Efficient Encodings for Document Ranking Vectors

Taher H. Haveliwala[*]
Stanford University

ABSTRACT

The rapid growth of the Web has led to the development of many techniques for enhancing search rankings by using precomputed numeric document attributes such as the estimated *popularity* or *importance* of Web pages. For efficient keyword-search query processing over large document repositories, it is vital that these auxiliary attribute vectors, containing numeric per-document properties, be kept in main memory. When only a small number of attribute vectors are used by the system (e.g., a document-length vector for implementing the cosine ranking scheme), a standard 4-byte, single-precision floating point representation for the numeric values suffices. However, for richer search rankings, which incorporate additional numeric attributes (e.g., a set of page-importance estimates for each page), it becomes more difficult to maintain all of the auxiliary ranking vectors in main memory. We propose *lossy* encoding schemes based on scalar quantization that efficiently encode auxiliary numeric properties, such as PageRank, an estimate of page importance used by the Google search engine. Unlike standard scalar quantization algorithms, which concentrate on minimizing the *numerical* distortion caused by lossy encodings, we seek to minimize the distortion of search-result rankings.

1. INTRODUCTION

Modern Web search engines incorporate a variety of numerical Web-page attributes in their search ranking functions in an attempt to bring order to the ever-growing Web. Given the massive repositories that Web search engines must index, with large numbers of concurrent users issuing queries to the system, developing memory-efficient encodings for these numerical attributes, so that they can be cached in main memory, is an increasingly important challenge.

An overview of a scalable keyword-search system helps make clear why per-document attributes, such as page popularity, must be maintained in main memory. As depicted in Figure 1, a typical Web search system utilizes an inverted text index $\mathcal{I}$ and a set of auxiliary ranking vectors $\{\vec{R_i}\}$. For concreteness, consider a system with only one such vector, $\vec{R_p}$, containing per-document popularity estimates. The index $\mathcal{I}$ contains information about the occurrences of terms in documents and is used to retrieve the set of document IDs for documents satisfying some query $\mathcal{Q}$. The index $\vec{R_p}$ is then consulted to retrieve the overall popularity score for each of these candidate documents. Using the information retrieved from $\mathcal{I}$ and $\vec{R_p}$, a composite document score is generated for each candidate result, yielding a final ranked listing.

The inverted index $\mathcal{I}$ is constructed offline and provides the mapping $\{t \rightarrow f_{dt}\}$, where f_{dt} describes the occurrence of term t in document d. In the simplest case, f_{dt} could be the within-document frequency of t. The number of random accesses to $\mathcal{I}$ needed to retrieve the necessary information for answering a query $\mathcal{Q}$ exactly equals the number of terms in the query, $|\mathcal{Q}|$. Because queries are typically small, consisting of only a few words, it is practical to keep the index $\mathcal{I}$ on-disk and perform $|\mathcal{Q}|$ seeks for answering each query.

The auxiliary index $\vec{R_p}$ is also constructed offline, and provides the mapping $\{d \rightarrow r_d\}$, where r_d is the popularity of document d according to some computed notion of popularity. Note that in contrast to $\mathcal{I}$, the index $\vec{R_p}$ provides *per-document* information. The search system typically must access $\vec{R_p}$ once for *each* candidate document of the result set, which could potentially be very large. These random accesses would be prohibitively expensive, unless $\vec{R_p}$ can be kept entirely in main memory. Whereas the query length is the upper bound for the accesses to $\mathcal{I}$, the number of candidate results retrieved from $\mathcal{I}$ is the upper bound for accesses to $\vec{R_p}$. One way to reduce the number of random accesses required is to store the attribute values of $\vec{R_p}$ in $\mathcal{I}$ instead; e.g., create an index $\mathcal{I}'$ that provides the mapping $\{t \rightarrow \{f_{dt}, r_d\}\}$. However, this requires replicating the value r_d once for each distinct term that appears in d, generally an unacceptable overhead especially if more than one numeric property is used.

Much work has been done on compressing $\mathcal{I}$, although comparatively less attention has been paid to effective ways of compressing auxiliary numeric ranking vectors such as $\vec{R_p}$. The typical keyword search system has only one such auxiliary ranking vector, $\vec{R_l}$ — the document lengths needed in computing the query-document cosine similarity [19] — and can be kept in main memory without much difficulty. However, for richer ranking schemes, such as PageRank and topic-sensitive PageRank [12, 6, 17, 8], which require consulting a *set* of auxiliary ranking vectors, much more consideration needs to be given to the encodings used for the attribute values.

Note that falling main memory prices do not alleviate the need for efficient encodings; increasingly affordable disk storage is leading to rapidly growing Web-crawl repositories, in turn leading to larger sets of documents that need to be indexed. Utilizing a rich set of per-document numeric ranking attributes for growing crawl repositories and growing

[*]Supported by NSF Grant IIS-0085896.

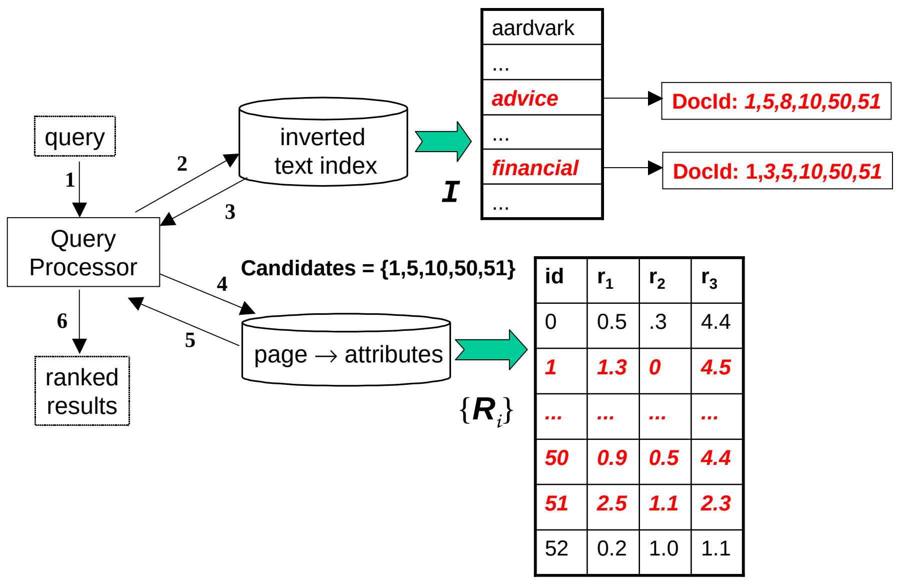

Figure 1: A simplified illustration of a search engine with a standard inverted text-index and 3 auxiliary numerical attributes for each document. Note that the number of random accesses to $\mathcal{I}$ is typically small, whereas the number of accesses to $\{\vec{R}_i\}$ is large. Our goal is to minimize the space needed for the data structure $\{\vec{R}_i\}$.

numbers of users thus continues to require efficient encoding schemes.

In this paper, we concentrate on developing efficient encodings for PageRank vectors, which are importance estimates for Web pages computed using the hyperlink graph of the Web [12, 6, 5, 17, 8]. In Section 2, we briefly review scalar quantization, which provides the framework for our approach, and introduce new distortion criteria for measuring quantizer performance. In Section 3, we discuss fixed-length encoding schemes and analyze their performance using a traditional numerical distortion measure. In Section 4, we investigate in detail distortion measures more appropriate for the case of search ranking functions, and analyze the performance of various quantization strategies both analytically and empirically. In Section 5, we extend our approach to consider variable-length encoding schemes and discuss their performance. We conclude with a review related work in Section 6.

