Analysis of Filtering for Similarity Search using Sketches

William Josephson Qin Lv Zhe Wang Moses Charikar Kai Li

Department of Computer Science
Princeton University, Princeton, NJ 08544

{*wkj, qlv, zhewang, moses, li*}*@CS.Princeton.EDU*

ABSTRACT

Sketches are tiny data structures that can be used to efficiently perform filtering high-dimensional data for similarity search. To design real systems with sketching techniques, an important design decision is the choice of sketch size given the targeted dataset size and desired filtering quality. Such design decisions need to be made without the dataset, or at least without the whole dataset. This paper provides analytical and experimental results to help system designers make such design decisions. We first show that the ℓ_1 distances between data objects in these datasets fits lognormal distributions well. We then present a rank-based filtering model for the sketching algorithm that uses Hamming distance to approximate ℓ_1 distance. Our experimental results show that the model gives conservative and good prediction for image, audio and 3D shape datasets. Finally, we show that the parameters of the model can be set with a small sample dataset and the resulting model can make good predictions for a large dataset.

1. INTRODUCTION

Content-based similarity search for massive amounts of feature-rich (non-text) data is difficult because such data is noisy and their feature vectors are high dimensional. Recent studies show that a promising approach is to use sketches constructed from feature vectors as compact metadata for a similarity search engine. At search time, sketches are used by a filtering step to quickly filter out unlikely answers, resulting in a much smaller candidate set which is then used for final ranking.

Filtering by itself is a useful mechanism for designing a similarity search engine for two main reasons. The first is that although recent studies have suggested indexing data structures such as Cover Tree[1] for k-Nearest-Neighbor (KNN) search of high-dimensional data, these data structures work well only with datasets that have low intrinsic dimensionality. In other words, the "curse of dimensionality" problem is still far from being solved. For data with high intrinsic dimensionality, filtering metadata can often be more efficient. The second is that content-based similarity search capability is often integrated with attribute-based (or annotation-based) search. A typical integration method is to perform an attribute-based search to produce an intermediate dataset which can be filtered efficiently into a candidate set for final ranking. The intermediate dataset size may not be large enough for any indexing method.

Properly sized sketches can greatly reduce the storage requirement for metadata and speed up similarity search while maintaining good search quality. An important design decision is sketch size in bits, given the desired filtering quality and the dataset size for a specific data type. Choose too few bits, and the distance estimates computed from the sketches will be inaccurate. Choose too many bits, and the sketches will needlessly waste storage space and CPU time.

Another parameter is the candidate set size after filtering. If the candidate set is too small, some good answers may not be included in the candidate set and search quality is poor. If the candidate set is too large, time is wasted on bad answers. To make the problem even worse, a system designer would ideally be able to determine these parameters at system initialization time when she knows only the proposed type of data with a small sample set and eventual target dataset size.

This paper presents three analytical and experimental results to help systems designers make such design decisions. The first result is to study how to model the distance distributions of various feature-rich datasets. We show that lognormal distribution models the ℓ_1 distance distributions of all our datasets (images, audio and 3D shape data) quite well and it fits much better than other common models such as the normal distribution. This result allows us to use lognormal distribution to model the distance distribution of a specific dataset.

The second set of results is to model the parameters of the sketching technique that uses Hamming distance to approximate ℓ_1 distance. This sketching technique has been shown to be useful and efficient to filter several kinds of data for similarity search. We present a rank-based filtering model for the sketch construction. We have shown, by experimenting with image, audio, and 3D shape datasets, that the model can conservatively predict the required sketch size, given desired filtering quality, target dataset size, and filtering result size. This result can also be used to determine the filtering result size given other parameters.

Our final study is to investigate how to use the rank-based filtering model to help systems designers make design decisions without the whole dataset. By conducting experiments with the three real datasets, our results show that the rank-based filtering model can perform well, yielding useful, conservative predictions for a large dataset if the parameters of the model are set with a small sample dataset. This result allows systems designers to build the model into a software tool.

Existing work on nearest neighbor search can be broadly classified into two categories: theoretical work based on worst-case analysis and empirical work based upon performance on actual datasets. Our work takes a middle road: we use information about the dataset to accurately model the performance of the filtering algorithm for nearest neighbor search. We are not aware of any previous work that incorporates such distance modeling into the analysis of these algorithms.

The remainder of this paper is organized as follows: Section 2 introduces the similarity search problem, sketching, the filtering technique using sketches, and the questions faced by system designers when using such techniques. Section 3 presents our ranked-based filtering model for analyzing and predicting system performance as a function of model parameters. Section 4 describes the methodology and datasets we have used to evaluate our analytical model experimentally. Section 5 compares the analytical and experimental results for parameter selection and extrapolation for each of the datasets. Finally, Section 6 summarizes related work and Section 7 concludes the paper.

2. FILTERING FOR SIMILARITY SEARCH

This section describes the similarity search problem, sketching algorithm, and filtering method using sketches that are considered in our analytical and experimental study.

2.1 Similarity Search

Informally, similarity search refers to searching a collection of objects to find objects similar to a query object. The objects we will be interested in are noisy, high-dimensional objects such as audio recordings and images. Here, similarity between objects refers to a human-perceived notion of similarity. This informal notion of similarity search is made concrete as follows: objects are represented by high-dimensional feature vectors and similarity is defined in terms of a distance metric on the underlying space of features. Given a query object, q, in this setting, the goal is to find nearby objects, r, such that the distance $d(q,r)$ is small. In particular, we may ask for all objects r within some chosen distance of the query point, or more often, we may ask for the k nearest neighbors of the query point. This latter formulation of the search problem is commonly referred to as the k-nearest neighbor (k-NN) problem.

Although the choice of how to extract features and which distance function to use are domain specific, it is frequently the case in practice that objects are represented by D-dimensional real-valued vectors and the perceptual distance between objects is modeled by one of the ℓ_p norms. For a pair of points $X = (X_1, \ldots, X_D)$ and $Y = (Y_1, \ldots, Y_D)$, these distance functions have the form:

$$d(X,Y) = \left(\sum_{1 \le i \le D} (X_i - Y_i)^p \right)^{1/p}$$

2.2 Sketching

The formalization of similarity search in terms of domain-specific feature extraction and k-nearest neighbor search allows for a simple mathematical description of the similarity search problem. However, domain-specific feature extraction algorithms for noisy, feature-rich data typically produce feature vectors in a high-dimensional vector space and nearest neighbor search in such spaces is notoriously difficult. Thus, it is necessary to attack the "curse of dimensionality" problem in order to design a practical similarity search engine for feature-rich data.

Inspired by dimension reduction ideas recently developed in the theory community, novel techniques have been proposed to convert high-dimensional feature vectors into compact representations called *sketches*. Sketches have two salient properties: their small size and the ability to estimate the distance between two feature vectors from their sketches. Given two feature vectors x and y and their sketches, $s(x)$ and $s(y)$, with respect to a distance function, d, there is an algorithm for computing an estimate of the distance $d(x,y)$ from $s(x)$ and $s(y)$ alone. When a sketch algorithm exists for a particular feature representation and domain-specific distance function, it allows a similarity search engine implementation to store much less metadata per object while retaining efficient and accurate similarity search. By design, the distance estimation algorithm on sketches is much faster than the exact distance calculation on the original feature vectors.

