moodbanao

UNIT- V

Text Search Algorithms: Introduction, Software text search algorithms, Hardware text search systems.

Information System Evaluation: Introduction, Measures used in system evaluation, Measurement example – TREC results.

Text Search Algorithms

Three classical text retrieval techniques have been defined for organizingitems in a textual database, for rapidly identifying the relevant items and foreliminating items that do not satisfy the search. The techniques are

1) Full text scanning (streaming)
2) Word inversion
3) Multiattribute retrieval

In addition to using the indexes as a mechanism for searching text ininformation systems, streaming of text was frequently found in the systems as anadditional search mechanism. The basic concept of a text scanning system is the ability for one or moreusers to enter queries, and the text to be searched is accessed and compared to thequery terms. When all of the text has been accessed, the query is complete.

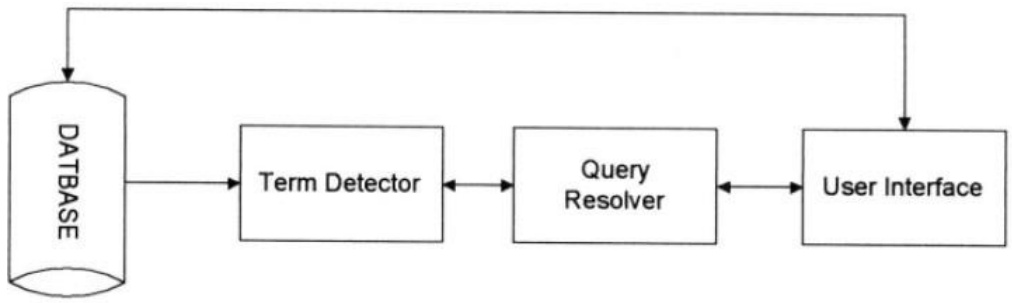


Figure 9.1 Text Streaming Architecture

The database contains thefull text of the items.The term detector is the special hardware/software that contains all of the terms being searched for and in some systems the logic betweenthe items. It will input the text and detect the existence of the search terms. It willoutput to the query resolver the detected terms to allow for final logical processingof a query against an item. The query resolver performs two functions.

It willaccept search statements from the users, extract the logic and search terms andpass the search terms to the detector. It also accepts results from the detector anddetermines which queries are satisfied by the item and possibility the weight associated with hit. The Query Resolver will

moodbanao

passinformation to the user interface that will be continually updating search status tothe user and on request retrieve any items that satisfy the user search statement. The worst casesearch for a pattern of *m* characters in a string of *n* characters is at least *n -m* + 1or a magnitude of O(*n*). Some of the original brute force methodscould require O(*n***m*) symbol comparisons. More recentimprovements have reduced the time to O(*n* + *m*).

In the case of hardware search machines, multiple parallel searchmachines (term detectors) may work against the same data stream allowing formore queries or against different data streams reducing the time to access thecomplete database. In software systems, multiple detectors may execute at thesame time.

There are two approaches to the data stream. In the first approach thecomplete database is being sent to the detector(s) functioning as a search of the
database. In the second approach random retrieved items are being passed to thedetectors. In this second case the idea is to perform an index search of the databaseand let the text streamer perform additional search logic that is not satisfied by theindex search.

Examples of limits of index searches are:

- Search for stop words
- Search for exact matches when stemming is performed
- Search for terms that contain both leading and trailing “don’t cares”
- Search for symbols that are on the inter-word symbol list (e.g., “ , ;)

The full text search function does not require anyadditional storage overhead. There is also the advantage where hits may bereturned to the user as soon as found. Typically in an index system, the completequery must be processed before any hits are determined or available. Streamingsystems also provide a very accurate estimate of current search status and time tocomplete the query. It is difficult to locate all the possible index valuesshort of searching the complete dictionary of possible terms.

Many of the hardware and software text searchers use finite stateautomata as a basis for their algorithms. A finite state automata is a logicalmachine that is composed of five elements:
I - a set of input symbols from the alphabet supported by the automata

moodbanao

S - a set of possible states

P - a set of productions that define the next state based upon the currentstate andinput symbol

S_0- a special state called the initial state

S_F- a set of one or more final states from the set **S**

9.2 Software Text Search Algorithms

In software streaming techniques, the item to be searched is read intomemory and then the algorithm is applied.

