# The Effect of Inverse Document Frequency Weights on Indexed Sequence Retrieval

Kevin C. O'Kane

Department of Computer Science

The University of Northern Iowa

Cedar Falls, Iowa 50613

okane@cs.uni.edu

http://www.cs.uni.edu/~okane

## Abstract

**Summary:** The IDF (Inverse Document Frequency) weight is a method to calculate a relative weighting factor for words used in natural language text indexing. The purpose of these experiments was to determine the effect of this technique when applied to retrieval of genomic sequences. This paper reports the results of an implementation of the IDF on the NCBI "nt" nucleotide data base and compares the results with other sequence retrieval systems. The implementation discussed here consists of a collection of open-source, C++ programs written for Linux that (1) build an inverted index of all sequences in a data base; (2) calculate a relative weight for each possible tuple of length k; (3) and retrieve, score, and rank, sequences from the data base in response to queries. Retrieval speeds are based primarily on query length rather than data base size.

## Introduction

Current access to genomic databases is mainly accomplished by means of heuristic-assisted pattern matching from flat or nearly flat files of stored sequences using programs such as BLAST [Altschul 1990] and FASTA [Pearson 2000]. The underlying data bases are growing rapidly with consequent deterioration of search times even on large, multiprocessor systems as current software tools reach their design limits.

BLAST and other similar systems pre-index each data base sequence according to short code letter words (generally, 3 letters for amino acid and 11 letters for nucleotide data bases). Queries are decomposed into similar short code words. The data base is scanned and those stored sequences having code words in common with the query are processed further to extend the initial code word match. Substitution matrices are frequently used with protein sequences to allow for evolutionary mutation. Statistical analysis of the results predict the degree to which an alignment is by chance, relative to the size of the data base. Scanning of the sequence indices is primarily sequential although many speed enhancing techniques are employed such as multiprocessor support.

The related area of natural language text indexing and retrieval has been studied since the mid-50's. In processing natural language text, the problem is to locate documents most closely matching a natural language query. To do this, a number of techniques have been developed to identify which terms in a document are most likely to be good indicators of content as opposed to terms that are poor content descriptors. By eliminating the poor descriptors and pre-indexing the documents by good descriptors, recall and precision can be enhanced. In this experiment, the

*inverse document frequency* [Salton 1983] weight was adapted for use with the NCBI “nt” nucleotide data base.

**Experiment Design**

Sequences from the NCBI "nt" non-redundant nucleotide data base were used. The “nt” data base (*ftp://ftp.ncbi.nih.gov/blast/db/FASTA*) was approximately 12 billion bytes in length at the time of the experiment and consisted 2,584,440 sequences in FASTA format.

The overall frequencies of occurrence of all possible 11 character tuples in each sequence in the data base were determined along with the number of sequences in which each unique word was found. A total of 4,194,299 unique words were identified, slightly less than the theoretical maximum of 4,194,304. The tuple size of 11 was initially selected as this is the default tuple size used in BLAST for nucleotide searches. The programs however, will accommodate other tuple lengths and the default tuple size for proteins is three.

The data base was processed according to the diagram in Figure 3. Each sequence in the “nt” data base was read and and decomposed into all possible words in the sequence string of length 11 by taking all 11 letter words beginning at starting position one, two and so on up to and including starting position eleven. Procedurally, given the vast amount of words produced, this was accomplished by producing multiple (about 110) intermediate files (*.words) of about 440 million bytes each, ordered alphabetically by word and listing, for each word, the relative reference number of the original sequence containing the word. A relative sequence number was used as it could be expressed in four bytes rather than an offset which would have required eight

bytes due to the size of the input data base (12 GB). A master table named *offset.table* was also produced that translates each relative sequence reference into an eight byte offset into the original data base. The multiple intermediate files were subsequently merged and and three files produced: (1) a large (40 GB) word-sequence table, *out.table*, giving, for each word, a list of the sequence references of those sequences in which the word occurs; (2) a file (*freq.bin*) containing the IDF weights for each word (53 MB); and (3) a file named *index* (76 MB) giving for each word the eight byte offset of the word's entry in *out.table*. Finally, *index* and *freq.bin* are merged into *ITABLE* (112 MB) which contains for each word its weight, offset, and a pointer to a list of aliases (not used with the "nt" data base).