In this paper, we focus on a recently proposed sketching technique [15], which constructs sketches as bit vectors from the original feature vectors such that the Hamming distance between bit vectors approximates the ℓ_1 distance between the original feature vectors. In fact, the sketch is designed to have higher resolution for smaller distances, making it suitable for nearest neighbor search. This sketching technique has proved useful in several application domains.

Briefly, the sketch algorithm works as follows. Let w_i be the "weight" (or importance) assigned by the domain-specific feature extraction algorithm to dimension i and let u_i be the maximum value for the i-th coordinate over all observed feature vectors. Likewise, let l_i be the minimum value for the i-th coordinate. Let D be the dimension of the feature vector, B be the final sketch size in bits, and H be the block size. At system startup time, $B \times H$ random (i, t_i) pairs are generated using Algorithm 1. At run time, the D-dimensional feature vector, x is converted into a B-bit bit vector using algorithm 2. For further details and a proof of correctness, we refer the reader to [15].

2.3 Filtering using Sketches

Algorithm 1 Generate $B \times H$ Random (i, t_i) Pairs

```
input:   $B, H, D, l[D], u[D], w[D]$
output:   $p[D], rnd\_i[B][H], rnd\_t[B][H]$
$p_i = w_i \times (u_i - l_i)$;     $for\ i = 0, \ldots, D-1$
normalize $p_i$     $s.t.\ \Sigma_{i=0}^{d-1}\ p_i = 1.0$
for $(b = 0; b < B; b++)$ do
    for $(h = 0; h < H; h++)$ do
        pick random number $r \in [0, 1)$
        find $i$     $s.t.\ \Sigma_{j=0}^{i-1}\ p_i <= r < \Sigma_{j=0}^{i}\ p_i$
        $rnd\_i[b][h] = i$
        pick random number $t_i \in [l_i, u_i]$
        $rnd\_t[b][h] = t_i$
    end for
end for
```

Algorithm 2 Convert Feature Vector to B-Bit Vector

```
input:   $v[D], B, H, rnd\_i[B][H], rnd\_t[B][H]$
output:   $bits[B]$
for $(b = 0; b < B; b++)$ do
    $x = 0$
    for $(h = 0; h < H; h++)$ do
        $i = rnd\_i[b][h]$;     $t_i = rnd\_t[b][h]$
        $y = (v_i < t_i\ ?\ 0 : 1)$
        $x = x \bigoplus y$
    end for
    $bits_b = x$
end for
```

Since sketches require little storage space and since the distance between query objects can be estimated from sketches efficiently, sketches can be used to implement a filtering query processor for similarity search. A filtering query processor first constructs a candidate set of result objects for a given query object on the basis of sketch distance. The candidate set size is chosen to be large enough such that it is likely to contain the k-nearest neighbors under the original distance on feature vectors. In effect, the construction of the candidate set "filters out" the vast majority of objects in the system that are far from the query object while still capturing the objects close to the query. Since sketches are small and distance estimation on sketches are very efficient, a simple, yet practical approach for generating this candidate set is a linear scan through the set of all sketches.

The second step in a filtering query processor is the ranking of the candidate set by the original distance metric on the original feature vector. This exact computation need only be carried out once for each point in the candidate set. The k-nearest neighbors in the candidate set under the original distance metric is then taken as the query result set. The underlying assumption in a filtering query processor is that the k-nearest neighbors in the candidate set is an accurate estimate of the k-nearest neighbors in the full data set. In practice, one must choose the candidate set to be large enough that it captures a sufficiently large fraction of the k-nearest neighbors under the original distance, but not so large that it adversely affects search engine performance. If the candidate set is too small, the query processor will be fast, but the search quality may be poor. On the other hand, if the candidate set is too big, the processor will waste time and resources on unlikely candidates. We can capture this inherent tradeoff between search quality and filter set size by asking what filter ratio is necessary to achieve a particular quality goal. If k is the number of results to return, a filter with filter ratio t will return a candidate set of size $t \times k$. A filtering query processor seeks to optimize t for a given fraction of the k-nearest neighbors in the final result set.

A system designer who adopts the filtering approach to similarity search must choose not only a particular domain-specific feature representation and distance function, but also an appropriate sketching algorithm and a set of parameters for sketching and filtering. More specifically, we are interested in answering the following questions:

- What is an appropriate choice for the sketch size, B?
- What is the best H value for XORing when constructing sketches?
- What filter ratio, t, should the filter have?
- How do the preceding parameters change as the input data set size grows?

The rest of the paper presents our analytical and experiment results to answer these questions.

3. ANALYTICAL MODEL

We use the following notation:

- B: sketch size in bits
- k: number of similar objects to return
- t: filter ratio – *i.e.* filtered set size is $k \times t$
- H: XOR block size in bits for sketching
- S: the set of domain-specific feature vectors
- D: the dimensionality of vectors in S
- $d(x, y)$: the domain-specific distance function on $x, y \in S$
- $s^0(x)$: the $H \times B$-bit sketch of $x \in S$ before XORing
- $s(x)$: the B-bit sketch of the feature vector $x \in S$
- $d_s(x, y)$: the sketch distance between $x, y \in S$

We now describe a simple analytical model for filtering using the ℓ_1 sketch of Section 2.2. This model provides a basis for system designers to choose appropriate parameter values for a sketch-based filtering similarity search query processor. In particular, for a given data set size, N, and result set size, k, the model predicts the relationship between recall, filter ratio (t), sketch size (B), and XOR block size (H). Thus, the model allows a system designer to choose the system parameters in anticipation of future growth.

In the following description let S be a set of N objects, each represented by a D-dimension feature vector. Given objects

q and r, let $d(q,r)$ be the *feature distance* between q and r, $s(q)$ and $s(r)$ be the sketches of q and r, respectively, and $d_s(q,r)$ the *sketch distance* between q and r. We define the *rank* of r given q to be the number of points in S that precede r when objects are ordered in increasing order of feature distance from q. For a fixed query q, let r_i denote the ith object in S in this ordering. Similarly, we define the *sketch rank* of r to be the number of points in S that precede r when objects are ordered in increasing order of sketch distance to q.

The goal is for the analytical model to answer the question: Given N, k fixed, as a function of t, B, and H, what fraction of the points $p \in S$ with rank at most k have sketch rank at most $k \times t$? We develop the model in a series of steps. First, we describe how we model the distribution of feature distances in the data set. Second, we obtain an expression for the distribution of the sketch distance as a function of the feature distance. Next, we model the distribution of the sketch rank of an object $r \in S$ for a query q as a function of its feature distance from q. This uses the distribution of feature distances in the data set and distribution of sketch distances. Finally, we use this model for the sketch rank to estimate the recall for a given filter ratio value. Each of these steps is described in the subsections that follow.

3.1 Distance Distribution

Since the sketch distance between two objects is related to the original feature vector distance, we first study the distribution of feature vector distances. For one particular query object, we calculate the feature vector distances of all the other objects in the dataset to this query object. The histogram of all the object feature distances forms the feature vector distance distribution for that particular query object. With the feature distance distribution known, we will be able to predict the sketch distance between the query object and rest of the objects using the analytic model described in the next section.