There are four major algorithms associated with software text search:

1) the brute force approach
2) Knuth-Morris-Pratt
3) Boyer-Moore, Shift-OR algorithm
4) Rabin-Karp.

Of all of the algorithms,Boyer-Moore has been the fastest requiring at most O(*n* + *m*) comparisons , Knuth-Pratt-Morris and Boyer-Moore both require O(*n*) preprocessing of search stringsThe Brute force approach is the simplest string matching algorithm. Theidea is to try and match the search string against the input text. If as soon as a mismatchis detected in the comparison process, shift the input text one position andstart the comparison process over. The expectednumber of comparisons whensearching an input text string of *n* characters for a pattern of *m* characters is

N_c=c/c-1(1-1/c^m)*(n-m+1)+O(1)

Where$N_c$is the expected number of comparisons and c is the size of the alphabetfor the text.

Knuth-Pratt-Morris(KPM) algorithm

Pattern:

a	b	c	d	a	b	C
1	2	3	4	5	6	7

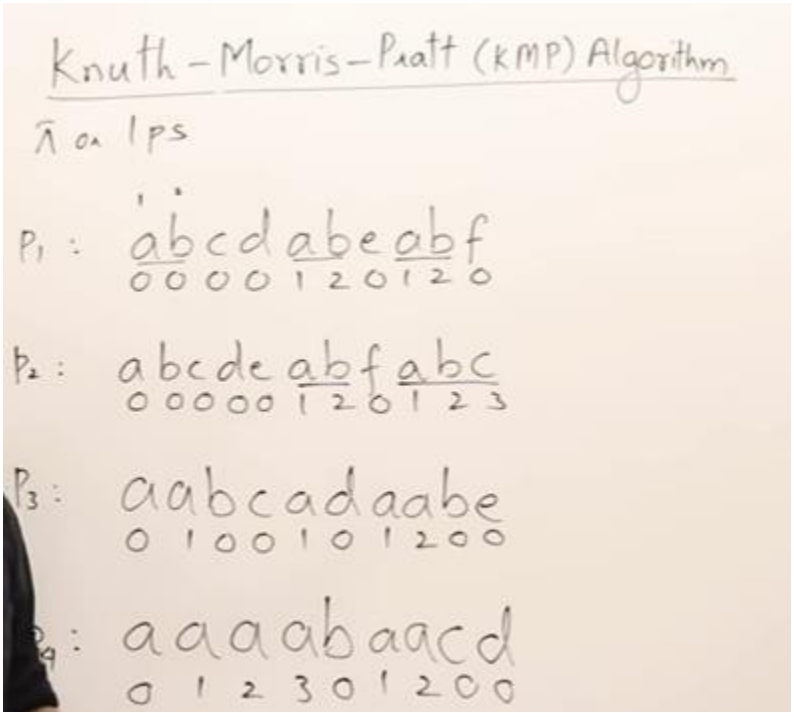
moodbanao

Now find out substrings as prefix, suffix by taking any number of characters from left to right and right to left.

Prefix: a, ab, abc, abcdetc

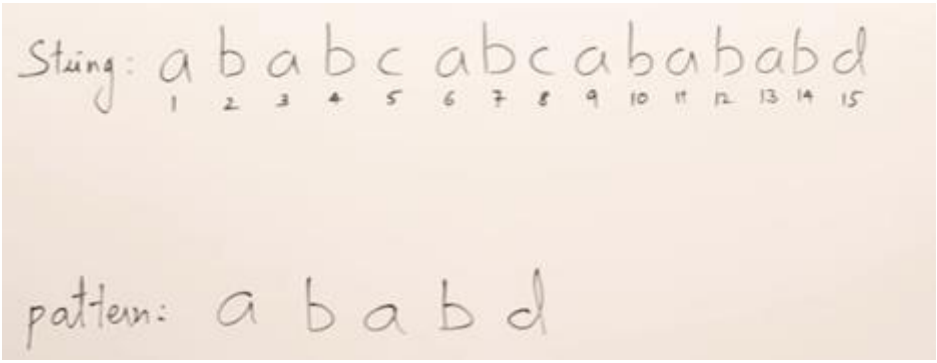
Suffix: c, bc, abc, dabcetc

From above prefix, suffix substrings we can observe a substring “abc” is there in both and also that is repeated twice in given pattern.

