Subsets and Supermajorities: Optimal Hashing-based Set Similarity Search

Thomas D. Ahle
BARC, IT University of Copenhagen
thdy@itu.dk

Jakob B. T. Knudsen
BARC, University of Copenhagen
jakn@di.ku.dk

***Abstract*—We formulate and optimally solve a new generalized Set Similarity Search problem, which assumes the size of the database and query sets are known in advance. By creating polylog copies of our data-structure, we optimally solve any symmetric Approximate Set Similarity Search problem, including approximate versions of Subset Search, Maximum Inner Product Search (MIPS), Jaccard Similarity Search and Partial Match.**

Our algorithm can be seen as a natural generalization of previous work on Set as well as Euclidean Similarity Search, but conceptually it differs by optimally exploiting the information present in the sets as well as their complements, and doing so asymmetrically between queries and stored sets. Doing so we improve upon the best previous work: MinHash [J. Discrete Algorithms 1998], SimHash [STOC 2002], Spherical LSF [SODA 2016, 2017] and Chosen Path [STOC 2017] by as much as a factor $n^{0.14}$ in both time and space; or in the near-constant time regime, in space, by an arbitrarily large polynomial factor.

Turning the geometric concept, based on Boolean supermajority functions, into a practical algorithm requires ideas from branching random walks on $\mathbb{Z}^2$, for which we give the first non-asymptotic near tight analysis.

Our lower bounds follow from new hypercontractive arguments, which can be seen as characterizing the exact family of similarity search problems for which supermajorities are optimal. The optimality holds for among all hashing based data structures in the random setting, and by reductions, for 1 cell and 2 cell probe data structures.

I. Introduction

Set Similarity Search (SSS) is the problem of indexing sets (or sparse boolean data) to allow fast retrieval of sets, similar under a given similarity measure. The sets may represent one-hot encodings of categorical data, "bag of words" representations of documents, or "visual/neural bag of words" models, such as the Scale-invariant feature transform (SIFT), that have been discretized. The applications are ubiquitous across Computer Science, touching everything from recommendation systems to gene sequences comparison. See [1], [2] for recent surveys of methods and applications.

Set similarity measures are any function, s that takes two sets and return a value in $[0, 1]$. Unfortunately, most variants of Set Similarity Search, such as Partial Match, are hard to solve assuming popular conjectures around the Orthogonal Vectors Problem [3], [4], [5], [6], which roughly implies that the best possible algorithm is to not build an index, and "just brute force" scan through all the data, on every query. A way to get around this is to study Approximate SSS: Given a query, q, for which the most similar set y has similarity$(q, y) \geq s_1$, we are allowed to return any set y' with similarity$(q, y') > s_1$, where $s_2 < s_1$. In practice, even the best *exact* algorithms for similarity search use such an (s_1, s_2)-approximate[1] solution as a subroutine [8].

Euclidean Similarity Search, where the data is vectors $x \in \mathbb{R}^d$ and the measure of similarity is "Cosine", has recently been solved optimally — at least in the model of hashing based data structures [9], [10]. Meanwhile, the problem on sets has proven much less tractable. This is despite that the first solutions date back to the seminal MinHash algorithm (a.k.a. min-wise hashing), introduced by Broder et al. [11], [12] in 1997 and by now boasting thousands of citations. In 2014 MinHash was shown to be near-optimal for set intersection *estimation* [13], but in a surprising, recent development, it was shown not to be optimal for similarity *search* [14]. The question thus remained: What *is* the optimal algorithm for Set Similarity Search?

The question is made harder by the fact that previous algorithms study the problem under different similarity measures, such as Jaccard, Cosine, or Braun-Blanquet similarity. The only thing those measures have in common is that they can be defined as a function f of the sets sizes, the universe size, and the intersection size. In other words, similarity$(q, y) = f(|q|, |y|, |q \cap y|, |U|)$ where $|U|$ is the size of the universe from which the sets are taken. In fact, any symmetric measure of similarity for sets must be defined by those four quantities.

Hence, to fully solve Set Similarity Search, we avoid specifying a particular similarity measure, and instead define the problem solely from those four parameters. This generalized problem is what we solve optimally in this paper, for all values of the four parameters:

Definition 1 (The (w_q, w_u, w_1, w_2)-GapSS problem). *Given some universe U and a collection $Y \subseteq \binom{U}{w_u|U|}$ of $|Y| = n$ sets of size $w_u|U|$, build a data structure that for any query set $q \in \binom{U}{w_q|U|}$: either returns $y' \in Y$ with $|y' \cap q| > w_2|U|$; or determines that there is no $y \in Y$ with $|y \cap q| \geq w_1|U|$.*

[1]By classical reductions [7] we can assume s_1 is known in advance.

For the problem to make sense, we assume that $w_q|U|$ and $w_u|U|$ are integers, that $w_q, w_u \in [0,1]$, and that $0 < w_2 < w_1 \le \min\{w_q, w_u\}$. Note that $|U|$ may be very large, and as a consequence the values w_q, w_u, w_1, w_2 may all be very small.

At first sight, the problem may seem easier than the version where the sizes of sets may vary. However, the point is that making polylog(n) data-structures for sets and queries of progressively bigger sizes,[2] immediately yields data structures for the original problem. Similarly, any algorithm assuming a specific set similarity measure also yields an algorithm for (w_q, w_u, w_1, w_2)-GapSS, so our lower bounds also hold for all previously studied SSS problems.

Example 1: As an example, assume we want to solve the Subset Search Problem, in which we, given a query q, want to find a set y in the database, such that $y \subseteq q$. If we allow a two-approximate solution, GapSS includes this problem by setting $w_1 = w_u$ and $w_2 = w_1/2$: The overlap between the sets must equal the size of the stored sets; and we are guaranteed to return a y' such that at least $|q \cap y'| \ge |y|/2$.

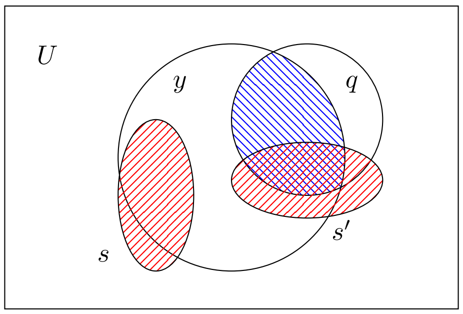
Example 2: In the (j_1, j_2)-Jaccard Similarity Search Problem, given a query, q, we must find y such that the Jaccard Similarity $|q \cap y|/|q \cup y| > j_2$ given that a y' exists with similarity at least j_1. After partitioning the sets by size, we can solve the problem using GapSS by setting $w_1 = \frac{j_1(w_q+w_u)}{1+j_1}$ and $w_2 = \frac{j_2(w_q+w_u)}{1+j_2}$. The same reduction works for any other similarity measure with polylog(n) overhead.

The version of this problem where $w_2 = w_q w_u$ is similar to what is in the literature called *"the random instance"* [16], [17], [18]. To see why, consider generating $n-1$ sets independently at random with size $w_u|U|$, and a "planted" pair, (q, y), with size respectively $w_q|U|$ and $w_u|U|$ and with intersection $|q \cap y| = w_1|U|$. Insert the size $w_u|U|$ sets into the database and query with q. Since q is independent from the $n-1$ original sets, its intersection with those is strongly concentrated around the expectation $w_q w_u|U|$. Thus, if we parametrize GapSS with $w_2 = w_q w_u + o(1)$, the query for q is guaranteed to return the planted set y.

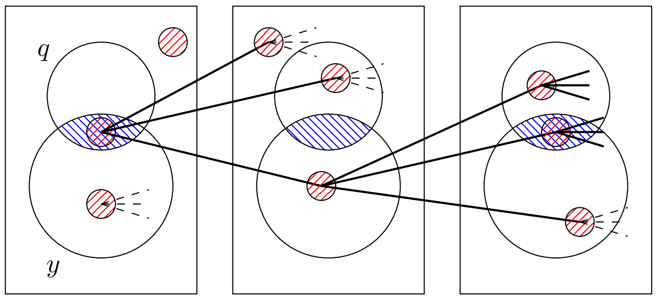
There is a tradition in the Similarity Search literature for studying such this independent case, in part because *it is expected that one can always reduce to the random instance*, for example using the techniques of "data-dependent hashing" [19], [9]. However, for such a reduction to make sense, we would first need an optimal "data-independent" algorithm for the $w_2 = w_q w_u$ case, which is what we provide in this paper. We discuss this further in the Related Work section.

For generality we still define the problem for all $w_2 \in (0, w_1)$, our upper bound holds in this general setting and so does the lower bound Theorem 2.

[2]For details, see [14] Section 5. A similar reduction, called "norm ranging", was recently shown at NeurIPS to give state of the art results for Maximum Inner Product Search in $\mathbb{R}^d$ [15], suggesting it is very practical.



(a) Two cohorts, y and q with a large intersection (blue). The first representative set, s, favours y, while the second, s', favours both y and q.



(b) Branching random walk run on two cohorts q and y. The bold lines illustrate paths considered by sets, while the dashed lines adorn paths only considered by only one of x or y. Here q has a higher threshold ($t_q = 2/3$) than y ($t_u = 1/2$), so q only considers paths starting with two favourable representatives.

Figure 1: The representative sets, coloured in red, are scattered in the universe to provide an efficient space partition for the data.

We give our new results in Section I-B and our new lower bounds in Section I-C, but first we would like to sketch the algorithm and some probabilistic tools used in the theorem statement.

