NASA Indexing Benchmarks: Evaluating Text Search Engines

Sandra L. Esler
<s.l.esler@larc.nasa.gov>

Michael L. Nelson*
<m.l.nelson@larc.nasa.gov>

NASA Langley Research Center
Mail Stop 185
Hampton, VA 23681-0001
1-757-864-8511

Journal of Network and Computer Applications, 20(4), October, 1997, pp. 339-353.

* correspondence author

NASA Indexing Benchmarks: Evaluating Text Search Engines

Sandra L. Esler
Michael L. Nelson
NASA Langley Research Center, Hampton, Virginia

The current proliferation of on-line information resources underscores the requirement for the ability to index collections of information and search and retrieve them in a convenient manner. This study develops criteria for analytically comparing the index and search engines and presents results for a number of freely available search engines.

A product of this research is a toolkit capable of automatically indexing, searching, and extracting performance statistics from each of the focused search engines. This toolkit is highly configurable and has the ability to run these benchmark tests against other engines as well.

Results demonstrate that the tested search engines can be grouped into two levels. Level one engines are efficient on small to medium sized data collections, but show weaknesses when used for collections 100MB or larger. Level two search engines are recommended for data collections up to and beyond 100MB.

1.0 Introduction and Background

With the increased utilization of on-line information services, particularly the World Wide Web and related Internet technologies, information collections have grown beyond the point where sequential listings of information are feasible. Users expect on-line searchable archives, or *digital libraries*, to be available for any sizable collection of information. To accommodate these needs, it is necessary to implement efficient text search engines within sites containing a measurable amount of information.

A search engine consists of two principle components: a gatherer and an inference engine [1]. The gatherer operates by periodically retrieving meaningful data from the document collection. This data is then extracted, or filtered (i.e., low semantic stopwords like “and”, “the”, and “or” are usually ignored) to be transformed into an index that serves as the knowledge base for the inference engine. The inference engine interprets the user’s request and produces a summary of matches relevant to the search term discovered within the knowledge base. Figure 1 depicts the general structure and flow of operations within a search engine.

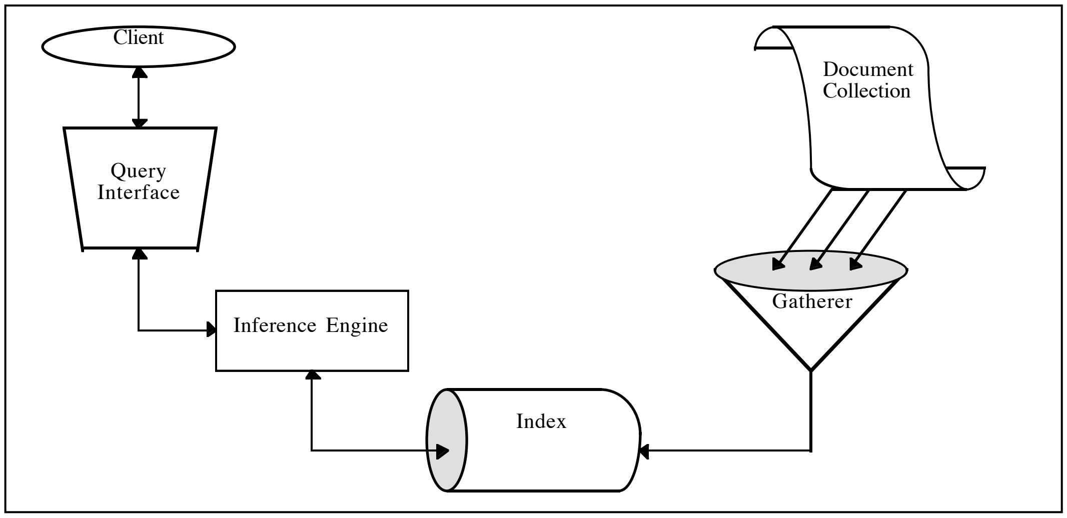

Figure 1. The general structure and flow of a search engine.

There is a direct correlation between disk usage and processing speed. Packages maintaining relatively small indexes tend to process requests slowly, whereas packages that maintain large indexes produce faster results [2]. The most significant difference among search engines is the index structure. Commonly chosen is an inverted index which arranges every occurrence of each word in a table indexed by that word using a tree or hash table structure [3]. This creates a large index (typically 50% - 300% the size of the data collection), but yields fast queries because the tables searched are limited by the query term [3]. Another potential drawback of an inverted index is that it is typically unable to perform approximate matching or wildcard searching. Alternative structures can generate indexes as small as two percent the size of the data collection, but render query times that are typically slower than inverted indexes [3]. It is critical to consider available local resources, specifically disk space and CPU time, when choosing a package.

In this paper, we present the NASA Indexing Benchmarks (NIB) , a simple series of sample data collections and timing utilities to extract performance characteristics of various systems for comparison against digital library requirements. The packages chosen for this initial testing include freeWAIS-sf, FFW, Isite, Harvest, and SWISH. All are exclusively freely available (i.e., no commercial packages are tested) and are not SQL based. However, the test data and routines are general enough to be used with any system. We do consider non-text search engines. NIB differs from the TREC benchmarks [4] in that NIB focuses primarily on space and time measurements, where TREC focuses on correctness and relevance issues.