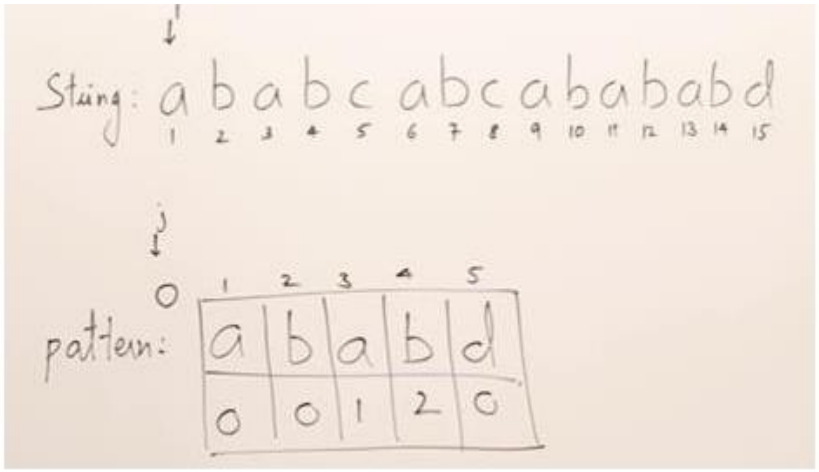


Example:

Given string and pattern is



Now construct table with repeated characters of pattern like following.



Here ‘a’ is repeated so keep index 1 below it, ‘b’ is repeated so keep index 2 below it and rest all ‘0’s. Now start search process.

Step-1: Initialize string index as I, pattern index as j. Start j from adding ‘0’ index.

Step-2: Compare string[i] and pattern [j+1]i,e, ‘a’ and ‘a’ both are matching so move both i and j to next position.

Step-3: Now compare string[i] and pattern [j+1] i.e, b and b matching so move both i, j to next position

When i=5 and j=4 string[i]=c and pattern[j+1]=d. here not matching then move j to its index location. i.e, 2. So now j position is pattern[2].

Now compare string[i]=c and pattern[j+1]=a. here not matching then move j to its index location i.e., 0. So now j position is pattern[0]. J is now on 0 position so we cannot move therefore move now I to next location i.e. 6.

Note: Here we can observe only j is moving back but not i. I is moving only in the forward direction.

Step-4: Repeat the process till find a match

https://www.youtube.com/watch?v=V5-7GzOfADQ **Boyer**

Moore Algorithm:

Step-1: Construct 'Bad Match Table'

Step-2: Compare right most character of pattern with given string based on the 'value' of bad match table

Step-3: If mismatch then shift the pattern to the right position corresponding to the 'value' of bad match table

While constructing bad match table use following formula for value. value=length of pattern-index-1 and last value=length of pattern

moodbanao

Constructing Bad Match Table

Example 1:

- Pattern – 'teammast'
- Text – 'welcometoteammast'

T E A M M A S T Length = 8

Index : 0 1 2 3 4 5 6 7

Letter	T	E	A	M	S	*
Value						

Constructing Bad Match Table

T E A M M A S T Length = 8

Index : 0 1 2 3 4 5 6 7

Letter	T	E	A	M	S	*
Value	7	6	5			

A = 8 - 2 - 1

Here the letter 'A' is occurring twice so replace the latest value by old one. In the same way for M also. T is the last character in pattern so its value=8(length of pattern)

Constructing Bad Match Table

T E A M M A S T Length = 8

Index : 0 1 2 3 4 5 6 7

Letter	T	E	A	M	S	*
Value	7	6	5	4		

M = 8 - 3 - 1

Constructing Bad Match Table

T E A M M A S T Length = 8

Index : 0 1 2 3 4 5 6 7

Letter	T	E	A	M	S	*
Value	7	6	5	3		

M = 8 - 4 - 1

moodbanao

Constructing Bad Match Table

T E A M M A S T Length = 8

Index : 0 1 2 3 4 5 6 7

Letter	T	E	A	M	S	*
Value	8	6	2	3	1	

T = 8 Last letter = length, if not already defined.