2. SCALAR QUANTIZATION

2.1 Quantization Rules

An excellent introduction to quantization can be found in [3]; we give only a brief review here. Let $\mathcal{C} \subset \mathbb{R}$; for our work, assume $\mathcal{C}$ is finite. A quantizer is a function $q(x) : \mathbb{R} \to \mathcal{C}$ that partitions $\mathbb{R}$ into a set $\mathcal{S}$ of intervals and maps values in the same interval to some common *reproduction value* in $\mathcal{C}$. In other words, $q(x)$ maps real values to some approximation of the value. Let $n = |\mathcal{C}|$. As the values in $\mathcal{C}$ can be indexed from 0 to $n-1$, one way to compactly represent values in the range of $q(x)$ is with fixed-length codes of length $\lceil log_2 n \rceil$ bits, in conjunction with a codebook mapping the fixed-length codes to the corresponding reproduction value. Let $\hat{x} = q(x)$. Given the sequence $\{a_i\}$ of real numbers as input, a compression algorithm based on fixed-length scalar quantization would output the sequence of l-bit codewords $\hat{a}_i$, along with the codebook mapping each *distinct* codeword to its corresponding reproduction value.

The error that results from quantizing a particular input value x to the reproduction value $\hat{x}$ is typically quantified by a *distortion* measure. We consider distortion measures in Section 2.2.

The simplest fixed-length encoding simply partitions the domain of possible values into n cells of uniform width using a uniform quantizer u_n, where n is typically chosen to be a power of 2. A more complex quantizer could use a nonuniform partition to lower the distortion. Alternatively, instead of using nonuniform partitions, the input values can be transformed with a nonlinear function $G(x)$, called a compressor, then uniformly quantized using u_n. The inverse function $G^{-1}(x)$ can be used for reconstructing an approximation to the original value. Such a quantizer $G^{-1}(u_n(G(x)))$ is called a *compander*[1]; it is known that any fixed-length, nonuniform quantizer can be implemented by an equivalent compander [3]. For simplicity, unless otherwise noted, we will define quantization strategies as companders.

Quantizers can also make use of variable-length codes for the elements in set $\mathcal{C}$. If shorter codewords are assigned to the elements in $\mathcal{C}$ that more frequently correspond to values in the input data being compressed, the *average* codeword length can be reduced. The simplest scheme would use a Huffman code for $\mathcal{C}$, using the known or estimated frequency of the elements in $\mathcal{C}$ to generate the optimal Huffman codes.

[1]Short for *compressor, expander*. Note that companders save on the need for explicit codebooks, as the partitioning of the domain is uniform. $G^{-1}(x)$ takes the place of the codebook.

2.2 Measuring Distortion

The scalar quantization literature in general considers the loss in numerical precision when comparing the expected distortion of quantization schemes. For instance, the most commonly used measure of distortion for a value is the squared error:

$$d(x, q(x)) = (x - q(x))^2 \tag{1}$$

The inaccuracy of a particular quantization function q for a particular set of input data is then the mean distortion, denoted $D(q)$. If $d(x, q(x))$ is the squared error as defined above, then $D(q)$ is referred to as the *mean squared error*, or MSE.

However, in the case of document ranking, the numerical error of the quantized attribute values themselves are not as important as the effect of quantization on the *rank order* induced by these attributes over the results to a search query. In our work, we show that distortion measures based on induced rankings of search-query results lead to different choices for optimality.

Assume each document in the corpus has k associated numerical ranking attributes. Note that some of these attributes, such as PageRank, are precomputed and stored, and some are query-specific and hence generated at query-time. As the goal of quantization is to reduce space requirements of the precomputed indexes, it is used only on the precomputed attributes. The attributes can be used in one of two ways to rank a set of documents that are candidate results to some search query.[2] The scenarios we consider are:

1. Each of the k attributes can be used separately to rank the candidate result documents to generate k intermediate rank orders, which are then *aggregated* to generate a final rank order over the candidates [1].

2. The values for the k attributes for the documents can be combined numerically to form a composite score, which is then used to rank the set of candidate documents.

Under Scenario 1, quantization has a very simple effect on the intermediate rank orders. Quantization can map similar values to the same cell, but can never swap the relative order of two values; for any two values x and y, and any quantizer q, we know that $x < y \Rightarrow q(x) \leq q(y)$. Thus, an intermediate rank order using a quantized version of an attribute differs from the intermediate rank order using the original attribute only through the introduction of false ties.[3] This property suggests the following distortion measure. Let the distortion of a quantizer on a particular attribute, for a particular candidate result set of size m, be measured as the sum of squares of the number of candidate documents mapped to the same cell, normalized so that the maximum distortion is 1. Assuming the original values for the attribute were distinct, this distortion is closely related to the fraction of document pairs in the result set that are falsely tied. More formally, let $\mathcal{R}$ be the query-result set, with $m = |\mathcal{R}|$, and let X_i be the number of documents *in* $\mathcal{R}$ mapped to cell i for the attribute under consideration. The distortion of an n-cell quantizer on the set $\mathcal{R}$ is given by:

$$Distortion(q_j, \mathcal{R}) = \frac{1}{m^2} \sum_{0}^{n-1} X_i^2 \tag{2}$$

We refer to this distortion measure as TDist. We can evaluate the relative performance of different quantizers based on the expectation (or average over some test query set) of the above distortion. Different search models for computing this expectation are given in Section 4. The empirical performance of different quantizers over a test set of queries on this distortion measure are presented in Section 4.5.

Under Scenario 2, we cannot use the above distortion measure. The error can no longer be measured solely through artificial ties. In the final rankings induced by the composite score, the relative ordering of documents can be different. We proceed by defining a more suitable distortion measure, KDist, based on techniques for comparing rank orders described in [1]. Consider two partially ordered lists of URLs, τ and τ_q, each of length m, corresponding to rankings induced by exact and approximate composite scores, resp. Let U be the union of the URLs in τ and τ_q. If δ is $U - \tau$, then let τ' be the extension of τ, where τ' contains δ appearing after all the URLs in τ.[4] We extend τ_q analogously to yield τ'_q. We define our distortion measure KDist as follows:

$$\text{KDist}(\tau, \tau_q) = \frac{|\{(u,v) : \tau', \tau'_q \text{ disagree on order of } (u,v), u \neq v\}|}{(|U|)(|U|-1|)} \tag{3}$$

In other words, $\text{KDist}(\tau, \tau_q)$ is the probability that τ and τ_q disagree on the relative ordering of a randomly selected pair of distinct nodes $(u, v) \in U \times U$. We present the empirical performance of different quantizers on the above distortion measure in Section 4.5.

3. FIXED-LENGTH ENCODING SCHEMES

In this section, we discuss fixed-length encoding schemes, describe the optimal encoding under the MSE distortion measure, and give the empirical MSE-performance of various fixed-length encoding schemes. The "best" fixed-length quantizer q can be chosen by answering the following three questions:

1. What is the appropriate measure of distortion $D(q)$ for the application?

2. How many cells should the partition have? In other words, what is the appropriate choice for n, noting that (a) the codeword length is given by $\lceil log_2 n \rceil$, and (b) smaller n will lead to higher distortion.

3. For a particular n, what compressor function $G(x)$ should be used to minimize distortion?

Answering Question 2 is simply a matter of choosing a codeword length that will allow the encoded ranking vector to fit in the available memory. If we answer Question 1 by choosing the MSE, then results from the quantization literature ([14, 3]), let us choose the optimal compressor function

[2]E.g., the candidate result set might consist of documents that contain all of the query terms.

[3]The final rankings, after rank aggregation, may of course differ in more complex ways, depending on how the aggregation is done. If in such a case, measuring the distortion of final rankings is desired, the distortion measure of Scenario 2 is more appropriate.

[4]The URLs *within* δ are not ordered with respect to one another.

based on the distribution of the input values, leading to the answer for Question 3. We begin with this case (i.e., where $D(q)$ is the MSE), and then discuss the use of more appropriate distortion measures in Section 4.

The optimal compressor function $G(x)$ (i.e., the $G(x)$ which will minimize the quantization MSE) is determined by the probability density function (pdf) of the input data, $p(y)$. In particular, the optimal compressor is given by the following [14]:

$$G(x) = c \cdot \int_{-\infty}^{x} p(y)^{1/3} dy \qquad (4)$$

Fortunately, the entire ranking vector that we would like to encode is available, so we can determine $p(y)$ easily. In Section 3.1, we look at the distribution of values of PageRank vectors in our dataset, and in Section 3.2, we discuss the MSE performance of 6 fixed-length coding schemes.