One of the goals for our approach is to predict the sketch performance when the dataset size changes. In order to do this, we predict the distribution of object distances in a data set using the distribution of distances in a smaller sample. The basis for this is the hypothesis that every data type is associated with an underlying object distance distribution. The particular distances observed in a specific data set can be viewed as a sample of the distribution associated with the data type. For example, in Figure 1, we compare the distance distribution with the full dataset with that of a uniformly sampled dataset with only one-tenth of the data points.

One subtlety in modeling distances is that the distribution of distances from different query objects can be different and using a single distribution for them can lead to errors. The distance distributions for different query objects have similar shapes but are peaked at different points. Since our data objects have a natural bound on each dimension, the objects are contained in a high dimensional rectangle. The location of the query object in this high dimensional rectangle will affect the peak of the feature distance distribution. In order to model this variation, we pick a random sample of 100 query objects and use their distance distributions to approximate the overall distance distributions. We compared this approach with using a single average distance distribution. The latter did not perform as well as the approach that explicitly models the variation in object distance distributions.

Further, we approximate the empirical individual query object distance distributions by a distribution with a closed form expression. The details of this appear in Section 5.1.

3.2 Sketch Distance Distribution

Given a dataset S, let w_i, u_i, l_i be the weight, upper bound and lower bound of the i-th dimension, respectively. Let $T = \sum_i w_i \times (u_i - l_i)$. Using the sketch algorithm of Section 2.2, for every object $r \in S$, we construct the initial bit vector $s^0(r)$ of length $B \times H$. For a fixed query point q, consider object $r \in S$ and let $x = d(q,r)/T$. The probability that $s^0(q)$ disagrees with $s^0(r)$ in the j-th bit is:

$$\Pr[s^0_j(q) \neq s^0_j(r)] = d(q,r)/T = x$$

After XORing contiguous H-bit blocks of s^0 to produce the final B-bit sketch, the probability that the two sketches differ in bit j is:

$$\Pr[s_j(q) \neq s_j(r)] = p(x) = \frac{1}{2}\left(1 - (1-2x)^H\right) \qquad (1)$$

Thus, the probability that the two B-bit sketches $s(q)$ and $s(r)$ differ in exactly b bits is given by the binomial distribution:

$$\Pr[d_s(q,r) = b] = p(x,b) = \binom{B}{b} p(x)^b \, (1 - p(x))^{B-b} \qquad (2)$$

where $p(x)$ is given by equation (1). This formula gives the probability distribution of the sketch distance as a function of the feature distance.

3.3 Rank Distribution

Consider an object $r \in S$. We would like to estimate the sketch rank of r, *i.e.* the number of objects that precede r when we order all objects in S in increasing order of sketch distance to query object q. A key assumption in this calculation is that the sketch distances are independent of each other. While this assumption is not completely accurate, it is a reasonable approximation. As we discuss later, this leads to a conservative estimate on the quality of the filtering results. We also assume that in the ordering by sketch distances, objects with the same sketch distance are ordered randomly, that is, for two objects with the same sketch distance, the probability that one precedes the other is exactly $1/2$.

The sketch rank of r is dependent on the sketch distance $d_s(q,r)$. Consider the event $d_s(q,r) = b$. Note that the probability of this event is a function of the feature distance $d(q,r)$ and is calculated in (2). Consider an object $r' \in S$ such that $d(q,r')/T = x$ and let $s = d_s(q,r')$ be the sketch distance of r'. Let $P(x,b)$ be the probability that r' is ranked lower (*i.e.* closer to q) than r when $d_s(q,r) = b$. Note that

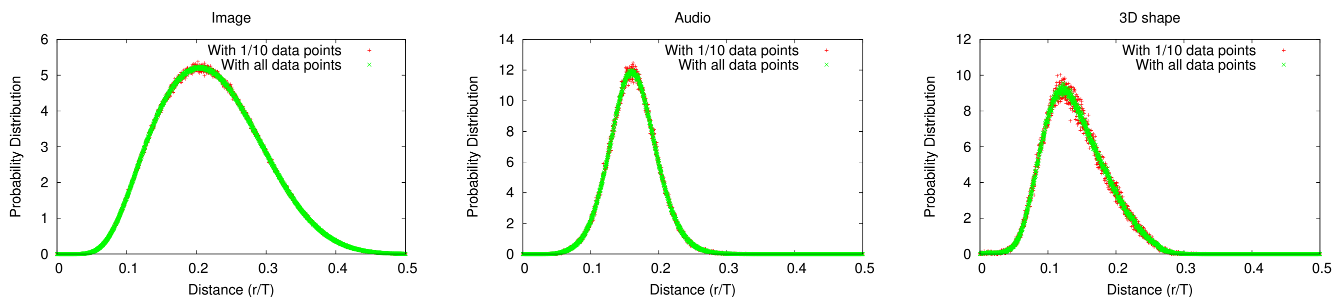

Figure 1: Compare Distance Distribution of Full Dataset and 1/10 of Dataset.

this is a function of $x = d(q, r')/T$ and the value b of $d_s(q, r)$.

$$\begin{aligned} P(x, b) &= \Pr[s < b] + \frac{1}{2}\Pr[s = b] \\ &= \sum_{i=0}^{b-1} \Pr[s = i] + \frac{1}{2}\Pr[s = b] \qquad (3) \\ &= \sum_{i=0}^{b-1} p(x, i) + \frac{1}{2} p(x, b) \qquad (4) \end{aligned}$$

Let $\text{rank}\,(r)$ denote the sketch rank of r. $\text{rank}\,(r)$ is the sum of indicator random variables $Y(r_i, r)$, one for every object $r_i \in S$. The indicator variable $Y(r_i, r)$ for $r_i \in S$ corresponds to the event that r_i precedes r in the ordering by sketch distance. Our independence assumption implies that given a value for $d_s(q, r)$, all these variables are independent. Let $x_i = d(q, r_i)/T$. Note that $\Pr[Y(r_i, r) = 1 | d_s(q, r) = b] = P(x_i, b)$ computed in (4). The expected value and variance of $\text{rank}\,(r)$ are given by

$$\begin{aligned} \text{E}(\text{rank}\,(r)\,|d_s(q, r) = b) &= \sum_{i=1}^{N} P(x_i, b) \\ \text{Var}(\text{rank}\,(r)\,|d_s(q, r) = b) &= \sum_{i=1}^{N} P(x_i, b) - [P(x_i, b)]^2 \end{aligned}$$

When we use the feature distance distribution model, let $f(x)$ to be the probability density function for the distances, *i.e.* $\int_{x_1}^{x_2} f(x)dx$ is the fraction of points $r' \in S$ such that $d(q, r')/T \in [x_1, x_2]$. We can replace the summation over all data points with an integration over the distance distribution:

$$\text{E}(\text{rank}\,(r)\,|d_s(q, r) = b) = N \int_0^1 P(x, b) f(x) dx$$

$$\text{Var}(\text{rank}\,(r)\,|d_s(q, r) = b) = N \int_0^1 (P(x, b) - P(x, b)^2) f(x) dx$$