The intended audience for this paper is the information provider (perhaps "webmaster") that maintains a large information architecture, but must consider providing a subset of the information as a searchable archive. The objective of this paper is report the strengths and weaknesses of common search engines, demonstrating which engines should be used in popular environments.

2.0 NIB Test Criteria

We have developed a standard test criterion to evaluate the performance profile of a search engine. This consists of a series of metrics and timing utilities which predict a search engine's performance in a standard environment. It is intended to enable webmasters and system administrators to rate the ability of a search engine's success given their system architecture and users' needs.

2.1 Search Options and their Evaluation

Search options allow requests to be more limited and precise. The webmaster or system administrator must decide which options are most applicable to their system's structure and their users' needs. Specific options studied are as follows:

(a) Boolean operators include conjunctive words such as "AND" which requires that all combined terms are found within the document and "OR" which simply requires that at least one of the terms is present. Boolean operators are useful, but can be too inclusive ("OR") or too restrictive ("AND") when searching for common terms within large document collections [5].

(b) Proximity searches are an even more restrictive form of "AND" operators because they require that the terms occur within a given distance. This can be especially useful when searching for Proper names such at "George Washington". This helps to prevent the false positives of a document about "George Smith who resides in Washington".

(c) Applicability scores predict the usefulness of the document to the user [6]. This is designed to save the client from retrieving irrelevant documents (i.e., large documents that make one insignificant reference to the query term).

(d) "At least" searches can be useful, especially when applicability scores are not returned, because clients can set the specific minimum number of term occurrences required within a document.

(e) Wildcard searches allow truncated words to be used as search terms. This is especially necessary if the extraction technique does not implement stemming. Stemming allows the client to discover different versions of a word. It is implemented by truncating words to their common root (i.e., *computer* and *computing* would both be truncated to *comput*) at the time of indexing [7]. Wildcard searches also allow matches that would otherwise not be discovered due to spelling errors. The importance of a search engine that can bypass spelling errors has increased due to the popularity of document scanning. Current optical character recognition devices possess an error rate of 1-5% [3].

(f) Field queries allow specific divisions within the document to be searched. For instance, a user searching technical reports may want to restrict the search terms to the title or author fields.

(g) Approximate matching can be used to locate words that "look like" or "sound like" the search term [8]. This is also a powerful tool for discovering data despite spelling errors.

(h) Relevance feedback allows queries to automatically be reconstructed, based on results that the user determines to be "similar" to the over-all target [6]. Terms in results marked as relevant are emphasized whereas terms in irrelevant results are de-emphasized. For instance, when searching for the topic of "Biology" the first two matches may be about 1) "People in Biology" and 2) "Events in Biology". If you select the first match, then your next query will be automatically customized to include keywords related to "People in Biology" instead of the previously generic term "Biology".

2.1.2 Time for a query

The time for a query is dependent on the index size, the specific query task, and other factors. The indexes are extracted from data collections ranging from 10K to 100MB containing from 1 to 15,000 unique files. Sixteen query formats were chosen from common cases seen with the NASA Langley Research Center Intranet Search Engine. Because of minute variation within results, each test was run five times across each data collection (see table 1 for query formats). The number of query term occurrences plays a tremendous role in the processing speed, so tests are performed on both common (terms that occur in at least 10% of the data collection) and uncommon (terms that occur in less than 10% of the data collection) query sets (see section 5.0 for specific queries tested). The mean is then computed and used to indicate the query performance of a search engine.

term
term AND term
term OR term
term AND term AND term
term OR term OR term
term AND term OR term
wildcard search
misspelled word

Table 1. Specific Query Combinations Chosen.

2.1.3 Result Clarity

Some packages provide simple file names as results while others also provide detailed summaries; table 2 contains examples of query returns gathered from each system's default format. More descriptive results, such as what Harvest and SWISH return clarify the discovered data, thus ensuring that the client spends minimum time viewing irrelevant data.

Engine	Query Result for: Jupiter AND Saturn
freeWAIS-sf	Score: 87, lines: 22 'saturn.html /local/www/documents/NIB/TEST/DATABASE/1K'

ffw	Url= /local/www/documents/NIB/TEST/DATABASE/1K/saturn.html |
	Title=Saturn and Satellites | Date=1970-1-1-0-0-0

Harvest	http URL: saturn.html host: acsb-www.larc.nasa.gov path: /NIB/TEST/DATABASE/1K/saturn.html Description: Saturn and Satellites Matched Line: url# @FILE {http://acsb-www.larc.nasa.gov/NIB/TEST/DATABASE/1K/saturn.html} Matched Line: title#title{21}: Saturn and Satellites Matched Line: headings#headings {21}: Saturn and Satellites Matched Line: body#SATURN Beyond Jupiter comes Saturn, a giant famous for its Matched Line: body#Saturn is similar to Jupiter, but 16 percent smaller Matched Line: body#the only moon in the solar system with a thick atmosphere. Saturn was Matched Line: description#description{21}: Saturn and Satellites
Isite	Score=100 File= /local/www/documents/NIB/TEST/DATABASE/1K/saturn.html