Boyer Moore Example

Letter	T	E	A	M	S	*
Value	8	6	2	3	1	8

Mismatch here so move 8 characters right hand side

Boyer Moore Example

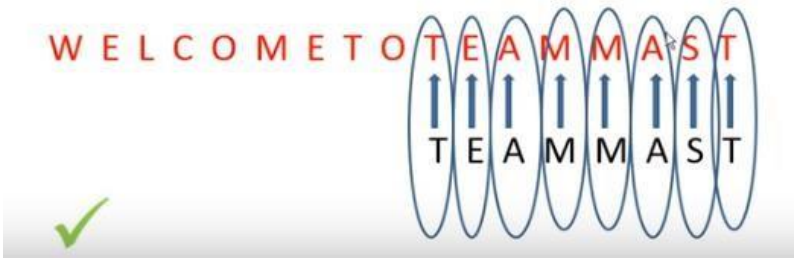
Letter	T	E	A	M	S	*
Value	8	6	2	3	1	8

Mismatch so move 1 character to the right hand side

moodbanao

Boyer Moore Example

Letter	T	E	A	M	S	*
Value	8	6	2	3	1	8



Letter	T	O	H	*
Value	1	2	5	5

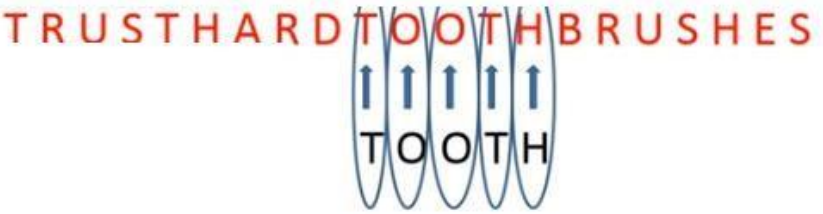
Boyer Moore Example

Letter	T	O	H	*
Value	1	2	5	5

Boyer Moore Example

Letter	T	O	H	*
Value	1	2	5	5

moodbanao



9.3 Hardware Text Search Systems

Software text search is applicable to many circumstances but has encountered restrictions on the ability to handle many search terms simultaneously against the same text and limits due to I/O speeds. One approach that off loaded the resource intensive searching from the main processors was to have a specialized hardware machine to perform the searches and pass the results to the main computer which supported the user interface and retrieval of hits. Since the searcher is hardware based, scalability is achieved by increasing the number of hardware search devices.

Another major advantage of using a hardware text search unit is in the elimination of the index that represents the document database. Typically the indexes are 70% the size of the actual items. Other advantages are that new items can be searched as soon as received by the system rather than waiting for the index to be created and the search speed is deterministic.

Figure 9.1 represents hardware as well as software text search solutions.The arithmetic part of the system is focused on the term detector. There has been three approaches to implementing term detectors: parallel comparators or associative memory, a cellular structure, and a universal finite state automata.

When the term comparator is implemented with parallel comparators, each term in the query is assigned to an individual comparison element and input

data are serially streamed into the detector. When a match occurs, the term comparator informs the external query resolver (usually in the main computer) by setting status flags.

Specialized hardware that interfaces with computers and is used to search secondary storage devices was developed from the early 1970s with the most recent product being the **Parallel Searcher (previously the Fast Data Finder).** The typical hardware configuration is shown in Figure 9.9 in the dashed box. The speed of search is then based on the speed of the I/O.

moodbanao

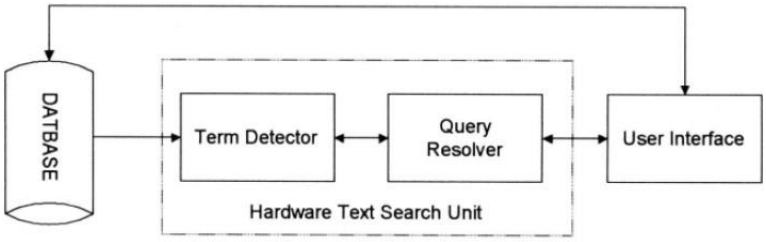


Figure 9.9 Hardware Text Search Unit

One of the earliest hardware text string search units was the ***Rapid Search*** Machine developed by General Electric. The machine consisted of a special purpose search unit where a single query was passed against a magnetic tape containing the documents. A more sophisticated search unit was developed by Operating Systems Inc. called the ***Associative File Processor (AFP).*** It is capable of searching against multiple queries at the same time. Following that initial development, OSI, using a different approach, developed the ***High SpeedText Search (HSTS) machine***. It uses an algorithm similar to the Aho- Corasick software finite state machine algorithm except that it runs three parallel state machines. One state machine is dedicated to contiguous word phrases, another for imbedded term match and the final for exact word match.