Given the fact that N is usually of the order of hundreds of thousands, the distribution of $\text{rank}\,(r)$ (for a specific value of $d_s(q, r)$) is approximately normal by the Central Limit Theorem. The normal distribution parameters can be determined by $\text{E}(\text{rank}\,(r)\,|d_s(q, r) = b)$ and $\text{Var}(\text{rank}\,(r)\,|d_s(q, r) = b)$. Thus the probability that $\text{rank}\,(r)$ is at most M can be expressed as

$$\Pr[\text{rank}\,(r_k) \leq M | d_s(q, r) = b] = \int_0^M f(y; \mu_b, \sigma_b) dy$$

where

$$\mu_b = \text{E}[\text{rank}\,(r)\,|d_s(q, r) = b]$$
$$\sigma_b = \sqrt{\text{Var}[\text{rank}\,(r)\,|d_s(q, r) = b]}$$
$$f(y; \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(y-\mu)^2/2\sigma^2}$$

Now, we can write the distribution of $\text{rank}\,(r)$ as a mixture of normal distributions, one for each value of $d_s(q, r)$. The distribution for b is weighed by $\Pr[d_s(q, r) = b]$. This gives us the distribution of $\text{rank}\,(r)$ and allows us to calculate the probability that $\text{rank}\,(r)$ is at most M as follows:

$$\Pr[\text{rank}\,(r) \leq M] = \sum_{b=0}^{B} \Pr[d_s(q, r) = b] \int_0^M f(y; \mu_b, \sigma_b) dy$$

We overload the notation somewhat and use $\text{rank}\,(x)$ for $x \in [0, 1]$ to denote the sketch rank of an object $r \in S$ such that $d_s(q, r)/T = x$. Note that $\Pr[d_s(q, r) = b] = p(x, b)$. Using the previous expression for $\Pr[\text{rank}\,(r) \leq M]$, we get

$$\Pr[\text{rank}\,(x) \leq M] = \sum_{b=0}^{B} p(x, b) \int_0^M f(y; \mu_b, \sigma_b) dy$$

3.4 Search Quality Estimation

Once we have an expression for the rank distribution for objects $r \in S$, for a given filter set size M, the expected fraction of the k nearest neighbors being included in the filtered set (i.e. the recall) can be computed as:

$$\text{Recall} = \frac{1}{k} \sum_{j=1}^{k} \Pr[\text{rank}\,(r_j) \leq M]$$

When the feature distance distribution is used, the recall can be calculated as:

$$\text{Recall} = \frac{N}{k} \int_0^{x_0} \Pr[\text{rank}\,(x) \leq M] f(x) dx$$

where x_0 can be derived from:

$$k = N \int_0^{x_0} f(x) dx$$

Note that the sketch distributions and rank distributions are computed based on a single query object q. As a result, the search quality estimate (recall) could vary a lot for different query points. To ensure that the results are representative of the entire data set, we use multiple representative query objects to model the distance distribution. To estimate overall search quality, we average the recall value computed using the distance distributions for each of these query objects.

4. EVALUATION

We have employed three kinds of feature-rich datasets to validate our models. To evaluate the filtering quality, we have used average recall in which the gold standard is the results computed with the original distance function.

4.1 Datasets

We have studied three kinds of data: images, audio, and 3D shapes. Table 1 provides a summary of the dataset sizes and the number of dimensions in the domain-specific feature vector representations.

Dataset	Number of Feature Vectors	Dimension
image	662,317	14
audio	54,387	192
3D shape	28,775	544

Table 1: Dataset Sizes and Dimensions.

4.1.1 Image Data

The image dataset used in our study is drawn from the Corel image collection [4] which contains about 60,000 images. The main reason to choose this dataset is that it has become a standard data set for evaluating content-based image retrieval algorithms.

We used the Region-Based Image Retrieval (RBIR) approach [19] to segment each image into multiple homogeneous regions based on color and texture. Each image region is represented by a 14-dimensional feature vector: nine dimensions for color moments and five dimensions for bounding box information. The bounding box is the minimum rectangle covering a segment and is characterized by five features: aspect ratio (width/height), bounding box size, area ratio (segment size/bounding box size), and segment centroids. We use weighted ℓ_1 distance on the 14-dimensional feature vectors to determine similarity. The images in the collection are segmented into about 660,000 regions, each of which is represented by a feature vector.

4.1.2 Audio Data

Our audio dataset is drawn from the DARPA TIMIT collection [8]. The TIMIT collection is an audio speech database that contains 6,300 English sentences spoken by 630 different speakers with a variety of regional accents. We chose this dataset also because it is available to the research community.

We break each sentence into smaller segments and extract features from each segment. For each audio segment, we use the Marsyas library [20] to extract feature vectors. We begin by using a 512-sample sliding window with variable stride to obtain 32 windows for each segment and then extract the first six MFCC parameters from each window to obtain a 192 dimensional feature vector for each segment. We use weighted ℓ_1 distance on the 192-dimensional feature vectors to determine similarity. As a result, the sentences of the dataset are into about 54,000 word segments, and extract one feature vector per word segment.

4.1.3 3D Shape Models

We have chosen to use the Princeton Shape Benchmark as our 3D shape dataset [18] and the Spherical Harmonic Descriptor (SHD) proposed by Princeton Shape Retrieval and Analysis group as the domain-specific feature [12]. The Princeton Shape Benchmark dataset is a mixture of about 29,000 3D polygonal models gathered from commercial viewpoint models, De Espona Models, Cacheforce models and from the Web. Each model is represented by a single feature vector, yielding about 29,000 feature vectors in total.

Each 3D model in the Princeton Shape Benchmark is represented by an Object File Format file with the polygonal surface geometry of the model. The models are first normalized, then placed on a $64 \times 64 \times 64$ axial grid. Thirty-two spheres of different diameters are used to decompose each model. Up to order 16 spherical harmonic coefficients are derived from the intersection of model with each of the 32 spherical shells. By concatenating all the spherical descriptors of a 3D model in a predefined order, we get a $32 \times 17 = 544$-dimensional shape descriptor for each 3D model. Although the original SHD algorithms used ℓ_2 distance as the similarity metric, we have found that ℓ_1 distance delivers similar search quality. As a result, in this study we use ℓ_1 distance on the 544-dimensional feature vector.

4.2 Evaluation Metrics and Method

We have conducted two types of experimental studies in this paper: distribution model fitting and filtering model validation.

For distribution model fitting, we use some common distribution functions to fit the distance distribution and compare the fitted distance function with the real distribution. In order to validate the fit, we compare the residuals after the least squared fitting and also plot the result for visual inspection. Moreover, we put each fitted distribution function into our model and compare their results with the result using real distance distribution to determine the best distribution function.

For filtering model validation, we compare the filtering results predicted by the model with those by an implementation of the sketch-based filtering algorithm. The filtering qualities are measured against the *gold standard* of each dataset, which are computed by a brute-force approach over the entire dataset using the original distance function. Specifically, we compare the recall value at filter ratio t predicted by our model with that computed experimentally. The recall at filter ratio t is the fraction of the k nearest neighbors to the query point that appear in the first $t \times k$ objects found by the filtering search algorithm. An ideal filtering algorithm will have a recall value of 1.0 at a filter ratio of 1. In practice, a good filter is one that achieves a satisfactory recall with a small filter ratio.