The IDF weights (*freq.bin)* $\boldsymbol{W_i}$ for each word *i* were calculated by taking the base 10 logarithm, multiplied by 10 and truncated to the nearest integer, of the total number of sequences (*N*) divided by the number of sequences in which each word occurred ($DocFreq_i$):

$$W_i = (\textbf{int})\ 10 * \mathrm{Log}_{10} ( N / \mathrm{DocFreq}_i )$$

This weight yields higher values for words whose distribution is more concentrated and lower values for words whose use is more widespread. Thus, words of broad context are weighted lower than words of narrow context.

For retrieval, each query sequence was read and decomposed into 11 character words. These words were reduced to a numeric equivalent and this was used to index *ITABLE*. Entries in a master scoring vector corresponding to data base sequences were incremented by the weight of the word if the word occurred in the sequence and if the weight of the word lay within a specified

range. Ultimately, when all words were processed, entries in the master sequence vector were normalized according to the length of the underlying sequences and to the length of the query. Finally, the master sequence vector was sorted and the top scoring entries were either printed or submitted to a Smith-Waterman [Smith 1981] procedure for more detailed scoring and then re-sorted and printed. Optionally, the Smith-Waterman alignments details can be printed and the selected sequences can be extracted from the data base and stored in a separate output file for additional post-processing.

**Results**

Figure 1 shows a graph of aggregate word frequency by weight. The height of each bar reflects the total number of instances of all words of a given weight in the data base. The bulk of the words, as is also the case with natural language text [Salton 1983, O'Kane 2004], reside in the middle range. The slight bulge on the right side of the otherwise uniform bell shaped curve was, however, unexpected. Figure 2 is a graph of the number of distinct words at each weight. The twin peaks were also unexpected, based on experience in natural language indexing. The two distinct peaks suggest the possible presence of two "vocabularies" with overlapping bell curves.

Five hundred test queries were randomly generated from the "nt" data base by (1) randomly selecting 500 sequences whose length was between 200 and 800 letters; (2) from each of these, extracting a random contiguous subsequence between 200 and 400 letters; and (3) randomly mutating 1 letter out of 12.

The test queries were processed and scored by the indexing program with weighting enabled and

disabled and also by BLAST. The output of each consisted of 500 sequence title lines ordered by score. The results are summarized in Table 1 and Figures 4 and 5. In Figures 4 and 5, larger bars further to the left indicate better performance (ideally, a single large bar at position 1). The *Average Time* includes post processing of the results by a Perl program. The *Average Rank* and *Median Rank* refer to the average and median positions, respectively, in the output of the sequence from which a query was originally derived. A lower number indicates better performance. The bar at position 60 indicates all ranks 60 and above as well as sequences not found.

When running in unweighted mode, all words in a query were weighted equally and sequences containing those words were scored exclusively on the unweighted cumulative count of the words in common with the query vector. When running in weighted mode, query words were used for indexing if they fell within the range of weights being tested and data base sequences were scored on the sum of the weights of the terms in common with the query vector and normalized for length.

Figure 5 shows results obtained using the 500 random sequences using indexing only and no weights. The graph in Figure 4 shows significantly better results for the same query sequences with weighted indexing enabled (see also Table 1). When all weighted words are used, the performance declines as reflected in Table 1 indicated by the legends “Weighted 65+ and “Weighted All”.

Subsequently, multiple ranges of weights were tested with the same random sequences. These tests are reflected in Table 1. In these tests, only words within certain weight ranges were used.

The primary indicators of success were the *Average Rank* and the number of sequences found and not found. From these results, it appears that optimal performance can be obtained using weights in the general range of 85 to 100 which corresponds to the down slope of the second peak in Figure 2. An unexpected similar improvement in performance can also be seen in those tests that encompassed the downslope of the first peak in Figure 2, as seen in range 70-80.

On larger query sequences (5,000 to 6,000 letters), the IDF weighted method performed slightly better than BLAST. On 25 long sequences randomly generated as noted above, the IDF method correctly ranked the original sequence first 24 times, and once at rank 3. BLAST, on the other hand, ranked the original sequence first 21 times while the remaining 4 were ranked 2, 2, 3 and 4. Average time per query for the IDF method was 47.4 seconds and the average time for BLAST was 122.8 seconds.