SWISH	#SWISH format 1.1 search words: jupiter AND saturn
	#Name: Index of a 1K Database
	#Saved as: 1K.swish
	#Counts: 38 words, 1 files
	#Indexed on: 03/06/96 07:47:11 EDT
	#Description: Index of a 1K Database for NIB Timing Tests
	#Pointer: http://acsb-www.larc.nasa.gov/NIB/TEST/CGI/swish.cgi
	#Maintained by: Sandra L. Esler (s.l.esler@larc.nasa.gov)
	Score=1000 /local/www/documents/NIB/TEST/DATABASE/1K/saturn.html "Saturn and Satellites" 855

Table 2. Example of discovered data results.

A subjective four point rating value was defined and used in our tests.

Results Attribute	*Clarity Value*
file name	1
applicability score	1
document location	1
summary	1

Table 3. Four Point Rating Value for Results Clarity.

2.2 Gathering Ability

The gathering ability phase is intended to demonstrate the efficiency of the index development. This is intended to predict which search engine performs best in given environments.

2.2.1 Index Size

Index size alone does not represent a measurable strength or weakness, it is affected by the method of data extraction used to gather information from the document collection. Index size is one of the most important roles in the performance of a search engine. Within any language there are approximately 10^6 to 10^7 frequently queried search terms composed of technical words and names [2]. An index which represents only a small percentage of all text (i.e., descriptive terms like adjectives and adverbs are usually not used as query terms) thus grows much slower than the entire body of the document collection. If the collection is queried for a small number of search terms, then using a full text (inverted) index would waste a tremendous amount of disk space because it indexes every word present in the text. However, if the document collection is queried for non-standard words then it is necessary to use a full text index, otherwise the prediction algorithms will filter out the words before they can be placed within the knowledge base. Thus the system administrator or webmaster must first determine both the document collection content and the intended user's purpose before index size result can be correctly interpreted.

Figure 2 demonstrates differences among various indexing approaches. SWISH indexes only select text, eliminating other words found within the document that are not defined within its dictionary (such as the html tags body, title, src, www, etc.). On the other hand, freeWAIS conducts a full index giving it the capability to search structured fields within document such as "title=Saturn and Satellites". The summary format used within Harvest generally produces the most descriptive results because of the sentence structure that is maintained along with the fields extracted (such as title, keywords, images, headings, and body).

SWISH: select text index	freeWAIS-sf: full text index	Harvest: summary format
# SWISH format 1.1 # Name: Index of a 1K Database # Saved as: 1K.swish # Counts: 38 words, 1 files # Indexed on: 03/06/96 08:47:11 EDT # Description: This is a full index of a 1K Database. # Pointer: http://acsb-www/harvest/TEST/CGI/shish.cgi # Maintained by: Sandra L. Esler (s.l.esler@larc.nasa.gov) adventure: atmosphere: brilliant: comes: copyright: diameter: encyclopedia: especially: famous: giant: jupiter: knowledge: large: massive: moon: moons: noteworthy: november: percent: pioneer: planet: reserved: rights: rings: satellites: saturn: search: similar: smaller: solar: spacecraft: system: thick: third: titan: visited:voyager: /local/www/documents/harvest/TEST/DATABASE/1K/saturn.html "Saturn and Satellites" 855	10 128 16 1980 1996 1k 48 adventure atmosphere bgcolor body border br brilliant cgi comes copy copyright database diameter documents encyclopedia especially famous ffffff giant gif h1 harvest head height hr href html images img inc index jupiter ka knowledge large local logo massive moon noteworthy november percent pioneer planet reserved rights rings satellites saturn search similar small smaller solar spacecraft src system test thick third titan title visited voyager width www	title{21}: Saturn and Satellites keywords{24}: encyclopedia search images{19}: /images/logo_ka.gif headings{21}: Saturn and Satellites body{551}: bgcolor="#ffffff"> [Search the Encyclopedia] November 1980 SATURN Beyond Jupiter comes Saturn, a giant planet famous for its brilliant rings. Saturn is similar to Jupiter, but 16 percent smallerin diameter and one-third as massive. It has many moons, but only one of them, Titan, is large. Titan is especially noteworthy because it's the only moon in the solar system with a thick atmosphere. Saturn was first visited by the Pioneer 10 spacecraft, and then by Voyager 1 and 2. Copyright 1996 Knowledge Adventure, Inc. All Rights Reserved.

Figure 2. Examples of select text, full index, and summary index formats.

Index size is gathered from the following data collection sizes: 10K, 100K, 1MB, 10MB, and 100MB. Each statistic is then normalized in terms of α data collection kilobytes and ϕ generated index kilobytes to demonstrate the growth pattern of each search engine. Normalization N is acquired by

$$N = \left[\left(\frac{\phi}{\alpha} \right) \bullet 100 \right]. \tag{1}$$

For instance, if the index size is 5MB and the data collection size is 100MB, then the index is 5% that of the data collection.