The method used in our experimental evaluation is to pick one hundred objects uniformly at random from each dataset as queries. For each query object, we use the domain-specific feature vector distance to compute the k nearest neighbors. We then fix the size of the sketch produced by the sketching algorithm and for each object in the dataset generate a sketch of that size, and use the sketches to compute the filtered candidate set of $t \times k$ objects and calculate the fraction of the k nearest neighbors appearing in this set. Since the sketching algorithm is itself randomized, we take the average recall over ten instances of the sketch algorithm. Finally, for each sketch size, we report the average recall value at a fixed filter ratio t over the one hundred randomly chosen query objects and compare that recall to the recall predicted by our model.

5. EXPERIMENTAL RESULTS

We are interested in answering the following three questions:

- What distribution model approximates the distance distributions of real datasets well?
- How well can the analytical model help system designers choose design parameters such as sketch size and filter ratio?
- How well can the analytical model predict for large dataset if its parameters are set with a small sample dataset?

This section reports the experimental answers to these questions.

5.1 Distance Distribution Model

In this section, we explore the possibility of using a simple distance distribution to model the real data distance distribution. This way, we can model the system with fewer parameters (typically two) rather than the full distance distribution. And we can also reuse the model distribution when the dataset size grows.

We have tried several common distance distributions to model the observed real distance distribution from the dataset. The distributions we have tried include: normal and lognormal distributions and polynomial functions. Since sketches can approximate the raw distance well, objects that are much further away than the k-th nearest neighbor have much less impact on the overall search quality than the ones that are closer. As a result, the distance distribution close to the k nearest neighbors are the most important in the overall result.

In our experiments, we use the first $2 \times k \times t$ data points in the full distance distribution to fit the statistical models, and then use the models to extrapolate to the full set of distances. We use nonlinear least-squares fitting to find the fit parameters. Using this method, the lognormal distribution gives us the smallest residuals among the fitting functions under consideration. As shown in Figure 2, the lognormal distribution fits the data, even when extrapolated to the full dataset.

We have also validated the choice of the distribution model by comparing the filtering result generated from the real distance distribution with that generated by the model distribution. In Figure 3, we showed the filtering result using different distribution model together with the real distance distribution. We can see that the lognormal distribution generates the closest trend to the real distance distribution.

Our results show that it is possible to model the real distance distribution with the lognormal distribution. The extrapolated lognormal distribution fits the real distribution well, and the modeling results also confirm the choice of lognormal distribution.

5.2 Choosing Filtering Parameters

The goal of the analytical model is to help systems designers choose design parameters properly. The following reports our experimental results to see how well the model can help systems designers choose sketch size B, XOR block size H, filter ratio t, and result size k.

Choosing Sketch Size B

Sketch size in bits B is perhaps the most crucial parameter in a sketch-based filtering mechanism for similarity search. Finding a good sketch size for a real system with a given expected dataset size will deliver high filter quality with minimal storage requirement.

Figure 4 shows the trend of filtering quality for different sketch sizes. The recall at a filter ratio of 10 is used to compare the experimental result with our analytical model. As expected, using more bits in a sketch leads to higher filtering quality.

Suppose we wish to build a system that achieves a recall of 0.9. Our results show that the sketch sizes must be about 160 bits, 256 bits and 128 bits for the image, audio, and 3D shape dataset respectively. The amount of storage required for sketch storage are substantially smaller than the feature vector storage requirement. We use these sketch sizes in the following experiments.

Notice that our analytical model conservatively predicts the average recall in all cases and the predicted trend is consistent with the experimental results. This implies that the model is appropriate for conservative estimates for real systems designs. We will discuss the reasons for the consistent underestimation in Section 3.3.

Choosing XOR Bits H

Choosing the right H is important, because a better H value could give better filtering quality without increasing the sketch size and thus the storage requirement. Although the best H value is data type dependent, our analytical model can help choose the value without experimenting with real data.

Figure 5 shows that our analytical model predicts similar H values to the experimental results. For the image dataset, both predicted and experimental results indicate the the best H value is 3. For audio dataset, the model predicts that the best H value is 3 and the experimental results show

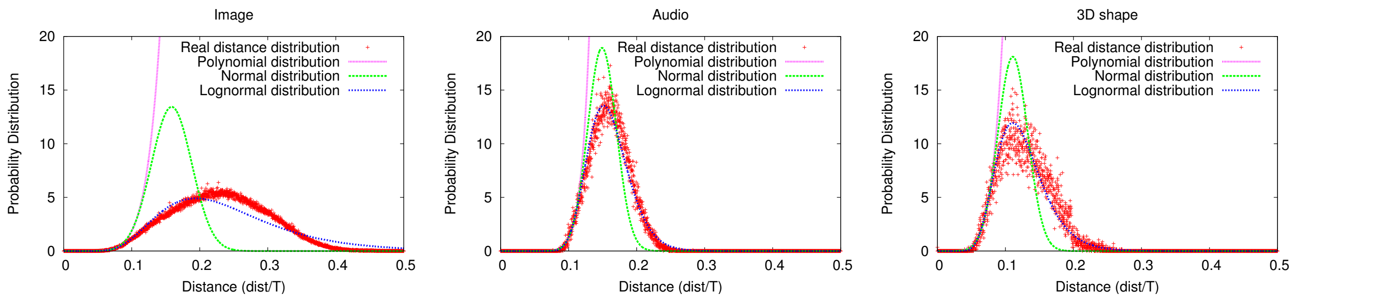

Figure 2: Compare the Real Distance Distribution with Different Distribution Models

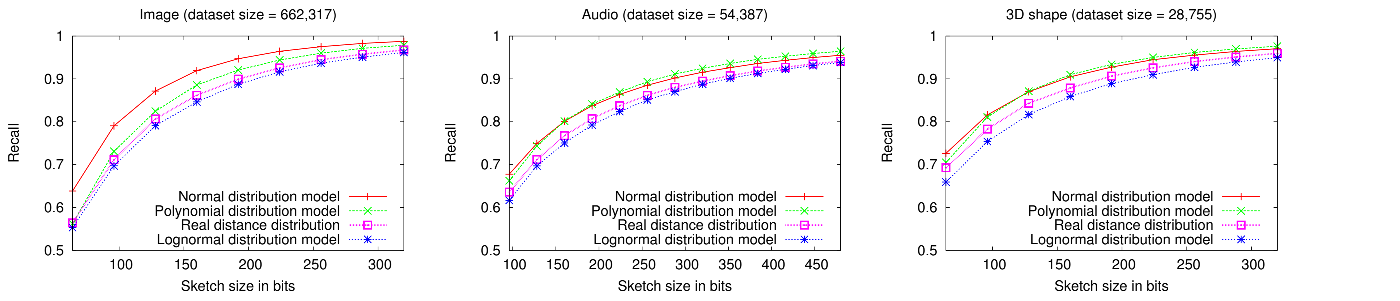

Figure 3: Filter Quality with Different Distribution Models. (H = 3, k = 100, t = 10)

that the best is 2. For 3D shape data both indicate that the best H value is 4. As before, the model predictions are consistently conservative.

