

  It is noted that PointSum achieves slightly lower relative error than range estimation given the same k setting. The reason is that PointSum utilizes all values in the point shifted sketch for estimating, which is equivalent to merging *without* truncating at k-th hash value. As opposed to the truncated merge operation applied to GSISE$_r$ index, PointSum will consume up to more than λ times space.To be more specific, in Figure 2(e), when query gap is 100, the range sketch required for GSISE-20k is $2 \times 20k$, while for PointSum, the cost is $20 \times 20k$. However, the PointSum-20k only decrease the error by less than 0.05. Considering the tremendous performance gain in efficiency and storage, the tiny sacrifice in relative error can be ignored.

  Figure 2(f) shows similar trend as Figure 2(c). Larger k will lead to smaller relative error. The relative error decreases rapidly at the initial increasing of k, then flattens after $k > 10,000$. PointSum has smaller relative error than GSISE methods, however, the relative errors are almost the same when k is large. Interestingly, the storage cost of PointSum-1k of gap $100 (20 \times 1k = 20k)$ and of GSISE-10k of gap $100 (2 \times 10k = 20k)$ are the same, so are the relative error of PointSum-1k of gap 100 and GSISE-10k, which are 0.267 and 0.266, respectively.

6.4 Top-K Related Keywords Mining

The experiment results of top-K problem are presented in Figures 2(g) to 2(o).

- **Time**. In Figures 2(g), 2(j) and 2(m), we investigate the response time by varying K of top-K from 1 to 100. In general, we can see that the average query processing time is under 25 milliseconds, which is quite small compared to Exact$_{\text{topk}}$. In fact, as we use large web page corpus, the Exact$_{\text{topk}}$ algorithm must run on the cluster in a batch mode for the given queries, which takes 6 hours to scan through the original documents in order to output top-K results. In each figure, by increasing the query gaps from 20 to 100, we observe that the query processing time increases slightly. This is due to the processing of top-K range estimation needs to count the number of common sketch values in the hash table. Larger query gap will result in larger intersection size, which means more common sketch values when looking up the hash table. Same reason goes for the increasing query processing time when k increases.
- **Recall**. We show the recall of our algorithms in different top-K settings in Figures 2(h), 2(k) and 2(n). We can observe that the recall grows with the increase of k. Larger k incurs smaller relative error, consequently improves the recall of the top-K algorithms. Furthermore, it is observed that query gap has no influence on the recall, which shows great stability over different query gap settings.
- **Extended Recall**. As shown in Figure 2(g) and Figure 2(m), the processing time of returning top 1 result and top 100 results are similar. This gives us the motivation of measuring extended recall. Figures 2(i), 2(l) and 2(o), present the extended recall. The extension rate is up to 4. In Figure 2(i), when extension rate is larger than 3, the recall of GSISE-15k and GSISE-20k both reach 90%. In other figures, all GSISE-20k experiments can reach more than 80% of recall when extension rate is 4. GSISE-1k performs inadequate even when extension rate reaches 4.

  In summary, our top-K estimation algorithm shows substantial performance advantage over exact methods in efficiency. Meanwhile, the recall of our algorithms can reach around 90% with slight extension of returned top-K results.