Inparallel with that development effort, GE redesigned their Rapid Search Machine into the ***GESCAN unit***. The GESCAN system uses a text array processor (TAP) that simultaneously matches many terms and conditions against a given text stream the TAP receives the query information from the user's computer and directly access the textual data from secondary storage. The TAP consists of a large cache memory and an array of four to 128 query processors. The text is loaded into the cache and searched by the query processors (Figure 9.10). Each query processor is independent and can be loaded at any time. A complete query is handled by each query processor.

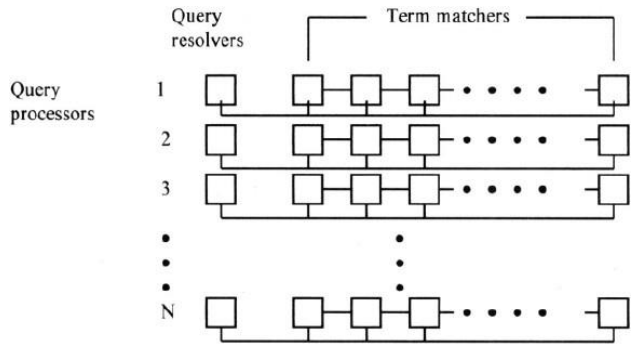


Figure 9.10 GESCAN Text Array Processor

A query processor works two operations in parallel; matching query terms to input text and Boolean logic resolution. Term matching is performed by a series of character cells each containing one character of the query. A string of character cells is implemented on the same LSI chip and the

moodbanao

chips can be connected in series for longer strings. When a word or phrase of the query is matched, a signal is sent to the resolution sub-process on the LSI chip. The resolution chip is responsible for resolving the Boolean logic between terms and proximity requirements. If the item satisfies the query, the information is transmitted to the users computer.

The text array processor uses these chips in a matrix arrangement as shown in Figure9.10. Each row of the matrix is a query processor in which the first chip performsthe query resolution while the remaining chips match query terms. The maximum number of characters in a query is restricted by the length of a row while the number of rows limit the number of simultaneous queries that can be processed.

Another approach for hardware searchers is to augment disc storage. The*augmentation is a generalized associative search* element placed between the read and write heads on the disk. The content addressable segment sequential memory (CASSM) system uses these search elements in parallel to obtain structured data from a database. The CASSM system was developed at the University of Florida as a general purpose search device. It can be used to perform string searching across the database. Another special search machine is the ***relational associative processor (RAP)*** developed at the University of Toronto. Like CASSM performs search across a secondary storage device using a series of cells comparing data in parallel.

The ***Fast Data Finder (FDF)*** is the most recent specialized hardware text search unit still in use in many organizations. It was developed to search text and has been used to search English and foreign languages. The early Fast Data Finders consisted of an array of programmable text processing cells connected in series forming a pipeline hardware search processor. The cells are implemented using a VSLI chip. In the TREC tests each chip contained 24processor cells with a typical system containing 3600 cells. Each cell will be a comparator for a single character limiting the total number of characters in a query to the number of cells.

The cells are interconnected with an 8-bit data path and approximately 20- bit control path. The text to be searched passes through each cell in a pipeline fashion until the complete database has been searched. As data is analyzed at each cell, the 20 control lines states are modified depending upon their current state and the results from the comparator. An example of a Fast Data Finder system is shown inFigure 9.11.