Choosing Filter Ratio t

In a real system, filter ratio may have a great impact on the system performance. The smaller the filter ratio a search system needs, the fewer data objects it must rank with the original, more expensive distance function. A good filtering mechanism achieves a high recall with t as close as possible to 1.

Figure 6 shows the results of using the analytical model to estimate the filter quality using different filter ratio for $k = 100$. The results show that the analytical model predicts the filtering quality quite well. Our analytical model predicts the same or similar knee points for all datasets. The knee points are all around a filter ratio of 5, although there is a larger gap between the predicted and experimental results for the 3D shape dataset.

Choosing Result Set Size k

Since different systems may need to return different numbers of results for each query, it is important to understand how recall varies as a function of the result set size k given a fixed filter ratio t. In real systems, the result set sizes typically depend on the user interface. For example, an image search system may return 20 thumbnails per page, whereas an audio search system might return 10 results per page.

Figure 7 shows the average recall results predicted by our analytical model and those of experimental results with $t = 10$. The results show that the model predicts the filtering quality conservatively, while the trend by the model is consistent with the experimental results. Since the gap between the experimental result and model remains almost constant, the model can help the system designer to estimate the loss of recall when smaller result set size is used. In both cases, the filtering qualities are not very sensitive to different result set sizes for a fixed filter ratio.

5.3 Extrapolating to Larger Dataset Size

When building a real system, it is common not to have the full dataset available at the initial deployment. It is important to be able to choose system parameters with only a small sample dataset, and have some performance and quality guarantees as the dataset size grows. Our analytical model is useful in this scenario since the system designer cannot conduct full-scale experiments to figure out the parameters to be used.

In order to validate our model's prediction, we conducted an experiment that simulates dataset growth. For each dataset, we used a small subset of the full dataset to configure our model: that is, we only use one tenth of the total data objects to model the distance distributions and then use the model parameters derived from small dataset to predict the filter quality when dataset size grows. The result is compared with the experimental results, where more and more data points in the dataset are included in each experiment to simulate the growing dataset.

Figure 8 shows the filtering quality with different dataset sizes. In each plot, the first data point corresponds to the small sample dataset that we use to derive our model parameters; the following data points labeled as “lognormal distribution model” are the projected results using the model. The experimental results are also shown in the same plot.

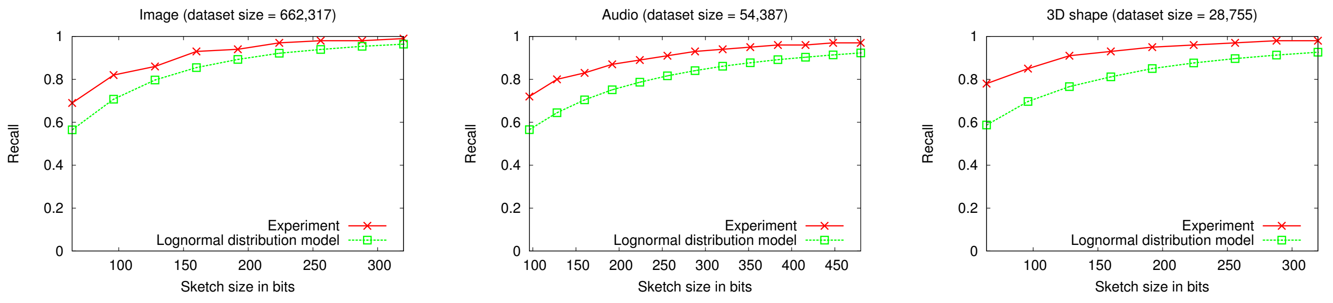

Figure 4: Filter Quality vs Sketch Size. (H = 3, k = 100, t = 10)

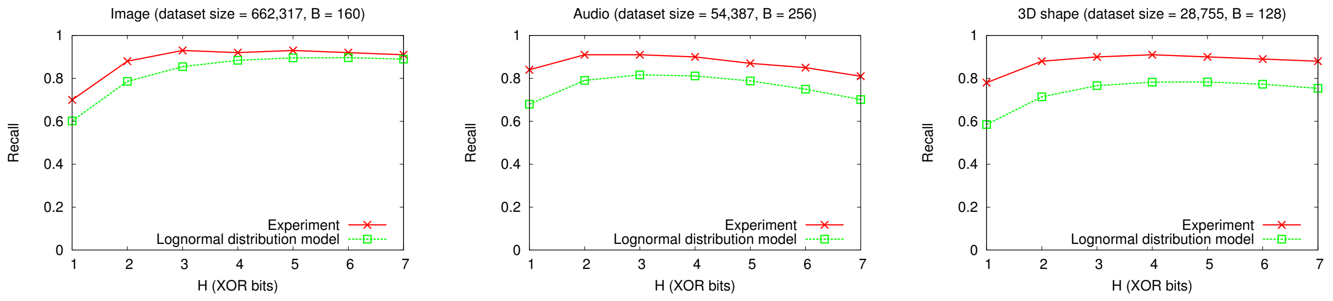

Figure 5: Filter Quality vs XOR Block Size H. (k = 100, t = 10)

The results show that the filtering quality degrades gradually as the dataset grows larger. The model can give a good prediction on the degree of quality degradation as the dataset size grows. The prediction works better when the sample dataset size is reasonably large as seen in the image dataset. For other datasets, the degradation prediction is more conservative, but conservative estimates are more acceptable than optimistic in real systems designs.

5.4 Discussion

For all the figures showing the recall value, we noticed a consistent underestimate of the model result compared with the experimental result. In fact the underestimate of the model is largely due to the simplified independence assumption of the model – *i.e.* the assumption the sketch distances of objects are independent of the k-th nearest neighbor's sketch distance.

In section 3.3, we assumed that the sketch distances for different objects $r \in S$ are independent This assumption simplifies the model, but in reality, there is some dependency that the model ignores. In fact, when r's sketch distance is large, the other sketch distances are also likely to be large and vice versa. In other words, there is a small positive correlation between sketch distances. In order to understand how such a correlation arises, it is instructive to consider the 1-dimensional case. Consider objects r and r' such that $d(q, r) \leq d(q, r')$. The algorithm to generate the bit vector sketch picks random thresholds for a dimension and checks if the coordinate in that dimension is above or below the threshold. The bit thus generated for q and r is different if the random threshold separates q and r. If this happens, it also likely to separate q and r'. If the sketch distance of r is large, then q and r must have been separated by several such randomly picked thresholds. But then it is likely that q and r' are also separated by these thresholds. Thus, the sketch distance of r' is likely to be large.

The positive correlation between sketch distances results in the rank of r_k being lower than that predicted by the independent model. In order to understand this, consider the extreme situation where the positive correlation is of the following form: for i randomly chosen in $[0, 100]$, each sketch distance is equal to the value at the i-th percentile of its individual distribution. In this case, objects with higher feature distance than the kth nearest neighbor r_k will never have their sketch distance lower than the sketch distance of r_k. The effect of positive correlation is similar, but less extreme than the situation described above. In other words, the positive correlation lowers the probability that objects further than the kth nearest neighbor will have their sketch distance lower than the sketch distance of r_k.