3.1 Data Distribution

To compute the right hand side of Equation 4, we examine the relative frequency distribution $p(y)$ of the values of the PageRank ranking vectors that we want to encode. PageRank is an iterative computation performed over the link graph of the Web that assigns an estimate of *importance* to every page on the Web [12]. A topic-sensitive extension to PageRank [6] modifies the link-graph computations to assign an estimate of importance, *with respect to particular topics*, to every page on the Web.

We used as our dataset the Stanford WebBase [7] crawl repository of 120 million pages, containing a total of 360 million distinct URLs. This latter count includes pages that were linked to from crawled pages, but were not themselves crawled. Note that using a standard 4-byte floating point representation, the PageRank vector for these 360 million pages requires 1.34GB of space. Figure 2 shows the distribution of the standard PageRank values for this dataset on a log-log plot. In plotting the distribution, we used logarithmic binning, with the counts normalized by the bin width. In other words, when computing the relative counts shown on the y-axis, we used bins of equal width on a logarithmic scale, and divided the counts by the actual width of the bin. This technique was needed, as the data is very sparse for high rank values.[5] The distribution appears to follow a power-law with exponent close to 2.17, similar to the findings of Pandurangan et al. [13] on a different dataset. They also provide a graph-generation model that explains this observed property of the Web.

The topic-specific rank vectors constructed following the methodology proposed in [6] behave similarly. For instance, the values for the PageRank vector generated with respect to the COMPUTERS topic follow the distribution shown in Figure 3; the other topic-specific PageRank scores we measured also follow a similar distribution, but are not shown here. Note that the power-law fit is not quite as close, with the slope steepening noticeably in the tail. The best-fit power-law exponents for the topic-specific PageRank distributions all ranged between 1.7 and 1.8. Because developing efficient encodings for the topic-specific rank vectors is analogous to developing encodings for the standard PageRank vector, the remaining discussion focuses solely on encodings for the standard PageRank vector.

[5]A similar approach is used in [15].

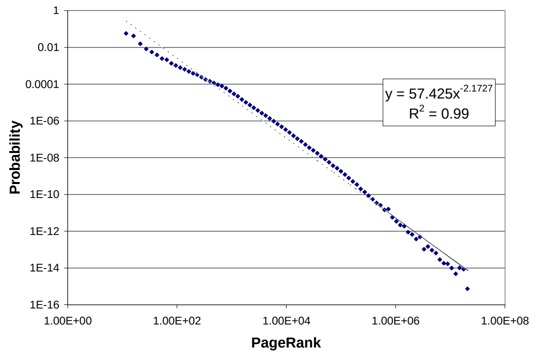

Figure 2: PageRank distribution on a log-log scale. The crawl graph included 360M URLs generated from a 120M page repository. The best-fit power-law curve is shown, with a slope close to -2.1, in agreement with prior findings.

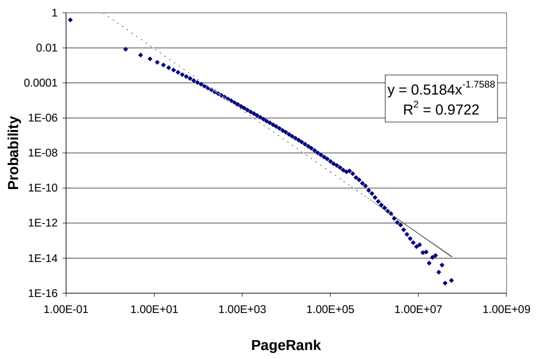

Figure 3: COMPUTERS-biased PageRank distribution on a log-log scale. The slope is close to -1.8, less steep than the ordinary PageRank distribution.

When computing the optimal compressor function for the standard PageRank vector, for minimizing the MSE, we compute Equation 4 with the pdf $p(y) = k \cdot y^{-2.17}, y > y_{min}$.

3.2 MSE Performance of Fixed-Length Schemes

In this section, we compare the performance of various fixed-length encoding schemes using the mean-squared-error (MSE) measure. Except for the **equal_depth** strategy, the quantizers are implemented as companders. A summary of the strategies we consider is given in Table 1. To illustrate the behavior of the quantization strategies on the PageRank values for the 360 million URLs, Figure 4[6] shows the relative number of input points mapped to each cell (i.e., the depth of each cell) for 6 different strategies when using 256 cells. The y-axis is a log scale, to facilitate comparison. The figure depicts how the various compressor functions transform the input data, whose distribution was earlier shown in Figure 2. Because of PageRank's power-law distribution, we see that for the **linear** (i.e., uniform-width) strategy most cells are

[6]For clarity, in all of the graphs that follow, the order of the entries in the graph legend reflects the relative position of the corresponding curves in the graph. Curve markers are consistent across the graphs as far as possible.

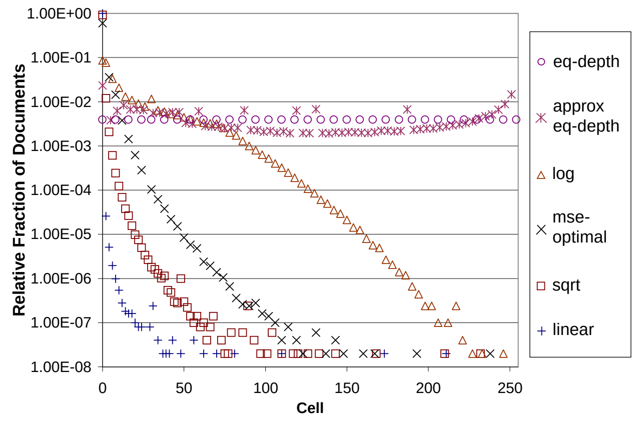

Figure 4: Relative cell counts for the various strategies for 8-bit codes (i.e., 256 cells). The y-axis gives the fraction of the values in the input mapped to each cell.

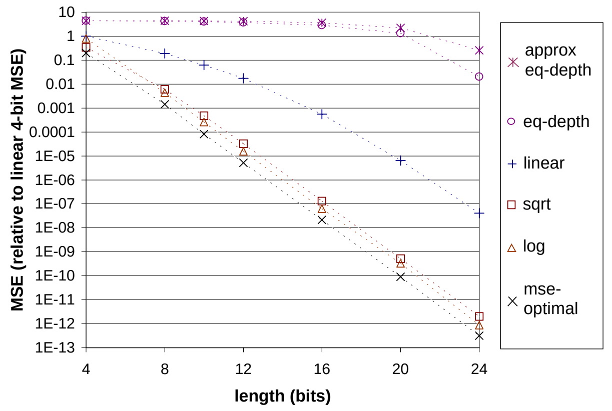

Figure 5: The MSE of 6 different strategies, for codeword lengths varying from 4 to 24. The MSE axis is on a log scale, and is normalized so that the MSE of the 4-bit linear compander is 1.

empty. The other 4 strategies use nonuniform partitions to divide up the range of possible PageRank values.

To compare the strategies in the traditional numeric fashion, we computed the MSE for each strategy for encoding the standard PageRank vector. We varied the number of cells used from 2^4 to 2^{24}; i.e., the number of bits necessary for a fixed-length code varied from 4 to 24 bits per value. Figure 5 graphs the MSE vs. code-length for each of the strategies. We see that **mse_optimal** performs the best, as expected, with **log** and **sqrt** not far behind. We will see in the following section, however, that if we use a distortion measure based on the induced rankings of query results, rather than the MSE, the choice of optimal strategy differs.

4. OPTIMIZING FIXED-LENGTH ENCODINGS FOR RANK-BASED DISTORTION MEASURES

We now discuss how we to choose the optimal quantization rule in the context of search ranking, under various distortion measures and probabilistic models for the keyword-search task. In general, unless both the search model and distortion measure are fairly simple, analytically deriving the optimal quantization rule becomes complex. We derive optimal quantization rules for simple cases, and rely solely on experimental data for more complex cases.