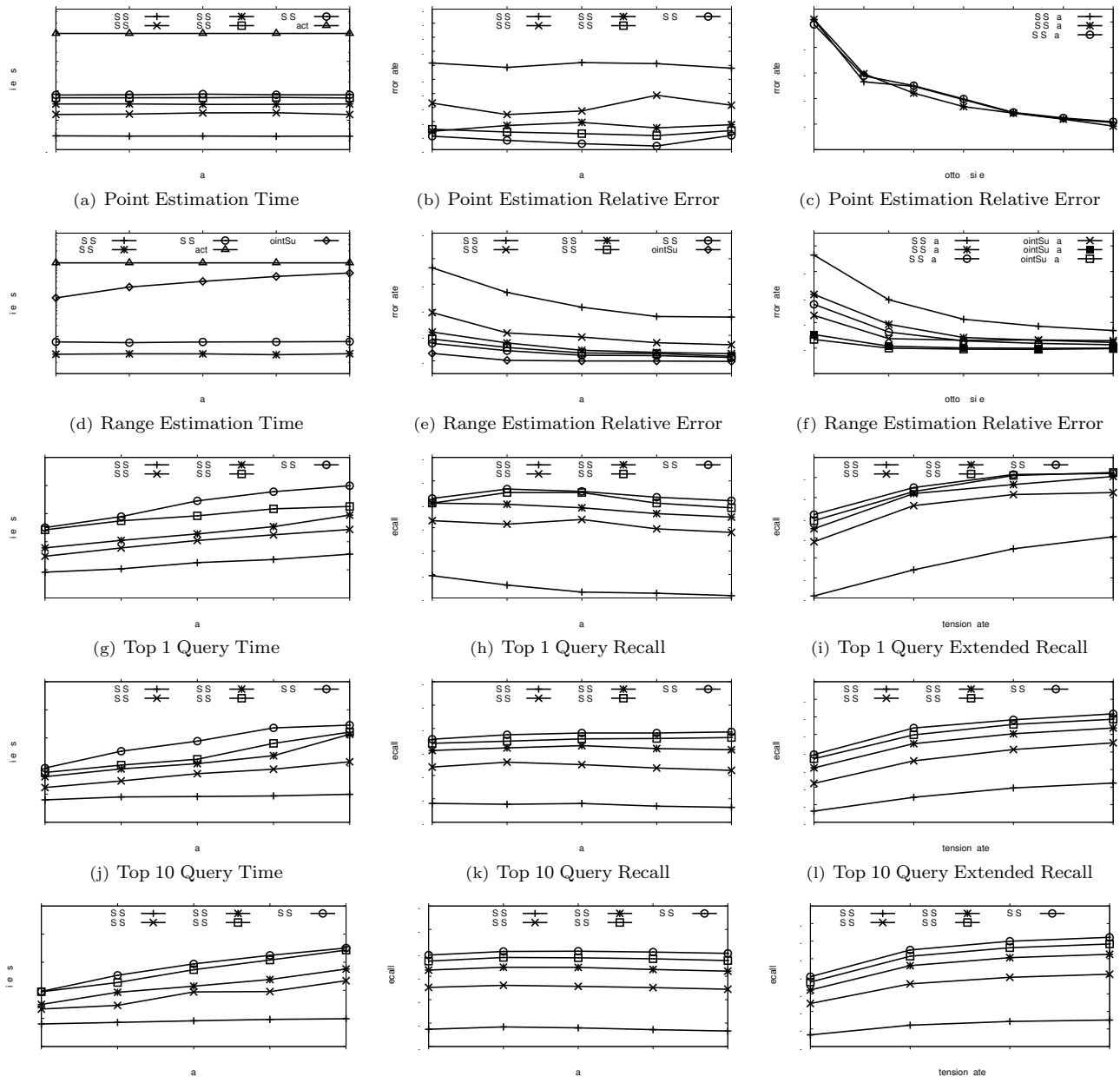
7. CONCLUSIONS

In this paper, we formally define the GSISE problem, for both point and range gap constraints. We propose space and time efficient estimation methods, based on bottom-k sketches and its extension to multisets. In addition, our estimation methods provide the probabilistic quality guarantees. We also apply our technique to the problem of finding top-K related keyword, by combining our estimation technique with the use of an inverted index. Our experiments using half a billion documents empirically verify the effectiveness and efficiency of the proposed methods.

Acknowledgements. We would like to thank Stefan Böttcher for helpful discussions and comments during our project cooperation. This work has been supported by Go8-DAAD Project RG123842, UNSW ECR/FRG PS35163, and ARC Discovery Projects DP130103401 and DP130103405.