A. Supermajorities

In Social Choice Theory a supermajority is when a fraction strictly greater than $1/2$ of people agree about something.[3] In the analysis of Boolean functions a t-supermajority function $f : \{0,1\}^n \to \{0,1\}$ can be defined as 1, if a fraction $\ge t$ of its arguments are 1, and 0 otherwise. We will sometimes use the same word for the requirement that a fraction $\le t$ of the arguments are 1.[4]

The main conceptual point of our algorithm is the realization that an optimal algorithm for Set Similarity Search

[3]"America was founded on majority rule, not supermajority rule. Somehow, over the years, this has morphed into supermajority rule, and that changes things." – Kent Conrad.

[4]It turns out that simply staying within a factor $1 \pm o(1)$ of t is also sufficient. This is similar to Dubiner [20].

must take advantage of the information present in the given sets, as well as that present in their complement. A similar idea was leveraged by Cohen et al. [21] for Set Similarity *Estimation*, and we show in the full version [22] that the classical MinHash algorithm can be seen as an average of functions that pull varying amounts of information from the sets and their complements. In this paper, we show that there is a better way of combining this information and that doing so results in an optimal hashing based data structure for the entire parameter space of random instance GapSS.

This way of combining this information is by supermajority functions. While on the surface they will seem similar to the threshold methods applied for time/space trade-offs in Spherical LSF [10], our use of them is very different. Where [10] corresponds to using small $t = 1/2 + o(1)$ thresholds (essentially simple majorities) our t may be as large as 1 (corresponding to the AND function) or as small as 0 (the NOT AND function). This way they are a sense as much a requirement on the complement as it is on the sets themselves.

The algorithm (idealized): While our data structure is technically a tree with a carefully designed pruning rule, the basic concept is very simple.

We start by sampling a large number of "representative sets" $R \subseteq \binom{U}{k}$. Here roughly $|R| \approx n^{\log n}$ and $k \approx \log n$. Given family $Y \subseteq \binom{U}{w_u|U|}$ of sets to store, which we call "cohorts", we say that $r \in R$ "t-favours" the cohort y if $|y \cap r|/|r| \geq t$. Representing sets as vectors in $\{0,1\}^d$, this is equivalent to saying $f_t(r \cap y) = 1$, where f_t is the t-supermajority function. (If t is less than w_u, the expected size of the overlap, we instead require $|y \cap r|/|r| \leq t$.)

Given the parameters $t_q, t_u \in [0,1]$, the data-structure is a map from elements of R to the cohorts they t_u-favour. When given a query $q \in \binom{U}{w_q|U|}$, (a $w_q|U|$ sized cohort), we compare it against all cohorts y favoured by representatives $r \in R$ which t_q-favour q (that is $|q \cap r|/|r| \geq t_q$). This set $R_{t_q}(q)$ is much smaller than $|R|$ (we will have $|R_{t_q}(q)| \approx n^{\varepsilon}$ and $E[|R_{t_u}(y) \cap R_{t_q}(q)|] \approx n^{\varepsilon-1}$), so the filtering procedure greatly reduces the number of cohorts we need to compare to the query from n to n^{ε} (where $\varepsilon = \rho_q < 1$ is defined later.)

The intuition is that while it is quite unlikely for a representative to favour a given cohort, and it is *very* unlikely for it to favour two given cohorts (q and y). So if it does, the two cohorts probably have a substantial overlap. Figure 1a has a simple illustration of this principle.

In order to fully understand supermajorities, we want to understand the probability that a representative set is simultaneously in favour of two distinct cohorts given their overlap and representative sizes. This paragraph is a bit technical and may be skipped at first read. Chernoff bounds in $\mathbb{R}$ are a common tool in the community, and for iid. $X_i \sim \text{Bernoulli}(p) \in \{0,1\}$ the sharpest form (with a matching lower bound) is $\Pr[\sum X_i \geq tn] \leq \exp(-n\,\mathrm{d}(t \,\|\, p))$,[5] which uses the binary KL-Divergence $\mathrm{d}(t \,\|\, p) = t \log \frac{t}{p} + (1-t) \log \frac{1-t}{1-p}$. The Chernoff bound for $\mathbb{R}^2$ is less common, but likewise has a tight description in terms of the KL-Divergence between two discrete distributions: $\mathrm{D}(P \,\|\, Q) = \sum_{\omega \in \Omega} P(\omega) \log \frac{P(\omega)}{Q(\omega)}$ (summing over the possible events). In our case, we represent the four events that can happen as we sample an element of U as a vector $X_i \in \{0,1\}^2$. Here $X_i = \left[\begin{smallmatrix}1\\1\end{smallmatrix}\right]$ means the ith element hit both cohorts, $X_i = \left[\begin{smallmatrix}1\\0\end{smallmatrix}\right]$ means it hit only the first and so on. We represent the distribution of each X_i as a matrix $P = \left[\begin{smallmatrix} w_1 & w_q - w_1 \\ w_u - w_1 & 1 - w_q - w_u + w_1 \end{smallmatrix}\right]$, and say $X_i \sim \text{Bernoulli}(P)$ iid. such that $\Pr[X_i = \left[\begin{smallmatrix}1-j\\1-k\end{smallmatrix}\right]] = P_{j,k}$. Then $\Pr[\sum X_i \geq \left[\begin{smallmatrix}t_q\\t_u\end{smallmatrix}\right] n] \approx \exp(-n\,\mathrm{D}(T \,\|\, P))$ where $T = \left[\begin{smallmatrix} t_1 & t_q - t_1 \\ t_u - t_1 & 1 - t_q - t_u + t_1 \end{smallmatrix}\right]$ and $t_1 \in [0, \min\{t_u, t_q\}]$ minimizes $\mathrm{D}(T \,\|\, P)$. (Here the notation $\left[\begin{smallmatrix}x\\y\end{smallmatrix}\right] \geq \left[\begin{smallmatrix}t_u\\t_q\end{smallmatrix}\right]$ means $x \geq t_u \wedge y \geq t_q$.)

The optimality of Supermajorities for Set Similarity Search is shown using a certain correspondence we show between the Information Theoretical quantities described above, and the hypercontractive inequalities that have been central in all previous lower bounds for similarity search.

These bounds above would immediately allow a cell probe version of our upper bound Theorem 1, e.g. a query would require $n^{\frac{\mathrm{D}(T_1 \| P_1) - \mathrm{d}(t_q \| w_q)}{\mathrm{D}(T_2 \| P_2) - \mathrm{d}(t_q \| w_q)}}$ probes, where $P_i = \left[\begin{smallmatrix} w_i & w_q - w_1 \\ w_u - w_i & 1 - w_q - w_u + w_i \end{smallmatrix}\right]$ and T_i defined accordingly. The algorithmic challenge is that for optimal performance, $|R|$ must be in the order of $\Omega(n^{\log n})$, and so checking which representatives favour a given cohort takes super polynomial time!

The classical approach to designing an oracle to efficiently yield all such representatives, , is a product-code or "tensoring trick". The idea, (used by [23], [24]), is to choose a smaller $k' \approx \sqrt{k}$, make k/k' different R'_i sets of size $n^{\sqrt{\log n}}$ and take R as the product $R'_1 \times \cdots \times R'_{k/k'}$. As each R' can now be decoded in $n^{o(1)}$ time, so can R. This approach, however, in the case of Supermajorities, has a big drawback: Since $t_q k'$ and $t_u k'$ must be integers, t_q and t_u have to be rounded and thus distorted by a factor $1 + 1/k'$. Eventually, this ends up costing us a factor $w_1^{-k/k'}$ which can be much larger than n. For this reason, we need a decoding algorithm that allows us to use supermajorities with as large a k as possible!

We instead augment the above representative sampling procedure as follows: Instead of independent sampling sets, we (implicitly) sample a large, random height k tree, with nodes being elements from the universe. The representative sets are taken to be each path from the root to a leaf. Hence, some sets in R share a common prefix, but mostly they are

[5]A special case of Hoeffding's inequality is obtained by $\mathrm{d}(p + \varepsilon \,\|\, p) \geq 2\varepsilon^2$, Pinsker's inequality.

still independent. We then add the extra constraint that *each of the prefixes of a representative has to be in favour of a cohort*, rather than only having this requirement on the final set. This is the key to making the tree useful: Now given a cohort, we walk down the tree, pruning any branches that do not consistently favour a supermajority of the cohort. Figure 1b has a simple illustration of this algorithm. This pruning procedure can be shown to imply that we only spend time on representative sets that end up being in favour of our cohort, while only weakening the geometric properties of the idealized algorithm negligibly.

While conceptually simple and easy to implement (modulo a few tricks to prevent dependency on the size of the universe, $|U|$), the pruning rule introduces dependencies that are quite tricky to analyze sufficiently tight. The way to handle this will be to consider the tree as a "branching random walk" over $\mathbb{Z}_+^2$ where the value represents the size of the representative's intersection with the query and a given set respectively. The paths in the random walk at step i must be in the quadrant $[t_q i, i] \times [t_u i, i]$ while only increasing with a bias of $\left[\begin{smallmatrix} w_q \\ w_u \end{smallmatrix}\right]$ per step. The branching factor is carefully tuned to just the right number of paths survive to the end.

The aspect of the pruning is a very important property of our algorithm and conceptually departs from previous methods.

Previous Locality Sensitive Filtering, LSF, algorithms [14], [18] can be seen as trees with pruning, but their pruning is on the individual node level, rather than on the entire path. This makes a big difference in which space partitions can be represented, since pruning on node level ends up representing the intersection of simple partitions, which can never represent Supermajorities in an efficient way.