An outline of this section follows. In Section 4.1, we introduce a simplified model of the keyword search process. We analytically derive the optimal quantization strategy for this model in Section 4.2, and then extend the derivation to richer models of search. Section 4.3 describes a technique to approximate the optimal strategy using a simple compander. In Section 4.4, we present empirical results describing the distribution of our data that provides justification for our simplified models. Section 4.5 presents experimental results illustrating the performance of the quantization strategies under various ranking models and corresponding distortion measures.

4.1 Retrieval and Ranking Model

We begin with a greatly simplified model of keyword search to allow for the analysis of the effects of quantization on query-result rankings, and later discuss extensions.

The first part of the model describes the retrieval of the candidate documents for a query. Let $\mathcal{D}$ be the set of documents in the Web-crawl repository. **Retrieve**$(\mathcal{D}, \mathcal{Q})$ is defined as the operation that returns the set $\mathcal{R} \subset \mathcal{D}$ consisting of documents that satisfy the query $\mathcal{Q}$.[7] For simplicity, we model the operation **Retrieve**$(\mathcal{D}, \mathcal{Q})$ as generating a random sample of size M from $\mathcal{D}$, with each element of $\mathcal{D}$ having an equal probability of appearing in the result set.

The second part of the model describes the ranking of the documents. Consider a single auxiliary ranking vector that is used to rank the documents in $\mathcal{R}$; e.g., assume that the candidate results will be ranked solely by their PageRank, ignoring any additional information available. Also assume that all full-precision PageRank values for the candidate documents are distinct.[8] The full-precision PageRank values $R_p(d)$ induce a total ordering over the set $\mathcal{R}$. If we use quantized PageRank values $q(R_p(d))$, then a weak ordering is induced instead. In other words, the relative order of the documents in $\mathcal{R}$ are preserved except for false ties between documents with PageRank values mapped to the same quantizer cell.

The third part of the model is the distortion measure used to judge the rank order inaccuracy caused by quantization. In the simplified scenario being developed, this measure consists of penalizing the false ties, as described in Scenario 1 of Section 2.2.

4.2 Derivation of Optimal Quantizers

For the simple model just discussed, we now derive the optimal quantizer. In particular, consider the case where **Retrieve**$(\mathcal{Q}, \mathcal{D})$ samples M documents from a repository $\mathcal{D}$ uniformly at random, without replacement. Let X_i be the number of query results mapped to cell i.

The number of documents returned by **Retrieve**$(\mathcal{Q}, \mathcal{D})$ will be different for different queries; i.e., M is a random

[7]E.g., $\mathcal{R}$ could be the set of documents that contain all of the query terms in $\mathcal{Q}$.
[8]This assumption fails to hold only for values close to the minimum.

Table 1: A description of 6 quantization strategies we compare.

Strategy	Description
linear	A uniform quantizer (partition uses cells of equal width)
sqrt	Compander with $G(x) \propto \sqrt{x}$
log	Compander with $G(x) \propto \log x$
mse_optimal	Compander with $G(x) \propto x^{-2.17}$
approx_eq_depth	Compander that approximates equal depth partitions, as described in Section 4.3
eq_depth	Quantizer where partition assigns an equal number of points to each cell

variable. For now, consider the case where $M = m$ for some constant m. As described in Section 2.2, we let the distortion of a particular result of length m be measured by the sum of squares of the number of points in the same cell, normalized so that the maximum distortion is 1; in other words, we measure the distortion of the results $\mathcal{R}$, where $|\mathcal{R}| = m$, using a quantizer with n cells as:

$$Distortion_m(q_j) = \frac{1}{m^2}\sum_{0}^{n-1} X_i^2 \tag{5}$$

We can treat $\mathcal{D}$, the documents in the corpus, as a multi-type population, with n types. The type of each document is simply the cell it is mapped to by the quantizer q_j. Let N_i represent the number of points in the input data set[9] that the quantizer maps to cell i (i.e., the count of each type), and let N be the total number of input values (i.e., $N = \sum N_i$). Because the operation **Retrieve**$(\mathcal{D}, \mathcal{Q})$ samples from the population $\mathcal{D}$ uniformly at random, without replacement, $< X_0, \ldots, X_{n-1} >$ follows the multivariate hypergeometric distribution, with parameters m and $< N_0, \ldots, N_{n-1} >$. We assume that $|\mathcal{D}| \gg |\mathcal{R}|$, so that the multinomial distribution (i.e., the distribution that would arise if **Retrieve** sampled *with* replacement), with parameters m and $< \frac{N_0}{N}, \ldots, \frac{N_{m-1}}{N} >$, is a reasonable approximation.[10] The task of finding the optimal n-cell quantizer is reduced to choosing cell depths which minimize the expectation of Equation 5. Note that linearity of expectation allows us to consider each X_i separately, even though they are not independent:

$$E[Distortion_m] = \frac{1}{m^2}E[\sum X_i^2] \tag{6}$$

$$= \frac{1}{m^2}\sum E[X_i^2] \tag{7}$$

Since each X_i follows a binomial distribution, $E[X_i^2]$ is easy to find. Letting $p_i = \frac{N_i}{N}$, and using the known mean and variance of binomial random variables [4], we see that

$$E[X_i] = mp_i \tag{8}$$

$$var[X_i] \equiv E[X_i^2] - E[X_i]^2 \tag{9}$$

$$var[X_i] = mp_i(1-p_i) \tag{10}$$

$$E[X_i^2] = mp_i(1-p_i) + (mp_i)^2 = mp_i + m(m-1)p_i^2 \tag{11}$$

[9] I.e., the PageRank vector we are compressing.

[10] The approximation has no impact on the final solution; see Appendix A for the full derivation using the multivariate hypergeometric distribution

So we need to find the p_i that minimizes

$$E[Distortion_m] = \frac{1}{m^2}\sum E[X_i^2] \tag{12}$$

$$= \frac{1}{m^2}\sum_i \left(mp_i + m(m-1)p_i^2\right) \tag{13}$$

$$= \frac{1}{m} + \frac{m-1}{m}\sum_i p_i^2 \tag{14}$$

The above is equivalent to minimizing $\sum_i p_i^2$ subject to the constraint $\sum_i p_i = 1$. Lagrange multipliers can be used to show that the optimal solution is given by

$$p_i^* = \frac{1}{n} \tag{15}$$

$$N_i^* = Np_i^* = \frac{N}{n} \tag{16}$$

In other words, an equal-depth partition scheme that places equal numbers of points in each cell minimizes the expected distortion of the query results for the TDist distortion measure.

The above considered the case where M, the number of results, was fixed to some constant m. However, different queries have different numbers of results, so that M is a random variable. However since Equation 16 is independent of m, the optimal solution in the case where M varies is also given by Equation 16.

We now discuss several extensions to make our model of the operation **Retrieve** more realistic. Consider the case where the candidate query-results are first pruned based on a threshold for their cosine similarity to the query, then ranked purely by the quantized PageRank $q(R_p)$. The intuition behind this model is that the ranking function first chooses a set of documents thought to be relevant to the query, and then ranks these relevant candidates by their popularity. Our experiments showed virtually no correlation between the PageRank of a document, and its cosine similarity to queries.[11] Thus, since the *pruned* candidate set is expected to follow the same distribution as the *raw* candidate set, the optimal solution is unchanged in this new model.

