Set Similarity Search for Skewed Data

Samuel McCauley [†] Jesper W. Mikkelsen [‡] Rasmus Pagh[†]

Abstract

Set similarity join, as well as the corresponding indexing problem *set similarity search*, are fundamental primitives for managing noisy or uncertain data. For example, these primitives can be used in data cleaning to identify different representations of the same object. In many cases one can represent an object as a sparse 0-1 vector, or equivalently as the set of nonzero entries in such a vector. A set similarity join can then be used to identify those pairs that have an exceptionally large dot product (or intersection, when viewed as sets). We choose to focus on identifying vectors with large *Pearson correlation*, but results extend to other similarity measures. In particular, we consider the indexing problem of identifying correlated vectors in a set S of vectors sampled from $\{0,1\}^d$. Given a query vector $\mathbf{y}$ and a parameter $\alpha \in (0,1)$, we need to search for an α-correlated vector $\mathbf{x}$ in a data structure representing the vectors of S. This kind of similarity search has been intensely studied in worst-case (non-random data) settings.

Existing theoretically well-founded methods for set similarity search are often inferior to heuristics that take advantage of *skew* in the data distribution, i.e., widely differing frequencies of 1s across the d dimensions. The main contribution of this paper is to analyze the set similarity problem under a random data model that reflects the kind of skewed data distributions seen in practice, allowing theoretical results much stronger than what is possible in worst-case settings. Our indexing data structure is a recursive, data-dependent partitioning of vectors inspired by recent advances in set similarity search. Previous data-dependent methods do not seem to allow us to exploit skew in item frequencies, so we believe that our work sheds further light on the power of data dependence.

1 Introduction

Data management is increasingly moving from a world of well-ordered, curated data sets to settings where data may be noisy, incomplete, or uncertain. This requires primitives that are able to work with notions of "approximate match", as opposed to the exact matches used in standard hash indexes and in equi-joins. Such functionality is particularly challenging to scale when data is high-dimensional, informally because of the *curse of dimensionality* which makes it hard to organize data in such a way that approximate matches can be efficiently identified. In this paper we consider the fundamental primitive *set similarity join* and more specifically the indexing problem *set similarity search*. A set similarity join is used to identify pairs of sets that are similar in the sense that they have an "exceptionally large intersection". Many notions of "exceptionally large intersection" exist, but from a theoretical point of view they are essentially equivalent [18]. We will consider the encoding of sets as sparse 0-1 vectors, and use Pearson correlation as the measure of similarity; we give more details below.

The information retrieval and database communities have extensively worked on designing scalable algorithms for similarity join, see e.g. the recent book by Augsten and Böhlen [10]. The state-of-the-art for practical set similarity join algorithms is reflected in the recent mini-survey and comprehensive empirical evaluation of Mann et al. [31]. The best methods in practice are ones that exploit the significant *skew* in frequencies of set elements that exists in many data sets. When there is insufficient skew these methods become inefficient, and in the worst case they degenerate to a trivial brute-force algorithm. In contrast, strong theoretical results are known when *randomization* and *approximation* of distances is allowed,

[*] The research leading to these results has received funding from the European Research Council under the European Union's 7th Framework Programme (FP7/2007-2013) / ERC grant agreement no. 614331. BARC, Basic Algorithms Research Copenhagen, is supported by the VILLUM Foundation grant 16582.

[†] BARC and IT U. Copenhagen, Copenhagen, Denmark. {samc,pagh}@itu.dk

[‡] IT U. Copenhagen, Copenhagen, Denmark. jesperwm@gmail.com

e.g. [2, 18, 22, 25, 46]. Even though existing randomized algorithms for set similarity search are superior to commonly used heuristics for difficult data distributions with small skew, it is clear that the heuristics will work much better (in theory and in practice) when the skew is large enough.

In this paper we target this disconnect between theory and practice, presenting a new data structure that, in a certain way, can *interpolate* between the best existing methods for small skew and heuristics that work well with large skew. Our message is that modeling skew can lead to algorithms that takes advantage of structure in data in a way that is theoretically justified. This complements recent advances in the theory of *data dependent* methods for high-dimensional search, where clustering structure in data (rather than skew) is exploited to achieve faster algorithms, even in the worst case [6, 8].

Motivating example.

Suppose we wish to search n d-dimensional boolean vectors $\mathbf{x}^1, \ldots, \mathbf{x}^n$ chosen from the "harmonic" distribution where the kth bit is set with probability $\Pr[\mathbf{x}^j_k = 1] = 1/k$, independently for each $\mathbf{x}^j$ and each $k \in \{1, \ldots, d\}$. The boolean vectors represent subsets of $\{1, \ldots, d\}$. For consistency with the rest of the paper, we refer to these elements as vectors, but use some set notation for simplicity; i.e. $|\mathbf{x}|$ is used to represent the Hamming weight of $\mathbf{x}$.

Assuming (for now) $\log d \gg \log n$, all vectors have Hamming weight close to the expectation $|\mathbf{x}| \approx \sum_{k=1}^{d} 1/k \approx \ln d$ with high probability by Chernoff bounds. We wish to search for a vector $\mathbf{x}^{j^*}$ that is correlated with a query vector $\mathbf{q}$ such that $|\mathbf{x}^{j^*} \cap \mathbf{q}| \geq i_1|\mathbf{q}|$, for some parameter $i_1 \in (0, 1)$.

Define $i_2|\mathbf{q}| = \mathbf{E}[|\mathbf{x}^j \cap \mathbf{q}|] = \sum_{k \in \mathbf{q}} 1/k$ to be the expected intersection size between $\mathbf{q}$ and $\mathbf{x}^j$ for $j \neq j^*$, $i_2 < i_1$. Assuming $i_2|\mathbf{q}| \gg \log n$ we will have $|\mathbf{x}^j \cap \mathbf{q}| \approx i_2|\mathbf{q}|$ for all $j \neq j^*$ with high probability. In this setting it is known how to perform the search in expected time roughly n^ρ, where $\rho = \log(i_1)/\log(i_2)$, and this bound is tight for LSH-like techniques [18].

However, we can do better for skewed distributions by splitting the search problem into two parts: split $\mathbf{q}$ into two equal-sized vectors $\mathbf{q}^{\text{frequent}}$ and $\mathbf{q}^{\text{rare}}$, defined as the first (resp. last) half $d/2$ bits of $\mathbf{q}$. Note that for every choice of parameter ℓ, we either have $|x^{j^*} \cap \mathbf{q}^{\text{frequent}}| \geq \ell|\mathbf{q}|$ or $|x^{j^*} \cap \mathbf{q}^{\text{rare}}| \geq (i_1 - \ell)|\mathbf{q}|$, so the original search problem can be solved by performing searches for a set with a large overlap either with $\mathbf{q}^{\text{frequent}}$ or $\mathbf{q}^{\text{rare}}$. Let

$$i_{\text{frequent}} = \mathbf{E}[|\mathbf{x}^j \cap \mathbf{q}^{\text{frequent}}|]/|\mathbf{q}|, \text{ and}$$
$$i_{\text{rare}} = \mathbf{E}[|x^j \cap \mathbf{q}^{\text{rare}}|]/|\mathbf{q}|,$$

such that $i_2 = i_{\text{frequent}} + i_{\text{rare}}$. Now the combined cost of the two searches becomes approximately $n^{\rho_{\text{frequent}}} + n^{\rho_{\text{rare}}}$, where

$$\rho_{\text{frequent}} = \log(\ell)/\log(i_{\text{frequent}}), \text{ and}$$
$$\rho_{\text{rare}} = \log(i_1 - \ell)/\log(i_{\text{rare}}) \ .$$

Choosing k to balance the two terms we get a faster query time whenever $i_{\text{frequent}} \gg i_{\text{rare}}$, i.e., when the distribution of elements in $\mathbf{q}$ has large skew.

The example shows that skew can be exploited, but it remains unclear how to do this in a principled way. This question was the starting point for this paper.

Probabilistic viewpoint.

Establishing correlation between random variables is a fundamental and well-studied problem in statistics, but computational aspects of this problem are far from settled. In a breakthrough paper [42], Greg Valiant addressed the so-called *light bulb problem* originally posed in [43]:

> *Given a set of n vectors $S \subseteq \{0, 1\}^d$ chosen uniformly and independently at random, with the exception of one pair of distinct vectors $\mathbf{x}, \mathbf{y} \in S$ that have correlation $\alpha > 0$, identify the vectors $\mathbf{x}$ and $\mathbf{y}$.*

If d is sufficiently high (e.g. $d \gg \log(n)/\alpha$) the correlated vectors $\mathbf{x}$ and $\mathbf{y}$ are, with high probability, the only pair with an inner product of around $(1 + \alpha)d/4$, while all other pairs have inner product around $d/4$. From an algorithmic perspective the problem then becomes that of finding the pair of vectors with a significantly higher inner product. The problem of searching for correlated vectors has been intensely studied in recent years, in theory [26, 27, 4, 3, 2, 1, 32] and in practice [38, 39, 40, 37, 28, 21, 20, 41]. Practical solutions often address the *search* version, known as *maximum inner product search*, where the vector $\mathbf{y}$ is given as a query (often denoted $\mathbf{q}$) and the task is to search for the correlated vector $\mathbf{x}$ in a data structure enabling fast search among the vectors in S.

The light bulb problem is perhaps the cleanest and most fundamental correlation search problem. However, vectors of real-life data sets are usually not well described by a uniform distribution over $\{0,1\}^d$. Instead, such vectors are often sparse (assuming without loss of generality that 0 has probability at least 1/2 in each coordinate), and the fraction of vectors having the value 1 in the ith coordinate varies greatly with i, often following e.g. a Zipfian distribution (see Section 8). This kind of *skew* is exploited by practical solutions to correlation search [11, 44, 31], since high correlation between vectors $\mathbf{x}$ and $\mathbf{y}$ will often be "witnessed" by $\mathbf{x}_i = \mathbf{y}_i = 1$, where the set $\{\mathbf{z} \in S \mid \mathbf{z}_i = 1\}$ is small. On the other hand, such methods do not perform well when the skew is small. In this paper we explore the computational problem of identifying correlations in random data with skew, focusing on the search version of the problem. Generalizing and modifying recent worst-case efficient data structures, we are able to get a smooth trade-off between "hard" queries and data sets with no skew, and "easy" queries and data sets of the kind often encountered in practice.

To model skewed data we adopt the model of Kirsch et al. [29] that was previously used to give statistical guarantees on data mining algorithms. We are not aiming at statistical guarantees, but instead use this model as an interesting "middle ground" between uniformly random and worst-case settings when analyzing algorithms dealing with high-dimensional data. Conceptually this model:

- is expressive enough to model real-world data (such as the feature vectors that are ubiquitous in machine learning) much better than random data,
- avoids the pessimism of worst-case analysis, yet
- is simple enough to be tractable to analyze.