B. Upper Bounds

As discussed, the performance of our algorithm is described in terms of KL-divergences. To ease understanding, we give a number of special cases, in which the general bound simplifies. The bounds in this section assume w_q, w_u, w_1, w_2 are constants. See the full version [22] for a version without this assumption.

Theorem 1 (Simple Upper Bound)**.** *For any choice of constants $w_q, w_u \geq w_1 \geq w_2 \geq 0$ and $1 \geq t_q, t_u \geq 0$ we can solve the (w_q, w_u, w_1, w_2)-GapSS problem over universe U with query time $\tilde{O}(n^{\rho_q} + w_q|U|) + n^{o(1)}$ and auxiliary space usage $\tilde{O}(n^{1+\rho_u})$, where*

$$\rho_q = \frac{\mathrm{D}(T_1 \,\|\, P_1) - \mathrm{d}(t_q \,\|\, w_q)}{\mathrm{D}(T_2 \,\|\, P_2) - \mathrm{d}(t_q \,\|\, w_q)},$$

$$\rho_u = \frac{\mathrm{D}(T_1 \,\|\, P_1) - \mathrm{d}(t_u \,\|\, w_u)}{\mathrm{D}(T_2 \,\|\, P_2) - \mathrm{d}(t_q \,\|\, w_q)}.$$

and T_1, T_2 are distributions with expectation $\left[\begin{smallmatrix} t_q \\ t_u \end{smallmatrix}\right]$ minimizing respectively $\mathrm{D}(T_1 \,\|\, P_1)$ and $\mathrm{D}(T_2 \,\|\, P_2)$, as described in Section I-A.

The two bounds differ only in the $\mathrm{d}(t_q \,\|\, w_q)$ and $\mathrm{d}(t_u \,\|\, w_u)$ terms in the numerator. The thresholds t_q and t_u can be chosen freely in $[0,1]^2$. Varying them compared to each other allows a full space/time trade-off with $\rho_q = 0$ in one end and $\rho_u = 0$ (and $\rho_q < 1$) in the other. Note that for a given GapSS instance, there are many (t_q, t_u) which are not optimal anywhere on the space/time trade-off. Using Lagrange's condition $\nabla\rho_q = \lambda\nabla\rho_u$ one gets a simple equation that all optimal (t_q, t_u) trade-offs must satisfy. As we will discuss later, it seems difficult to prove that a solution to this equation is unique, but in practice, it is easy to solve and provides an efficient way to optimize ρ_q given a space budget $n^{1+\rho_u}$. Figure 2 and Figure 3 provides some additional intuition for how the ρ values behave for different settings of GapSS.

Regarding the other terms in the theorem, we note that the $\tilde{O}$ hides only $\log n$ factors, and the additive $n^{o(1)}$ term grows as $e^{O(\sqrt{\log n \log\log n})}$, which is negligible unless $\rho_q = 0$. We also note that there is no dependence on $|U|$, other than the need to store the original dataset and the additive $w_q|U|$, which is just the time it takes to receive the query. The main difference between this theorem and the full version is that the full theorem does not assume the parameters (w_q, w_u, w_1, w_2) are constants but consider them potentially very small. In this more realistic scenario, it becomes very important to limit the dependency on factors like w_1^{-1}, which is what guides a lot of our algorithmic decisions.

Example 1: Near balanced ρ values.: As noted, many pairs (t_q, t_u) are not optimal on the trade-off, in that one can reduce one or both of ρ_q, ρ_u by changing them. The pairs that are optimal are not always simple to express, so it is interesting to study those that are. One such particularly simple choice on the Lagrangian is $t_q = 1 - w_u$ and $t_u = 1 - w_q$.[6] This point is special because the values of t_q and t_u depend only on w_u and w_q, while in general they will also depend on w_1 and w_2. In this setting we have $T_i = \left[\begin{smallmatrix} 1-w_q-w_u+w_i & w_u-w_i \\ w_u-w_i & w_i \end{smallmatrix}\right]$, which can be plugged into Theorem 1.

In the case $w_q = w_u = w$ we get the balanced ρ values $\rho_q = \rho_u = \log(\frac{w_1}{w}\frac{1-w}{1-2w+w_1}) \big/ \log(\frac{w_2}{w}\frac{1-w}{1-2w+w_2})$ in which case it is simple to compare with Chosen Path's ρ value of $\log(\frac{w_1}{w}) \big/ \log(\frac{w_2}{w})$. Chosen Path on balanced sets was shown in [14] to be optimal for w, w_1, w_2 small enough, and we see that Supermajorities do indeed recover this value for that range.

Example 2: Subset/superset queries.: If $w_1 = \min\{w_u, w_q\}$ and $w_2 = w_u w_q$ we can take $t_q = \frac{-\alpha}{w_q - w_u} + \frac{w_q(1-w_u)}{w_q-w_u}$ and $t_u = \frac{w_u(1-w_u)w_q(1-w_q)}{w_q-w_u}\alpha^{-1} - \frac{w_u(1-w_q)}{w_q-w_u}$ for

[6]To make matters complicated, this *is* a simple choice *and* on the Lagrangian, but that doesn't prove another point on the Lagrangian won't reduce both ρ_q and ρ_u and thus be better. That we have a matching lower bound for the algorithm doesn't help, since it only matches the upper bound for (t_q, t_u) minimal in Theorem 1. In the case $w_q = w_u$ we can, however, prove that this t_q, t_u pair is optimal.

any $\alpha \in [w_1 - w_q w_u, \max\{w_u, w_q\} - w_q w_u]$. This represents one of the cases where we can solve the Lagrangian equation to get a complete characterization of the t_q, t_u values that give the optimal trade-offs. Note that when $w_1 = w_u$ or $w_1 = w_q$, the P matrix as used in the theorem has 0's in it. The only way the KL-divergence $\mathrm{D}(T \,\|\, P)$ can then be finite is by having the corresponding elements of T be 0 and use the fact that $0 \log \frac{0}{q}$ is defined to be 0 in this context.

Example 3: Linear space/constant time.: Setting t_1 in $T_1 = \begin{bmatrix} t_1 & t_q - t_1 \\ t_u - t_1 & 1 - t_q - t_u + t_1 \end{bmatrix}$ such that either $\frac{t_1}{w_1} = \frac{t_q - t_1}{w_q - w_1}$ or $\frac{t_1}{w_1} = \frac{t_u - t_1}{w_u - w_1}$ we get respectively $\mathrm{D}(T_1 \,\|\, P_1) = \mathrm{d}(t_q \,\|\, w_q)$ or $\mathrm{D}(T_1 \,\|\, P_1) = \mathrm{d}(t_u \,\|\, w_u)$. Theorem 1 then yields algorithms with either $\rho_q = 0$ or $\rho_u = 0$ corresponding to either a data structure with $\approx e^{\tilde{O}(\sqrt{\log n})}$ query time, or with $\tilde{O}(n)$ auxiliary space. Like [10] we have $\rho_q < 1$ for any parameter choice, even when $\rho_u = 0$. For very small w_q and $w_u < \exp(-\sqrt{\log n})$ there are some extra concerns which are discussed after the main theorem.

C. Lower Bounds

Results on approximate similarity search are usually phrased in terms of two quantities: (1) The "query exponent" $\rho_q \in [0, 1]$ which determines the query time by bounding it by $O(n^{\rho_q})$; (2) The "update exponent" $\rho_u \in [0, 1]$ which determines the time required to update the data structure when a point is inserted or deleted in Y and is given by $O(n^{\rho_u})$. The update exponent also bounds the space usage as $O(n^{1+\rho_u})$. Given parameters (w_q, w_u, w_1, w_2), the important question is for which pairs of (ρ_q, ρ_u) there exists data structures.

Previous work split into two models: (1) Cell probe lower bounds [25], [26], [10] and (2) Lower bounds in restricted models [27], [28], [29], [10], [14]. The common restricted models are the LSH model [30], the LSF model [24] and the most general "list of points" model formulated by [10]. This last model contains all known Similarity Search data structures, except for the so-called "data-dependent" algorithms. (It is however conjectured [18] that data-dependency does not help on random instances (recall this corresponds to $w_2 = w_q w_u$), which is the setting of Theorem 3.)

We show two main lower bounds: (1) A symmetric LSF bound that requires $w_q = w_u$ and $\rho_q = \rho_u$ and (2) A list-of-points bound that requires the "random setting" $w_2 = w_q w_u$. The second type is tight everywhere, but quite technical. The first type meanwhile is quite simple to state, informally:

Theorem 2. *If $w_q = w_u = w$ and $\rho_u = \rho_q = \rho$, any data-independent LSF data structure must use space $n^{1+\rho}$ and have query time n^ρ where $\rho \geq \log(\frac{w_1 - w^2}{w(1-w)}) \big/ \log(\frac{w_2 - w^2}{w(1-w)})$.*

We note that previous symmetric bounds [31], [14] were only asymptotic, whereas our lower bound holds over the entire range of $0 < w_2 < w_1 < w < 1$. By comparison with $\rho = \log(\frac{w_1(1-w)}{w(1-2w+w_1)}) \big/ \log(\frac{w_2(1-w)}{w(1-2w+w_2)})$ from Example 1 in the Upper Bounds section, we see that the lower bound is sharp when $w, w_1, w_2 \to 0$[7] and also for $w_1 \to w$, since $w(1 - 2w + w_1) = w(1 - w) - w(w - w_1)$. However, for $w_2 = w^2$ (the random instance), Theorem 2 just says $\rho \geq 0$, which means it tells us nothing.