Word sizes other than 11 were tested but with mixed results. Using a word longer than 11 greatly increases the number of words and intermediate file sizes while a smaller value results in too few words relative the the number of sequences to provide full resolution.

**Conclusions**

The results indicate that using IDF weights improves indexed retrieval of nucleotide sequences and is generally twice as fast as BLAST on queries of length up to 6000 letters. Based on these results, more sophisticated and time consuming indexing techniques may also yield benefits. These include hierarchical sequence clustering, synonym recognition, vocabulary clustering and other from the area of natural language indexing. The unexpected anomalies in the graphs of the

word frequencies by weight noted above require additional study to determine the nature of the phenomena.

The software for the IDF indexer and retrieval phase is available without charge in source code form under the open-source GNU General Public License (GPL) for Linux and Cygwin (for Windows) at www.cs.uni.edu/~okane. It is part of a Linux-based toolkit of software designed to aid in the fast retrieval and matching of databases of up to 256 terabytes in size.

**References**

Altschul SF, Gish W, Miller W, Myers EW, Lipman DJ. (1990) Basic local alignment search tool. *J. Mol. Biol*. 215:403-10.

O'Kane, K.C.; and Lockner, M. J. (2004) Indexing genomic sequence libraries, *Information Processing and Management*, 41:265-274.

Pearson, W. R. (2000) Flexible sequence similarity searching with the FASTA3 program package. *Methods Mol. Biol.* 132:185-219.

Salton, G. (1983), *Introduction to Modern Information Retrieval,* McGraw-Hill (New York 1983).

Smith, T.F. & Waterman, M.S. (1981) Identification of common molecular subsequences. *J. Mol. Biol*. 147:195-197