1.1 Our Results

We assume that data vectors are sampled from a distribution $\mathcal{D}$ (see Section 2 for details). Let S be the set of n data vectors sampled independently from $\mathcal{D}$. We parameterize how close a query is to its nearest neighbor using a parameter α. For $\alpha > 0$ a query $\mathbf{q}$ that is α-correlated with some $\mathbf{x} \sim \mathcal{D}$ can be defined as follows: Let $\mathbf{n} \sim \mathcal{D}$ be a "noise vector", and independently let

$$\mathbf{q}_i = \begin{cases} x_i & \text{with probability } \alpha \\ n_i & \text{with probability } 1-\alpha \ . \end{cases}$$

For a data vector $\mathbf{x} \sim \mathcal{D}$ we define $p_i = \Pr[\mathbf{x}_i = 1]$. We assume $p_i < 1/2$ for all i and that each bit of $\mathbf{x}$ is sampled independently.

Our results involve two additional parameters. First, we follow previous literature (e.g. [18, 17, 7]) in parameterizing our running time by a constant ρ. Generally, ρ would only depend on the similarity of the planted close points (α in this case); for our problem ρ is a function of α and $\mathcal{D}$ (and the query in the adversarial case). Secondly, we assume that there is a large constant C satisfying $\sum_{i\in[d]} p_i = C\log n$. We require C to be large both to ensure correctness[1] and to achieve our target running time.

Theorem 1. *Consider a dataset S of n vectors sampled from $\mathcal{D}$ and let C satisfy $\sum_{i\in[d]} p_i \geq C\log n$.*

Assume that $\mathbf{q}$ is α-correlated with $\mathbf{x}$ for some $\alpha > 0$.

Our data structure returns $\mathbf{x}$ on query $\mathbf{q}$ with high probability. Let ρ satisfy

$$\sum_{i\in[d]} \frac{p_i^{1+\rho}}{p_i(1-\alpha)+\alpha)} = \sum_{i\in[d]} p_i.$$

Then for every $\epsilon > 0$ there exists a sufficiently large C such that each query has expected cost $O(dn^{\rho+\epsilon})$, and the data structure requires expected $O(n^{1+\rho+\epsilon} + dn)$ space.

[1] Specifically, to ensure that our data structure is correct with high probability.

Discussion.

In the balanced case where all probabilities p_i are identical, we recover the time bounds of the recently proposed ChosenPath algorithm [18], which are known to be optimal in this setting. In the very unbalanced case where some p_i are $\Omega(1)$, some p_i are $O(1/n)$, and the expected number of items of both kinds are comparable, we match the well-known *prefix filter* algorithm [11], which beats ChosenPath in this setting. For skew between these extremes we get strict improvements over existing methods. (See Section 7 and Figure 1 for further discussion.)

Techniques.

Our data structure is a natural, recursive variant of ChosenPath that is able to exploit skew by varying the recursion depth over the branches of the recursion tree and by aggressively favoring choices based on the given distribution that are more likely distinguish close and far elements. We stress that ChosenPath is not able to exploit skew, and in fact has the same tight running time guarantee independent of the data distribution. Because we cut the depth of the tree earlier based on the skew of the distribution, we must tighten the previous analysis to handle sampling without replacement, while also parameterizing our performance based on skew. This leads to significant challenges.

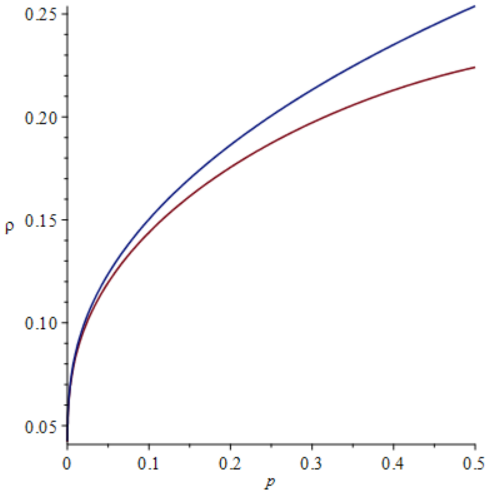

Figure 1: The red line gives the ρ value of our data structure when the distribution is such that half the bits are set to 1 with probability p and the other half is set to 1 with probability $p/8$, and the sought-for correlation is $\alpha = 2/3$. The blue line gives the ρ-value achieved by Chosen Path. Prefix filtering has a ρ-value of 1 in this case and therefore not included in the figure. We see that even though Chosen Path achieves the optimal ρ-value for solving the (b_1, b_2)-approximate similarity problem when considering worst-case inputs, we can achieve a better ρ-value when the input distribution is skewed.

Adversarial queries.

It may not always be reasonable to assume that a query is random and α-correlated, as in Theorem 1. For this reason we analyze the setting where the query may be adversarially chosen. In Section 3 we give a data structure that adapts to the difficulty of the query, matching existing worst-case bounds for "hard" queries, while being much faster for "easy" queries. Our result uses a similarity function $B(\mathbf{x}, \mathbf{q})$ (defined later) to parameterize how close the query is to the nearest element of the data set.

Theorem 2. *Consider a dataset S of n vectors sampled from $\mathcal{D}$ and let C satisfy $\sum_{i\in[d]} p_i \ge C \log n$. Let ρ_u satisfy*

$$\sum_{i\in[d]} p_i^{1+\rho_u} = b_1 \sum_{i\in[d]} p_i.$$

For any query $\mathbf{q}$ of size $|\mathbf{q}| \ge C \log n$ satisfying $B(\mathbf{q}, \mathbf{x}) \ge b_1$ for some $\mathbf{x} \in S$, let $\rho(q)$ satisfy

$$\sum_{i\in\mathbf{q}} p_i^{\rho(q)} = b_1 \sum_{i\in\mathbf{q}} p_i.$$

Then for any $\varepsilon > 0$ there exists a sufficiently large C such that our data structure requires $O(n^{1+\rho_u+\varepsilon} + dn)$ expected space, can be built in $O(dn^{1+\rho_u+\varepsilon})$ time, and can perform a search in time $O(dn^{\rho(q)+\varepsilon})$, returning $\mathbf{x}$ with probability[2] *at least $1/2$.*

[2] This probability can be increased using independent repetitions.

Similarity joins.

Our results immediately apply to the problem of database similarity joins [34, 9, 30, 45, 10, 31, 2, 22].

Many similarity join algorithms work using (essentially) repeated similarity search queries; see e.g. [31, 34, 24]. This method is equally effective here. Assume that we want to find all similar pairs between sets R and S, and that the actual join size (i.e. the number of close pairs) is much smaller than R or S.[3] Then Theorem 2 implies that we can preprocess S in time $O(d|S|^{1+\rho})$ and find all pairs in time $O(d|R||S|^{\rho})$. The same idea extends to Theorem 1 as well.

1.2 Related work

Assume in the following that d is large enough that the empirical correlation closely matches the true correlation.

Correlation search on the unit sphere.

After shifting vectors to have zero mean and normalizing them (which does not affect correlation), searching for a vector $\mathbf{x} \in S$ that is α-correlated with a query vector $\mathbf{y}$ boils down to a search problem on the unit sphere S^{d-1}: Given $\mathbf{y} \in S^{d-1}$ find $\mathbf{x} \in S$ such that $\mathbf{x}\cdot\mathbf{y} \geq \alpha$. In turn, this is equivalent to near neighbor search under Euclidean distance on the unit sphere, which has been studied extensively. For "balanced" data structures with space (and construction time) $\tilde{O}(n^{1+\rho})$ and query time $\tilde{O}(n^{\rho})$ the state-of-the-art is $\rho = \frac{1-\alpha}{1+\alpha}$ [5], using a non-trivial generalization of Charikar's famous LSH for angular distances [15]. Generalizations to various time-space trade-offs are also known [17, 7].

Correlation search on sparse vectors.

For sparse binary vectors, the framework of *set similarity search* captures search for different notions of similarity/correlation [16]. For vectors of fixed Hamming weight there is a 1-1 correspondence between Pearson correlation and standard set similarity measures such as Jaccard similarity and Braun-Blanquet similarity. It is known that methods such as MinHash [14, 13] that specifically target sparse vectors yield better search algorithms than more general methods when the fraction β of non-zero entries is bounded by a sufficiently small constant. Recently it was shown that MinHash can be improved in this setting, yielding balanced similarity search data structure with $\rho = \log(\beta + \alpha(1-\beta))/\log\beta$ [18].

Batched search.

The first subquadratic algorithm for the light bulb problem was given by Paturi et al. [36], yielding time $n^{2-\Theta(\alpha)}$. Valiant [42] showed that it is possible to remove α from the exponent, obtaining substantially subquadratic running time even for small values of α. Karppa et al. strenthened Valiant's result to time $O(n^{1.6}\text{poly}(1/\alpha))$ [26], for $\log n \ll d = n^{o(1)}$ (where the constant 1.6 reflects current matrix multiplication algorithms). A somewhat slower, but deterministic, subquadratic algorithm was subsequently presented [27]. With the exception of [36] these methods rely on fast rectangular matrix multiplication and seem inherently only applicable to batched search problems.

Worst-Case search.

Much of previous theoretical work on set similarity search has focused on the worst case, where the point set and queries are adversarial rather than being drawn from a known distribution. The Chosen Path algorithm of Christiani and Pagh focuses on Braun-Blanquet similarity, which is the metric we use here.[4] In particular, if the dataset contains a point p where q and p have Braun-Blanquet similarity at least b_1, the data structure returns a point p' such that q and p have Braun-Blanquet similarity at least b_2. Their data structure achieves space $\tilde{O}(n^{1+\rho})$ and query time $\tilde{O}(n^{\rho})$ for $\rho = (\log b_1)/\log b_2$. This result is a strict improvement over the classic MinHash algorithm [14, 13].

[3] If the join is large we can extend this idea via parameterizing by the join size; see e.g. the time bounds in [34, 22].

[4] We show that Pearson correlated vectors are very likely to have high Braun-Blanquet similarity in Lemma 10.

Lower bounds.

Cell probe data structure lower bounds achievable by current techniques are polylogarithmic in data size (assuming polylogarithmic word size). Tight bounds are only known for small query time, see e.g. [35] — for subquadratic space this leaves a large gap to the polynomial upper bounds that can be achieved by similarity search techniques. However, good *conditional* lower bounds based on the strong exponential time hypothesis (SETH) [23] are known. Ahle et al. [2] showed hardness of approximate maximum inner product search under SETH, but only for inner products very close to zero. Recently, Abboud, Rubinstein, and Williams [1] significantly improved this result, and in particular showed, again assuming SETH, that maximum inner product requires near-linear time even for vectors in $\{0,1\}^d$ and allowing a large $2^{(\log d)^{1-o(1)}}$ approximation factor. This means that there is little hope of obtaining strong algorithms in the worst case, even for batched search problems. Of course, as the upper bounds for the light bulb problem show, the average case is easier.

Heuristics.

Above we have discussed related work with a *theoretical* emphasis. Many heuristics exist, especially for sparse vectors. Some of the most widely used ones are based on exact, deterministic filtering techniques, such as prefix filtering [11] which evaluates the similarity of every pair of vectors that share a 1 in a position that has a small fraction of 1s. This is effective when vectors are sparse and *skewed* in the sense that there are many positions with a small fraction of 1s. We refer to Mann et al. [31] for an overview of exact, heuristic techniques.