For the random instances, we give an even stronger lower bound, which gets rid of the restrictions $w_q = w_u$ and $\rho_q = \rho_u$. This lower bound is tight for any $0 < w_q w_u < w_1 < \min\{w_q, w_u\}$ in the list-of-points model:

Theorem 3. *Consider any list-of-point data structure for the $(w_q, w_u, w_1, w_q w_u)$-GapSS problem over a universe of size d of n points with $w_q w_u d = \omega(\log n)$, which uses expected space $n^{1+\rho_u}$, has expected query time $n^{\rho_q - o_n(1)}$, and succeeds with probability at least 0.99. Then for every $\alpha, \beta \in [0, 1]$ with $\alpha + \beta = 1$ there exists t_q, t_u such that*

$$\alpha\rho_q + \beta\rho_u \geq \alpha \frac{\mathrm{D}(T \,\|\, P) - \mathrm{d}(t_q \,\|\, w_q)}{\mathrm{d}(t_u \,\|\, w_u)} + \beta \frac{\mathrm{D}(T \,\|\, P) - \mathrm{d}(t_u \,\|\, w_u)}{\mathrm{d}(t_u \,\|\, w_u)},$$

where $P = \begin{bmatrix} w_1 & w_q - w_1 \\ w_u - w_1 & 1 - w_q - w_u + w_1 \end{bmatrix}$ and $T \ll P$ minimizes $\mathrm{D}(T \,\|\, P)$ given $\mathop{\mathrm{E}}_{X \sim T}[X] = \left[\begin{smallmatrix} t_q \\ t_u \end{smallmatrix}\right]$.

Note that for $w_2 = w_q w_u$, the term $\mathrm{D}(T_2 \,\|\, P_2)$, in Theorem 1, splits into $\mathrm{d}(t_q \,\|\, w_q) + \mathrm{d}(t_u \,\|\, w_u)$, and so the upper and lower bounds perfectly match. This shows that for any linear combination of ρ_q and ρ_u our algorithm obtains the minimal value. By continuity of the terms, this equivalently states as saying that no list-of-points algorithm can get a better query time than our Theorem 1, given a space budget imposed by ρ_u. [8]

Example 1: Choices for t_q and t_u: As in the upper bounds, it is not easy to prove that a particular choice of t_q and t_u minimizes the lower bound. Setting $t_q = 1 - w_u$ and $t_u = 1 - w_q$ the expression in Theorem 3 we obtain the same value as in Example 1 of the upper bound, however it could be (though we conjecture not) that another set of thresholds would reduce both the upper and lower bound.

The good news is that the hypercontractive inequality by Oleszkiewicz [32], can be used to prove certain optimal choices on the space/time trade-off.[9] In particular we will show that for $w_q = w_u = w$ the choice $t_q = t_u = 1 - w$ is optimal in the lower bound, and matches exactly the value

[7]As $w, w_1, w_2 \to 0$ we recover the lower bound $\rho \geq \log(\frac{w_1}{w}) / \log(\frac{w_2}{w})$ obtained for Chosen Path in [14].

[8]It is easy to see that $\rho_u = 0$ minimizes $\alpha\rho_q + (1 - \alpha)\rho_u$ when $\alpha = 0$, and similarly $\rho_u = \rho_{max}$ minimizes $\alpha\rho_q + (1 - \alpha)\rho_u$ when $\alpha = 1$, where ρ_{max} is the minimal space usage when $\rho_q = 0$. Furthermore, we note that when we change α from 0 to 1, then ρ_u will continuously and monotonically go from 0 to ρ_{max}. This shows that for every $\rho_u \in [0, \rho_{max}]$ there exists an α such that $\alpha\rho_q + (1 - \alpha)\rho_u$ is minimized, where ρ_q is best query time given the space budget imposed by ρ_u.

[9]The generalizations by Wolff [33] could in principle expand this range, but they are only tight up to a constant in the exponent.

$\rho = \log\left(\frac{w_1(1-w)}{w(1-2w+w_1)}\right) / \log(\frac{w_2(1-w)}{w(1-2w+w^2)})$ from Example 1 in the Upper Bounds section.

For cell probe lower bounds, we can use the framework of Panigrahy et al. [25], [26], [34]. Using the hypercontractive inequalities shown in this paper with this framework, as well (as the extension by [10]), we can show, unconditionally, that no data structure, which probes only 1 or 2 memory locations[10], can improve upon the space usage of $n^{1+\rho_u}$ obtained by Theorem 1 as we let $\rho_q = 0$. In particular, this shows that the near-constant query time regime from Example 3 in the Upper Bounds is optimal up to $n^{o(1)}$ factors in time and space.

D. Technical Overview

The main realization of this paper, is that all set similarity search problems can be stated in terms of a single geometric question, which can be answered optimally. Once proven, the main challenge is to turn this geometry into an efficient algorithm.

Supermajorities – why do they work?: Representing sets as binary vectors $x \in \{0,1\}^{|U|}$, it is natural to assume the best Similarity Search data structure for Euclidean and binary data — Spherical LSF — should be the best choice for sets as well. Unfortunately this mapping throws away two key properties of the data: that the vectors are sparse, and that they are non-negative. Algorithms like MinHash, which were specifically designed for sets, take advantage of the sparsity by entirely disregarding the remaining universe, U. This is seen by the fact that adding new elements to U never changes the MinHash of a set. Meanwhile Spherical LSF takes the inner product between x and a Gaussian vector scaled down by $1/\sqrt{|U|}$, so each new element added to U, in a sense, lowers the "sensitivity" to x.

In an alternative situation we might imagine $|x|$ being nearly as big as $|U|$. In this case we would clearly prefer to work with $U \setminus x$, since information about an element that is left out, is much more valuable than information about an element contained in x. What Supermajorities does can be seen as perfectly balancing how much information to include from x with how much to include from $U \setminus x$.

As a side effect the extra flexibility afforded by our approach allows balancing the time required to perform queries with the size of the database. It is perhaps surprising that this simple balancing act is enough to be optimal across all hashing algorithms as well as 1 cell and 2 cell probe data structures.

The results turn out to be best described in terms of the KL-divergences $\mathrm{D}(T \,\|\, P) - \mathrm{d}(t_q \,\|\, w_q)$ and $\mathrm{D}(T \,\|\, P) - \mathrm{d}(t_u \,\|\, w_u)$, which are equivalent to $\mathrm{D}(T_{XY} \,\|\, P_{Y|X}T_X)$ and $\mathrm{D}(T_{XY} \,\|\, P_{X|Y}T_Y)$. Here P_{XY} is the distribution of a coordinated sample from both a query and a dataset, P_X and P_Y are the marginals, and T_{XY} is roughly the distribution of samples conditioned on having a shared representative set. Intuitively these describe the amount of information gained when observing a sample from T_{XY} given a belief that X (resp. Y) is distributed as T and Y (resp. X) is distributed as P.[11]

Branching Random Walks: Making Supermajorities a practical algorithm (rather than just cell probe), requires, as discussed in the introduction, an efficient decoding algorithm of which representative sets overlap with a given cohort. Such oracles have been studied carefully in the literature, and since the LSH forest in 2005 [35] a common idea has been trees with independent pruning in each leaf. Our method is the first to significantly depart from this idea: While still a tree, our pruning is highly dependent across the levels of the tree, carrying a state from the root to the leaf which needs be considered by the pruning as well as the analysis. In "branching random walk", the state is represented in the "random walk", while the tree is what makes it branching. While considered heuristically in [24], such a stateful oracle has not before been analysed, partly because it wasn't necessary. For Supermajorities, meanwhile, it is crucially important for getting sub linear query times for very small sets.

The approach from [36], [24], [10] when applied to our scheme would correspond to making our representatives have size just $\sqrt{k}$ (so there are only $|R'| \approx e^{\tilde{O}(\sqrt{\log n})}$ of them,) and then make $R'^{\otimes\sqrt{k}}$ our new R. Since R' can be decoded in $n^{o(1)}$ time, and the second step can be made to take only time proportional to the output, this works well for some cases. This approach has two main issues: (1) There is a certain overhead that comes from not using the optimal filters, but only an approximation. However, this gives only a factor $e^{\tilde{O}(\sqrt{\log n})}$, which is usually tolerated. Worse is (2): Since the thresholds $t_q k$ and $t_u k$ have to be integral, using representative sets of size $\sqrt{k}$ means we have to "repair" them by a multiplicative distortion of approximately $1 \pm 1/\sqrt{k}$, compared to $1 \pm 1/k$ for the "real" filters. This turns out to cost as much as $w_1^{-\sqrt{k}}$ which can easily be much larger than the polynomial cost in n. In a sense, this shows that supermajority functions must be applied to measure the entire representative part of a cohort at once! This makes tensoring not well fit for our purposes.

A pruned branching random walk on the real line can be described in the following way. An initial ancestor is created with value 0 and form the zeroth generation. The people in the ith generation give birth Δ times each and independently of one another to form the $(i+1)$th generation. The people in the $(i+1)$th generation inherit the value, v, of their parent plus an independent random variable X. If ever $v + X < 0$, the child doesn't survive. After k generations, we expect

[10]For 1 probe, the word size can be $n^{o(1)}$, whereas for the 2 probe argument, the word size can only be $o(\log n)$ for the lower bound to hold.

[11]The use of information theory may remind readers of the Entropy LSH approach by [16], but the methods are similar in name only.

by linearity $\Delta^k \Pr[\forall_{i\le k} \sum_{j\in[i]} X_i \ge 0]$ people to be alive, where X_i are iid. random variables as used in the branching. A pruned 2d-branching random walk is simply one using values $\in \mathbb{R}^2$.