We conducted an experiment with the real dataset which clearly demonstrates such a dependence. In the experiment, we repeated the sketch construction 100,000 times and observed the relationship between sketch distance s of a particular data point r and the sketch distance s_k of the kth nearest neighbor r_k. Figure 9 shows the result. The points show the average value of s when s_k takes different values and the dashed line shows the constant s value expected with independent model. We can see that there is a small positive correlation between r's sketch distance s and r_k's sketch distance s_k. Figure 10 further shows the experimental result where the (empirically observed) probability distributions of r's sketch distance is plotted for two different values

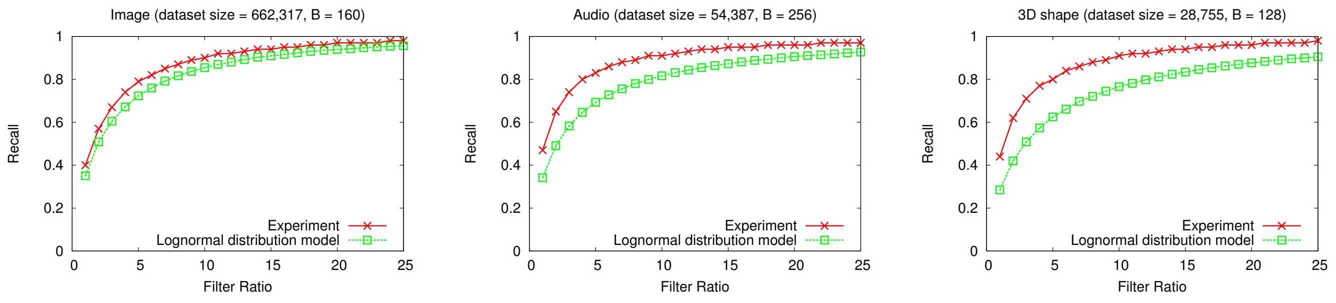

Figure 6: Filter Quality vs Filter Ratio t. (H = 3, k = 100)

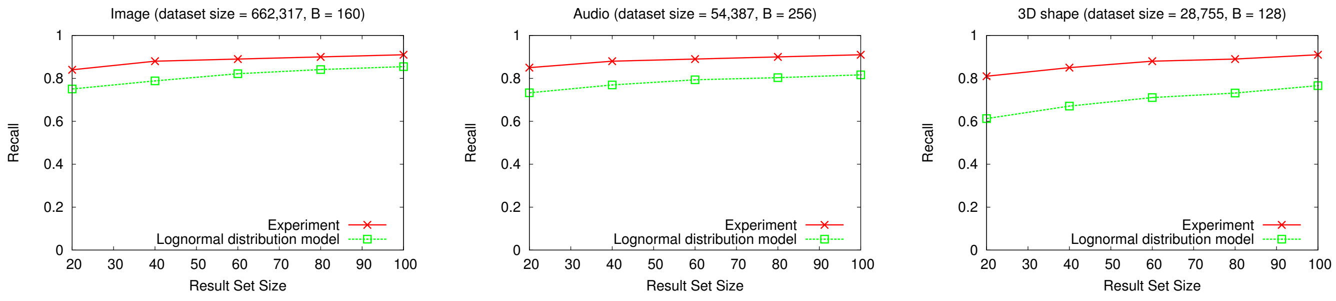

Figure 7: Filter Quality vs Result Set Size k. (H = 3, t = 10)

of s_k.

This data dependence affects the expected probability of r overtaking r_k. The experimental result shows that the probability of that particular data point r overtaking r_k is 0.125 while our independent model's prediction is 0.167 according to Equation 4. The higher rank prediction of r_k's sketch distance of our model will generate lower recall value in the quality score at filter ratio t and cause a consistent underestimate to the experimental results.

In order to accurately model the dependence of data object r's sketch distance r on r_k's sketch distance r_k, much more information about the data set is needed: value distributions on each dimension, data value dependence between different objects on each dimensions, *etc.* While this might give more accurate predictions, it is much harder to obtain reliable estimates of such fine grained information about the data set. Also it is unclear how well a model that incorporates such detailed information can be extrapolated to larger data set sizes. We have decided to adopt a simpler model in this paper that captures the essence of the experiment. Although our model gives a consistently low estimate of recall, it matches the general trend of the experimental results very well.

6. RELATED WORK

Similarity search is typically formulated as a k-nearest neighbor search problem. The exact form of this problem suffers from the "curse of dimensionality" – either the search time or the search space is exponential in dimension d [6, 16]. As a result, researchers have instead focused on finding approximate nearest neighbors whose distances from the query point are at most $1 + \epsilon$ times the exact nearest neighbor's distance. Our filtering scheme for similarity search can be seen as an approach to approximate nearest-neighbor search. For a general overview of dimension reduction techniques, see Imola Fodor's survey [7].

Our filtering scheme for similarity search draws on a number of recent advances in the theory community in the construction of compact data structures ("sketches") and in general dimension reduction. It is often possible to produce sketches such that a particular distance function on the original data may be quickly estimated from the corresponding sketches with provable bounds on the error. The classical Johnson-Lindenstrauss lemma [11] shows that any set of n points in ℓ_2 can be mapped down to $O(\log n)/\epsilon^2$ dimensions via random projections such that all distances are preserved within a factor of $1 + \epsilon$. This can be viewed as a sketch construction for the ℓ_2 norm and has several applications. The original proof of Johnson and Lindenstrauss was subsequently simplified by a sequence of later papers. Early work on sketches includes the min-wise independent permutation sketches for filtering near-duplicate documents that Broder *et al.* developed for the AltaVista search engine [3, 2]. Subsequent work by Indyk and Motwani [10, 9] introduced the notion of locality-sensitive hash functions selected so that the collision probability is higher for pairs of objects that are closer in some suitable sense. Such families are useful for constructing compact data structures for nearest-neighbor search. The LSH functions described by Indyk and Motwani are for the binary Hamming space. Kushilevitz, Ostrovsky and Rabani [14] developed a hashing scheme to distinguish between pairs of objects with ℓ_1 distances above and below a given threshold – the sketch construction of

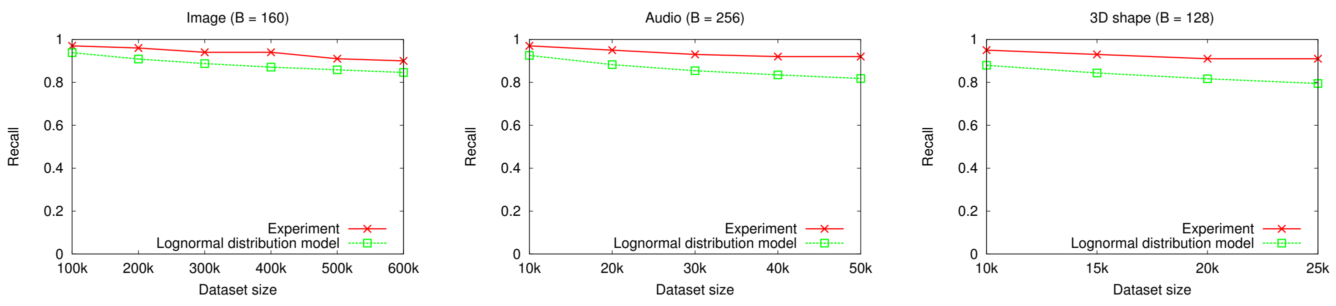

Figure 8: Filter Quality vs Dataset Size (H = 3, k = 100, t = 10)