moodbanao

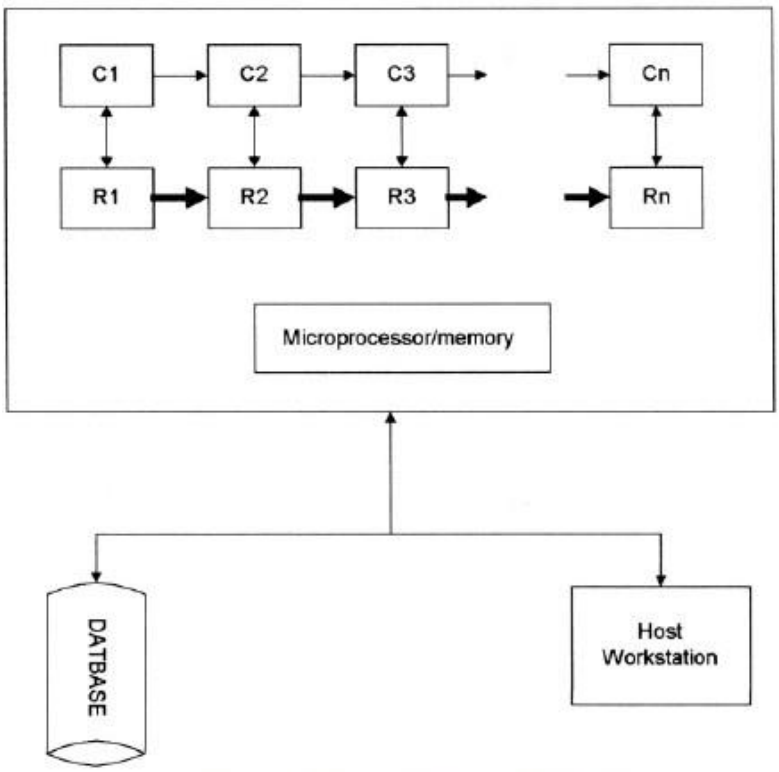


Figure 9.11 Fast Data Finder Architecture

moodbanao

A cell is composed of both a register cell (Rs) and a comparator (Cs).The input from the Document database is controlled and buffered by the micro process/memory and feed through the comparators. The search characters are stored in the registers. The connection between the registers reflect the control lines that are also passing state information.Groups of cells are used to detect query terms, along with logic between the terms, by appropriate programming of the control lines. When a pattern match is detected, a hit is passed to the internal microprocessor that passes it back to the host processor, allowing immediate access by the user to the Hit item.

The functions supported by the Fast data Finder are:

- Boolean Logic including negation
- Proximity on an arbitrary pattern
- Variable length "don't cares"
- Term counting and thresholds
- Fuzzy matching
- Term weights
- Numeric ranges

Information System Evaluation

The creation of the annual Text Retrieval Evaluation Conference (TREC) sponsored by the Defense Advanced Research Projects Agency (DARPA) and the National Institute of Standards and Technology (NIST) changed the standard process of evaluating information systems. The conference provides a standard database consisting of gigabytes of test data, search statements and the expected results from the searches to academic researchers and commercial companies for testing of their systems. This has placed a standard baseline into comparisons of algorithms.

In recent years the evaluation of Information Retrieval Systems and techniques for indexing, sorting, searching and retrieving information have become increasingly important.

There are many reasons to evaluate the effectiveness of an Information Retrieval System:

- To aid in the selection of a system to procure
- To monitor and evaluate system effectiveness
- To evaluate query generation process for improvements
- To provide inputs to cost-benefit analysis of an information system

moodbanao

- To determine the effects of changes made to an existing information system.

Measures Used in System Evaluations

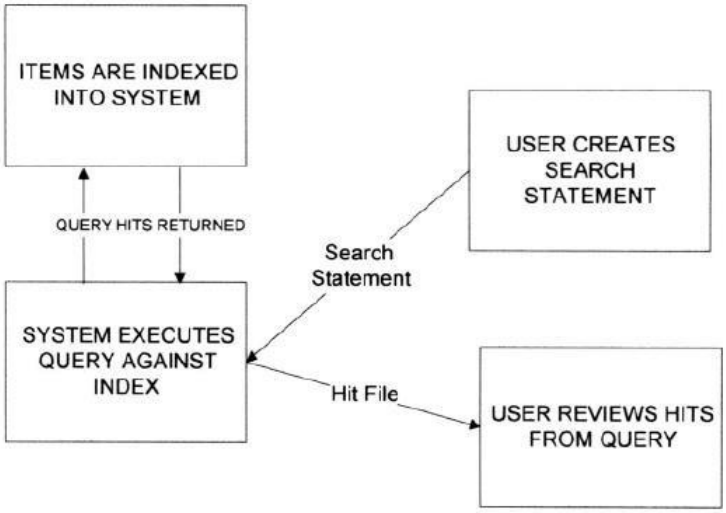


Figure 11.1 Identifying Relevant Items

Measurements can be made from two perspectives: user perspective and system perspective. Techniques for collecting measurements can also be objective or subjective. An objective measure is one that is well-defined and based upon numeric values derived from the system operation. A subjective measure can produce a number, but is based upon an individual users judgments.