Branching random walks have been analysed before in the Brownian motion literature [37]. They are commonly analysed using the second-moment method, however, as noted by Bramson [38]: "an immediate frontal assault using moment estimates, but ignoring the branching structure of the process, will fail." The issue is that the probability that a given pair of paths in the branching process survives is too large for standard estimates to succeed. If the lowest common ancestor of two nodes manages to accumulate much more wealth than expected, its children will have a much too high chance of surviving. For this reason we have to counterintuitively *add extra pruning when proving the lower bound* that a representative set survives. More precisely, we prune all the paths that accumulate much more than the expected value. We show that this does not lower the probability that a representative set is favour by much, while simultaneously decreasing the variance of the branching random walk a lot. Unfortunately, this adds further complications, since ideally, we would like to prune every path that gets below the expectation. Combined with the upper bound this would trap the random walks in a band to narrow to guarantee the survival of a sufficient number of paths. Hence instead, we allow the paths to deviate by roughly a standard deviation below the expectation.

Output-sensitive set decoding: In our algorithm we are careful to not have factors of $|U|$ and $|X|$ (the size of the sets) on our query time and space bounds. When sampling our tree, at each level we must pick a certain number, Δ, of elements from the universe and check which of them are contained in the set being decoded. This is an issue, since Δ may be much bigger than $X \cap \Delta$, and so we need an "output-sensitive" sampling procedure. We do this by substituting random sampling with a two-independent hash function $h : U^k \to [q]$, where q is a prime number close to $|U|$. The sampling criterion is then $h(r \circ x) \le \Delta$, where $\circ$ is string concatenation. The function $h(r)$ can be taken to be $\sum_{i=1}^k a_i x_i + b \pmod q$ for random values $a_1, \dots, a_k, b \in [q]$, so we can expand $h(r\circ x)$ as $h(r) + a_k x \pmod q$.

$$
\begin{aligned}
\text{Now } &\{x \in X \mid (h(r\circ x) \bmod q) < \Delta\} \\
&= \{x \in X \mid (h(r) + a_k x \bmod q) < \Delta\} \\
&= \cup_{i=0}^{\Delta-1} \{x \in X \mid a_k x \equiv \Delta - h(r) \bmod q\} \\
&= \{x \in X \mid (a_k x \bmod q) \in [-h(r), \Delta - h(r)] \bmod q\},
\end{aligned}
$$

where the last equation is adjusted in case $(-h(r) \bmod q) > (\Delta - h(r) \bmod q)$. By pre-computing $\{a_k x \bmod q \mid x \in X\}$ (just has to be done one for each of roughly $\log n$ levels in the tree), and storing the result in a predecessor data-structure (or just sorting it), the sampling can be done it time proportional to the size of its output.

Lower Bounds and Hypercontractivity: The structure of our lower bounds is by now standard: We first reduce our lower bound to random instances by showing that with high probability the random instances are in fact an instance of our problem. For this to work, we need $w_w\,|U| = \omega(\log n)$ and in particular $|U| = \omega(\log n)$, so we get concentration around the mean. This requirement is indeed known to be necessary, since the results of [24], [39] break the known lower bounds in the "medium dimension regime" when $|U| = O(\log n)$.

The main difference compared to previous bounds is that we study Boolean functions on so-called p-biased spaces, where the previous lower bounds used Boolean functions on unbiased spaces. This is necessary for us to lower bound every parameter choice for GapSS. In particular we are interested in tight hypercontractive inequalities on p-biased spaces. We say that a distribution $\mathcal{P}_{XY}$ on a space $\Omega_X \times \Omega_Y$ is (r,s)-hypercontractive if

$$\mathop{\mathrm{E}}_{\mathcal{P}_{XY}}[f(X)g(Y)] \le \mathop{\mathrm{E}}_{\mathcal{P}_X}[f(X)^r]^{1/r} \mathop{\mathrm{E}}_{\mathcal{P}_Y}[g(Y)^s]^{1/s}$$

for all functions $f : \Omega_X \to \mathbb{R}$ and $g : \Omega_Y \to \mathbb{R}$, where $\mathcal{P}_X$ and $\mathcal{P}_Y$ are the marginal distributions on the spaces Ω_X and Ω_Y respectively. On unbiased spaces, the classic Bonami-Beckner inequality [40], [41] gives a complete understanding of the hypercontractivity. Unfortunately, this is not the case for p-biased spaces where the hypercontractivity is much less understood, with [32] and [33] being state of the art. We sidestep the issue of finding tight hypercontractive inequalities by instead showing an equivalence between hypercontractivity and KL-divergence, which is captured in the following lemma:[12]

Lemma I.1. *Let $\mathcal{P}_{XY}$ be a probability distribution on a space $\Omega_X \times \Omega_Y$ and let $\mathcal{P}_X$ and $\mathcal{P}_Y$ be the marginal distributions on the spaces Ω_X and Ω_Y respectively. Let $s, r \in [1, \infty)$, then the following is equivalent*

1) *For all functions $f : \Omega_X \to \mathbb{R}$ and $g : \Omega_Y \to \mathbb{R}$,*

$$\mathop{\mathrm{E}}_{\mathcal{P}_{XY}}[f(X)g(Y)] \le \mathop{\mathrm{E}}_{\mathcal{P}_X}[f(X)^r]^{1/r} \mathop{\mathrm{E}}_{\mathcal{P}_Y}[g(Y)^s]^{1/s}.$$

2) *For all probability distributions $\mathcal{Q}_{XY} \ll \mathcal{P}_{XY}$,*

$$\mathrm{D}(\mathcal{Q}_{XY} \,\|\, \mathcal{P}_{XY}) \ge \tfrac{1}{r}\mathrm{D}(\mathcal{Q}_X \,\|\, \mathcal{P}_X) + \tfrac{1}{s}\mathrm{D}(\mathcal{Q}_Y \,\|\, \mathcal{P}_Y),$$

where $\mathcal{Q}_X$ and $\mathcal{Q}_Y$ be the marginal distributions on the spaces Ω_X and Ω_Y respectively

The main technical argument needed for proving Lemma I.1 is that, for all probability distributions $\mathcal{P}, \mathcal{Q}$, where $\mathcal{Q}$ is absolutely continuous with respect to $\mathcal{P}$, and all functions ϕ,

$$\mathrm{D}(\mathcal{Q} \,\|\, \mathcal{P}) + \log \mathop{\mathrm{E}}_{X\sim\mathcal{P}}[\exp(\phi(X))] \ge \mathop{\mathrm{E}}_{X\sim\mathcal{Q}}[\phi(X)].$$

[12]It appears that one might prove a similar result using [42] and [43].

This is attributed to Donsker and Varadhan [44].

We use Lemma I.1 together with the "Two-Function Hypercontractivity Induction Theorem" [28], which shows that if $\mathcal{P}_{XY}^{\otimes n}$ is (r, s)-hypercontractive if and only if $\mathcal{P}_{XY}$ is (r, s)-hypercontractive. This implies that $\operatorname*{E}_{\mathcal{P}_{XY}^{\otimes n}}[f(X)g(Y)] \leq \operatorname*{E}_{\mathcal{P}_{X}^{\otimes n}}[f(X)^r]^{1/r} \operatorname*{E}_{\mathcal{P}_{Y}^{\otimes n}}[g(Y)^s]^{1/s}$ for all functions f, g if and only $\mathrm{D}(\mathcal{Q}_{XY} \,\|\, \mathcal{P}_{XY}) \geq \frac{1}{r}\mathrm{D}(\mathcal{Q}_X \,\|\, \mathcal{P}_X) + \frac{1}{s}\mathrm{D}(\mathcal{Q}_Y \,\|\, \mathcal{P}_Y)$ for all probability distributions $\mathcal{Q}_{XY}$. In the proof of Theorem 3 we have $\mathcal{P}_{XY} = \left[\begin{smallmatrix} w_1 & w_q - w_1 \\ w_u - w_1 & 1 - w_q - w_u + w_1 \end{smallmatrix}\right]$ and consider all the probability distributions $\mathcal{Q}_{XY} \ll \mathcal{P}_{XY}$ minimizing $\mathrm{D}(\mathcal{Q}_{XY} \,\|\, \mathcal{P}_{XY})$ given $\operatorname*{E}_{\mathcal{Q}_X}[X] = \left[\begin{smallmatrix} t_q \\ t_u \end{smallmatrix}\right]$.

The obtained inequalities can be used directly with the framework by Panigrahy et al. [25] to obtain bounds on "Robust Expansion", which has been shown to give lower bounds for 1-cell and 2-cell probe data structures, with word size $n^{o(1)}$ and $o(\log n)$ respectively.

The Directed Noise Operator: We extend the range of our lower bounds further, by studying a recently defined generalization of the p-biased noise operator [45], [46], [47], [48]. This "Directed Noise Operator", $T_\rho^{p_1 \to p_2} : L_2(\{0,1\}^d, \pi_{p_1}^{\otimes d}) \to L_2(\{0,1\}^d, \pi_{p_2}^{\otimes d})$ has the property $\widehat{T_\rho^{p_1 \to p_2} f}^{(p_2)}(S) = \rho^{|S|} \hat{f}^{(p_1)}(S)$ for any $S \subseteq [d]$, where $\hat{f}^{(p)}(S)$ denotes the p-biased Fourier coefficient of f. Just like the Ornstein Uhlenbeck operator, we show that $T_\sigma^{p_2 \to p_3} T_\rho^{p_1 \to p_2} = T_{\rho\sigma}^{p_1 \to p_3}$ and that $T_\rho^{p_2 \to p_1}$ is the adjoint of $T_\rho^{p_1 \to p_2}$. By connecting this operator to our hypercontractive theorem, we can integrate the results by Oleszkiewicz and obtain provably optimal points on the (t_q, t_u) trade-off.