A second extension to the model is to make the random sampling nonuniform. In other words, each document can have a different probability of being chosen as a candidate result. In this case, the hypergeometric distribution no longer applies since different objects of a given type have different probabilities of being chosen. We could assume that the result set $\mathcal{R}$ is constructed by a sequence of m multinomial trials (i.e., sampling with replacement). Let $p(d)$

[11] The exact correlations are not given here, but were all close to zero. This result is expected, since PageRank is a purely link-based, *query-independent* estimate of page importance.

be the probability of document d being chosen during a trial. Let $p(cell_i) = \sum_{d_j \in cell_i} p(d_j)$. Then the random vector $< X_0, \ldots, X_{n-1} >$ follows the multinomial distribution with parameters m and $< p(cell_0), \ldots, p(cell_{n-1}) >$. The previous multinomial assumption holds if in addition to the requirement $|\mathcal{D}| \gg |\mathcal{R}|$, we also stipulate that no document dominates the probability mass of its cell. If $p(d)$ is extremely nonuniform among documents with similar values of the attribute being quantized, then sampling *with* replacement is no longer a good approximation to sampling *without* replacement. If the multinomial approximation does hold, then a derivation, similar to the above, shows that an *equiprobable* partition is optimal. In other words, instead of making the depths of all cells constant, we make the probability mass assigned to each cell constant:

$$\sum_{d_j \in cell_i} p(d_j) = \frac{1}{n} \tag{17}$$

In the general case, where the final rankings are generated using arbitrary ranking functions that numerically combine the scores from several ranking vectors, developing a probabilistic model for analytically deriving a solution becomes difficult; for such cases, we currently rely on empirical results, measuring the average distortion across a large number of sample queries.

4.3 Approximating Equal-Depth Partitioning

Using an equal-depth partition, although optimal for the TDist distortion measure, could lead to additional overhead. In the encoding phase, the true equal-depth scheme would require a binary search through the interval endpoints to determine the appropriate cell for each input value. Since the encoding step is performed offline, the cost is acceptable. However, in the decoding step, if the reproduction value for a particular cell is needed, a true equal-depth partition scheme requires a codebook that maps from cells to cell centroids, leading to additional space as well as processing costs. We show how we can approximate an equal-depth partition by using a simple compander, with a compressor function derived from the distribution of the underlying data, thus eliminating both the need for binary searches when encoding, and the need for a codebook at runtime.

If the input data values were distributed uniformly, we would intuitively expect that a uniform partition would be similar to an equal-depth partition. We confirm this intuition as follows. Let N be the total number of points, and let N_i be the number of points that fall in cell i, for a quantizer with n cells. If the input points are distributed uniformly at random, then clearly each N_i follows the binomial distribution with parameters $(N, \frac{1}{n})$. Thus, the expected number of points in each cell is simply $\mu = E[N_i] = \frac{N}{n}$. The probability that N_i falls within a tight range of this expectation is high for large N, with $n \ll N$. For instance, for $N = 10^8, n = 10^6$, and using the normal approximation for N_i, we get that $Pr[.8\mu \leq N_i \leq 1.2\mu] \approxeq 0.95$.

Note, however, that the input data is *not* uniform. In particular, as we saw in Section 3.1, PageRank closely follows a power-law distribution. However, we can devise a compressor function $G_{eq}(x)$ that transforms the data to follow a uniform distribution, which we can then uniformly quantize, thus approximating an equal-depth quantizer. Let X be a random variable following the power-law distribution, with exponent 2.17. I.e., the pdf $f(x)$ for X is $f(x) = kx^{-2.17}$. Equivalently, if x_{min} is the minimum possible rank, and x_{max} is the maximum possible rank, the cumulative distribution function (cdf) [12] is $F(x) = \frac{k}{1.17}(x_{min}^{-1.17} - x^{-1.17})$. The normalization constant c is chosen so that $F(x_{max}) = 1$. We would like to find a function $G_{eq}(x)$ such that $G_{eq}(X)$ corresponds to a uniform distribution, i.e., a $G_{eq}(x)$ such that

$$Pr[G_{eq}(X) \leq y] = y \tag{18}$$

But it is easy to see that in fact, $F(x)$ itself is such a function, since the cumulative distribution of $F(X)$ is:[13]

$$Pr[F(X) \leq x] = Pr[X \leq F^{-1}(x)] = F(F^{-1}(x)) = x \tag{19}$$

Thus, a function $G_{eq}(x)$ that will transform the PageRank data to uniformly distributed data is the cdf $F(x)$, assuming that the PageRank distribution is a close fit for the power-law. This transformation allows us to eliminate the explicit codebook, instead using $G_{eq}(x)$ and $G_{eq}^{-1}(x)$ as the compressor and expander functions, resp., to approximate an equal-depth partition. The empirical distribution of $G_{eq}(X)$, shown in Figure 4 as the series **approx_equal_depth**, does indeed appear to be roughly uniform. The results that will be described in Section 4.5 show that in practice, this approximation leads to performance similar to that of exact equal-depth partitioning.

4.4 Data Distribution: Corpus vs. Query Results

In Section 4.2, for deriving the optimal quantizer for the TDist distortion measure (Equation 6), we assumed that the operation $\mathbf{Retrieve}(\mathcal{Q}, \mathcal{D})$ could be modeled as a uniform random sample of $\mathcal{D}$. We now present empirical data showing that this assumption is reasonable. We plotted the PageRank distribution of the raw query results for each of 86 test queries; in every case, the PageRank distribution closely matched the distribution of PageRank values in the corpus $\mathcal{D}$ as a whole. We display the distribution of PageRank values for the results of two representative queries in Figure 6.

The distributions were not an *exact* match, however, leading to the possibility that equiprobable and equal-depth partitions will not behave identically. We randomly partitioned our set of test queries into two halves. Using the first set, we measured the distribution of PageRank values of documents in the results for the queries. Figure 7 shows that this distribution deviates slightly from the distribution of PageRank values in the corpus as a whole; the distribution of PageRank values when restricted to documents that appear as raw candidate results seems to follow a power-law distribution with exponent close to 2.0. In other words, under our simplified model of the operation **Retrieve**, higher rank documents have a slightly higher probability of appearing in raw query results.[14]

[12] The cdf is simply $\int_{x_{min}}^{x} pdf(y)dy$

[13] Note that $\forall_x f(x) > 0$ implies that $F(x)$ is strictly increasing, and thus invertible.

[14] By "raw query results", we simply mean the set of documents containing all of the query terms for some query.

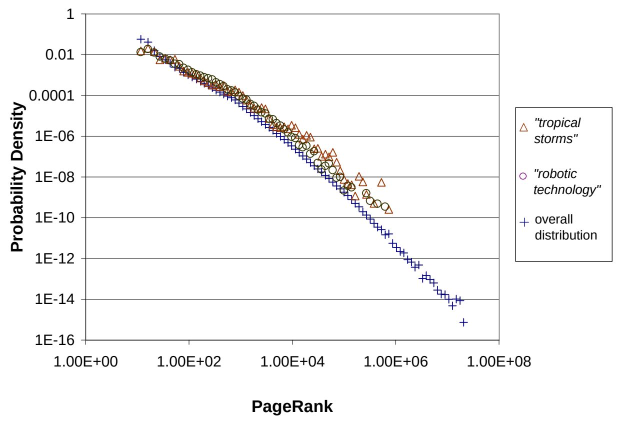

Figure 6: Comparison of the PageRank distribution for the corpus as a whole, to the set of pages containing the query terms "tropical storms", and the set of pages containing the query terms "robotic technology". As expected, the PageRank distribution for the raw conjunctive query results is close to the distribution on the overall corpus.

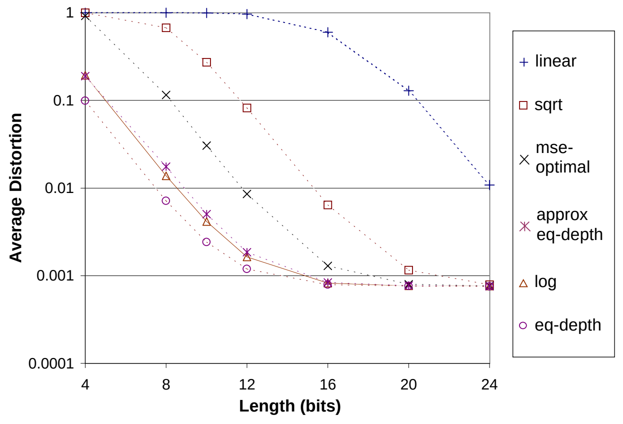

Figure 8: The average distortion for the various strategies when using the TDist distortion measure (Equation 6) over the full lists of query results, for 86 test queries.