2 Model

Our model closely follows the one in [29] — we describe it here for completeness. The elements in our set are d-dimensional boolean vectors $\mathbf{x} \in \{0,1\}^d$. These vectors represent a subset of items from a universe $U = \{1,\dots,d\}$. We generally consider our elements to be boolean vectors; however, we express some of our operations using set notation for convenience (for example, $\mathbf{x} \cap \mathbf{q}$ represents the 1 bits in common between $\mathbf{x}$ and $\mathbf{q}$).

We are given a distribution $\mathcal{D} = \mathcal{D}[p_1,\dots,p_d]$ over $\{0,1\}^d$ defined as follows: if $\mathbf{x}$ is a vector drawn from $\mathcal{D}$, then $\Pr[\mathbf{x}_i = 1] = p_i$ independently for each $i \in [d]$. We denote by $\mathcal{D}^n = \mathcal{D}^n[p_1,\dots,p_n]$ the distribution obtained by sampling n vectors independently from $\mathcal{D}$. The probabilities $p_1,\dots,p_d$ are assumed to be known to the algorithm. We assume that all item-level probabilities are at most 1/2. The particular value 1/2 is not important, and all of our results holds as long as there is some constant $M < 1$ such that all item-level probabilities are bounded by M.

Definition 3 (Correlation). *Let* $\alpha \in [0,1]$*. Fix* $\mathbf{x} \in \{0,1\}^d$*. We say that* $\mathbf{q}$ *is* α-correlated to $\mathbf{x}$ with respect to $\mathcal{D}$*, written* $\mathbf{q} \sim \mathcal{D}_\alpha(\mathbf{x})$*, if* $\mathbf{q}$ *is a random vector drawn as follows: For each* $i \in [d]$ *independently, with probability* α *let* $\mathbf{q}_i = \mathbf{x}_i$*, and with probability* $1-\alpha$ *let* $\mathbf{q}_i \sim$ *Bernoulli*(p_i)*.*

If $\mathbf{x} \sim \mathcal{D}$ and $\mathbf{q} \sim \mathcal{D}_\alpha(\mathbf{x})$, then the distribution of $\mathbf{q}$ is exactly $\mathcal{D}$—in particular, $\Pr(\mathbf{q}_i = 1) = p_i$. Furthermore, for each $i \in [d]$, the random variables $\mathbf{q}_i$ and $\mathbf{x}_i$ have Pearson correlation α.

Our goal is to create an efficient data structure for vectors sampled from $\mathcal{D}^n$ which takes advantage of possible skew in the item-level probabilities. Our performance bounds (query and preprocessing times) are expected, and depend both on $\mathcal{D}^n$ and the data structure's random choices. Following [18], we will use Braun-Blanquet similarity,

$$B(\mathbf{x},\mathbf{q}) = \frac{|\mathbf{x} \cap \mathbf{q}|}{\max\{|\mathbf{x}|,|\mathbf{q}|\}},$$

as our measure for similarity between sets. This similarity measure is closely related to Jaccard similarity, see [18] for details. We now formally define the two versions of the problem that we are considering:

Adversarial query

Given a dataset $S \sim \mathcal{D}^n[p_1,\dots,p_d]$ of n items sampled from $\mathcal{D}$ and a similarity threshold b_1, preprocess S into a data structure with the following capability: Given $\mathbf{q} \in \{0,1\}^d$, return $\mathbf{x} \in S$ such that $B(\mathbf{x},\mathbf{q}) \geq b_1$

if such an $\mathbf{x}$ exists. The data structure must succeed with probability at least $1 - o_n(1)$ over its own internal randomness. In particular, the probability that the data structure succeeds must be independent of the dataset $S \in \mathcal{D}^n$.

Correlated query

Given a dataset $S \sim \mathcal{D}^n[p_1, \dots, p_d]$ of n items sampled from $\mathcal{D}$ and a correlation threshold $0 < \alpha \leq 1$, preprocess S into a data structure with the following capability: Let $\mathbf{x} \in S$ and let $\mathbf{q} \sim \mathcal{D}_\alpha(\mathbf{x})$ be α-correlated to $\mathbf{x}$ with respect to $\mathcal{D}$. Then, given the query $\mathbf{q}$, the data structure must return $\mathbf{x}$. The data structure must succeed in doing so with probability at least $1 - o_n(1)$ over the choice of $\mathbf{q}$, the randomness of the dataset S, and its own internal randomness.[5]

As usual, the part of the success probability that depends only on the data structure's random choices may be boosted by a small number of repetitions. Thus, for adversarial queries, it suffices to build a data structure with success probability, say, 1/2, and for correlated queries, it suffices that the part of the success probability depending only on the data structure's random choices is at least 1/2. However, the part of the success probability depending on the query and the dataset cannot be boosted by repetitions. Instead, we need to design a data structure such that under some reasonable assumptions on the item-level probabilities, we achieve the desired success probability.

We say that an event occurs **with high probability** if it occurs with probability $O(1/n^c)$ for a tunable[6] constant c.

We make use of the following weighted Chernoff bound found in e.g. [33, Ex. 4.14].

Lemma 4. *Let $X_1, \dots, X_n$ be independent random variables. Let $a_1, \dots, a_n \in [0, 1]$, $p_1, \dots, p_n \in [0, 1]$, and assume that $\Pr[X_i = a_i] = p_i$, $\Pr[X_i = 0] = 1 - p_i$ for $i \in [n]$. Furthermore, assume that $a_i \leq a$ for all $i \in [n]$, and let $S_n = \sum_{i=1}^n X_i$. Then,*

$$\Pr\left[S_n \geq (1+\varepsilon)\,\mathbf{E}\left[S_n\right]\right] \leq \exp\left(-\frac{\varepsilon^2\,\mathbf{E}[S_n]}{3a}\right), \text{ and}$$
$$\Pr\left[S_n \leq (1-\varepsilon)\,\mathbf{E}\left[S_n\right]\right] \leq \exp\left(-\frac{\varepsilon^2\,\mathbf{E}[S_n]}{2a}\right).$$

3 Data Structure

The data structure follows the locality-sensitive *mapping*, or *filtering*, framework [12, 18, 17]. In this framework, each element $\mathbf{x}$ is mapped to a *set* of filters $F(\mathbf{x})$. This is distinct from locality-sensitive hashing, in which $\mathbf{x}$ would be mapped to a single hash value.

While locality-sensitive filtering is distinct from locality-sensitive hashing, preprocessing and searching follows essentially the same high-level idea.

To search for a query $\mathbf{q}$ we iterate through each filter $f \in F(\mathbf{q})$. For each such filter, we test all vectors $\mathbf{x} \in S$ such that $f \in F(\mathbf{x})$. In other words, we iterate through each vector that has a filter in common with $\mathbf{q}$, and test the similarity of $\mathbf{x}$ and $\mathbf{q}$. If we find a sufficiently close $\mathbf{x}$ we return it; otherwise we return failure after exhausting all $f \in F(\mathbf{q})$.

Thus, the goal of preprocessing is to make it easy to find elements that have a filter in common with a given query. For each filter f mapped to by some element of S, we store a list of all $\mathbf{x}'$ such that $f \in F(\mathbf{x}')$. These lists can be stored and accessed easily (i.e. in a hash table), and this method takes space linear in $\sum_{x \in S} |F(\mathbf{x})|$. We can preprocess quickly by calculating $F(\mathbf{x})$ for all $\mathbf{x} \in S$.

Thus, the goal of our data structure is to define a randomized mapping of vectors to sets of filters satisfying:

- If $\mathbf{x}$ and $\mathbf{q}$ have low similarity, then $F(\mathbf{x}) \cap F(\mathbf{q})$ is small in expectation (guaranteeing small space and fast execution).

[5] Clearly, some assumptions on the item-level probabilities of $\mathcal{D}^n$ are needed for this to be possible. For all of our results, we will assume that $\sum_{i \in [d]} p_i$ is sufficiently large.

[6] We can make c arbitrarily small by adjusting other parameters. For example, in the above discussion, we can boost the probability of success from 1/2 to $O(1/n^c)$ using $\Theta(c \log n)$ independent repetitions.

- If $\mathbf{x}$ and $\mathbf{q}$ have high similarity, then $F(\mathbf{x}) \cap F(\mathbf{q})$ is non-empty with high probability (guaranteeing correctness).

Computing $F(\mathbf{x})$: the choice of paths

Our data structure is based on the Chosen Path data structure of Christiani and Pagh [18]. For our data structure, each $f \in F(\mathbf{x})$ corresponds to a path, i.e., an ordered sequence $(i_1, \ldots, i_\ell) \subseteq [d]^\ell$ where each $i_j \in [d]$ is the index of one of the d dimensions. The construction of $F(\mathbf{x})$ ensures that if $f \in F(\mathbf{x})$, then it must hold that $\mathbf{x}_i = 1$ for all $i \in f$. We say that the path f was *chosen* by $\mathbf{x}$ if $f \in F(\mathbf{x})$.

The data structure comes with a (deterministic) function s which maps each vector $\mathbf{x} \in \{0,1\}^d$, path-length j and bit $i \in [d]$ to a *threshold* $s(\mathbf{x}, j, i) \in [0,1]$. The choice of s depends on the desired application and will be specified later on. In particular, s is how our data structure adapts to the distribution—previous data structures essentially used a constant function for s.[7]

When initializing our data structure, we once and for all select k hash functions, $h_1, \ldots, h_k$, where $h_j : [d]^j \to [0,1]$, each chosen independently from a family $\mathcal{H}$ of pairwise independent hash functions. These hash functions are fixed throughout.

We now explain how to recursively compute the set of paths $F(\mathbf{x})$. Initially, let $F_0(\mathbf{x})$ consist of the empty path. We recursively grow the paths one step at a time; in particular, $F_j(\mathbf{x})$ contains paths of length j. To demonstrate our recursive process, let $v = (i_1, \ldots, i_j)$ be a path in $F_j(\mathbf{x})$. If $\prod_{k=1}^{j} p_{i_k} \leq 1/n$, we stop recursing, and v is a filter of $\mathbf{x}$. Otherwise, we independently consider each set bit i of $\mathbf{x}$ which is not already in v. With probability $s(\mathbf{x}, j, i)$, we concatenate i to the end of v; this results in a new filter $v' \in F_{j+1}(\mathbf{x})$ with $v' = v \circ i$ (where $\circ$ denotes concatenation). This probabilistic choice is made using $h_{j+1}(v \circ i)$.

We formally define this recursive process using the following equation.

$$F_{j+1}(\mathbf{x}) = \left\{ v \circ i \;\middle|\; \begin{array}{l} v = (i_1, \ldots, i_j) \in F_j(\mathbf{x}), \\ \displaystyle\prod_{i_k=1}^{j} p_{i_k} > 1/n, \\ i \in \mathbf{x} \setminus v, \\ h_{j+1}(v \circ i) < s(\mathbf{x}, j, i) \end{array} \right\}.$$

Finally, we define $F(\mathbf{x})$ to be the union of all paths that stopped recursing.

$$F(\mathbf{x}) = \bigcup_{j=1}^{k} \left\{ v = (i_1, \ldots, i_j) \in F_j(\mathbf{x}) \;\middle|\; \prod_{k=1}^{j} p_{i_k} \leq 1/n \right\}.$$

We can calculate $F(\mathbf{x})$ in $O(d|F(\mathbf{x})|)$ time.