We show that for p-biased distributions over $\{0,1\}^n$, we can add the following line to the list of equivalent statements in Lemma I.1:

3) For all functions $f : \{0,1\}^n \to \mathbb{R}$ it holds $\|T_\rho^{p_1 \to p_2} f\|_{L_{s'}(p_1)} \leq \|f\|_{L_r(p_2)}$.

The operator allows us to prove some optimal choices for r and s in Lemma I.1 (and by effect for t_q and t_u.)

Another use of T is in proving lower bounds outside of the random instance $w_2 = w_q w_u$ regime. Using the power means inequality over p-biased Fourier coefficients, we show the relation

$$\left(\langle T_\alpha^{p \to p} f, f \rangle_{L_2(p)} / \|f\|_{L_2(p)}^2\right)^{1/\log(1/\alpha)} \leq \left(\langle T_\beta^{p \to p} f, f \rangle_{L_2(p)} / \|f\|_{L_2(p)}^2\right)^{1/\log(1/\beta)}.$$

which is allows comparing functions under two different noise levels. This is stronger than hypercontractivity, even though we can prove it in fewer instances. The proof can been seen as a variation of [31] and we get a lower bound with a similar range, but without asymptotics and for Set Similarity instead of Hamming space Similarity Search.

E. Related Work

For the reasons laid out in the introduction, we will compare primarily against approximate algorithms. The best of those are all able to solve GapSS, thus making it easy to draw comparisons. The methods known as Bit Sampling [30] and SimHash (Hyperplane rounding) [49], while sometimes better than MinHash[11] and Chosen Path [14] are always worse (theoretically) that Spherical LSF, so we won't perform a direct comparison to those.

It should be noted that both Chosen Path and Spherical LSF both have proofs of optimality in the restricted models. However these proofs translated to only a certain region of the (w_q, w_u, w_1, w_2) space, and so they may nearly always be improved.

Arguably the largest break-through in Locality Sensitive Hashing, LSH, based data structures was the introduction of *data-dependent* LSH [19], [9], [50]. It was shown how to reduce the general case to an instance in which many LSH schemes work better. Using those data structures on GapSS with $w_2 > w_q w_u$ will often yield better performance than the algorithms described in this paper. However, in the "random instance" case $w_2 = w_q w_u$, which is the main focus of this paper, data-dependency has no effect, and so this issue won't show up much in our comparisons.

We note that even without a reduction to the random instance, for many practical uses, it is natural to assume such "independence" between the query and most of the dataset. Arguably this is the main reason why approximate similarity search algorithms have gained popularity in the first place. In practice, some algorithms for Set Similarity Search take special care to handle "skew" data distributions [51], [52], [53], in which some elements of the Universe are heavily over or under-represented. This reduces the remaining dataset to a random instance.

Many of the algorithms, based on the LSH framework, all had space usage roughly $n^{1+\rho}$ and query time n^ρ for the same constant ρ. This is known as the "balanced regime" or the "LSH regime". Time/space trade-offs are important, since $n^{1+\rho}$ can sometimes be too much space, even for relatively small ρ. Early work on this was done by Panigrahy [16] and Kapralov [54] who gave smooth trade-offs ranging from space $n^{1+o(1)}$ to query time $n^{o(1)}$. A breakthrough was the use of LSF (rather than LSH), which allowed time/space trade-offs with sublinear query time even for near linear space and small approximation [17], [23], [18].

Comparison to Spherical LSF: We use "Spherical LSF" as a term for the algorithms [24] and [17], but in particular section 3 of [10], which has the most recent version. The algorithm solves the (r, cr)-Approximate Near Neighbour problem, in which we, given a dataset $Y \subseteq \mathbb{R}^d$ and a query $q \in \mathbb{R}^d$ must return $y \in Y$ such that $\|q - y\| < cr$ or determinate that there is no $y' \in Y$ with $\|y - q\| \leq r$.

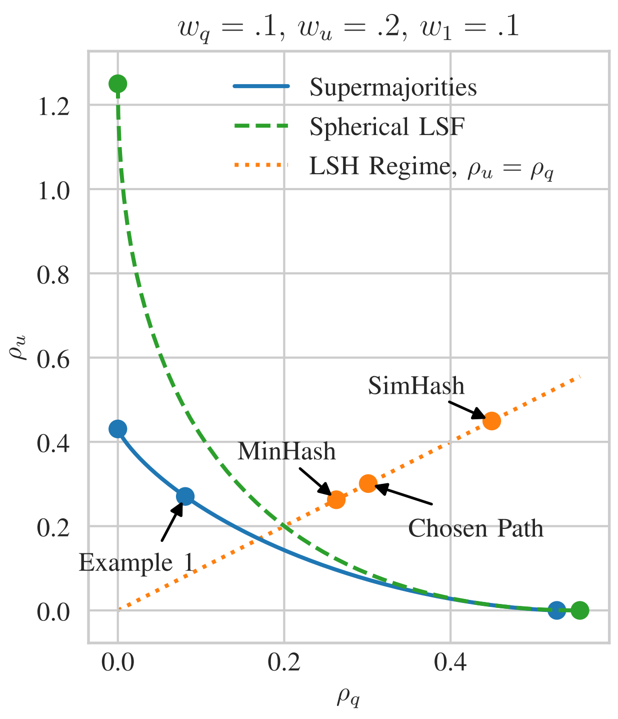


Figure 2: Comparison to Spherical LSF: Plots of the achievable ρ_q (time exponent) and ρ_u (space exponent) achievable with Theorem 1. The plots are drawn in the "random setting", $w_2 = w_q w_u$ where Spherical LSF and Data-Dependent LSH coincide.

The algorithm is a tree over the points, P. At each node they sample T i.i.d. Gaussian d-dimensional vectors $z_1, \dots, z_T$ and split the dataset up into (not necessarily disjoint) "caps" $P_i = \{p \in P \mid \langle z_i, p\rangle \geq t_u\}$. They continue recursively and independently until the expected number of leaves shared between two points at distance $\geq cr$ is $\approx n^{-1+\varepsilon}$.

The real algorithm also samples includes some caps that are dependent on an analysis of the dataset. This allows obtaining a query time of $n^{1/(2c^2-1)}$, for all values of r, rather than only in the "random instance", which, for data on the sphere, corresponds to $r = 1/(\sqrt{2}c)$. (To see this, notice that $rc = 1/\sqrt{2}$, which is the expected distance between two orthogonal points on a sphere.)

Whether we analyse the data-independent algorithm or not, however, a key property of Spherical LSF is that each node in the tree is independent of the remaining nodes. This allows a nice inductive analysis. In comparison, in our algorithm, the nodes are not independent. Whether a certain node gets pruned, depends on which elements from the universe were sampled at all the previous nodes along the path from the root. One could imagine doing Spherical LSF with a running total of inner products along each path, which would make the space partition more smooth, and possible better in practice. Something along these lines was indeed suggested in [24], however it wasn't analysed, as for Spherical LSF *the inner products at each node are continuous, and the thresholds can be set at any precision.*

It is clear that Spherical LSF can solve GapSS – one simply needs an embedding of the sets onto the sphere. The embedding $x \mapsto x/\|x\|_2$ was considered by [14] while other authors have considered $x \mapsto (2x-1)/\sqrt{d}$ and various asymmetric embeddings [55]. We would like to find the most efficient embedding to get a fair comparison. However, we don't know how to do this optimally over all possible embeddings, which include using MinHash and possibly somehow emulating Supermajorities.[13] We instead show that the embedding $x \mapsto (x-w)/\sqrt{w(1-w)}$ is the the most efficient *affine* embedding. The proof can be found in the full version [22]. In Figure 2 and Figure 3 the ρ-values of Spherical LSF are obtained using this optimal embedding.

From the figures, we see the two main cases in which Spherical LSF is suboptimal. As the sets get very small $(w_q, w_u, w_1 \to 0)$ the ρ value in the LSH regime goes to 1, whereas Supermajorities (as well as MinHash and Chosen Path) still obtain good performance. Similarly in the asymmetric case $w_q \neq w_u$, as we make ρ_q very small, the performance gap between Supermajorities and Spherical LSF can grow to arbitrarily large polynomial factors.

Comparison to MinHash: Given a random function $h : \mathcal{P}(\{1,\dots,d\}) \to [0,1]$, the MinHash algorithm hashes a set $x \subseteq \{1,\dots,d\}$ to $m_h(x) = \arg\min_{i\in x} h(i)$. One can show that $Pr[m_h(x) = m_h(y)] = J(x,y) = \frac{|x\cap y|}{|x\cup y|}$. Using the LSH framework by Indyk and Motwani [30] this yields a data structure for Approximate Set Similarity Search over Jaccard similarity, J, with query time dn^ρ and space usage $n^{1+\rho} + dn$, where $\rho = \frac{\log j_1}{\log j_2}$ and j_1 and j_2 define the gap between "good" and "bad" search results. As Jaccard similarity is a set similarity measure, it is clear that MinHash yields a solution to the GapSS problem with $\rho_q = \rho_u = \log \frac{w_1}{w_q+w_u-w_1} \big/ \log \frac{w_2}{w_q+w_u-w_2}$. Similarly, and that any solution to GapSS can yield a solution to Approximate SSS over Jaccard similarity.