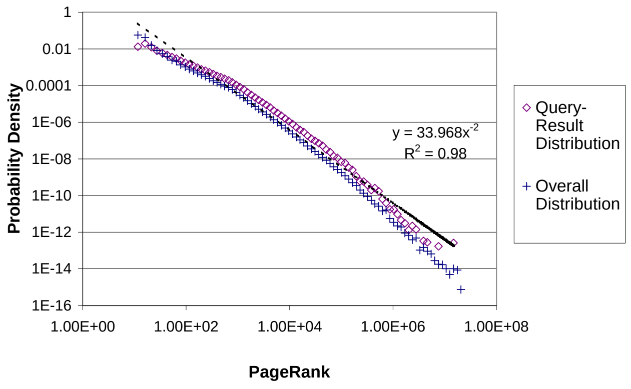

Figure 7: Comparison of the PageRank distribution for the repository as a whole, to the distribution for the set of results to 43 of the test queries.

4.5 Empirical Performance Under Rank-Based Distortion Measures

We now explore the empirical performance of various quantization schemes on sample query results. Our test set of 86 queries consisted of 36 queries compiled from previous papers and 50 queries created using the titles from the TREC-8 topics 401-450 [11]. Using a text index for our repository, we retrieved, for each query, the URLs for all pages that contain all of the words in the query.

Figure 8 plots the average (over the 86 query results) of the distortion for the 6 strategies when using the TDist distortion measure. We see that as expected, the **equal_depth** strategy performs the best for all codelengths. Also note that the **approx_equal_depth** and **log** strategies also perform similarly. Notably, the **mse_optimal** strategy, which was optimal when using the MSE distortion criteria, is no longer the optimal choice – this result signifies the need to consider appropriate notions of distortion when choosing quantization strategies for search rankings.

In Section 4.2, we argued that for the TDist distortion measure, for a retrieval model in which documents in the corpus have different probabilities of appearing in the results, an equiprobable, rather than an equal-depth, partition is superior. As mentioned in Section 4.4, we noticed a slight correlation between the PageRank and the probability of appearing in the raw candidate result set. To test the performance of the **equal_prob** strategy, we implemented the compander described in Section 4.3, replacing the pdf with $f(x) = kx^{-2.0}$. On the subset of test queries that were not used in estimating the power-law exponent, we measured the performance, using the TDist distortion measure, of this approximation to an equiprobable partition. The results are shown in Figure 9, and demonstrate that the (approximate) equiprobable scheme improves upon the (approximate) equal-depth scheme. However, the true equal-depth scheme still performs the best by a small margin.

Figure 10 plots the average TDist distortion when the query results are first pruned to include only the top 100 documents based on the pure cosine similarity to the search query, then ranked using only the quantized PageRank value. As expected, the results match the results of Figure 8; since cosine similarity is uncorrelated with PageRank, the optimal strategy is unchanged from that of Figure 8.

Our third query result scenario and distortion measure are as follows. Let τ be the ordered list of the top 100 documents when query results are ranked by the composite score $cos_{\mathcal{Q}d} \cdot r_d$; let τ_q be the ordered list of the top 100 documents when query results are ranked by $cos_{\mathcal{Q}d} \cdot q(r_d)$ for some quantizer q. Note that $\tau \neq \tau_q$ because $q(r_d)$ is less precise than r_d. We measure distortion using the KDist measure described in Section 2.2. As shown in Figure 11, in this scenario, the **log** strategy performs the best for all codelengths in minimizing the mean KDist distortion.

The previous results demonstrate the importance of using a distortion measure suited to the ranking strategy used by the search engine when choosing a quantization strategy.

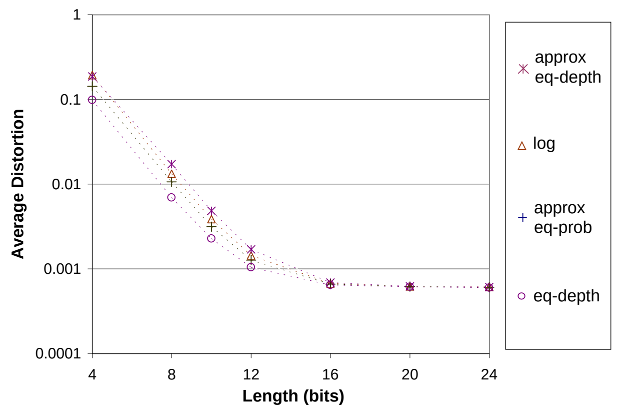

Figure 9: The approximate equiprobable partition outperforms the approximate equal-depth partition on the TDist **distortion measure. Documents with high PageRank had slightly higher probabilities of being candidate results. The true equal-depth partition strategy still has the least distortion.**

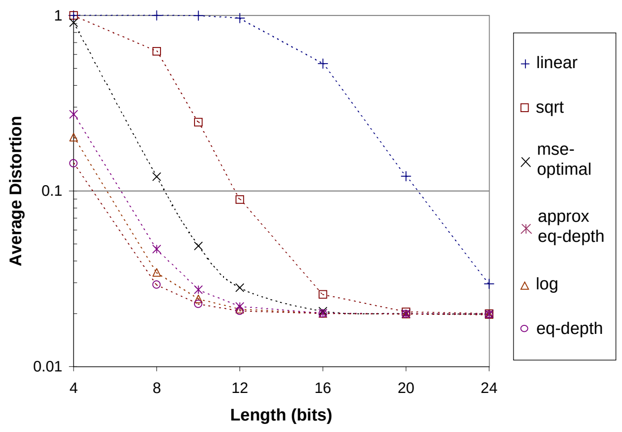

Figure 10: The average distortion for the various strategies when using the TDist **distortion measure over pruned lists of query results, for 86 test queries. The pruned lists consisted of the top 100 results based on the cosine similarity of the documents to the query.**

5. VARIABLE-LENGTH ENCODING SCHEMES

Fixed length encoding schemes are much simpler to support in the implementation of the ranking function, because the attribute values are at easily computed offsets into the attribute vector. Assuming consecutively assigned document identifiers, and a fixed-length encoding using l bits per codeword, the value for the attribute for document i is at bit location $i \times l$. Variable-length encodings have the potential to reduce the *average* codeword lengths, and thus the overall storage requirement for the ranking vector. However, the downside is a more complex decoding process, which is less efficient and may not be practical, depending on the search engine's performance criteria. In particular, to retrieve the attribute values when the ranking vector is encoded with a variable-length scheme, a sparse index[15] is needed to allow the lookup of the block containing the desired value. Furthermore, all the values in that block preceding the desired value would also need to be decoded. For a large-scale search engine supporting millions of users, this additional overhead may not be affordable. In this section, we first explore the effectiveness of variable-length schemes in minimizing storage, and then investigate the additional runtime costs for decoding variable-length codes.

5.1 Variable-Length Encoding Performance

To investigate the effectiveness of variable-length schemes, we computed the average Huffman codelengths for the quantization schemes previously discussed in Sections 3 and 4. When the MSE distortion of Figure 5 is plotted against the Huffman codelength, rather than the fixed codelength, the uniform quantization strategy, **linear**, becomes the best performer, as shown in Figure 12.[16] The variable-length encoding for the cells eliminates the inefficiencies of uniform quantization. This effect carries over to the result-based distortion measures as well, as shown in the replotting of Figure 8, given as Figure 13. Note that the equal-depth approaches derive no benefit from Huffman coding – the average codelength is reduced precisely when the cell depths are *non*uniform. Note that the average codeword lengths shown in these graphs does not include the memory required for the additional indexes needed at runtime for efficient decoding; we discuss decoding requirements in Section 5.2.

These empirical results indicate that if a variable-length encoding scheme is used to generate codes for the cells, a uniform quantizer performs similarly to the optimal quantizer, under the distortion measures we used.