Preprocessing

During the preprocessing, we randomly select the k hash functions and compute $F(\mathbf{x})$ for each $\mathbf{x}$. Then, we use a standard dictionary data structure to construct an inverted index such that for each $f \in \cup_{\mathbf{x} \in S} F(\mathbf{x})$, we can look up $\{\mathbf{x} \in S \colon f \in F(\mathbf{x})\}$.

Answering a query

For a given query $\mathbf{q}$ we compute its chosen paths $F(\mathbf{q})$ as described above (using the same hash functions as in the preprocessing). For each $f \in F(\mathbf{q})$, we then compute $B(\mathbf{x}, \mathbf{q})$ for every $\mathbf{x}$ which chose the path f, i.e., for which $f \in F(\mathbf{x})$. If an $\mathbf{x}$ with similarity at least b_1 is found then we return $\mathbf{x}$. If we have exhausted all candidates without finding such an $\mathbf{x}$, we report that no high-similarity vector was found.

[7] As mentioned previously, this is not the only distinguishing technical detail. We must also sample without replacement and adjust the path length dynamically based on the probabilities of the sampled bits.

4 Correctness

We begin with a structural lemma on the number of paths two vectors $\mathbf{x}$ and $\mathbf{q}$ have in common. We will use this lemma to prove correctness of both of our data structures. This lemma follows the same high-level idea of the proof of correctness used in [18], but must be generalized to handle the distribution-dependent choices made by the data structures. The proof has been moved to Section 10.

Lemma 5. *Suppose that for $\mathbf{x} \in \{0,1\}^d$ and $\mathbf{q} \in \{0,1\}^d$ the following holds: For every $1 \le j \le k$ and every $v = (i_1, \ldots, i_j) \in [d]^j$, we have*

$$\sum_{i \in (\mathbf{x} \cap \mathbf{q}) \setminus v} \min\{s(\mathbf{x}, j, i), s(\mathbf{q}, j, i)\} \ge 1. \tag{1}$$

Then, $\Pr[F(\mathbf{x}) \cap F(\mathbf{q}) \neq \emptyset] \ge 1/\log n$.

The following lemma is used to prove performance of our algorithms. We use an inductive argument to bound the number of paths generated by the data structure. This argument depends crucially on both the thresholds $s(\mathbf{x}, j, i)$ and the distribution-dependent stopping rule.

Lemma 6. *Let $\mathbf{x} \in \{0,1\}^d$, and let ρ be such that $\sum_{i \in \mathbf{x}} p_i^\rho s(\mathbf{x}, j, i) \le c$. Then $\mathbf{E}[|F(\mathbf{x})|] = O(n^\rho c^{\log n})$. Furthermore, the expected time spend on computing $F(\mathbf{x})$ is $O(n^\rho c^{\log n} |\mathbf{x}|)$*

Proof. For $v \in F_j(\mathbf{x})$, define the random variables $Y(\mathbf{x}, v, i) = \mathbf{1}_{h_j(v \circ i) \le s(\mathbf{x}, j, i)}$. Let $F_j^t(\mathbf{x}) \subseteq F_j(\mathbf{x})$ be the set of paths $v \in F_j(\mathbf{x})$ such that $\sum_{i \in v} \log 1/p_i \le t$. Furthermore, let $F_j^t(\mathbf{x}, i) \subseteq F_j^t(\mathbf{x})$ be the set of paths $v \in F_j^t(x)$ for which $i \notin v$. We claim that $\mathbf{E}[|F_j^t|]$ is at most $2^{\rho t} c^j$. The proof is by induction over j and t. Note that for every $v \in F_{j+1}^t$, there must exist $i \in \mathbf{x}$ and $v' \in F^{t - \log(1/p_i)}(\mathbf{x}, i)$ such that $v = v \circ i'$ and $Y(\mathbf{x}, v, i) = 1$. Thus,

$$\begin{aligned}
\mathbf{E}[|F_{j+1}^t|] &= \mathbf{E}\left[\sum_{i \in \mathbf{x}} \sum_{v \in F_j^{t-\log(1/p_i)}(\mathbf{x}, i)} Y(\mathbf{x}, v, i)\right] \\
&= \mathbf{E}\left[\sum_{i \in \mathbf{x}} \sum_{v \in F_j^{t-\log(1/p_i)}(\mathbf{x}, i)} \mathbf{E}[Y(\mathbf{x}, v, i)]\right] \\
&= \mathbf{E}\left[\sum_{i \in \mathbf{x}} \sum_{v \in F_j^{t-\log(1/p_i)}(\mathbf{x}, i)} s(\mathbf{x}, j, i)\right] \\
&\le \sum_{i \in \mathbf{x}} \mathbf{E}[|F_j^{t-\log(1/p_i)}(\mathbf{x}, i)|] s(\mathbf{x}, j, i) \\
&\le \sum_{i \in \mathbf{x}} \mathbf{E}[|F_j^{t-\log(1/p_i)}(\mathbf{x})|] s(\mathbf{x}, j, i) \\
&\le \sum_{i \in \mathbf{x}} 2^{\rho(t - \log(1/p_i))} c^j s(\mathbf{x}, j, i) \\
&\le 2^{\rho t} c^j \sum_{i \in \mathbf{x}} p_i^\rho s(\mathbf{x}, j, i) \\
&\le 2^{\rho t} c^{j+1}.
\end{aligned}$$

Now, for every $v \in F(\mathbf{x})$ there must exist j and $i \in \mathbf{x}$ such that $v = v' \circ i$ for some $v' \in F_j^{\log n}(\mathbf{x})$. (The $\log n$

follows from taking the log of both sides of $\prod_{i_k} p_{i_k} \leq 1/n$). It follows that

$$\begin{aligned}
\mathbf{E}[|F(\mathbf{x})|] &= \mathbf{E}\left[\sum_{j=0}^{\log n}\sum_{i\in\mathbf{x}}\sum_{v\in F_j^{\log n}(\mathbf{x},i)} Y(\mathbf{x},v,i)\right] \\
&\leq \sum_{j=0}^{\log n}\mathbf{E}[|F_j^{\log n}|]\sum_{i\in x} s(\mathbf{x},j,i) \\
&\leq 2^{\rho\log n}c^{\log(n)+1}\sum_{i\in\mathbf{x}} s(\mathbf{x},j,i) \\
&= O(n^\rho c^{\log n}).
\end{aligned}$$

We now bound the expected time for computing $F(\mathbf{x})$. To this end, note that we spend at most $O(|\mathbf{x}|)$ time in each recursive step, and that the expected number of recursive steps is at most $\sum_{j=0}^{\log n}\mathbf{E}[|F_j^{\log(n)}|] = O(n^\rho c^{\log n})$. □

The next lemma shows that because we stop each branch of our recursive process once the expected number of vectors from S which choses this path is constant, it follows that the expected query time is linear in $|F(\mathbf{q})|$.

Lemma 7. *Let* $\mathbf{q}\in\{0,1\}^d$, *and let* ρ *be such that* $\sum_{i\in\mathbf{x}} p_i^\rho s(\mathbf{q},j,i) \leq c$. *Furthermore, let* $S\sim\mathcal{D}^n[p_1,\ldots,p_d]$. *Then* $\mathbf{E}[\sum_{\mathbf{x}\in S}|F(\mathbf{q})\cap F(\mathbf{x})|] = O(n^\rho c^{\log n})$.

Proof. From Lemma 6, we know that $\mathbf{E}[|F(\mathbf{q})|] = O(n^\rho c^{\log n})$. Let $v = (i_1,\ldots,i_j) \in F(\mathbf{q})$ be any path chosen by $\mathbf{q}$. By definition, $\Pr[v\in F(\mathbf{x})] \leq \Pr[\mathbf{x}_{i_1} = 1\wedge\cdots\wedge\mathbf{x}_{i_j} = 1] \leq 1/n$. Thus,

$$\begin{aligned}
\mathbf{E}\left[\sum_{\mathbf{x}\in S}|F(\mathbf{q})\cap F(\mathbf{x})|\right] &= \mathbf{E}\left[\sum_{v\in F(\mathbf{q})}\sum_{\mathbf{x}\in S}\mathbf{1}_{\{v\in F(\mathbf{x})\}}\right] \\
&= \mathbf{E}\left[\sum_{v\in F(\mathbf{q})}\sum_{\mathbf{x}\in S}\mathbf{E}[\mathbf{1}_{\{v\in F(\mathbf{x})\}}]\right] \\
&\leq \mathbf{E}\left[\sum_{v\in F(\mathbf{q})}\sum_{\mathbf{x}\in S} 1/n\right] \\
&= \mathbf{E}[|F(\mathbf{q})|] \\
&= O(n^\rho c^{\log n}).
\end{aligned}$$

□

5 Adversarial Queries

For the adversarial case, we set $s(\mathbf{x},j,i) = \frac{1}{b_1|\mathbf{x}|-j}$. Thus, the sampling thresholds only depend on $|\mathbf{x}|$ and the number of bits already contained in the path v. The remainder of the data structure follows the description in Section 3.

5.1 Correctness

We begin with a proof of correctness, giving a guarantee that if there exists an $\mathbf{x}$ similar to a query $\mathbf{q}$, then $\mathbf{x}$ and $\mathbf{q}$ will share a filter (so $\mathbf{x}$ will be found by our algorithm).

Fix $\mathbf{x}$ and $\mathbf{q}$ such that $B(\mathbf{x},\mathbf{q}) \geq b_1$ and fix $v = (i_1,\dots,i_j) \in [d]^j$. The assumption $B(\mathbf{x},\mathbf{q}) \geq b_1$ is equivalent to $|\mathbf{x} \cap \mathbf{q}| \geq b_1 \max\{|\mathbf{x}|,|\mathbf{q}|\}$. Thus,

$$\begin{aligned}
\sum_{i \in (\mathbf{x}\cap\mathbf{q})\setminus v} \min\{s(\mathbf{x},j,i), s(\mathbf{x},j,i)\} &= \sum_{i \in (\mathbf{x}\cap\mathbf{q})\setminus v} \frac{1}{b_1 \max\{|\mathbf{x}|,|\mathbf{q}|\} - j} \\
&= \frac{|\mathbf{x}\cap\mathbf{q}| - j}{b_1 \max\{|\mathbf{x}|,|\mathbf{q}|\} - j} \\
&\geq 1.
\end{aligned}$$

Correctness follows immediately from Lemma 5.

5.2 Performance guarantees

Query time

We bound the query time using a constant ρ that depends only on b_1 and $\mathcal{D}$.