MinHash has been very popular, since it gives a good, all-round algorithm for Set Similarity Search, that is easy to implement. In Figure 3 we see how MinHash performant for different settings of GapSS. In particular we see that when solving the Superset Search problem, which is a common use case for MinHash, our new algorithm obtains quite a large polynomial improvement, except when the Jaccard similarity between the query and the sought after superset is nearly 0 (which is hardly an interesting situation.)

[13]We would also need some sort of limit on how much time the embedding takes to perform.

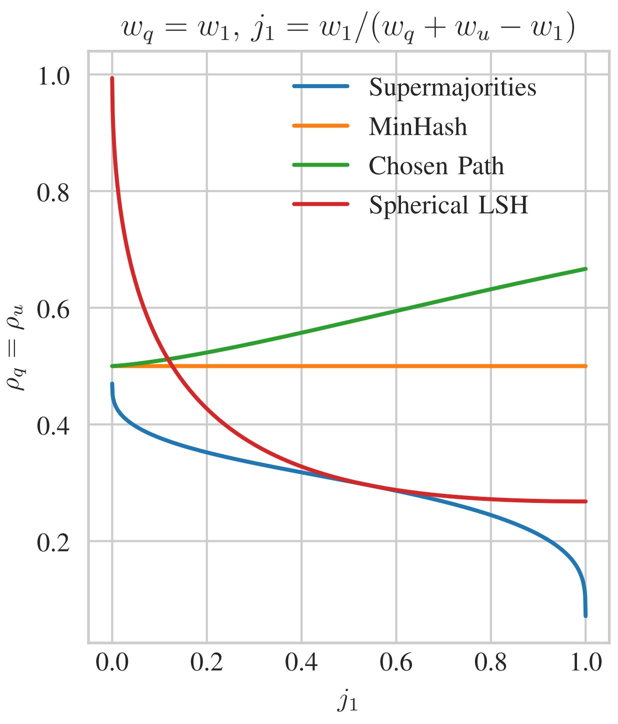


Figure 3: Comparison to MinHash: Varying the Jaccard similarity, j_1, among close sets, while fixing the exponent of MinHash at $\rho = .5$ in the subset search instance, $w_q = w_1$.

MinHash is quite different from the other algorithms considered in this section. For some more intuition of why MinHash is not optimal for Approximate Set Similarity Search, we show in the full paper [22] that MinHash can be seen as an average of a family of Chosen Path like algorithms. We also show that an average is always worse than simply using the best family member, which implies that MinHash is never optimal.

Comparison to Chosen Path: The Chosen Path algorithm of [14], is virtually identical to Supermajorities, when parametrized with $t_q = t_u = 1$. Similar to Spherical LSF and our decoding algorithm, they build a tree on the datasets. For each node they sample iid. Elements $x_1, x_2, \dots \in U$ from the universe, and split the data into (not necessarily disjoint) subsets $P_i = \{p \in P \mid x_i \in p\}$. They again continue recursively and independently until the expected number of leaves shared between two dissimilar points is sufficiently small.

The case $t_q = t_u = 1$ however, turns out to be a very special case of our algorithm, because one can decide which leaves of the tree to prune, without knowledge of what happened previously on the path from the root to the node. This allows a nice inductive analysis of Chosen Path based on second moments, which is a classic example literature on branching processes. Meanwhile, for our general algorithm, we need to analyse the resulting branching random walk, a conceptually much different beast.

Doing the analysis, one gets a data structure for Approximate Set Similarity Search over Braun-Blanquet similarity, $B(x, y) = \frac{|x \cap y|}{\max\{|x|,|y|\}}$, with query time $|q|n^\rho$ and auxiliary space usage $n^{1+\rho}$, where $\rho = \frac{\log b_1}{\log b_2}$ and b_1 and b_2 define the gap between "good" and "bad" search results. Since $t_q = t_u = 1$ is sometimes the optimal choice for Supermajorities, it is clear that we must sometimes coincide in performance with Chosen Path. In particular, this happens as $w_q = w_u$ and $w_q, w_u, w_1 \to 0$. This is also one of the case where our lower bound Theorem 2 is sharp, which confirms, in addition to the lower bound in [14] that both algorithms are sharp for LSF data structures in this setting.

In the case $w_q = w_u$ the ρ value of Chosen Path can be equivalently written in terms of Jaccard similarities as $\log \frac{2j_1}{1+j_1} \big/ \log \frac{2j_2}{1+j_2}$, which is always smaller than the $\log j_1 / \log j_2$ obtained by MinHash. (This value, $2j/(1+j)$, is also known as the Sørensen-Dice coefficient of two sets.) However, in the case $w_q \neq w_u$ Chosen Path can be much worse than MinHash. In [14] it was left as an open problem whether MinHash could be improved upon in general. It is a nice result that the balanced ρ value of Supermajorities (when $\rho_q = \rho_u$) can be shown (numerically) to always be less than or equal to $\log \frac{2j_1}{1+j_1} \big/ \log \frac{2j_2}{1+j_2}$, even when $w_q \neq w_u$.

Partial Match (PM) and Super-/Subset queries (SQ): Partial Match asks to pre-process a database D of n points in $\{0,1\}^d$ such that, for all query of the form $q \in \{0,1,*\}^d$, either report a point $x \in D$ matching all non-$*$ characters in q or report that no such x exists. A related problem is Super-/Subset queries, in which queries are on the form $q \in \{0,1\}^d$, and we must either report a point $x \in D$ such that $x \subseteq q$ (resp. $q \subseteq x$) or report that no such x exists. Folklore reductions show that they are all equivalent to the subset query problem.

The classic approach, studied by Rivest [56], is to split up database strings like `supermajority` and file them under `s`, `u`, `p` etc. Then when given query like `set` we take the intersection of the lists `s`, `e`, `t`. Sometimes this can be done faster than brute force searching each list. He also considered the space heavy solution of storing all subsets, and showed that when $d \leq 2\log n$, the trivial space bound of 2^d can be somewhat improved.

Indyk, Charikar and Panigrahy [57] also studied the exact version of the problem, and gave, for each $c \in [n]$, an algorithm with $O(n/2^c)$ time and $n2^{O(d \log^2 d \sqrt{c/\log n})}$ space, and another with $O(dn/c)$ query time and nd^c space. Their approach was a mix between the shingling method of Rivest, building a look-up table of size $\approx 2^{\Omega(d)}$, and a brute force search. These bounds manage to be non-trivial for $d = \omega(\log n)$, however only slightly. (e.g. $n/\operatorname{poly}(\log n)$

time with polynomial space.)

There has also been a large number of practical papers written on Partial Match / Subset queries or the equivalent batch problem of subset joins [58], [59], [60], [61], [62]. Most of these use similar methods to the above, but save time and space in various places by using bloom filters and sketches such as MinHash [11] and HyperLogLog [63].

F. Conclusion

The full paper, besides proofs of the above claims, contains further reductions showing the relation to MinHash and other well known schemes. By showing that supermajorities solve set similarity optimally for any set similarity measure, we not only unify and explain the performance of the previous literature, but also recover major performance improvements and space/time trade-offs.

REFERENCES

[1] S.-S. Choi, S.-H. Cha, and C. C. Tappert, "A survey of binary similarity and distance measures," *Journal of Systemics, Cybernetics and Informatics*, vol. 8, no. 1, pp. 43–48, 2010.

[2] L. Jia, L. Zhang, G. Yu, J. You, J. Ding, and M. Li, "A survey on set similarity search and join." *International Journal of Performability Engineering*, vol. 14, no. 2, 2018.

[3] R. Williams, "A new algorithm for optimal 2-constraint satisfaction and its implications," *Theoretical Computer Science*, vol. 348, no. 2, pp. 357–365, 2005.

[4] T. D. Ahle, R. Pagh, I. Razenshteyn, and F. Silvestri, "On the complexity of inner product similarity join," in *Proceedings of the 35th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*. ACM, 2016, pp. 151–164.

[5] A. Abboud, A. Rubinstein, and R. Williams, "Distributed pcp theorems for hardness of approximation in p," in *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE, 2017, pp. 25–36.

[6] L. Chen and R. Williams, "An equivalence class for orthogonal vectors," in *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM, 2019, pp. 21–40.

[7] S. Har-Peled, P. Indyk, and R. Motwani, "Approximate nearest neighbor: Towards removing the curse of dimensionality." *Theory of computing*, vol. 8, no. 1, pp. 321–350, 2012.

[8] T. Christiani, R. Pagh, and M. Thorup, "Confirmation sampling for exact nearest neighbor search," *arXiv preprint arXiv:1812.02603*, 2018.

[9] A. Andoni and I. Razenshteyn, "Optimal data-dependent hashing for approximate near neighbors," in *Proceedings of the Forty-Seventh Annual ACM Symposium on Theory of Computing*. ACM, 2015, pp. 793–801.

[10] A. Andoni, T. Laarhoven, I. Razenshteyn, and E. Waingarten, "Optimal hashing-based time-space trade-offs for approximate near neighbors," in *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*. Society for Industrial and Applied Mathematics, 2017, pp. 47–66.

[11] A. Z. Broder, S. C. Glassman, M. S. Manasse, and G. Zweig, "Syntactic clustering of the web," *Computer Networks and ISDN Systems*, vol. 29, no. 8-13, pp. 1157–1166, 1997.

[12] A. Z. Broder, "On the resemblance and containment of documents," in *Compression and Complexity of Sequences 1997. Proceedings*. IEEE, 1997, pp. 21–29.

[13] R. Pagh, M. Stöckel, and D. P. Woodruff, "Is min-wise hashing optimal for summarizing set intersection?" in *Proceedings of the 33rd ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*. ACM, 2014, pp. 109–120.