5.2 Variable-Length Encoding Costs

Variable-length encoders outperform fixed-length encoders, when judged on the average codelength needed to achieve a particular distortion, for most of the distortion measures we have discussed. However, there is a substantial processing overhead at query time to decode the numeric attribute values. The driving motivation behind our work was to reduce the per-document attribute lookup cost by fitting the ranking vectors in main memory; variable-length encodings are only appropriate if fixed-length encodings are not sufficient to allow the attribute vectors to be stored in memory. We next discuss the offline and query-time costs of variable-length schemes compared to fixed-length schemes.

During the offline step, compression of the input data values using variable-length schemes requires first generating Huffman codes for the cells of the partition, and then generating a compressed version of the input by replacing each input value with the Huffman codeword assigned to the cell the value was mapped to. A fixed-length scheme does not require generating a Huffman code — the intervals can be assigned sequential l-bit ids. However, the cost of generating a Huffman code is fairly low; using the implementation of [9], we were able to generate the codebook and compress the in-

[15]The index needs to be sparse, since otherwise any space savings from a variable-length coding scheme would be lost.
[16]That uniform-width quantizers are near-optimal for variable-length encoding is known for the MSE distortion measure [3].

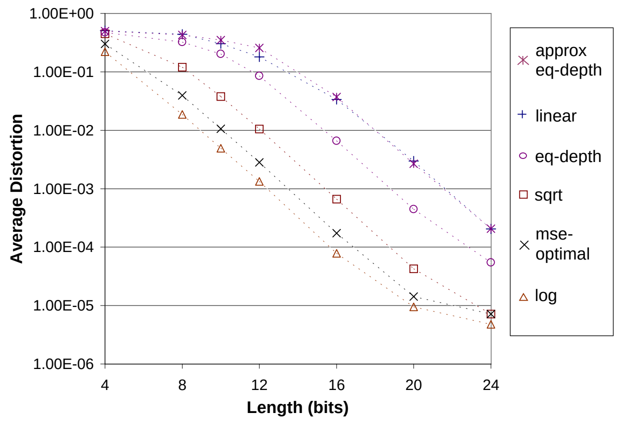

Figure 11: The average distortion for the various strategies when using the KDist **distortion measure over pruned lists of query results, for 86 test queries. The pruned lists consisted of the top 100 results ranked by the score $cos_{Qd} \cdot r_d$.**

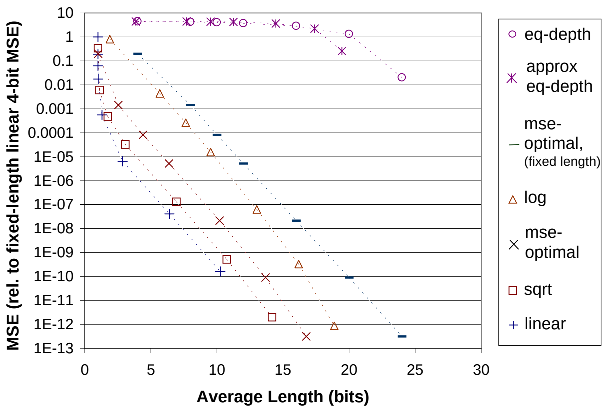

Figure 12: The MSE of 6 different strategies, plotted against the average codeword length used. The MSE axis is on a log scale, and is normalized so that the MSE of the 4-bit linear compander is 1. For comparison, the optimal fixed-length performer is also shown; the simple linear quantizer, using variable-length codes, performs better than the optimal fixed-length strategy.

put data (360M values, 1.34GB) in under 10 minutes using an AMD Athlon 1533MHz machine with a 6-way RAID-5 volume. Given the minimal impact of small variations in preprocessing cost, we did not explore further the offline overhead for variable-length encoding schemes.

The impact of additional *query-time* costs, however, is more significant. For both the fixed-length and variable-length scenarios, the query engine loads the entire sequence of quantized values into memory as a string b of bits. We assume that documents are identified by consecutively assigned document identifiers, so that the document with ID i is the ith value inserted into the bit string. In the case of a fixed-length scheme with codewords of length l, the attribute value associated with some document i is simply the value of the bit substring $b[i \times l, (i+1) \times l - 1]$. The only cost is a memory lookup; in the case where l is not the length of standard integer data type (e.g., 8, 16, or 32), a few bit shifts are also required.

When using a variable-length code, however, for a given document i, finding the exact location for its attribute value in the bit string becomes nontrivial. Decoding from the beginning of the string until finding the ith value is of course inefficient, making an index necessary. More specifically, since maintaining the exact offset into the bitstring for each value would completely negate the benefit of compression, we must use a sparse index which maintains offsets to blocks of values. Decoding the attribute value for document i requires decoding all the values from the beginning of the block up through the desired value. Thus, the decoding time is proportional the block size B; more precisely, the expected number of decodes is $B/2$. Using small blocks reduces the decoding time, but in turn increases the space usage of the sparse index. Figure 14 shows the decode time, in μs/document vs. block size, for 4 variable-length schemes. For comparison, the decode time for a fixed-length encoding scheme is also given. These times were measured on an AMD Athlon 1533MHz machine with 2GB of main memory. The additional space overhead, in bits/codeword, needed by the sparse index for 360M values for various block sizes is plotted in Figure 15.

The times may seem very small, making the variable-length schemes seem attractive; however, for a large-scale search engine, with thousands of concurrent active queries, where each query has thousands of candidate results requiring attribute value decodings for tens of attributes, the per-result decode time needs to be as inexpensive as possible. As an illustrative example, consider a search engine with 1 billion pages with a query workload of 10 queries/s. Assume that each document has a single numeric property (e.g., PageRank) that needs to be decoded for calculating final rankings. Also assume that the average query yields .01% of the repository as candidate results, so that the processing for each query requires retrieving the numeric properties for 100,000 documents. If a variable-length scheme is used, so that the decode time for a single attribute value for a single document requires 35 μs, decoding alone will require 3.5 seconds of CPU time per query, or equivalently, 35 machines are needed to handle the query workload (if decoding were the only cost in the system). If the decode time for is instead 1 μs per document (e.g., utilizing a fixed-length encoding scheme), only 0.1s is spent decoding for each query; equivalently, a single machine can handle the query workload. Of course there are other significant costs in the system in addition to attribute value decode time; discussing them in detail is beyond the scope of this paper. Our goal in this example is simply to provide some intuition as to why per-document decode times need to be kept very small for large-scale search engines.

6. RELATED WORK

There has been much work in the field of compression in the context of large-scale Web search. An excellent overview of text-index compression techniques can be found in [19]. Suel and Yuan [18] investigate strategies for compressing the Web hyperlink graph. Raghavan and Garcia-Molina [16] explore techniques for compressing the Web link graph in ways that allow for efficient query processing.

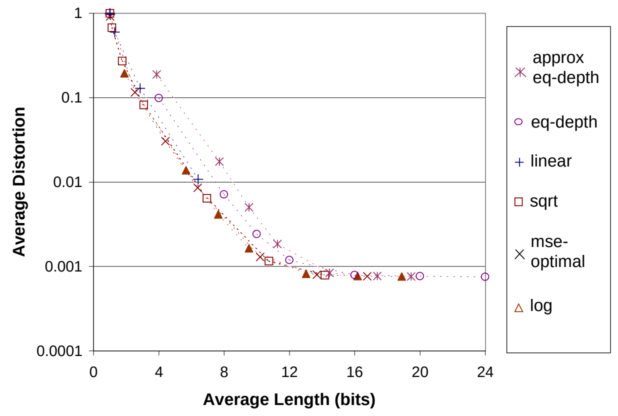

Figure 13: The average distortion for the variable length strategies when using the TDist distortion measure over the full lists of query results, for 86 test queries. The x-axis represents the *average* codeword lengths using a Huffman code.