Measurements with automatic indexing of items arriving at a system are derived from standard performance monitoring associated with any program in a computer (e.g., resources used such as memory and processing cycles) and time to process an item from arrival to availability to a search process. When manual indexing is required, the measures are then associated with the indexing process.

Response time is a metric frequently collected to determine the efficiency of the search execution. Response time is defined as the time it takes to execute the search. In addition to efficiency of the search process, the quality of the search results are also measured by precision and recall.

$$\text{Precision} = \frac{Number\_Retrieved\_Relevant}{Number\_Total\_Retrieved}$$

$$\text{Recall} = \frac{Number\_Retrieved\_Relevant}{Number\_Possible\_Relevant}$$

moodbanao

$$\text{Fallout} = \frac{Number\_Retrieved\_Nonrelevant}{Number\_Total\_Nonrelevant}$$

where *Number_Total_Nonrelevant* is the total number of non-relevant items in the database. Fallout can be viewed as the inverse of recall and will never encounter the situation of 0/0 unless all the items in the database are relevant to the search. It can

Another measure that is directly related to retrieving non-relevant items can be used in defining how effective an information system is operating. This measure is called Fallout and defined as:

There are other measures of search capabilities that have been proposed. A new measure that provides additional insight in comparing systems or algorithms is the "Unique Relevance Recall" (URR) metric. URR is used to compare more two or more algorithms or systems. It measures the number of relevant items that are retrieved by one algorithm that are not retrieved by the others:

$$\text{Unique\_Relevance\_Recall} = \frac{Number\_unique\_relevant}{Number\_relevant}$$

Number_unique_relevant is the number of relevant items retrieved that were not retrieved by other algorithms. When many algorithms are being compared, the definition of *uniquely* found items for a particular system can be modified, allowing a small number of other systems to also find the same item and still be considered unique. This is accomplished by defining a percentage **(P_u)** of the total number of systems that can find an item and still consider it unique. *Number_relevant* can take on two different values based upon the objective of the evaluation:

VALUE	INTERPRETATION
Total Number Retrieved Relevant (TNRR)	the total number of relevant items found by all algorithms
Total Unique Relevant Retrieved (TURR)	the total number of unique items found by all the algorithms

A	B	C	D	E	F	G	H	I	J	K	L	M
3	4	2	22	1	100	200	22	100	10	500	6	15

Activate W
Go to PC settir

Figure 11.2a Number Relevant Items

moodbanao

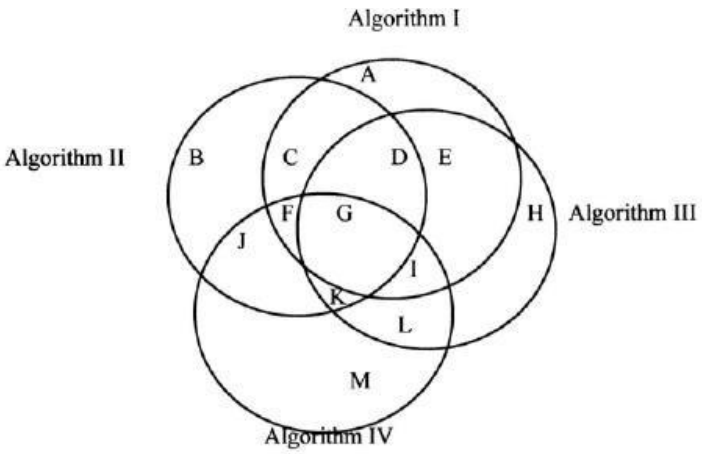


Figure 11.2b Four Algorithms With Overlap of Relevant Retrieved

actually found by the algorithm. Figure 11.2a and 11.2b provide an example of the overlap of relevant items assuming there are four different algorithms. Figure 11.2a gives the number of items in each area of the overlap diagram in Figure 11.2b. If a relevant item is found by only one or two techniques as a "unique item," then from the diagram the following values URR values can be produced:

Algorithm I - 6 unique items (areas A, C, E)
Algorithm II - 16 unique items (areas B, C, J)
Algorithm III - 29 unique items (areas E, H, L)
Algorithm IV - 31 unique items (areas J, L, M)