2.2.2 Time to Index

Index time is derived by ε time required to extract data from a collection of size α and to reformat it into the knowledge base. This is then normalized so that index size does not skew the results. Statistics are gathered from data collections of size: 10K, 100K, 1MB, and 100MB. Normalized speed S is obtained by:

$$S = \left(\frac{\alpha}{\varepsilon} \right). \tag{2}$$

If the real time required to index a 100MB data collection is 120 minutes, then each minute is capable of indexing approximately 0.83MBs. This is not intended to imply linear growth in index speed, but rather to normalize findings across search engines.

2.2.3 Locality Constraints

Locality of the index is important if it is desired to maintain the index independent from the data collection. This is necessary if the collection is very large, or if the search engine experiences high traffic. Locality constraints are noted for each search engine by reporting if they have remote indexing capability.

3.0 Search Engines Tested

The NIB test was performed on: freeWAIS-sf, FFW, Isite, Harvest, and SWISH. All packages are capable of compiling on most UNIX platforms and require a C++ compiler, g++ library, and Perl 4.0 (or better).

3.1 FreeWAIS-sf Version 2.0.65

http://ls6-www.informatik.uni-dortmund.de/ir/projects/freeWAIS-sf

FreeWAIS-sf was created at the University of Dortmund, Germany [8]. It is an extension of freeWAIS developed by Clearinghouse for Networked Information Discovery and Retrieval. The original WAIS prototype was release in 1988, and thus did not have internal support for HTML documents [9]. FreeWAIS-sf adds structured fields and a mechanism for custom indexing of arbitrary file formats, including HTML and non-ASCII documents.

3.2 FFW Version 2.3

http://www.nta.no/produkter/ffw/ffw.html

Freetext search For Web (FFW) was developed by the MultiTorg Project at Telenor Research, Norway [10]. The input parser is specifically designed for HTML documents, but can be modified for other formats. Norwegian and English versions are available.

3.3 Harvest / Glimpse Version 1.4

http://harvest.cs.colorado.edu

Harvest was developed at the University of Colorado. It is comprised of subsystems capable of gathering, extracting, reformatting, caching, replicating, and querying unstructured information on a web server [11]. Harvest uses GLobal IMPlicit SEarch (GLIMPSE), developed by the University of Arizona, as the default inference engine. It is a hybrid of grep-like and index based structures. Other engines such as Nebula and WAIS can also be used [12].

3.4 Isite Version 1.04

http://www.cnidr.org/ir/isite.html

Isite was created by Clearinghouse for Networked Information Discovery and Retrieval and it uses Isearch as its Index/Search Subsystem.

3.5 SWISH Version 1.1.1

http://www.eit.com/software/swish/swish.html

Simple Web Indexing System for Humans (SWISH) was developed at Enterprise Integration Technologies. It was specifically created for use on web sites.

4.0 Testing Procedures

All tests were performed on a dedicated SUN, SPARCstation-5, 64Mbytes main memory, running at 85MHz on Solaris 2.4 [UNIX(R) System V Release 4.0] with Perl 5.001, gcc 2.7.2, and 40MB swap space. Table 4 contains specific queries that were tested according to table 1.

It is necessary to stress that each search engine is highly customizable. For instance, Harvest can be used in conjunction with GLIMPSE, Nebula, or WAIS. GLIMPSE also has three different performance settings ranging from low disk space and slow query processing to high disk space and fast query processing. Search engines, like any software tool, must be flexible to the user's needs and computing proficiency. Each engine was implemented with the default settings. Customizing a search engine is not a complicated task, yet it should not be required for implementation.

Query	Saturation	Format	Query Example
1	*uncommon*	term	Anguilla
2	*common*	term	government
3	*uncommon*	term AND term	lobster AND limestone
4	*common*	term AND term	government AND resource
5	*uncommon*	term OR term	lobster OR limestone
6	*common*	term OR term	government OR resource
7	*uncommon*	term AND term AND term	lobster AND limestone AND tourism
8	*common*	term AND term AND term	government AND resource AND growth
9	*uncommon*	term OR term OR term	lobster OR limestone OR tourism
10	*common*	term OR term OR term	government OR resource OR growth
11	*uncommon*	term AND term OR term	lobster AND limestone OR tourism
12	*common*	term AND term OR term	government AND resource OR growth
13	*uncommon*	wildcard search	Angui*
14	*common*	wildcard search	govern*
15	*uncommon*	misspelled word	Anguila
16	*common*	misspelled word	govormant

Table 4. Specific Queries Tested.

5.0 Test Results

Table 5 and figure 3 report the results generated from the 1MB data collection queries across each of the search engines. Other data collection sizes show similar relative results between the various packages. It is important to note that Harvest is the only search engine capable of discovering misspelled words (see query task 15 and 16). A second important finding is that query results were greatly influenced by the data extraction technique (specifically stemming). FFW and SWISH do not implement stemming and therefore cannot match alternative versions of query terms. This is demonstrated in query tasks four and eight when the term 'resource' is not discovered within documents containing the word 'resources'.