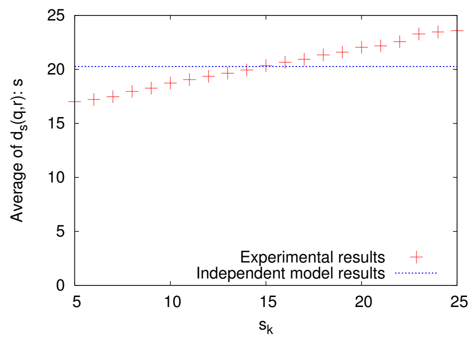

Figure 9: Dependence of s to s_k

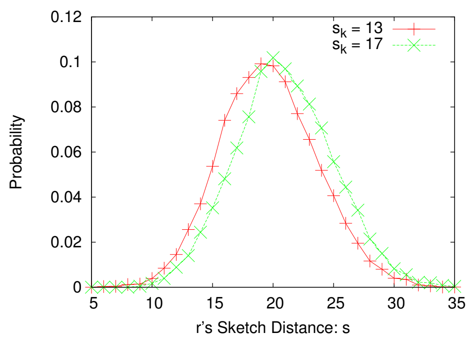

Figure 10: Probability Distribution of s with Different s_k

[15] adapts some of their ideas. Datar *et al.* [5] have also described a locality-sensitive hashing scheme for the ℓ_p norms based upon p-stable distributions. Sketching techniques also play an integral role in streaming algorithms, *i.e.* one-pass algorithms for very large data sets that store a very small amount of information. A survey of the work in this area is outside the scope of this paper. We refer the reader to the survey by Muthukrishnan [17]. The notions of doubling dimension and intrinsic dimensionality (see [13, 1]) have been used previously to capture the inherent complexity of data sets from the point of view of several algorithmic problems including nearest neighbor search. However these notions do not provide a fine-grained model for distance distributions and do not have enough information to accurately estimate the performance of filtering algorithms for nearest neighbor search.

7. CONCLUSIONS

This paper reports the results of modeling the parameters of using sketches to filter data for similarity search. The goal of our study is to help systems designers choose key design parameters such as sketch size and filtering result size. We validated our model with three feature-rich datasets including images, audio recordings, and 3D shape models. Our study shows three main results for sketches that use Hamming distance to approximate ℓ_1 norm distance:

- Lognormal distribution models the ℓ_1 distance distributions of all our datasets (images, audio and 3D shape data) quite well and it fits much better than other known models such as exponential, normal and polynomial distributions. This result allows us to use lognormal distribution to model the distance distribution of a specific dataset.
- We have proposed a rank-based filtering model for the sketch construction to use Hamming distance to approximate ℓ_1 distance. We have shown, by experimenting with image, audio, and 3D shape datasets, that this model can conservatively predict the required sketch size for required recall, given the dataset size and its filtering candidate set size.
- Using the lognormal distribution with its parameters derived from a small sample dataset, we show that the rank-based filtering model can be used to perform good predictions of sketch sizes for a large dataset.

Our experimental studies show that the rank-based filtering model predicts results close to experimental results. Although there are noticeable gaps between the predicted and experimental results for certain systems parameters, the predicted trends are consistent with the experimental results. Furthermore, the predictions from the model are consistently conservative in all cases.

8. REFERENCES

[1] A. Beygelzimer, S. Kakade, and J. Langford. Cover trees for nearest neighbor. http://hunch.net/~jl/projects/-cover_tree/icalp_3/icalp_1.ps.

[2] A. Broder, M. Charikar, A. Frieze, and M. Mitzenmacher. Min-wise independent

permutations. *Journal of Computer Systems and Sciences*, 60(3):630–659, 2000.

[3] A. Z. Broder, S. C. Glassman, M. S. Manasse, and G. Zweig. Syntactic clustering of the web. In *Proc. of the Sixth Int. World Wide Web Conf.*, pages 391–404, 1997.

[4] http://www.fotosearch.com/corel.

[5] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the 20th annual symposium on Computational geometry(SCG)*, pages 253–262, 2004.

[6] D. Dobkin and R. Lipton. Multidimensional search problems. *SIAM J. Computing*, 5:181–186, 1976.

[7] I. K. Fodor. A survey of dimension reduction techniques. Technical Report UCRL-ID-148494, Lawrence Livermore National Laboratory, 2002.

[8] J. S. Garofolo, L. F. Lamel, W. M. Fisher, J. G. Fiscus, D. S. Pallett, and N. L. Dahlgren. DARPA TIMIT acoustic-phonetic continuous speech corpus, 1993.

[9] A. Gionis, P. Indyk, and R. Motwani. Similarity search in high dimensions via hashing. In *Proc. of the 25th Int. Conf. on Very Large Databases*, pages 518–529, 1999.

[10] P. Indyk and R. Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *Proc. of the 30th Annual ACM Symposium on Theory of Computing*, pages 604–613, 1998.

[11] W. B. Johnson and J. Lindenstrauss. Extensions of lipschitz mapping into hilbert space. *Contemporary Mathematics*, 26:189–206, 1984.

[12] M. Kazhdan, T. Funkhouser, and S. Rusinkiewicz. Rotation invariant spherical harmonic representation of 3D shape descriptors. In *Proc. of the Eurographics Symposium on Geometry Processing*, 2003.

[13] R. Krauthgamer and J. R. Lee. Navigating nets: Simple algorithms for proximity search. In *Proc. of the 15th ACM Symposium on Discrete Algorithms*, pages 798–807, 2004.

[14] E. Kushilevitz, R. Ostrovsky, and Y. Rabani. Efficient search for approximate nearest neighbor in high dimensional spaces. *SIAM Journal of Computing*, 30(2):457–474, 2000.

[15] Q. Lv, M. Charikar, and K. Li. Image similarity search with compact data structures. In *Proc. of the 13th ACM Conf. on Information and Knowledge Management*, pages 208–217, 2004.

[16] S. Meiser. Point location in arrangements of hyperplanes. *Information and Computation*, 106(2):286–303, 1993.

[17] S. Muthukrishnan. Data streams: Algorithms and applications. *Foundations and Trends in Theoretical Computer Science*, 1(2), 2005.

[18] P. Shilane, M. Kazhdan, P. Min, and T. Funkhouser. The Princeton shape benchmark. In *Proc. of the Conf. on Shape Modelling and Applications*, 2004.

[19] A. W. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain. Content-based image retrieval at the end of the early years. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(12):1349–1380, 2000.

[20] G. Tzanetakis and P. Cook. *MARSYAS: A Framework for Audio Analysis*. Cambridge University Press, 2000.