Lemma 8. *For every $\varepsilon > 0$ there exists a constant C such that if $\sum_{i\in\mathbf{q}} p_i^\rho \leq b_1|\mathbf{q}|$ and $|\mathbf{q}| \geq C\log n$, then $\mathbf{E}[|F(\mathbf{q})|] = O(n^{\rho+\varepsilon})$.*

Proof. First, if $|\mathbf{q}| \geq C\log n$ for some $C > 1$, then

$$\begin{aligned}
s(\mathbf{q},j,i) &= \frac{1}{b_1|\mathbf{q}| - j} \\
&\leq \frac{1}{b_1|\mathbf{q}| - \log n} \\
&= \frac{1}{b_1|\mathbf{q}|}\frac{1}{1-(\log n)/(b_1|\mathbf{q}|)} \\
&\leq \frac{1}{b_1|\mathbf{q}|}\frac{1}{1-1/(b_1C)}.
\end{aligned}$$

Thus,

$$\begin{aligned}
\sum_{i\in\mathbf{q}} p_i^\rho s(\mathbf{q},j,i) &\leq \sum_{i\in\mathbf{q}} \frac{p_i^\rho}{b_1|\mathbf{q}|}\frac{1}{1-1/(b_1C)} \\
&\leq \frac{1}{1-1/(b_1C)}.
\end{aligned}$$

It follows from Lemma 6 that $\mathbf{E}[|F(\mathbf{q})|] = O\left(n^\rho\left(\frac{1}{1-1/(b_1C)}\right)^{\log n}\right)$. □

Preprocessing time

The following lemma bounds our expected per-element processing time. The proof has been moved to Section 10.

Lemma 9. *For every $\varepsilon > 0$ there exists a constant C such that if $\sum_{i\in[d]} p_i^{1+\rho} = b_1\sum_{i\in[d]} p_i$ and $\sum_{i\in[d]} p_i \geq C\log n$, then $\mathbf{E}[|F(\mathbf{x})|] = O(n^{\rho+\varepsilon})$ for $\mathbf{x} \sim \mathcal{D}[p_1,\dots,p_d]$.*

We multiply this by n to get the expected total preprocessing time for all elements. Since our data structure takes space linear in the total size of the stored filters, we similarly obtain $O(n^{1+\rho+\varepsilon})$ space.

6 Correlated Queries

We need samples from the distribution to have sufficient length such that we can guarantee that $B(\mathbf{q}, \mathbf{x}) > B(\mathbf{q}, \mathbf{x}')$ for $\mathbf{x}' \sim \mathcal{D}$ which is not correlated with $\mathbf{q}$ (see Lemma 10). Therefore, we assume there is a sufficiently large constant C that the expected size of an element drawn from $\mathcal{D}$ is at least $\sum_{i \in [d]} p_i \geq C \log n$. Similarly, we assume that all $p_i \leq \alpha/2$. We assume that $C \gg 1/\alpha$.

In the correlated case, we no longer need to sample vertices uniformly at random. The query $\mathbf{q}$ and the distribution $\mathcal{D}$ give us further information about where $\mathbf{x}$ and $\mathbf{q}$ are likely to intersect. In particular, we can calculate the conditional probability

$$\widehat{p_i} = \Pr(\mathbf{x}_i = 1 \mid \mathbf{q}_i = 1) = p_i(1 - \alpha) + \alpha.$$

We weight our chosen path choices by this conditional probability. We then increase each sampling probability by a small constant $1 + \delta = 1 + 3/(\sqrt{\alpha C})$; this will help us ensure correctness for filters. (We derive this constant in the proof of Lemma 11. A smaller constant is likely sufficient in practice, particularly for small α.) Thus, at round j, we sample each bit $i \in \mathbf{x}$, without replacement, with probability

$$s(\mathbf{x}, j, i) = \frac{1 + 3/\sqrt{\alpha C}}{\widehat{p_i} C(\log n) - j}.$$

With these new sampling probabilities, we again maintain paths as defined in Section 3. To answer a query $\mathbf{q}$, we look through all $\mathbf{x}' \in F(\mathbf{q})$, returning any $\mathbf{x}'$ that has similarity at least $b_1 = \alpha/1.3$.

6.1 Correctness

For a query $\mathbf{q}$, our data structure attempts to find the $\mathbf{x}$ that is α-correlated with $\mathbf{q}$ by finding a similar vector among the items stored for filters in $F(\mathbf{q})$. We begin by showing that with high probability, $B(\mathbf{x}, \mathbf{q}) > b_1$; meanwhile, for uncorrelated $\mathbf{x}'$, $B(\mathbf{x}', \mathbf{q}) < b_1$.

Lemma 10. *Assume $q \sim \mathcal{D}_\alpha(\mathbf{x})$. With high probability, $B(\mathbf{x}, \mathbf{q}) \geq \alpha/1.3$. Meanwhile, for all $\mathbf{x}' \in S$ not correlated with $\mathbf{q}$, $B(\mathbf{x}', \mathbf{q}) \leq \alpha/1.5$ with high probability.*

Proof. To begin, note that for any $\mathbf{x}' \sim \mathcal{D}$, $\mathbf{E}[|\mathbf{x}|] = C \log n$. By Chernoff bounds (Lemma 4 for example) and the union bound, $\sqrt{1.3}(C \log n) \geq \min\{|\mathbf{x}'|, |\mathbf{q}|\} \geq (C \log n)\sqrt{3}/2$ with high probability for all $\mathbf{x}'$ and $\mathbf{q}$.

First, consider the correlated pair: $\mathbf{q} \sim \mathcal{D}_\alpha(\mathbf{x})$. We have $\mathbf{E}[|\mathbf{x} \cap \mathbf{q}|] = \sum_i p_i^2(1 - \alpha) + p_i \alpha \geq \alpha C \log n$. Again by Chernoff, $|\mathbf{x} \cap \mathbf{q}| \geq \alpha(C \log n)/\sqrt{1.3}$ with high probability. Combining this with the above, $B(\mathbf{x}, \mathbf{q}) \geq \alpha/1.3$ with high probability.

Now, consider an uncorrelated pair $\mathbf{x}'$ and $\mathbf{q}$ drawn independently from $\mathcal{D}$. We have $\mathbf{E}[|x \cap \mathbf{q}|] = \sum_i p_i^2 \leq (\alpha C \log n)/2$. Using Chernoff bounds, $\mathbf{E}[|x \cap \mathbf{q}|] \leq (2/\sqrt{3})(\alpha C \log n)/2$ with high probability. Combining with the above, $B(\mathbf{x}, \mathbf{q}) \leq \alpha/1.5$ with high probability. □

Now we show that our sampling probabilities are large enough to guarantee that the two α-correlated vectors $\mathbf{x}$ and $\mathbf{q}$ are likely to share a path. We need to use new techniques beyond those in Section 5 (and beyond previous results) because we our proof must leverage that the close pair is chosen randomly. After all, if we ignore this aspect, we have reduced to the adversarial case—we want to improve those bounds.

We do this by showing that, with high probability, any given path during our recursive process satisfies the requirements of Lemma 5. Note that we prove correctness with probability at least $1 - 1/n^2$ for simplicity; we can obtain stronger bounds by slightly increasing δ.

Lemma 11. *Consider a path v of length at most $\log n$, and $q \sim \mathcal{D}_\alpha(\mathbf{x})$. Assume $C\alpha \geq 15$. Then, with probability at least $1 - 1/n^2$,*

$$\sum_{i \in (\mathbf{x} \cap \mathbf{q}) \setminus v} s(\mathbf{x}, |v|, i) \geq 1.$$

Proof. We begin by calculating the expected value.

$$\begin{aligned}\mathbf{E}\left[\sum_{i\in(\mathbf{x}\cap\mathbf{q})\setminus v} s(\mathbf{x},|v|,i)\right] &= \sum_{i\in[d]\setminus v} \Pr(i\in\mathbf{x}\cap\mathbf{q})\frac{1+\delta}{\widehat{p_i}C(\log n)-j}\\ &\geq \sum_{i\in[d]\setminus v} p_i\frac{(1+\delta)}{C(\log n)-j/\widehat{p_i}}\\ &\geq \frac{(1+\delta)(C\log n-\sum_{i\in v}p_i)}{C(\log n)-j}\\ &\geq 1+\delta.\end{aligned}$$

We have $\widehat{p_i}\geq\alpha$, so $s(\mathbf{x},|v|,i)\leq(1+\delta)/(\alpha C\log n)$. Then by Lemma 4,

$$\Pr\left(\sum_{x\in(\mathbf{x}\cap\mathbf{q})\setminus v} s(\mathbf{x},|v|,i)\leq 1\right)\leq\exp\left(-\frac{\alpha C\log n}{2}\left(\frac{\delta}{1+\delta}\right)^2\right).$$

Assume that $C\alpha\geq 15$. Then we assign

$$\delta=\frac{3}{\sqrt{\alpha C}}\geq\frac{2\sqrt{\ln 2/(\alpha C)}}{1-2\sqrt{\ln 2/(\alpha C)}}.$$

Substituting, we obtain the lemma. □

Applying union bound, all paths in our recursive process satisfy Lemma 5. Thus, these two lemmas imply that we return the α-correlated $\mathbf{x}$ with high probability.

6.2 Performance Guarantees

We now compute the expected number of paths chosen by an $\mathbf{x}\sim\mathcal{D}$. Since we have both $\mathbf{q}\sim\mathcal{D}$ and $\mathbf{x}\sim\mathcal{D}$ for all $\mathbf{x}\in S$, this lemma implies the query time and space bounds of Theorem 1 immediately.

Lemma 12. *For every $\varepsilon>0$ there exists a constant C such that if $\sum_{i\in[d]}\frac{p_i^{1+\rho}}{\widehat{p_i}}=\sum_{i\in[d]}p_i$ and $\sum_{i\in[d]}p_i=C\log n$, then $\mathbf{E}[|F(\mathbf{x})|]=O(n^{\rho+\varepsilon})$ for $\mathbf{x}\sim\mathcal{D}[p_1,\dots,p_d]$.*

Proof. We want to show that if C is sufficiently large, then $\mathbf{E}_{\mathcal{H},\mathbf{x}\sim\mathcal{D}}[|F(\mathbf{x})|]=O(n^{\rho+\varepsilon})$ where the expectation is over both the randomness $\mathcal{H}$ of the data structure and the randomness of $\mathbf{x}$. Since $\widehat{p_i}\geq\alpha$ and $j\leq\log n$, we have

$$\begin{aligned}s(\mathbf{x},j,i) &= \frac{1}{\widehat{p_i}C\log n-j}\\ &\leq \frac{1}{\widehat{p_i}C\log n-\log n}\\ &\leq \frac{1}{\widehat{p_i}C\log n}\cdot\frac{1}{1-\frac{\log n}{\widehat{p_i}C\log n}}\\ &\leq \frac{1}{\widehat{p_i}C\log n}\frac{1}{\alpha C}.\end{aligned}$$

Furthermore, by Lemma 4,

$$\begin{aligned}\Pr\left[\sum_{i\in\mathbf{x}}p_i^{\rho}/\widehat{p_i}\leq(1+C^{-1/3})\sum_{i\in[d]}p_i^{1+\rho}/\widehat{p_i}\right] &\geq 1-\exp\left(-\alpha\frac{(C^{-1/3})^2}{3}\sum_{i\in[d]}p_i^{1+\rho}/\widehat{p_i}\right)\\ &\geq 1-\exp\left(-\alpha\frac{C^{-2/3}}{3}C\log(n)\right)\\ &\geq 1-n^{\alpha\frac{b_1C^{1/3}}{3}}.\end{aligned}$$