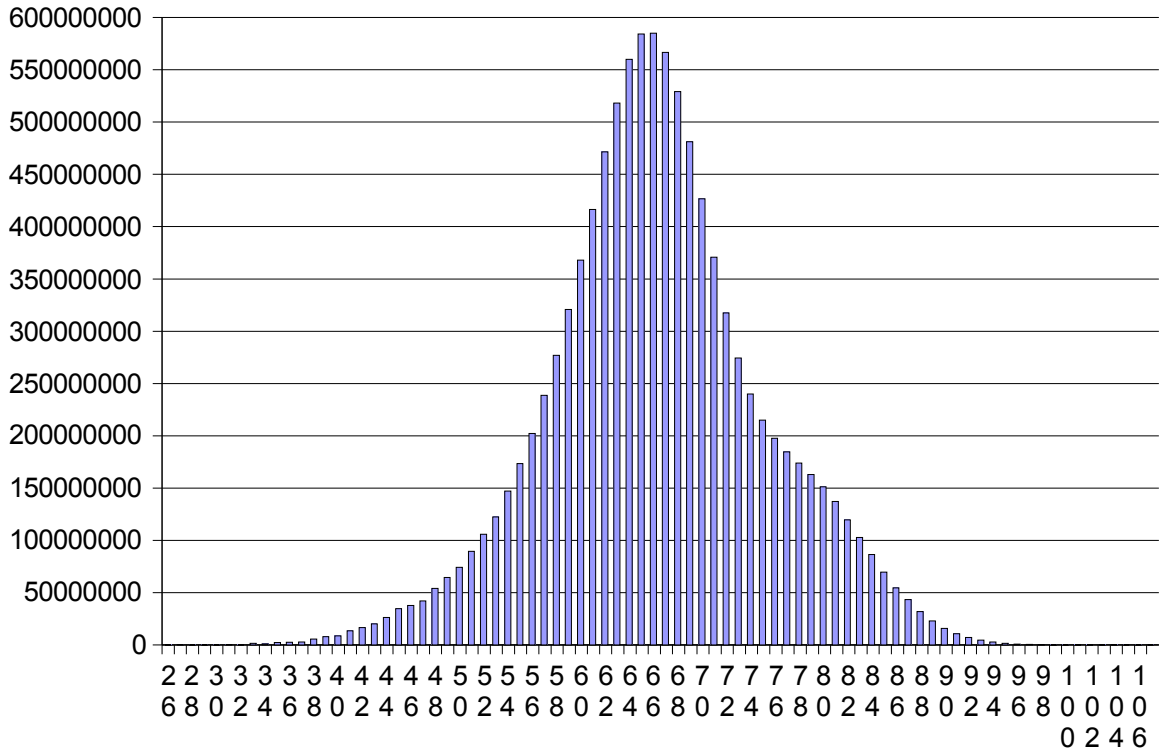


Figure 1

Aggregate Distribution of Words by Weight

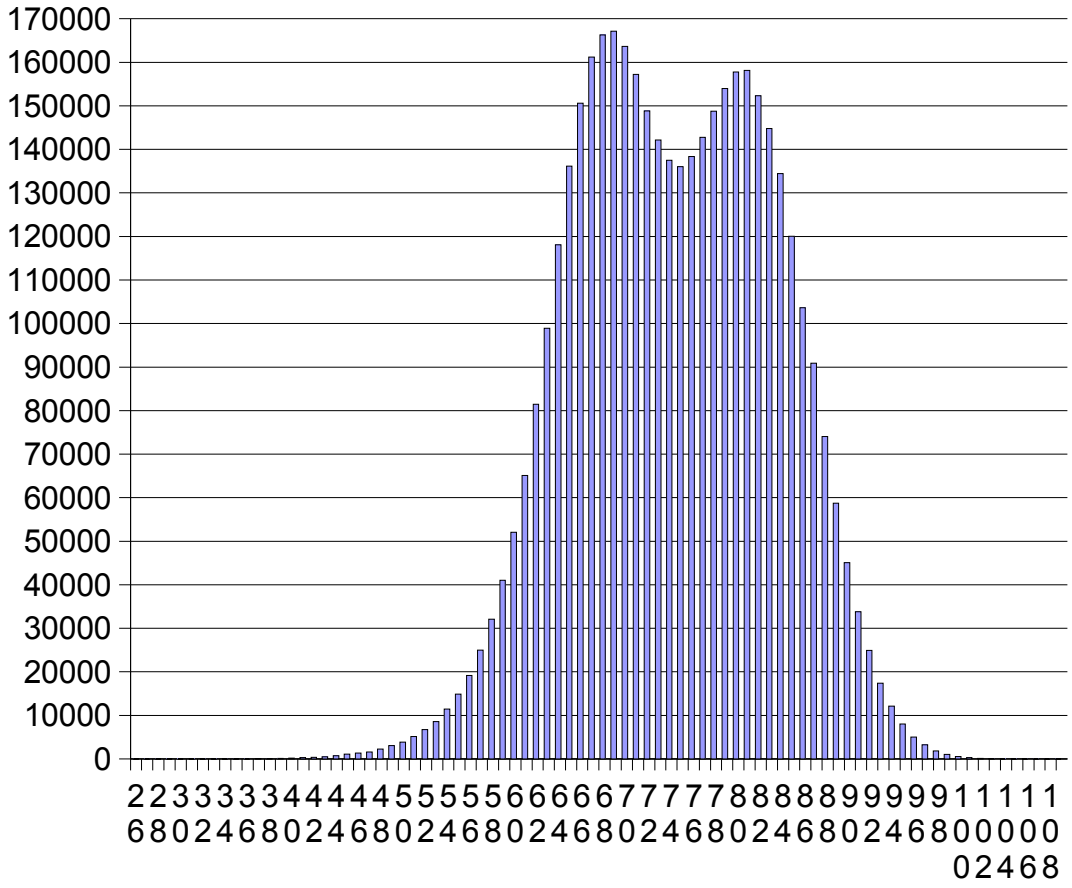


Figure 2

Number of Words at Each Weight

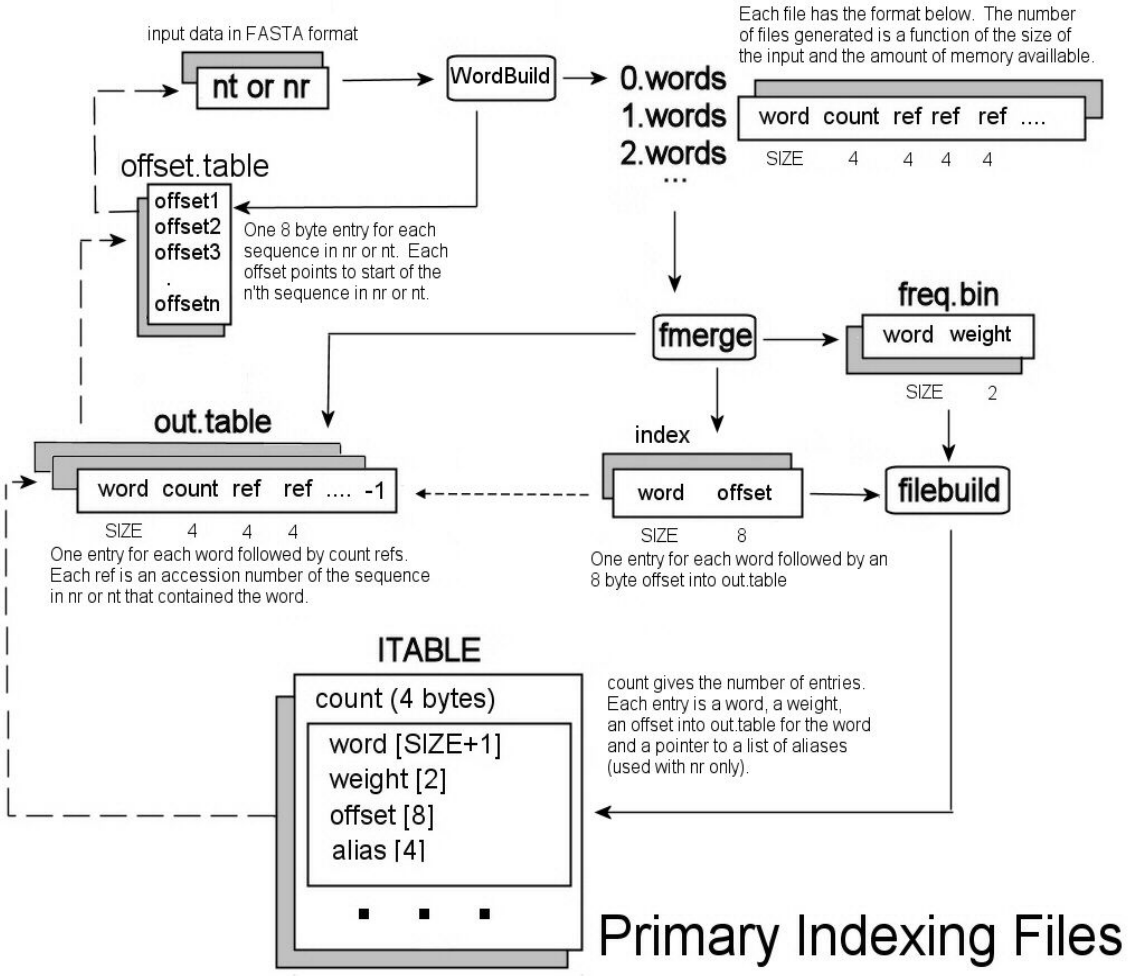


Figure 3

Indexing Procedures

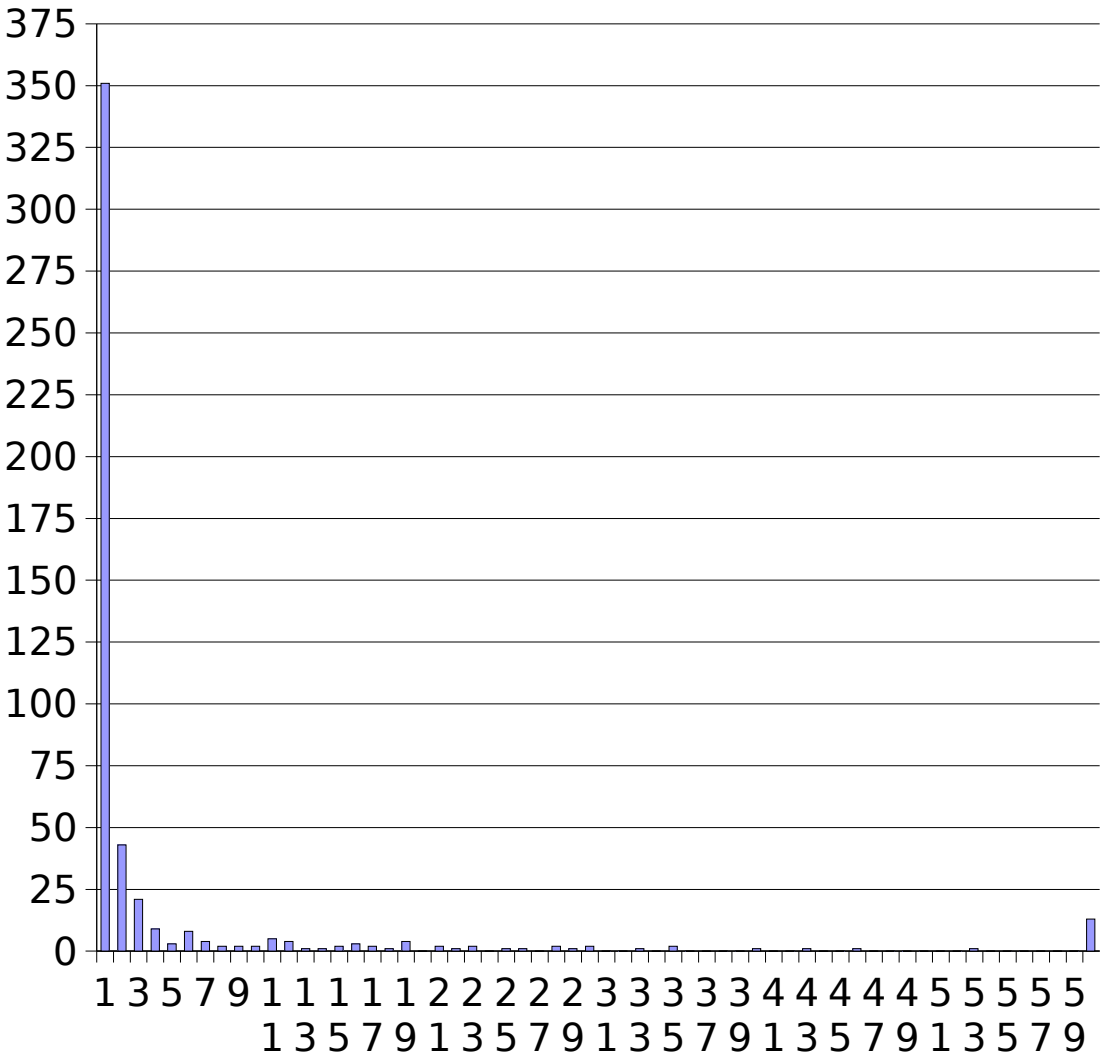


Figure 4

Retrieval Using Weighted 65-120 Index

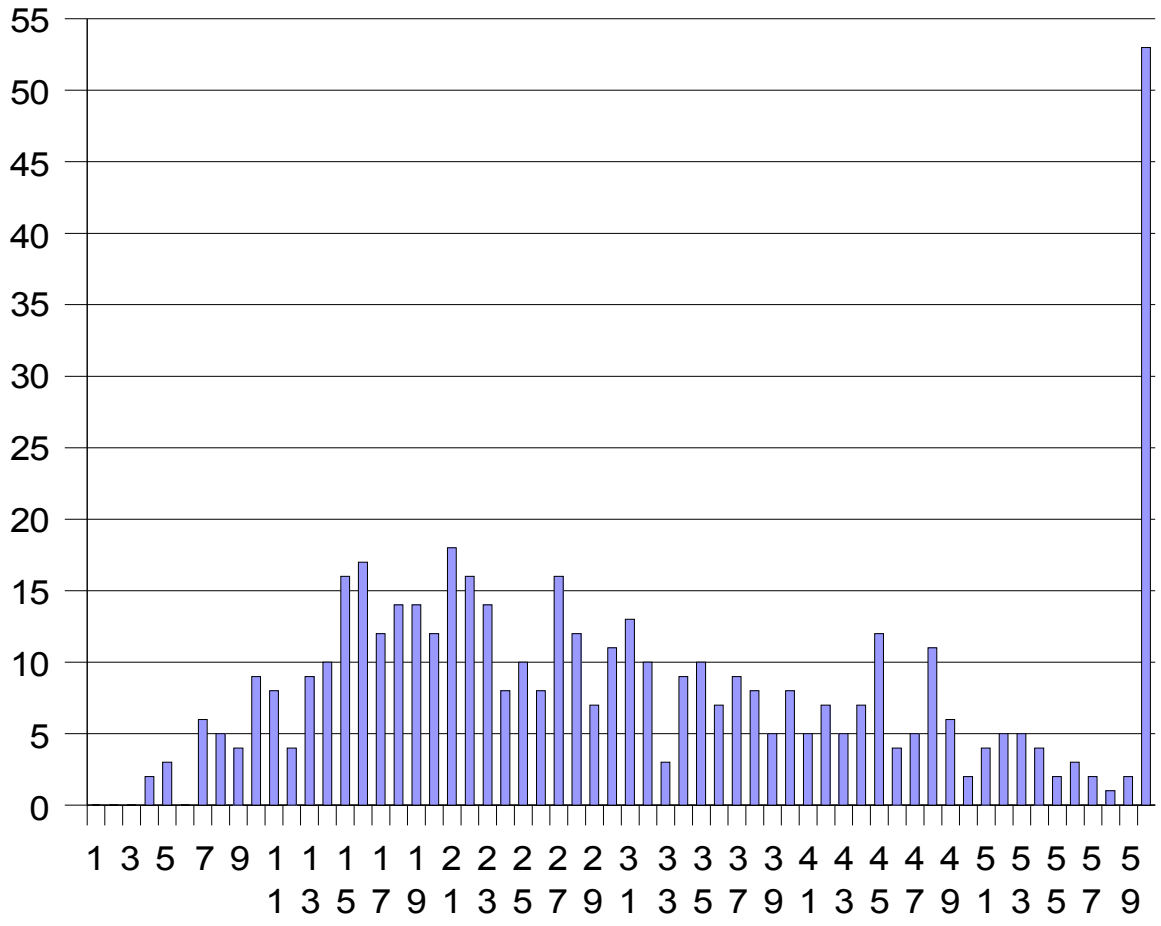