There is a direct correlation between search complexity and search speed within the tested engines; the packages that offer the most options also yield the slowest query times. This is demonstrated with the comparison of Harvest and SWISH results. Harvest offers the most options (i.e., remote indexing capability, stemming, discovery of misspelled words, etc.) and returns the slowest query times, whereas SWISH returns rapid query speeds and offers the least options (no remote indexing, no stemming, and no technique to resolve misspelled words). These statistics do not directly rank one engine over another, but confirm the importance of analyzing performance statistics carefully before choosing and implementing a search engine.

There are environments in which each search engine is best suited. For instance, if the data collection was generated by scanning documents, then due to optical character recognition's high error rate it is strongly recommended to implement an engine that resolves misspelled/misscanned words. On the other hand, if the data collection is one that the users will be familiar with, then the strength of fast query speed may carry a heavier weight than stemming or discovery of misspelled words.

Task	freeWAIS-sf	FFW	Harvest	Isite	SWISH
1	0.304 sec	0.120 sec	0.400 sec	0.180 sec	0.022 sec
2	0.403 sec	0.203 sec	0.500 sec	0.302 sec	0.042 sec
3	0.303 sec	0.143 sec	0.521 sec	0.381 sec	0.003 sec
4	0.303 sec	0.200 sec	1.900 sec	0.580 sec	0.061 sec
5	0.321 sec	0.141 sec	0.442 sec	0.383 sec	0.120 sec
6	0.423 sec	0.203 sec	0.583 sec	0.563 sec	0.100 sec
7	0.343 sec	0.181 sec	0.543 sec	0.583 sec	0.080 sec
8	0.321 sec	0.182 sec	2.041 sec	0.822 sec	0.061 sec
9	0.303 sec	0.200 sec	0.504 sec	0.621 sec	0.101 sec
10	0.344 sec	0.261 sec	0.662 sec	0.800 sec	0.083 sec
11	0.303 sec	0.181 sec	0.584 sec	0.603 sec	0.103 sec
12	0.344 sec	0.163 sec	2.861 sec	0.822 sec	0.082 sec
13	0.344 sec	0.163 sec	0.402 sec	0.201 sec	0.100 sec
14	0.401 sec	0.163 sec	0.522 sec	0.304 sec	0.202 sec
15	No Match	No Match	0.701 sec	No Match	No Match
16	No Match	No Match	0.641 sec	No Match	No Match

Table 5. 1MB Query Speed Raw Data Results.

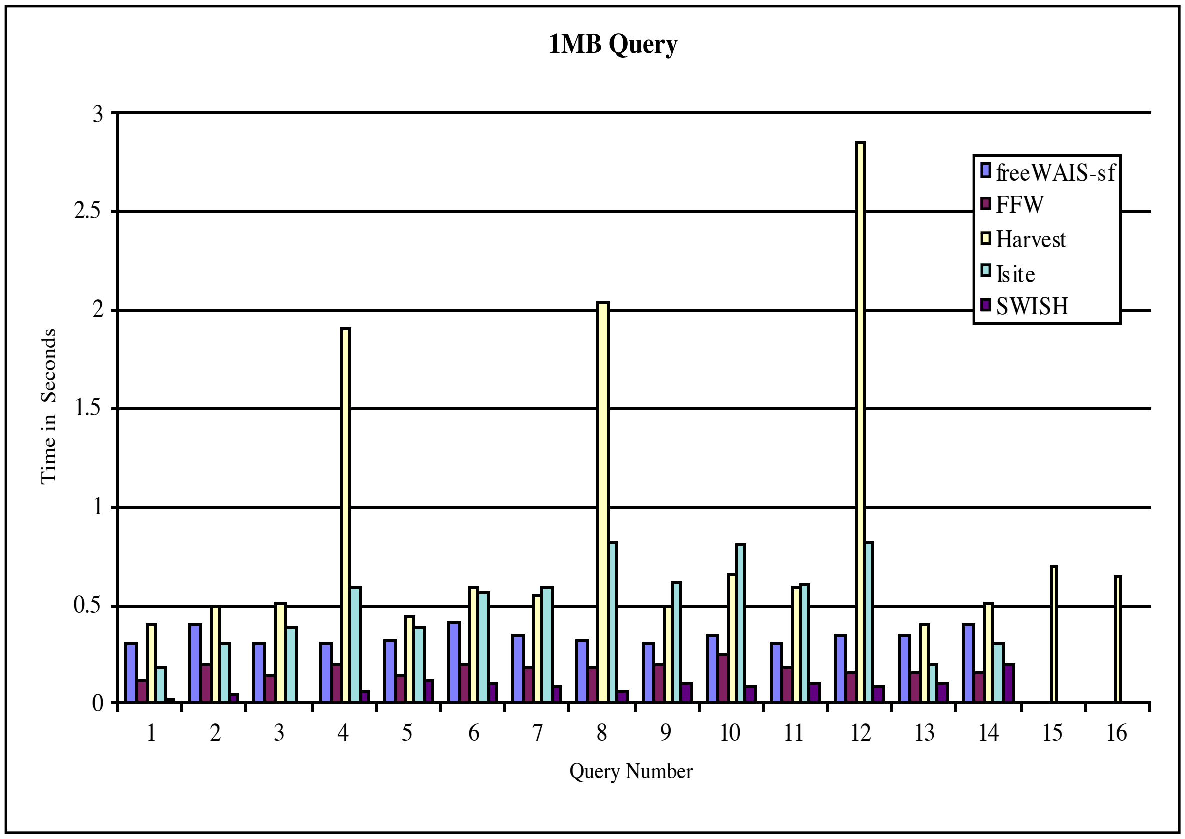

Figure 3. 1MB Query Speed Graph.