Thus, with probability $1 - n^{\Omega(C^{1/3})}$, we have

$$\begin{aligned}\sum_{i \in \mathbf{x}} p_i^\rho s(\mathbf{x}, j, i) &\leq \sum_{i \in \mathbf{x}} p_i^\rho \frac{1}{\widehat{p}_i C \log n} \frac{1}{\alpha C} \\ &\leq \frac{1 + C^{-1/3}}{C \log(n) \alpha C} \sum_{i \in [d]} p_i^{1+\rho} / \widehat{p}_i \\ &= \frac{1 + C^{-1/3}}{\alpha C}.\end{aligned}$$

Lemma 6 yields $\Pr_{\mathbf{x} \sim \mathcal{D}}[\mathbf{E}_{\mathcal{H}}[F(\mathbf{x})] \leq n^{\rho + \varepsilon}] \geq 1 - n^{\Omega(C^{1/3})}$. Since $\mathbf{E}_{\mathcal{H}}[|F(\mathbf{x})|]$ is polynomial in n for any $\mathbf{x}$, this implies that $\mathbf{E}_{\mathcal{H}, \mathbf{x} \sim \mathcal{D}}[|F(\mathbf{x})|] = O(n^{\rho + \varepsilon})$. □

7 Performance Comparison

The bounds achieved by our data structures are given as the solution to an equation which is not in closed form. In this section, we give some examples and intuition of how much speedup we achieve.

7.1 Adversarial query

If the query is adversarial, then the query time for a query $\mathbf{q}$ is determined by the smallest ρ such that

$$\sum_{i \in \mathbf{q}} p_i^\rho \leq b_1 |\mathbf{q}|.$$

We will now discuss to what extent our data structure is able to take advantage of possible skew in the item-level probabilities. In order to simplify the discussion, assume that $|\mathbf{q}| = \sum_{i \in [d]} p_i$ (so that $|\mathbf{q}|$ equals the expected value of $|\mathbf{x}|$ when $\mathbf{x} \sim \mathcal{D}$, meaning that all sets have roughly the same size). Now, for every $\varepsilon > 0$ there exists a C such that if $\sum_{i \in [d]} p_i \geq C \log n$, then $B(\mathbf{x}, \mathbf{q}) = \frac{1}{|\mathbf{q}|} \sum_{i \in \mathbf{q}} p_i \pm \varepsilon$ for every $\mathbf{x} \in S$ with high probability. Thus, we can solve the problem by solving the (b_1, b_2)-approximate Braun-Blanquet similarity search problem using the standard Chosen Path data structure, thereby obtaining a ρ-value which, for sufficiently large C, is arbitrarily close to

$$\rho_{\mathsf{CP}} = \frac{\log(b_1)}{\log\left(\frac{1}{|\mathbf{q}|} \sum_{i \in \mathbf{q}} p_i\right)}.$$

If the part of the input distribution relevant for the query $\mathbf{q}$ contains no skew, i.e., if $p_i = p$ for all $i \in \mathbf{q}$, then $\rho_{\mathsf{CP}} = \rho$. However, $\rho < \rho_{\mathsf{CP}} = \log(b_1)/\log(p)$ in all other cases.

To our knowledge there is no closed-form expression for the solution ρ to the equation $\sum_{i \in \mathbf{q}} p_i^\rho \leq b_1 |\mathbf{q}|$. Instead, we will compute ρ in a few settings to illustrate how skew influences our query time. Suppose that $\mathbf{q}$ consists of two types of bits: half the bits of $\mathbf{q}$ are set with probability p_a in a random set from S, and the other half is set with probability p_b. Furthermore, assume that $\sum_{i \in [d]} p_i = |\mathbf{q}| = \Theta(\log n)$.

Suppose first that the distribution has very significant skew, e.g., $p_a = 1/4$, $p_b = n^{-0.9}$. Assume that we are searching for a set in S with similarity at least, say, $b_1 = 1/3$. In this case, $\rho_{\mathsf{CP}} \geq \frac{\log(1/3)}{\log(1/8)} \geq 0.528$, whereas we obtain a ρ-value of $\rho = \frac{\log(2/3)}{\log(1/4)} + o_n(1) \leq 0.293$ for n sufficiently large. Prefix filtering does not give any non-trivial (worst-case) performance guarantee. This example shows that we can take advantage of the skew even when it is possible that the entire intersection between $\mathbf{q}$ and the close point $\mathbf{x}$ is within a part of $\mathbf{q}$ where all bits are set with the same probability in a datapoint. A more extreme example occurs if we increase b_1 to $b_1 = 2/3$ so that a possible close point $\mathbf{x}$ must share some of the bits for which the associated probability is $p_b = n^{-0.9}$. In this case, equation (7.1) is satisfied for ρ arbitrarily close to zero. Thus, we achieve a very fast query time (in particular, our query time will be $O(n^\varepsilon)$ for every constant $\varepsilon > 0$). In order to understand why, note that no path in $F(\mathbf{q})$ can contain more than two bits with associated item-level probability $n^{-0.9}$. Furthermore, since $b_1 > 1/2$, our sampling threshold $s(\mathbf{q}, j, i)$ is low enough that we

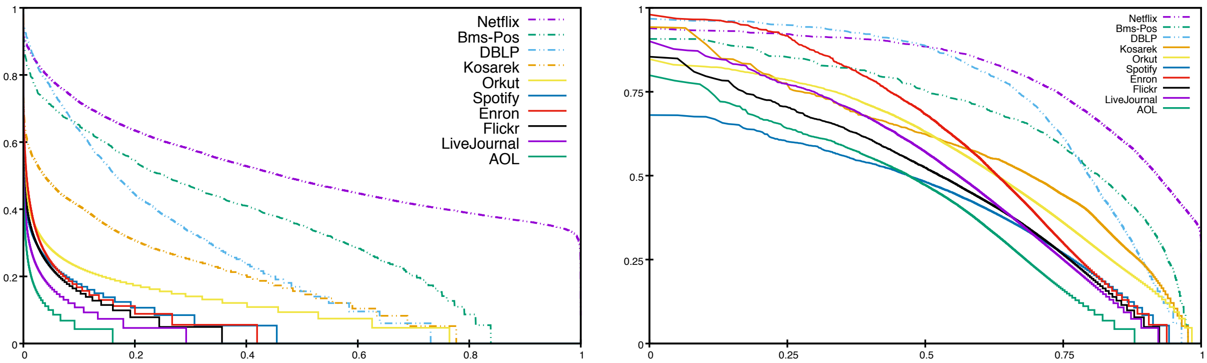

Figure 2: Frequency distributions of real datasets from the set similarity search benchmark of Mann et al. [31], with frequencies p_j in decreasing order, plotted in two ways. In both figures, for each $j \in [d]$, the y-axis denotes $1 + \log_n p_j$. On the left, the x-axis denotes j/d; on the right, the x-axis denotes $\log_d j$.

are very unlikely to create long paths consisting only of bits with associated probability $1/4$. Indeed, the expected size of a path in $F(\mathbf{q})$ is $O(1)$ in this case. Solving the problem by solving the (b_1, b_2)-approximate problem will yield a ρ value of $\rho_{\mathsf{CP}} = \log(2/3)/\log(1/8) = 0.194$. Prefix filtering will need $\Omega(n^{0.1})$ time to locate a possible close point if it exists.

7.2 Correlated query

When the query is α-correlated with a point $\mathbf{x}$ in our dataset S, the running time is (for $\sum_{i\in[d]} p_i$ sufficiently large) determined by the smallest ρ such that $\sum_{i\in[d]} p_i^{1+\rho}/\widehat{p}_i \leq \sum_{i\in[d]} p_i$ where $\widehat{p}_i = (1-\alpha)p_i + \alpha$.

As for the adversarial case, whenever there is significant skew in the input distribution, we can get very fast query time. Suppose for example that $\mathcal{D}[p_1, \ldots, p_d]$ is such that $4C\log(n)$ bits are set to 1 with probability $p_a = 1/4$ and $n^{9/10}C\log(n)$ bits are set to 1 with probability $p_b = n^{-9/10}$. If $\alpha = 2/3$, then we find that our expected query time is $O(n^{\varepsilon})$ for every constant $\varepsilon > 0$. On the other hand, prefix filtering takes $\Omega(n^{0.1})$ time.

Let us now consider examples where the input distribution is skewed but all probabilities are $\Omega(1)$, meaning in particular that prefix filtering does not give any non-trivial worst-case guarantees. For example, suppose that $C\log n$ bits are set to 1 with probability $p_a = \Theta(1)$ and $C\log n$ bits are set to 1 with probability $p_b = \Theta(1)$. We compare the performance of our data structure to that of using Chosen Path for solving the (b_1, b_2)-approximate problem where b_1 is the expected similarity between the correlated points and b_2 is the expected similarity between the query and an uncorrelated point. For $p_a = p$ and $p_b = p/8$ and $\alpha = 2/3$, we plot the corresponding ρ-values in Figure 1.

8 Skew in Real Data Sets

In this section we examine how the ideas that inspired our model apply to the real-life sparse data sets used to evaluate similarity set methods by Mann et al. [31].

Skew

For each data set we computed the empirical frequencies p_j, indexed such that frequencies decrease as j grows. In Figure 2 we show the distribution of frequencies p_j in two ways. On the left side we plot $\log_n(p_j n)$ against j/d, where n is the number of vectors and d is the dimension of the vectors. On the right side we show the distribution on a normalized log-log scale. As can be seen, all data sets display a significant skew. A plain Zipfian distribution would appear linear on this plot (with slope corresponding to the exponent in the distribution). Frequencies do not follow a Zipfian distribution, but are generally close to a "piecewise Zipfian" distribution. For almost all data sets, the frequencies outside of the very top can be thought of as being bounded by $p_j \leq n^{-\gamma}$ for some constant $\gamma > 0$.

Approximate Independence

Our theoretical analysis is based on the assumption of independence between the set bits (i.e. the items) of the random vectors, and more specifically on the inequality:

$$\Pr_{\mathbf{x}\in S}[\forall_{j\in I}\mathbf{x}_j = 1] \leq \prod_{j\in I} p_j, \text{ for } I \subseteq [d] \ . \tag{2}$$

The asymptotic analysis still applies if (2) holds only up to a constant factor. (Under some conditions we may even let this factor increase exponentially with $|I|$.) To shed light on whether this assumption is realistic we considered random size-2 and size-3 subsets I, respectively, and computed the expected number of vectors $\mathbf{x}$ satisfying $\forall_{j\in I}\mathbf{x}_j = 1$ under the independence assumption and in the data set, respectively. The ratio of these numbers is the constant factor needed, on average over all sets I, to make (2) hold.