Figure 5

Retrieval Using Unweighted Index (all words)

| Method | Average Time | Average Rank | Median Rank | Found | Not Found |
|---|---|---|---|---|---|
| Weights 40-50 | 17.30 | 7.31 | 1 | 500 | 0 |
| Weights 50-60 | 17.13 | 8.04 | 1 | 495 | 5 |
| Weights 60-70 | 17.23 | 6.46 | 1 | 497 | 3 |
| Weights 70-80 | 17.23 | 5.69 | 1 | 499 | 1 |
| Weights 80-90 | 17.38 | 8.25 | 1 | 499 | 1 |
| Weights 90-100 | 17.50 | 5.06 | 1 | 497 | 3 |
| | | | | | |
| Weights 40-65 | 17.28 | 7.31 | 1 | 500 | 0 |
| Weights 55-70 | 17.25 | 8.04 | 1 | 495 | 1 |
| Weights 70-85 | 17.19 | 6.47 | 1 | 497 | 3 |
| Weights 85-100 | 17.26 | 5.69 | 1 | 499 | 1 |
| | | | | | |
| Weights 40-60 | 17.26 | 7.31 | 1 | 500 | 0 |
| Weights 60-80 | 17.27 | 8.04 | 1 | 495 | 5 |
| Weights 80-100 | 17.34 | 6.47 | 1 | 497 | 3 |
| | | | | | |
| Weights 75-84 | 17.37 | 7.31 | 1 | 500 | 0 |
| Weighted 65-120 | 18.34 | 7.31 | 1 | 500 | 0 |
| Weighted 0-120 | 19.50 | 7.67 | 1 | 499 | 1 |
| BLAST | 41.74 | 6.12 | 1 | 499 | 1 |
| Unweighted | 23.6 | 38.27 | 29 | 492 | 8 |

Table 1

Comparison of Methods for 500 Randomized Sequence Queries