Results demonstrate that the tested search engines can be grouped into two levels. Level one engines are efficient on small to medium sized data collections, but show weakness when used for collections of

100MB or larger. Specific engines tested that fall into this group are: FFW, Isite, and SWISH. Indexing algorithms used in these engines require a tremendous amount of swap space (over 40K) when collecting from large databases. This large requirement causes the system to core dump due to virtual memory limits. If desired, the system administrator or webmaster can get around this by breaking large data collections into smaller modules. Individual modules can be indexed separately, and then merged together. However, this is not recommended because more powerful engines are available that can easily handle large collections.

Level two search engines are recommended for data collection up to and beyond 100MB. Examples of level two search engines are freeWAIS-sf and Harvest. These efficiently manage memory needed to index large collections.

5.1 Index Size Results

Table 6 and figure 4 contain statistics reflecting the size of indexes generated by each search engine using each data collection from 10K to 100MB. The general pattern is a gradual increase in efficiency once past the original 10K collection which is skewed by overhead.

Engine	10K	100K	1MB	10MB	100MB
freeWAIS-sf	24K	176K	768K	7080K	56000K
FFW	8K	40K	176K	664K	core dump
Harvest	5K	29K	168K	736K	2904K
Isite	4K	40K	344K	2904K	core dump
SWISH	8K	56K	304K	2816K	core dump

Table 6. Index Size Results.

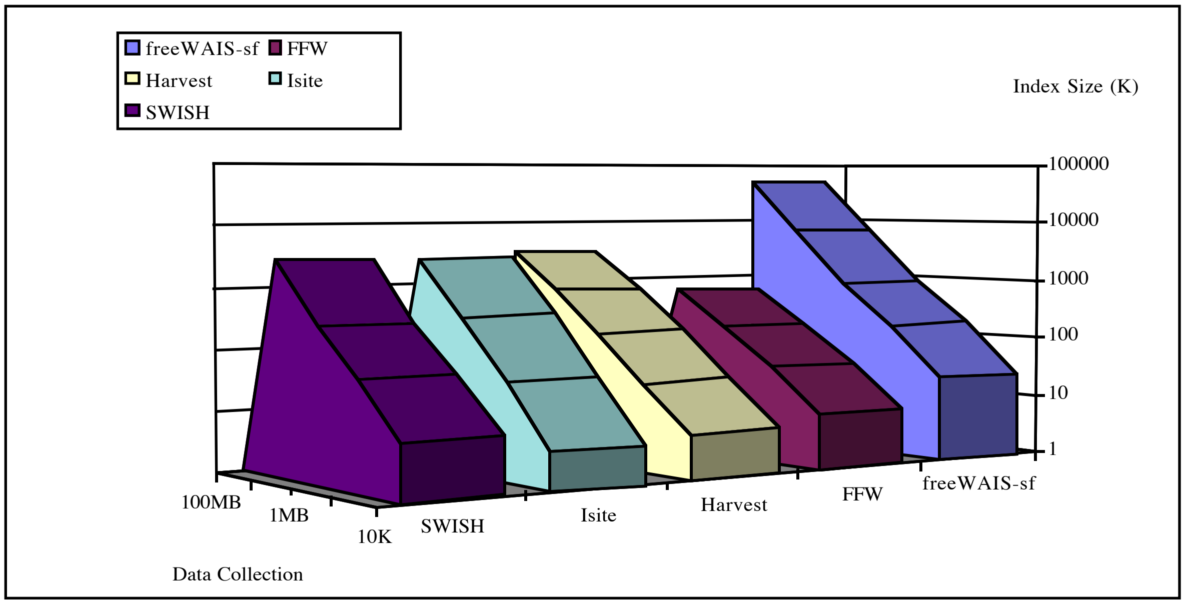

Figure 4. Index Size Results

5.2 Index Speed Results

Table 7 and figure 5 display the time to index values.

Engine	10K	100K	1MB	10MB	100MB
freeWAIS-sf	0:00:02.20 sec	0:00:06.10 sec	0:00:25.90 sec	0:04:44.40 sec	1:03:06.24 sec
FFW	0:00:01.30 sec	0:00:02.60 sec	0:00:22.80 sec	0:04:19.40 sec	core dump
Harvest	0:00:03.30 sec	0:00:04.10 sec	0:00:10.70 sec	0:01:35.20 sec	0:16:33.55 sec
Isite	0:00:01.10 sec	0:00:01.90 sec	0:00:12.30 sec	0:03:27.21 sec	core dump
SWISH	0:00:00.30 sec	0:00:02.80 sec	0:00:26.50 sec	0:05:50.10 sec	core dump

Table 7. Index Speed Results.