TNRR = A + B + C + • • • + M = 985
TURR = A + B + C + E + H + J + L + M = 61

Activate Window
Go to PC settings to ac

Algorithm	URR_{TNRR}	URR_{TURR}
Algorithm I	6/985 = .0061	6/61= .098
Algorithm II	16/985 = .0162	16/61= .262
Algorithm III	29/985 = .0294	29/61 = .475
Algorithm IV	31/985 = .0315	31/61 = .508

Other measures have been proposed for judging the results of searches:

Novelty Ratio: ratio of relevant and not known to the user to total relevant retrieved

Coverage Ratio: ratio of relevant items retrieved to total relevant by the user before the search

Sought Recall: ratio of the total relevant reviewed by the user after the search to the total relevant the user would have liked to examine

moodbanao

In some systems, programs filter text streams, software categorizes data or intelligent agents alert users if important items are found. In these systems, the Information Retrieval System makes decisions without any human input and their decisions are binary in nature (an item is acted upon or ignored). These systems are called binary classification systems for which effectiveness measurements are created to determine how algorithms are working (Lewis-95). One measure is the utility measure that can be defined as (Cooper-73):

$$U = \alpha*(\text{Relevant\_Retrieved}) + \beta*(\text{Non-Relevant\_Not Retrieved}) - \delta*(\text{Non-Relevant\_Retrieved}) - \gamma*(\text{Relevant\_Not Retrieved})$$

where α and β are positive weighting factors the user places on retrieving relevant items and not retrieving non-relevant items while δ and γ are factors associated with the negative weight of not retrieving relevant items or retrieving non-relevant items. This formula can be simplified to account only for retrieved items with β and γ equal to zero (Lewis-96). Another family of effectiveness measures called the E-measure that combines recall and precision into a single score was proposed by Van Rijsbergen (Rijsbergen-79).

Activate Windows

Measurement Example-TREC-Results

Until the creation of the Text Retrieval Conferences (TREC) by the Defense Advance Research Projects Agency (DARPA) and the National Institute of Standards and Technology (NIST), experimentation in the area of information retrieval was constrained by the researcher's ability to manually create a test database. One of the first test databases was associated with the Cranfield I and II tests (Cleverdon-62, Cleverdon-66). It contained 1400 documents and 225 queries.

It became one of the standard test sets and has been used by a large number of researchers. Other test collections have been created by Fox and Sparck Jones. There have been five TREC-conferences since 1992. TREC- provides a set of training documents and a set of test documents, each over 1 Gigabyte in size. It also provides a set of training search topics (along with relevance judgments from the database) and a set of test topics.

The researchers send to the TREC-sponsor the list of the top 200 items in ranked order that satisfy the search statements. These lists are used in determining the items to be manually reviewed for relevance and for calculating the results from each system. The search topics are "user need" statements rather than specific queries. This allows maximum flexibility for each researcher to translate the search statement to a query appropriate for their system and assists in the determination of whether an item is relevant.

moodbanao

The search Topics in the initial TREC-consisted of a Number, Domain (e.g., Science and Technology), Title, Description of what constituted a relevant item, Narrative natural language text for the search, and Concepts which were specific search terms.

In addition to the search measurements, other standard information on system performance such as system timing, storage, and specific descriptions on the tests are collected on each system. This data is useful because the TREC- objective is to support the migration of techniques developed in a research environment into operational systems. TREC-5 was held in November 1996. The results from each conference have varied based upon understanding from previous conferences and new objectives.

TREC-1 (1992) was constrained by researchers trying to get their systems to work with the very large test databases. TREC-2 in August 1993 was the first real test of the algorithms which provided insights for the researchers into areas in which their systems needed work. The search statements (user need statements) were very large and complex. They reflect long-standing information needs versus adhoc requests. By TREC-3, the participants were experimenting with techniques for query expansion and the importance of constraining searches to passages within items versus the total item.

TREC-4 introduced significantly shorter queries (average reduction from 119 terms in TREC-3 to 16 terms in TREC-4) and introduced five new areas of testing called “tracks” (Harman-96). The queries were shortened by dropping the title and a narrative field, which provided additional description of a relevant item. The multilingual track expanded TREC-4 to test a search in a Spanish test set of 200 Mbytes of articles from the “El Norte” newspaper.