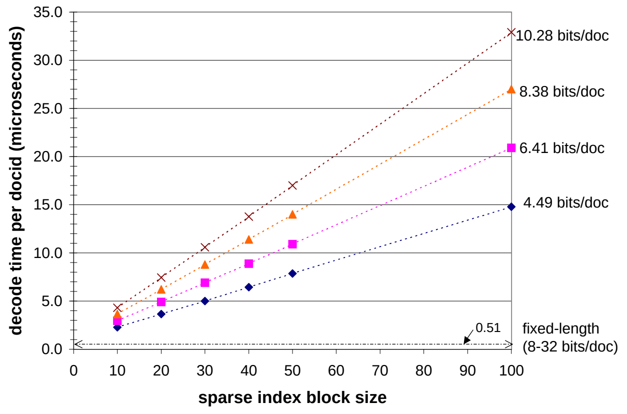

Figure 14: The decode time in microseconds per document for a PageRank vector quantized uniformly using variable-length codes with 4 different average codeword lengths. The decode time using a fixed-length code is also given for comparison.

Our work explores the development of lossy encodings for auxiliary numeric ranking vectors, where the quality of an encoding is judged by its effect on the final *rankings* induced over query results. An approach for efficiently encoding the document-length vector, needed for cosine computations, was studied in [10]. However, that work did not consider variable-length encodings, and did not provide analytic results for the behavior of the encodings under various models for query result distributions. The use of measures based on Kendall's-τ rank correlation for comparing document rankings has recently become popular [1, 6], and is given a thorough treatment by Fagin et al. in [2].

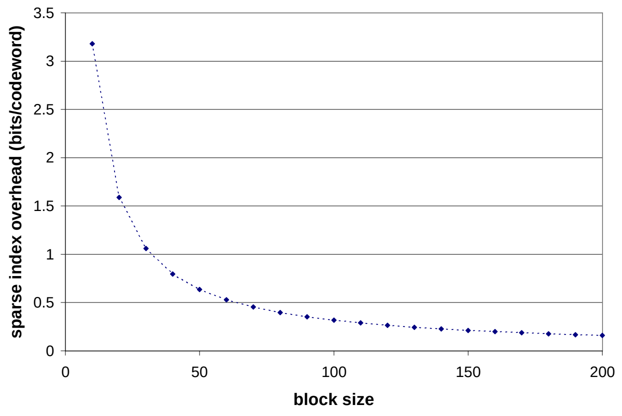

Figure 15: The additional space overhead needed by the sparse index, measured as bits/codeword, for block sizes ranging from 10 to 200.

7. ACKNOWLEDGMENTS

I would like to thank Professor Jeff Ullman for his comments and feedback. I would also like to thank Mayur Datar and Aristides Gionis for comments and useful discussions on the derivations given in Section 4.2 and Appendix A.

8. REFERENCES

[1] Cynthia Dwork, Ravi Kumar, Moni Naor, and D. Sivakumar. Rank aggregation methods for the web. In *Proceedings of the Tenth International World Wide Web Conference*, 2001.

[2] Ronald Fagin, Ravi Kumar, and D. Sivakumar. Comparing top k lists. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms*, 2003.

[3] Robert M. Gray and David L. Neuhoff. Quantization. *IEEE Transactions on Information Theory*, 44(6), October 1998.

[4] G. R. Grimmett and D. R. Stirzaker. *Probability and Random Processes*. Oxford University Press, New York, 1992.

[5] Taher H. Haveliwala. Efficient computation of PageRank. *Stanford University Technical Report 1999-31*, 1999.

[6] Taher H. Haveliwala. Topic-sensitive PageRank. In *Proceedings of the Eleventh International World Wide Web Conference*, 2002.

[7] J. Hirai, S. Raghavan, H. Garcia-Molina, and A. Paepcke. Webbase: A repository of web pages. In *Proceedings of the Ninth International World Wide Web Conference*, 2000.

[8] Glen Jeh and Jennifer Widom. Scaling personalized web search. *Stanford University Technical Report*, 2002.

[9] A. Moffat and J. Katajainen. In-place calculation of minimum-redundancy codes. In S.G. Akl, F. Dehne, and J.-R. Sack, editors, *Proc. Workshop on Algorithms and Data Structures*, pages 393–402, Queen's University, Kingston, Ontario, August 1995. LNCS 955, Springer-Verlag.

[10] Alistair Moffat, Justin Zobel, and Ron Sacks-Davis. Memory efficient ranking. *Information Processing and Management*, 30(6):733–744, November 1994.

[11] National Institute of Standards and Technology (NIST). *The 8th Text Retrieval Conference (TREC-8)*, 1999.

[12] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The PageRank citation ranking: Bringing order to the web. *Stanford Digital Libraries Working Paper*, 1998.

[13] Gopal Pandurangan, Prabhakar Raghavan, and Eli Upfal. Using PageRank to characterize web structure. In *Proceedings of the Eighth Annual International Computing and Combinatorics Conference*, 2002.

[14] P. Panter and W. Dite. Quantization in pulse-count modulation with nonuniform spacing of levels. In *Proceedings of IRE*, Jan. 1951.

[15] David Pennock, Gary Flake, Steve Lawrence, Eric Glover, and C. Lee Giles. Winner's don't take all: Characterizing the competition for links on the web. In *Proceedings of the National Academy of Sciences*, 2002.

[16] Sriram Raghavan and Hector Garcia-Molina. Representing web graphs. In *Proceedings of the 19th IEEE International Conference on Data Engineering (ICDE)*, 2003.

[17] Matthew Richardson and Pedro Domingos. *The Intelligent Surfer: Probabilistic Combination of Link and Content Information in PageRank*, volume 14. MIT Press, Cambridge, MA, 2002.

[18] Torsten Suel and Jun Yuan. Compressing the graph structure of the web. In *IEEE Data Compresseion Conference (DCC)*, March 2001.

[19] Ian H. Witten, Alistair Moffat, and Timothy C. Bell. *Managing Gigabytes*. Morgan Kaufmann Publishers, San Francisco, 1999.

APPENDIX

A. OPTIMAL QUANTIZER: DERIVATION USING THE MULTIVARIATE HYPERGEOMETRIC DISTRIBUTION

The derivation of the optimal quantizer under the distortion measure given in Equation 5, when the operation **Retrieve** is modeled as a uniform random sample, *without replacement*, is very similar to the derivation given in Section 4.2. As before, we want to minimize Equation 6, although each X_i now follows the hypergeometric, rather than the binomial, distribution. Using the known mean and variance for the hypergeometric distribution [4], with parameters N, N_i, and m, and letting $p_i = \frac{N_i}{N}$, we compute $E[X_i^2]$ as before:

$$E[X_i] = mp_i \tag{20}$$

$$var[X_i] \equiv E[X_i^2] - E[X_i]^2 \tag{21}$$

$$var[X_i] = mp_i(1-p_i)\frac{N-m}{N-1} \tag{22}$$

$$E[X_i^2] = mp_i(1-p_i)\frac{N-m}{N-1} + (mp_i)^2 \tag{23}$$

Plugging $E[X_i^2]$ back into Equation 6 and simplifying, we get

$$\begin{aligned}
E[Distortion_m] &= \frac{1}{m^2}\sum E[X_i^2] && (24)\\
&= \frac{1}{m^2}\sum_i \big(mp_i(1-p_i)\frac{N-m}{N-1} + (mp_i)^2\big) && (25)\\
&= \frac{1}{m^2}\sum_i \big(\frac{N-m}{N-1}(mp_i - mp_i^2) + m^2p_i^2\big) && (26)\\
&= \frac{N-m}{m(N-1)}\sum_i p_i + \big(1 - \frac{N-m}{m(N-1)}\big)\sum_i p_i^2 && (27)\\
&= \frac{N-m}{m(N-1)} + \big(1 - \frac{N-m}{m(N-1)}\big)\sum_i p_i^2 && (28)
\end{aligned}$$

Given values for N and m, the above is equivalent to minimizing $\sum_i p_i^2$ subject to the constraint $\sum_i p_i = 1$, leading to the solution given by Equation 16, i.e., an equal-depth partitioning scheme.