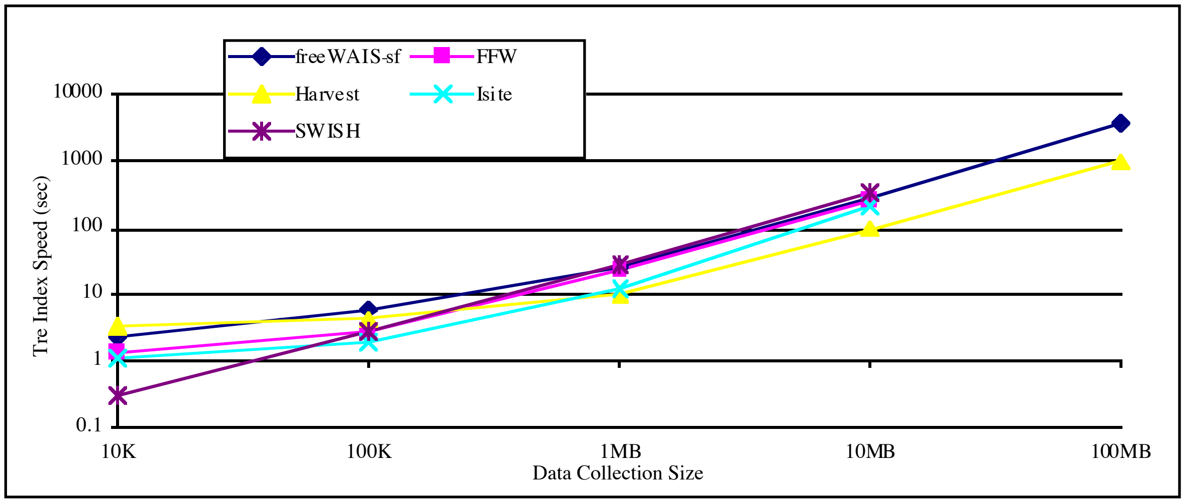

Figure 5. Index Speed.

5.3 Additional Extracted Characteristics

Table 8 contains a subjective score given to each search engine based on their results clarity (i.e., see table two). As previously stated, each search engine can be customized to display results in a different format. The results in this study were obtained by using the default setting of each package. Table 9 contains information regarding the search engine's capability to index remote data collections. This is definitely a positive attribute, but is only required if the data collection spans more than one server. Table 10 depicts which search options each package supports. Although freeWAIS-sf is shown to implement approximate matching, it is necessary to detail the fact that it is only available in place of string queries. Also, Harvest is shown to return applicability scores. It does not return "relevance scores" like most engines, but instead it returns the number of matched lines within a document.

Attribute	freeWAIS-sf	FFW	Harvest	Isite	SWISH
Title		1	1		1
Relevance	1			1	1
Location	1	1	1	1	1
Summary			1		
Total	2	2	3	2	3

Table 8. Results Clarity.

freeWAIS-sf	FFW	Harvest	Isite	SWISH
✓	✓	✓	✓	

Table 9. Remote Indexing Capability.

Search Options	freeWAIS-sf	FFW	Harvest	Isite	SWISH
Boolean Operators	1	1	1	1	1
Proximity Searches	1				
Applicability Scores	1		1	1	
"At least" Searches	1				
Wildcard Searches	1	1	1	1	1
Field Queries	1		1	1	
Approximate Matching	1		1		
Relevance Feedback	1				
Total	8	2	5	4	2

Table 10. Available Search Options.

6.0 NIB Toolkit

The NIB Toolkit is composed of two components that are capable of automating the entire index, search, and analysis task necessary for these tests. It can be obtained by contacting the authors.

The toolkit works by running the queries in section 5.0 over the included test data sets consisting of 10K, 100K, 1MB, 10MB, and 100MB databases. As the tests are run, statistical information is returned to the user. Figures 6 and 7 contain sample output that is generated by this system.

```
Indexing started at:    Wed Jun 26 08:09:50 EDT 1996
Indexing finished at:   Wed Jun 26 08:10:16 EDT 1996
Directory Indexed:      local/www/documents/NIB/TEST/DATABASE/1MB   (1024KB)
Total Collection:       1024K
Generated Index:        768K
Normalized Index:       75.00%
Index Speed:            25.9 seconds
Normalized Speed:       39.54 K/second
```

Figure 6. Statistical output generated by the indexing component of NIB.

```
Query: 5 'lobster or limestone'
Number of Matches: 2
Match: 1: Score:  375, lines: 292 'av.html   /local/www/documents/NIB/TEST/DATABASE/1MB/'
Match: 2: Score:   37, lines: 423 'al.html   /local/www/documents/NIB/TEST/DATABASE/1MB/'
Query Average: 0.32 seconds
Highest: 0.4 seconds
Lowest: 0.3 seconds
```

Figure 7. Statistical output generated by the querying component of NIB.