[14] T. Christiani and R. Pagh, "Set similarity search beyond minhash," in *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, 2017, pp. 1094–1107.

[15] X. Yan, J. Li, X. Dai, H. Chen, and J. Cheng, "Norm-ranging lsh for maximum inner product search," in *Advances in Neural Information Processing Systems*, 2018, pp. 2956–2965.

[16] R. Panigrahy, "Entropy based nearest neighbor search in high dimensions," in *Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*. Society for Industrial and Applied Mathematics, 2006, pp. 1186–1195.

[17] T. Laarhoven, "Tradeoffs for nearest neighbors on the sphere," *arXiv preprint arXiv:1511.07527*, 2015.

[18] A. Andoni, T. Laarhoven, I. Razenshteyn, and E. Waingarten, "Optimal hashing-based time-space trade-offs for approximate near neighbors," in *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM, 2017, pp. 47–66.

[19] A. Andoni, P. Indyk, H. L. Nguyen, and I. Razenshteyn, "Beyond locality-sensitive hashing," in *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*. Society for Industrial and Applied Mathematics, 2014, pp. 1018–1028.

[20] M. Dubiner, "Bucketing coding and information theory for the statistical high-dimensional nearest-neighbor problem," *IEEE Transactions on Information Theory*, vol. 56, no. 8, pp. 4166–4179, 2010.

[21] E. Cohen and H. Kaplan, "Leveraging discarded samples for tighter estimation of multiple-set aggregates," *ACM SIGMETRICS Performance Evaluation Review*, vol. 37, no. 1, pp. 251–262, 2009.

[22] T. D. Ahle and J. B. T. Knudsen, "Subsets and supermajorities: Optimal hashing-based set similarity search," *CoRR*, vol. abs/1904.04045, 2020. [Online]. Available: http://arxiv.org/abs/1904.04045

[23] T. Christiani, "A framework for similarity search with space-time tradeoffs using locality-sensitive filtering," in *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM, 2017, pp. 31–46.

[24] A. Becker, L. Ducas, N. Gama, and T. Laarhoven, "New directions in nearest neighbor searching with applications to lattice sieving," in *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM, 2016, pp. 10–24.

[25] R. Panigrahy, K. Talwar, and U. Wieder, "A geometric approach to lower bounds for approximate near-neighbor search and partial match," in *2008 49th Annual IEEE Symposium on Foundations of Computer Science*. IEEE, 2008, pp. 414–423.

[26] ——, "Lower bounds on near neighbor search via metric expansion," in *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*. IEEE, 2010, pp. 805–814.

[27] R. Motwani, A. Naor, and R. Panigrahi, "Lower bounds on locality sensitive hashing," in *Proceedings of the twenty-second annual symposium on Computational geometry*. ACM, 2006,

pp. 154–157.
[28] R. O'Donnell, *Analysis of boolean functions*. Cambridge University Press, 2014.
[29] A. Andoni and I. Razensteyn, "Tight lower bounds for data-dependent locality-sensitive hashing," in *32nd International Symposium on Computational Geometry (SoCG 2016)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2016.
[30] P. Indyk and R. Motwani, "Approximate nearest neighbors: towards removing the curse of dimensionality," in *Proceedings of the thirtieth annual ACM symposium on Theory of computing*. ACM, 1998, pp. 604–613.
[31] R. O'Donnell, Y. Wu, and Y. Zhou, "Optimal lower bounds for locality-sensitive hashing (except when q is tiny)," *ACM Transactions on Computation Theory (TOCT)*, vol. 6, no. 1, p. 5, 2014.
[32] K. Oleszkiewicz, "On a nonsymmetric version of the khinchine-kahane inequality," in *Stochastic inequalities and applications*. Springer, 2003, pp. 157–168.
[33] P. Wolff, "Hypercontractivity of simple random variables," *Studia Mathematica*, vol. 3, no. 180, pp. 219–236, 2007.
[34] M. Kapralov and R. Panigrahy, "Nns lower bounds via metric expansion for l_∞ and emd," in *International Colloquium on Automata, Languages, and Programming*. Springer, 2012, pp. 545–556.
[35] M. Bawa, T. Condie, and P. Ganesan, "Lsh forest: self-tuning indexes for similarity search," in *Proceedings of the 14th international conference on World Wide Web*, 2005, pp. 651–660.
[36] A. Andoni and P. Indyk, "Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions," in *Foundations of Computer Science, 2006. FOCS'06. 47th Annual IEEE Symposium on*. IEEE, 2006, pp. 459–468.
[37] Z. Shi, *Branching random walks*. Springer, 2015.
[38] J. D. Biggins, "Martingale convergence in the branching random walk," *Journal of Applied Probability*, vol. 14, no. 1, pp. 25–37, 1977.
[39] T. M. Chan, "Orthogonal range searching in moderate dimensions: kd trees and range trees strike back," in *33rd International Symposium on Computational Geometry (SoCG 2017)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017.
[40] A. Bonami, " 'E study of the fourier coefficients of the functions of $l\ p(g)$," in *Annals of the Fourier Institute*, vol. 20, no. 2, 1970, pp. 335–402.
[41] W. Beckner, "Inequalities in fourier analysis," *Annals of Mathematics*, pp. 159–182, 1975.
[42] C. Nair, "Equivalent formulations of hypercontractivity using information measures," in *International Zurich Seminar on Communications*, 2014, p. 42.
[43] E. Friedgut, "An information theoretic proof of a hypercontractive inequality," *Entropy*, no. 1/29, 2015.
[44] M. D. Donsker and S. R. S. Varadhan, "Asymptotic evaluation of certain markov process expectations for large time. iv," *Communications on Pure and Applied Mathematics*, vol. 36, no. 2, pp. 183–212, 1983.
[45] D. Ahlberg, E. Broman, S. Griffiths, and R. Morris, "Noise sensitivity in continuum percolation," *Israel Journal of Mathematics*, vol. 201, no. 2, pp. 847–899, 2014.
[46] A. Abdullah and S. Venkatasubramanian, "A directed isoperimetric inequality with application to bregman near neighbor lower bounds," in *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*, 2015, pp. 509–518.
[47] N. Lifshitz, "Hypergraph removal lemmas via robust sharp threshold theorems," *arXiv preprint arXiv:1804.00328*, 2018.
[48] P. Keevash, N. Lifshitz, E. Long, and D. Minzer, "Hypercontractivity for global functions and sharp thresholds," *arXiv preprint arXiv:1906.05568*, 2019.
[49] M. S. Charikar, "Similarity estimation techniques from rounding algorithms," in *Proceedings of the thiry-fourth annual ACM Symposium on Theory of Computing*. ACM, 2002, pp. 380–388.
[50] A. Andoni, I. Razenshteyn, and N. S. Nosatzki, "Lsh forest: Practical algorithms made theoretical," in *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM, 2017, pp. 67–78.
[51] C. Rashtchian, A. Sharma, and D. P. Woodruff, "Lsf-join: Locality sensitive filtering for distributed all-pairs set similarity under skew," *arXiv preprint arXiv:2003.02972*, 2020.
[52] Y. Zhang, X. Li, J. Wang, Y. Zhang, C. Xing, and X. Yuan, "An efficient framework for exact set similarity search using tree structure indexes," in *2017 IEEE 33rd International Conference on Data Engineering (ICDE)*. IEEE, 2017, pp. 759–770.
[53] S. McCauley, J. W. Mikkelsen, and R. Pagh, "Set similarity search for skewed data," in *Proceedings of the 37th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, 2018, pp. 63–74.
[54] M. Kapralov, "Smooth tradeoffs between insert and query complexity in nearest neighbor search," in *Proceedings of the 34th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*. ACM, 2015, pp. 329–342.
[55] A. Shrivastava and P. Li, "Asymmetric lsh (alsh) for sublinear time maximum inner product search (mips)," in *Advances in Neural Information Processing Systems*, 2014, pp. 2321–2329.
[56] R. L. Rivest, "Partial-match retrieval algorithms," *SIAM Journal on Computing*, vol. 5, no. 1, pp. 19–50, 1976.
[57] M. Charikar, P. Indyk, and R. Panigrahy, "New algorithms for subset query, partial match, orthogonal range searching, and related problems," in *International Colloquium on Automata, Languages, and Programming*. Springer, 2002, pp. 451–462.
[58] K. Ramasamy, J. M. Patel, J. F. Naughton, and R. Kaushik, "Set containment joins: The good, the bad and the ugly," in *VLDB*, 2000.
[59] S. Melnik and H. Garcia-Molina, "Adaptive algorithms for set containment joins," *ACM Transactions on Database Systems (TODS)*, vol. 28, no. 1, pp. 56–99, 2003.
[60] A. Goel and P. Gupta, "Small subset queries and bloom filters using ternary associative memories, with applications," *ACM SIGMETRICS Performance Evaluation Review*, vol. 38, no. 1, pp. 143–154, 2010.
[61] P. Agrawal, A. Arasu, and R. Kaushik, "On indexing error-tolerant set containment," in *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*. ACM, 2010, pp. 927–938.
[62] R. C. Fernandez, J. Min, D. Nava, and S. Madden, "Lazo: A cardinality-based method for coupled estimation of jaccard similarity and containment," in *ICDE*, 2019.
[63] P. Flajolet, É. Fusy, O. Gandouet, and F. Meunier, "Hyperloglog: the analysis of a near-optimal cardinality estimation algorithm," in *Discrete Mathematics and Theoretical Computer Science*. Discrete Mathematics and Theoretical Computer Science, 2007, pp. 137–156.