8. REFERENCES

[1] N. Alon, Y. Matias, and M. Szegedy. The space complexity of approximating the frequency moments. In *STOC*, 1996.
[2] J. Barbay, A. López-Ortiz, and T. Lu. Faster adaptive set intersections for text searching. In *WEA*, 2006.
[3] J. Barbay, A. López-Ortiz, T. Lu, and A. Salinger. An experimental investigation of set intersection algorithms for text searching. *Journal of Experimental Algorithmics*, 14, 2009.
[4] K. S. Beyer, P. J. Haas, B. Reinwald, Y. Sismanis, and R. Gemulla. On synopses for distinct-value estimation under multiset operations. In *SIGMOD Conference*, 2007.
[5] A. Z. Broder, M. Charikar, A. M. Frieze, and M. Mitzenmacher. Min-wise independent permutations (extended abstract). In *Symposium on the Theory of Computing*, 1998.
[6] E. Cohen. Size-estimation framework with applications to transitive closure and reachability. *J. Comput. Syst. Sci.*, 55(3), 1997.
[7] E. Cohen and H. Kaplan. Summarizing data using bottom-k sketches. In *PODC*, 2007.
[8] E. Cohen and H. Kaplan. Tighter estimation using bottom k sketches. *PVLDB*, 1(1), 2008.
[9] E. Cohen and H. Kaplan. Leveraging discarded samples for tighter estimation of multiple-set aggregates. In *SIGMETRICS/Performance*, 2009.
[10] G. Cormode and S. Muthukrishnan. An improved data stream summary: The count-min sketch and its applications. In *LATIN*, 2004.
[11] J. S. Culpepper and A. Moffat. Efficient set intersection for inverted indexing. *ACM Trans. Inf. Syst.*, 29(1), 2010.
[12] E. D. Demaine, A. López-Ortiz, and J. I. Munro. Adaptive set intersections, unions, and differences. In *SODA*, 2000.
[13] B. Ding and A. C. König. Fast set intersection in memory. *PVLDB*, 4(4), 2011.



(a) Point Estimation Time (b) Point Estimation Relative Error (c) Point Estimation Relative Error

(d) Range Estimation Time (e) Range Estimation Relative Error (f) Range Estimation Relative Error

(g) Top 1 Query Time (h) Top 1 Query Recall (i) Top 1 Query Extended Recall

(j) Top 10 Query Time (k) Top 10 Query Recall (l) Top 10 Query Extended Recall

(m) Top 100 Query Time (n) Top 100 Query Recall (o) Top 100 Query Extended Recall

Figure 2: Experiment Results

[14] D. Dubhashi and D. Ranjan. Balls and bins: A study in negative dependence. *Random Structures and Algorithms*, 13:99–124, 1996.

[15] R. Fagin. Combining fuzzy information from multiple systems. *J. Comput. Syst. Sci.*, 58(1), 1999.

[16] R. Fagin, A. Lotem, and M. Naor. Optimal aggregation algorithms for middleware. *J. Comput. Syst. Sci.*, 66(4), 2003.

[17] J.-L. Hou and C.-A. Chan. Method for keyword correlation analysis. US Patent 20050071365 A1, 2005.

[18] F. K. Hwang and S. Lin. A simple algorithm for merging two disjoint linearly-ordered sets. *SIAM J. Comput.*, 1(1), 1972.

[19] P. Indyk et al. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *STOC*, 1998.

[20] Y. Luo, W. Wang, X. Lin, X. Zhou, J. Wang, and K. Li. SPARK2: top-k keyword query in relational databases. *IEEE Trans. Knowl. Data Eng.*, 23(12), 2011.

[21] D. Okanohara and Y. Yoshida. Conjunctive filter: Breaking the entropy barrier. In *ALENEX*, 2010.

[22] R. Pagh, M. Stöckel, and D. P. Woodruff. Is min-wise hashing optimal for summarizing set intersection? In *PODS*, 2014.

[23] J. B. Rocha-Junior, A. Vlachou, C. Doulkeridis, and K. Nørvåg. Efficient processing of top-k spatial preference queries. *PVLDB*, 4(2), 2010.

[24] B. Schlegel, T. Willhalm, and W. Lehner. Fast sorted-set intersection using SIMD instructions. In *ADMS*, 2011.

[25] M. Sudhof, A. G. Emilsson, A. L. Maas, and C. Potts. Sentiment expression conditioned by affective transitions and social forces. In *SIGKDD*, 2014.

[26] D. Takuma and H. Yanagisawa. Faster upper bounding of intersection sizes. In *SIGIR*, 2013.

[27] M. Theobald, H. Bast, D. Majumdar, R. Schenkel, and G. Weikum. Topx: efficient and versatile top-k query processing for semistructured data. *VLDB J.*, 17(1), 2008.

[28] C. Xiao, W. Wang, X. Lin, and H. Shang. Top-k set similarity joins. In *ICDE*, 2009.