The toolkit can be customized for additional search engine systems by answering a few preliminary questions. This design is flexible and should work with most search engines, and is distributed in Perl for flexibility if additional modification is needed.

7.0 Conclusions

Due to the large growth of on-line information, it is necessary to accommodate the user with keyword searching capability via a search engine. NIB is a system of analytical measurements designed to extract performance characteristics from any given search engine. This research developed a toolkit capable of extracting these performance statistics automatically. It was specifically performed on the following search engines (but is easily customizable for other packages): freeWAIS-sf, FFW, Harvest, Isite, and SWISH. Results demonstrated that the search engines can be grouped into two levels. Level one engines, including FFW, Isite, and SWISH, do not function correctly when indexing data collections of 100MB or greater. They require a large amount of swap space (greater than 40MB) which most average sized servers are not capable of providing. However, level two search engines (including freeWAIS-sf and Harvest) are capable of functioning efficiently for these large data collections.

There is a direct correlation between search complexity and search speed within the tested engines; the packages that offer the most options also yield the slowest query times. This is evident when SWISH and Harvest are compared:

- SWISH offers the least options (i.e., no remote indexing, no stemming, and no technique to resolve misspelled words) but it returns results at an amazing speed.
- Harvest offers the most options (i.e., remote indexing capability, stemming, discovery of misspelled words, etc.) but has the slowest query times which average roughly .40 seconds than each SWISH query.

The NIB performance tests are not intended to rank one search engine over another, but rather they provide a means for determining which package is best suited for particular requirements. The system administer or webmaster must view the extracted performance characteristics carefully before implementing any search engine.

References

[1] Darren R. Hardy and Michael F. Schwartz 1996. Customized Information Extraction as a Basis for Resource Discovery. *ACM Transactions of Computer Systems,* 14(2), 171-199. *ftp://ftp.cs.colorado.edu/pub/cs/techreports/schwartz/Essence.Jour.ps.Z*

[2] C. Mic Bowman, Peter B. Danzig, Udi Manber, and Michael F. Schwartz 1994. Scaleable Internet Resource Discovery: Research Problems and Approaches. *Communications of the ACM,* 37(8), 98-107. *ftp://ftp.cs.colorado.edu/pub/cs/techreports/schwartz/RD.ResearchProblems.Jour.ps.Z*

[3] Udi Manber and Sun Wu 1994. GLIMPSE: A Tool to Search Through Entire File Systems. *Proceedings of the USENIX Winter Conference,* 23-32. *ftp://cs.arizonia.edu/reports/1993/TR93-34.ps.Z*

[4] Donna Harman 1994. Overview of the Third Text Retrieval Conference (TREC-3). *Proceedings of the Third Text Retrieval Conference (TREC-3)*, Gaithersburg, Maryland, November 2-4, 1994. *http://www-nlpir.nist.gov/TREC/trec3.papers/donnas.trec3paper.ps*

[5] Gary Marchionini and Diane Barlow 1994. A Comparison of Boolean-based Retrieval to the WAIS System for Retrieval of Aeronautical Information: Final Report. NASA CR-4569.

[6] Gerard Salton and Chris Buckley. Improving Retrieval Performance by Relevance Feedback 1993. *Journal of the American Society For Information Science*, 41(4), 388-297.

[7] Frequently Asked Questions and Answers about freeWAIS-sf. *http://www.cis.ohio-state.edu/hypertext/faq/usenet/wais-faq/freeWAIS-sf/faq.html*

[8] Ulrich Pfeifer, Norbert Fuhr, and Tung Huynh 1995. Searching Structured Documents with the Enhanced Retrieval Functionality of freeWAIS-sf and Sfgate. *Computer Networks and ISDN Systems*, 27, 1027-1036. *http://www.igd.fhg.de/www/www95/papers/47/fwsf/fwsf.html*

[9] Brewster Kahle, Harry Morris, Franklin Davis, Kevin Tine, Clare Hart, and Robin Palmer 1992. Wide Area Information Servers: An Executive Information System for Unstructured Files. *Electronic Networking: Research, Applications, and Policy,* 2(1), 59-68.

[10] FFW Freetext Search For Web. *http://www.nta.no/produkter/ffw/ffw.html*

[11] C Mic Bowman, Peter Danzig, Darren R. Hardy, Udi Manber, Michael F. Schwartz, "The Harvest Information Discovery and Access System," *Proceedings of the Second International World Wide Web Conference*, Chicago, IL, October 17-20, 1994, pp. 763-772. *ftp://ftp.cs.colorado.edu/pub/cs/techreports/schwartz/Harvest.Conf.ps.Z*

[12] C. Mic Bowman, Chandra Dharap, Mrinal Baruah, Bill Camargo, and Sunil Potti 1994. A File System for Information Management. *Proceedings of the Conference on Intelligent Information Management Systems*. *ftp://ftp.cse.psu.edu/pub/bowman/doc/iims.ps.Z*