The ratios are shown in Table 1. If the independence assumption were true, the ratios would be close to 1. As can be seen all data sets have some kind of positive correlation between dimensions meaning that there are more pairs/triples than the independence model predicts. However, for most of the data sets it seems reasonable to assume that these correlations are weak enough that (2) holds up to a constant factor independent of n, at least for small constant $|I|$. In those cases we expect our theoretical analysis based on independence to be indicative of actual running time.

Data set	$\|I\| = 2$	$\|I\| = 3$
AOL	1.2	3.9
BMS-POS	1.5	3.9
DBLP	1.4	2.3
ENRON	2.9	21.8
FLICKR	1.7	4.9
KOSARAK	7.1	269.4
LIVEJOURNAL	2.3	7.3
NETFLIX	3.1	24.0
ORKUT	4.0	37.9
SPOTIFY	24.7	6022.1

Table 1: We computed the ratio bewteen $\mathbf{E}_I[\Pr_{\mathbf{x}\in S}[\forall_{j\in I}\mathbf{x}_j = 1]]$ (expected observed number of vectors with 1s in I) and $\mathbf{E}_I[\prod_{j\in I} p_j]$ (expected number of vectors with 1s in I assuming independence). The expectations are computed for I chosen uniformly from subsets of $[d]$ of size 2 and 3, respectively.

9 Conclusion

Several open problems remain about how to handle nearest neighbor search in skewed datasets.

One natural question is how to relax the assumption that all item probabilities p_i are known to the algorithm beforehand. It seems likely that one can estimate each p_i to very high precision by counting the occurrences in the dataset itself, leading to the same asymptotic bounds. We note that the set similarity join algorithm in [19] avoids using or estimating p_i, but we do not know how to analyze its performance in our setting.

The performance examples in this paper deal with distributions with two types of item: very rare and very common items. In practice, one more often encounters distributions with much more gradual skew, such as a Zipf distribution. Unfortunately, sets selected using a Zipf distribution have very small expected size, which trivializes the asymptotics. It would be interesting to find a class of distributions that accurately characterizes the skew of real data while remaining interesting for asymptotic analysis. Such a distribution would be an important use case for our algorithm.

Finally, it would be interesting to relax the independence assumption. Our experiments gave some evidence that many of the datasets we studied have only mild dependencies, but for some this was not the case (particularly the Spotify dataset). In fact, the Spotify dataset has recently been observed to be a difficult case for a variant of the Chosen Path algorithm [19], possibly due to these correlations. It seems that if the correlations are "simple" and known ahead of time, there may be strategies to deal with them when sampling paths. Such an algorithm would loosen the independence assumption in our model, and has the potential to lead to much stronger performance in practice.

10 Omitted Proofs

Proof of Lemma 5. In order to simplify calculations, we are going to consider a conveniently chosen subset of $F(\mathbf{x}) \cap F(\mathbf{q})$. In order to define this subset, for $v \in F_j(\mathbf{x}) \cap F_j(\mathbf{q})$ and $i \in (\mathbf{x} \cap \mathbf{q}) \setminus v$ let,

$$s(v \circ i) = \frac{\min\{s(\mathbf{x}, j, i), s(\mathbf{q}, j, i)\}}{\sum_{i' \in (\mathbf{x} \cap \mathbf{q}) \setminus v} \min\{s(\mathbf{x}, j, i'), s(\mathbf{q}, j, i')\}}.$$

We define M_j and M exactly as $F_j(\mathbf{x}) \cap F_j(\mathbf{q})$ and $F(\mathbf{x}) \cap F(\mathbf{q})$, except that we replace the requirements $h_j(v \circ i) \leq s(\mathbf{x}, j, i)$ and $h_j(v \circ i) \leq s(\mathbf{q}, j, i)$ with the requirement $h_j(v \circ i) \leq s(v \circ i)$. It follows from the assumption (1) that M is indeed a subset of $F(\mathbf{x}) \cap F(\mathbf{q})$. We claim that $\Pr[M \neq \emptyset] \geq 1/\log n$. In order to show this, we will make use of the following second moment bound:

$$\Pr[M_j \neq \emptyset] \geq \frac{\mathbf{E}[|M_j|]^2}{\mathbf{E}[|M_j|^2]}. \tag{3}$$

From (3), it suffices to show $\mathbf{E}[|M_j|^2] \leq j + 1$ and $\mathbf{E}[|M_j|] = 1$ for all $j \geq 0$. We will prove that this holds by induction. Recall that we start with a single empty path, and so $|M_0| = 1$ with probability 1. Thus, the statement trivially holds for $j = 0$. Let $j > 0$. Define the random variable $Y(v \circ i) = \mathbf{1}_{h_j(v \circ i) \leq s(v \circ i)}$. Note that this random variable is independent of M_{j-1}. In particular, for every possible m_{j-1}, we have $\mathbf{E}[Y(v \circ i) | M_{j-1} = m_{j-1}] = \mathbf{E}[Y(v \circ i)] = s(v \circ i)$. Using this independence along with (1) yields

$$\begin{aligned}
\mathbf{E}[M_j] &= \mathbf{E}\left[\sum_{v \in M_{j-1}} \sum_{i \in (\mathbf{x} \cap \mathbf{q}) \setminus v} Y(v \circ i)\right] \\
&= \mathbf{E}\left[\sum_{v \in M_{j-1}} \sum_{i \in (\mathbf{x} \cap \mathbf{q}) \setminus v} \mathbf{E}[Y(v \circ i)]\right] \\
&= \mathbf{E}\left[\sum_{v \in M_{j-1}} \sum_{i \in (\mathbf{x} \cap \mathbf{q}) \setminus v} s(v \circ i)\right] \\
&\geq \mathbf{E}\left[\sum_{v \in M_{j-1}} 1\right] = \mathbf{E}[|M_{j-1}|].
\end{aligned}$$

Recall that the hash-function h_j was chosen from a pairwise independent family. Thus, for $v \circ i \neq v' \circ i'$, we have $\mathbf{E}[Y(v \circ i) Y(v' \circ i')] = \mathbf{E}[Y(v \circ i)]\, \mathbf{E}[Y(v' \circ i')]$. Then,

$$\begin{aligned}
\mathbf{E}[|M_j|^2] =&\ \mathbf{E}\left[\left(\sum_{v \in M_{j-1}} \sum_{i \in (\mathbf{x} \cap \mathbf{q}) \setminus v} Y(v \circ i)\right)^2\right] \\
=&\ \mathbf{E}\left[\sum_{v \in M_{j-1}, i \in (\mathbf{x} \cap \mathbf{q}) \setminus v} Y(v \circ i)^2\right] + \mathbf{E}\left[\sum_{\substack{v \in M_{j-1}, i \in (\mathbf{x} \cap \mathbf{q}) \setminus v \\ u \in M_{j-1}, i' \in (\mathbf{x} \cap \mathbf{q}) \setminus v' \\ v \circ i \neq v' \circ i'}} Y(v \circ i) Y(v' \circ i')\right] \\
=&\ \mathbf{E}\left[\sum_{v \in M_{j-1}, i \in (\mathbf{x} \cap \mathbf{q}) \setminus v} \mathbf{E}[Y(v \circ i)^2]\right] + \mathbf{E}\left[\sum_{\substack{v, v' \in M_{j-1} \\ i, i' \in (\mathbf{x} \cap \mathbf{q}) \setminus v \\ v \circ i \neq v' \circ i'}} \mathbf{E}[Y(v \circ i)]\, \mathbf{E}[Y(v' \circ i')]\right].
\end{aligned}$$

We have $\mathbf{E}[Y(v \circ i)] = s(v \circ i)$. Furthermore, $\sum_{i \in (\mathbf{x} \cap \mathbf{q}) \setminus v} s(v \circ i) \leq 1$ (the same applies with v', i' substituted for v, i respectively). Substituting,

$$\begin{aligned}
\mathbf{E}[|M_j|^2] = \ & \mathbf{E}\left[\sum_{v \in M_{j-1}, i \in (\mathbf{x} \cap \mathbf{q}) \setminus v} s(v \circ i)\right] + \mathbf{E}\left[\sum_{\substack{v \in M_{j-1}, i \in (\mathbf{x} \cap \mathbf{q}) \setminus v \\ u \in M_{j-1}, i' \in (\mathbf{x} \cap \mathbf{q}) \setminus v' \\ v \circ i \neq v' \circ i'}} s(v \circ i) s(v' \circ i')\right] \\
\leq \ & \mathbf{E}\left[\sum_{v \in M_{j-1}} 1\right] + \mathbf{E}\left[\sum_{v \in M_{j-1}} \sum_{v' \in M_{j-1}} 1\right] \\
= \ & \mathbf{E}[|M_{j-1}|] + \mathbf{E}[|M_{j-1}|^2] \\
\leq \ & 1 + j.
\end{aligned}$$

□

Proof of Lemma 9. Let $\mathbf{x} \sim \mathcal{D}[p_1, \ldots, p_d]$ and assume that $\sum_{i \in [d]} p_i \geq C \log n$ for some $C > 1$. We want to show that if C is sufficiently large, then $\mathbf{E}_{\mathcal{H}, \mathbf{x} \sim \mathcal{D}}[|F(\mathbf{x})|] = O(n^{\rho + \varepsilon})$ where the expectation is over both the randomness $\mathcal{H}$ of the data structure and the randomness of $\mathbf{x}$. By a Chernoff bound,

$$\begin{aligned}
\Pr\left[|\mathbf{x}| \geq (1 - C^{-1/3}) \sum_{i \in [d]} p_i\right] &\geq 1 - \exp\left(-\frac{(C^{-1/3})^2}{2} \sum_{i \in [d]} p_i\right) \\
&\geq 1 - \exp\left(-\frac{C^{-2/3}}{2} C \log(n)\right) \\
&\geq 1 - \exp(-C^{1/3} \log(n)/2) \\
&\geq 1 - n^{\frac{C^{1/3}}{2}}.
\end{aligned}$$

Furthermore,

$$\begin{aligned}
\Pr\left[\sum_{i \in \mathbf{x}} p_i^{\rho} \leq (1 + C^{-1/3}) \sum_{i \in [d]} p_i^{1+\rho}\right] &\geq 1 - \exp\left(-\frac{(C^{-1/3})^2}{3} \sum_{i \in [d]} p_i^{1+\rho}\right) \\
&= 1 - \exp\left(-\frac{C^{-2/3}}{3} b_1 \sum_{i \in [d]} p_i^{\rho}\right) \\
&\geq 1 - \exp\left(-\frac{C^{-2/3}}{3} b_1 C \log(n)\right) \\
&\geq 1 - n^{\frac{b_1 C^{1/3}}{3}}.
\end{aligned}$$

When both of these events occur, we have

$$\begin{aligned}
s(\mathbf{x}, j, i) &= \frac{1}{b_1 |\mathbf{x}| - j} \\
&\leq \frac{1}{b_1 |\mathbf{x}| - \log n} \\
&\leq \frac{1}{b_1 (1 - C^{-1/3}) \sum_{i \in [d]} p_i - \log n} \\
&= \frac{1}{b_1 \sum_{i \in [d]} p_i} \frac{1}{1 - C^{-1/3} - \log n / (b_1 \sum_{i \in [d]} p_i)} \\
&\leq \frac{1}{b_1 \sum_{i \in [d]} p_i} \frac{1}{1 - C^{-1/3} - 1/(b_1 C)}.
\end{aligned}$$

And therefore

$$\begin{aligned}\sum_{i\in\mathbf{x}} p_i^\rho s(\mathbf{x},j,i) &\le \sum_{i\in\mathbf{x}} p_i^\rho \frac{1}{b_1\sum_{i\in[d]} p_i}\frac{1}{1-C^{-1/3}-1/(b_1C)}\\ &\le \sum_{i\in[d]} p_i^{1+\rho} \frac{1}{b_1\sum_{i\in[d]} p_i}\frac{1+C^{-1/3}}{1-C^{-1/3}-1/(b_1C)}\\ &\le \frac{1+C^{-1/3}}{1-C^{-1/3}-1/(b_1C)}.\end{aligned}$$

Thus, according to Lemma 6, $\Pr_{\mathbf{x}\sim\mathcal{D}}[\mathbf{E}_{\mathcal{H}}[F(\mathbf{x})] \le n^{\rho+\varepsilon}] \ge 1-n^{\Omega(C^{1/3})}$. Since $\mathbf{E}_{\mathcal{H}}[|F(\mathbf{x})|]$ is polynomial in n for any $\mathbf{x}$, this implies that $\mathbf{E}_{\mathcal{H},\mathbf{x}\sim\mathcal{D}}[|F(\mathbf{x})|] = O(n^{\rho+\varepsilon})$. □

References

[1] A. Abboud, A. Rubinstein, and R. Williams. Distributed PCP theorems for hardness of approximation in P. In *Proc. 58th Symposium on Foundations of Computer Science (FOCS)*, pages 25–36. IEEE, 2017.

[2] T. D. Ahle, R. Pagh, I. Razenshteyn, and F. Silvestri. On the complexity of inner product similarity join. In *Proc. 35th Symposium on Principles of Database Systems (PODS)*, pages 151–164. ACM, 2016.

[3] J. Alman, T. M. Chan, and R. Williams. Polynomial representations of threshold functions and algorithmic applications. In *Proc. 57th Symposium on Foundations of Computer Science (FOCS)*, pages 467–476. IEEE, 2016.

[4] J. Alman and R. Williams. Probabilistic polynomials and hamming nearest neighbors. In *Proc. 56th Symposium on Foundations of Computer Science (FOCS)*, pages 136–150. IEEE, 2015.

[5] A. Andoni, P. Indyk, T. Laarhoven, I. Razenshteyn, and L. Schmidt. Practical and optimal LSH for angular distance. In *Proc. 28th Conference on Neural Information Processing Systems (NIPS)*, pages 1225–1233, 2015.

[6] A. Andoni, P. Indyk, H. L. Nguyen, and I. Razenshteyn. Beyond locality-sensitive hashing. In *Proc. 25th Symposium on Discrete Algorithms (SODA)*, pages 1018–1028. ACM-SIAM, 2014.

[7] A. Andoni, T. Laarhoven, I. Razenshteyn, and E. Waingarten. Optimal hashing-based time-space tradeoffs for approximate near neighbors. In *Proc. 28th Symposium on Discrete Algorithms (SODA)*, pages 47–66. ACM-SIAM, 2017.

[8] A. Andoni and I. Razenshteyn. Optimal data-dependent hashing for approximate near neighbors. In *Proc. 47th Symposium on Theory of Computing (STOC)*, pages 793–801. ACM, 2015.

[9] A. Arasu, V. Ganti, and R. Kaushik. Efficient exact set-similarity joins. In *Proc. 32nd International Conference on Very Large Data Bases (VLDB)*, pages 918–929, 2006.

[10] N. Augsten and M. H. Böhlen. Similarity joins in relational database systems. *Synthesis Lectures on Data Management*, 5(5):1–124, 2013.

[11] R. J. Bayardo, Y. Ma, and R. Srikant. Scaling up all pairs similarity search. In *Proc. 16th World Wide Web Conference (WWW)*, pages 131–140, 2007.

[12] A. Becker, L. Ducas, N. Gama, and T. Laarhoven. New directions in nearest neighbor searching with applications to lattice sieving. In *Proc. 27th Symposium on Discrete Algorithms (SODA)*, pages 10–24. ACM-SIAM, 2016.

[13] A. Z. Broder. On the resemblance and containment of documents. In *Proc. Compression and Complexity of Sequences*, pages 21–29. IEEE, 1997.

[14] A. Z. Broder, S. C. Glassman, M. S. Manasse, and G. Zweig. Syntactic clustering of the web. *Computer Networks and ISDN Systems*, 29(8):1157–1166, 1997.

[15] M. S. Charikar. Similarity estimation techniques from rounding algorithms. In *Proc. 34th Symposium on Theory of Computing (STOC)*, pages 380–388. ACM, 2002.

[16] S. Choi, S. Cha, and C. C. Tappert. A survey of binary similarity and distance measures. *J. Syst. Cybern. Informatics*, 8(1):43–48, 2010.

[17] T. Christiani. A framework for similarity search with space-time tradeoffs using locality-sensitive filtering. In *Proc. 28th Symposium on Discrete Algorithms (SODA)*, pages 31–46. ACM-SIAM, 2017.

[18] T. Christiani and R. Pagh. Set similarity search beyond minhash. In *Proc. 49th Symposium on Theory of Computing (STOC)*, pages 1094–1107. ACM, 2017.

[19] T. Christiani, R. Pagh, and J. Sivertsen. Scalable and robust set similarity join. In *Proc. 34th International Conference on Data Engineering (ICDE)*. IEEE, 2018. To appear.

[20] S. Dasgupta and K. Sinha. Randomized partition trees for exact nearest neighbor search. In *Proc. 26th Conference on Learning Theory (COLT)*, pages 317–337, 2013.

[21] S. Dasgupta and K. Sinha. Randomized partition trees for nearest neighbor search. *Algorithmica*, 72(1):237–263, 2015.

[22] X. Hu, Y. Tao, and K. Yi. Output-optimal parallel algorithms for similarity joins. In *Proc. 36th Symposium on Principles of Database Systems (PODS)*, pages 79–90. ACM, 2017.

[23] R. Impagliazzo, R. Paturi, and F. Zane. Which problems have strongly exponential complexity? *J. Comput. Syst. Sci.*, 63(4), 2001.

[24] Y. Jiang, D. Deng, J. Wang, G. Li, and J. Feng. Efficient parallel partition-based algorithms for similarity search and join with edit distance constraints. In *Proc. Joint EDBT/ICDT Workshops*, pages 341–348. ACM, 2013.

[25] M. Kapralov. Smooth tradeoffs between insert and query complexity in nearest neighbor search. In *Proc. 34th Symposium on Principles of Database Systems (PODS)*, pages 329–342. ACM, 2015.

[26] M. Karppa, P. Kaski, and J. Kohonen. A faster subquadratic algorithm for finding outlier correlations. In *Proc. 27th Symposium on Discrete Algorithms (SODA)*. ACM-SIAM, 2016.

[27] M. Karppa, P. Kaski, J. Kohonen, and P. Ó. Catháin. Explicit correlation amplifiers for finding outlier correlations in deterministic subquadratic time. In *Proc. 24th European Symposium on Algorithms (ESA)*, pages 52:1–52:17, 2016.

[28] O. Keivani, K. Sinha, and P. Ram. Improved maximum inner product search with better theoretical guarantees. In *Proc. International Joint Conference on Neural Networks (IJCNN)*, pages 2927–2934, 2017.

[29] A. Kirsch, M. Mitzenmacher, A. Pietracaprina, G. Pucci, E. Upfal, and F. Vandin. An efficient rigorous approach for identifying statistically significant frequent itemsets. *Journal of the ACM (JACM)*, 59(3):12, 2012.

[30] G. Li, D. Deng, J. Wang, and J. Feng. Pass-join: A partition-based method for similarity joins. *Proc. 37th International Conference on Very Large Data Bases*, 5(3):253–264, 2011.

[31] W. Mann, N. Augsten, and P. Bouros. An empirical evaluation of set similarity join techniques. *Proc. 42nd International Conference on Very Large Data Bases*, 9(9):636–647, 2016.

[32] A. May and I. Ozerov. On computing nearest neighbors with applications to decoding of binary linear codes. In *Proc. 34th International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT)*, pages 203–228, 2015.

[33] M. Mitzenmacher and E. Upfal. *Probability and computing - randomized algorithms and probabilistic analysis.* Cambridge University Press, 2005.

[34] R. Pagh, N. Pham, F. Silvestri, and M. Stöckel. I/O-efficient similarity join. In *Proc. 23rd European Symposium on Algorithms (ESA)*, pages 941–952, 2015.

[35] R. Panigrahy, K. Talwar, and U. Wieder. Lower bounds on near neighbor search via metric expansion. In *Proc. 51st Symposium on Foundations of Computer Science (FOCS)*, pages 805–814. IEEE, 2010.

[36] R. Paturi, S. Rajasekaran, and J. H. Reif. The light bulb problem. In *Proc. 2nd Workshop on Computational Learning Theory (COLT)*, pages 261–268, 1989.

[37] P. Ram and A. G. Gray. Maximum inner-product search using cone trees. In *Proc. 18th International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 931–939. ACM, 2012.

[38] A. Shrivastava and P. Li. Asymmetric LSH (ALSH) for sublinear time maximum inner product search (MIPS). In *Proc. 27th Conference on Neural Information Processing Systems (NIPS)*, pages 2321–2329, 2014.

[39] A. Shrivastava and P. Li. Asymmetric minwise hashing for indexing binary inner products and set containment. In *Proc. 24th International Conference on World Wide Web (WWW)*, pages 981–991, 2015.

[40] A. Shrivastava and P. Li. Improved asymmetric locality sensitive hashing (ALSH) for maximum inner product search (MIPS). In *Proc. 31st Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 812–821, 2015.

[41] C. Teflioudi, R. Gemulla, and O. Mykytiuk. LEMP: Fast retrieval of large entries in a matrix product. In *Proc. 41st International Conference on Management of Data (SIGMOD)*, pages 107–122. ACM, 2015.

[42] G. Valiant. Finding correlations in subquadratic time, with applications to learning parities and the closest pair problem. *J. ACM*, 62(2):13:1–13:45, 2015.

[43] L. G. Valiant. Functionality in neural nets. In *Proc. 7th National Conference on Artificial Intelligence (AAAI)*, pages 629–634, 1988.

[44] J. Wang, G. Li, and J. Feng. Can we beat the prefix filtering?: an adaptive framework for similarity join and search. In *Proc. 38th International Conference on Management of Data (SIGMOD)*, pages 85–96. ACM, 2012.

[45] C. Xiao, W. Wang, X. Lin, and J. X. Yu. Efficient similarity joins for near duplicate detection. In *Proc. International Conference on World Wide Web (WWW)*, pages 131–140, 2008.

[46] H. Zhang and Q. Zhang. Embedjoin: Efficient edit similarity joins via embeddings. In *Proceedings of International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 585–594. ACM, 2017.