

UNIT-V

Text Search Algorithms: Introduction, Software text searches Algorithms, Hardware text search systems. **Multimedia Information Retrieval:** Spoken Language Audio Retrieval, Non-Speech Audio Retrieval, Graph Retrieval, Imagery Retrieval, Video Retrieval

Text Search Algorithms Three classical text retrieval techniques have been defined for organizing items in a textual data base, for rapidly identifying the relevant items and for eliminating items that do not satisfy the search.

The techniques are

- 1) Full text scanning (streaming)
- 2) Word inversion
- 3) Multi attributes retrieval

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

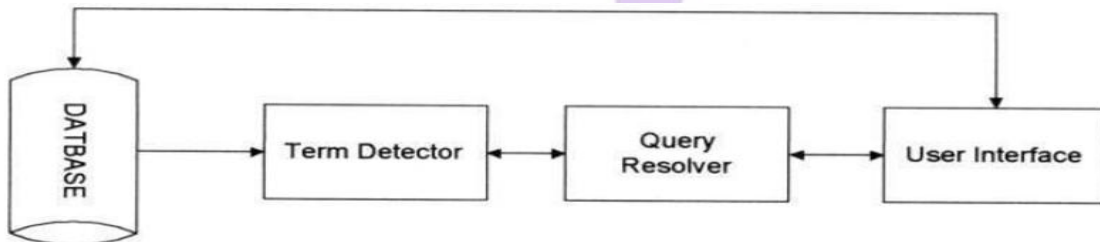


Figure 9.1 Text Streaming Architecture

The data base contains the full 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 between the items. It will in put the text and detect the existence of the search terms. It will output to the query resolver the detected terms to allow for final logical processing of a query against an item. The query resolver performs two functions. It will accept search statements from the users, extract the logic and search terms and pass the search terms to the detector. It also accepts results from the detector and determines which queries are satisfied by the item and possibility the weight associated with hit. The Query Resolver will pass information to the user interface that will be continually updating search status to the user and request retrieve any items that

Some of the original brute force methods could require $O(n*m)$ symbol comparisons. More improvements have reduced the time to $O(n+m)$. In the case of hardware search machines, multiple parallel search machines (term detectors) may work against the same data stream allowing for more queries or against different data streams reducing the time to access the complete database. In software systems, multiple detectors may execute at the same time.

There are two approaches to the data stream. In the first approach the complete data base is being sent to the detector(s) functioning database. In the second approach random retrieved items are being passed to the detectors. In this second case the idea is to perform an index search of the data base and let the text streamer perform additional search logic that is not satisfied the index 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 care” Search for symbols that are on the inter-word symbol list (e.g., “;”) The full text search function does not require any additional storage overhead. There is also the advantage where hits may be returned to the user as soon as found. Typically, in an index system, the complete query must be processed before any hits are determined or available. Streaming systems also provide a very accurate estimate of current search status and time to complete the query. It is difficult to locate all the possible index values short of searching the complete dictionary of possible terms. Many of the hardware and software text searchers use finite state automata as a basis for their algorithms. A finite state automaton is a logical machine that is composed of five elements:

I-a set of input symbols from the alphabet supported by the automata

S-a set of possible states

P-a set of productions that define the next state based up on the current state and input symbol

S₀-a special state called the initial state, **S_F**-a set of one or more final states from the sets

Software Text Search Algorithms In software streaming techniques, the item to be searched is read into memory and then the algorithm is applied. There are four major algorithms associated with software text search: the brute force approach Knuth-Morris-Pratt Boyer-Moore, Shift-OR algorithm 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 strings The Brute force approach is the simplest string-matching algorithm. The idea history and match the search string against the input text. input text one position and start the comparison process over. The expected number of comparisons when searching an input text string of n characters for a pattern of m characters.

$$N_c = \frac{c}{c-1} (1 - \frac{1}{c^m}) * (n-m+1) + O(1)$$

Where N_c is the expected number of comparisons and c is the size of the alphabet for the text.

Knuth-Pratt-Morris (KPM)algorithm

Pattern:

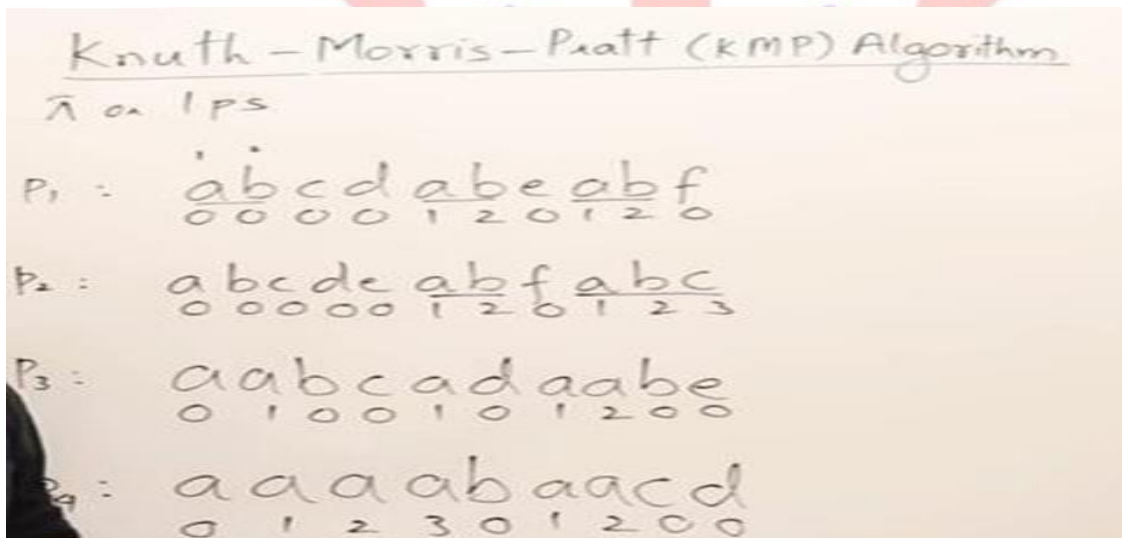
abcdabC12

34567

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

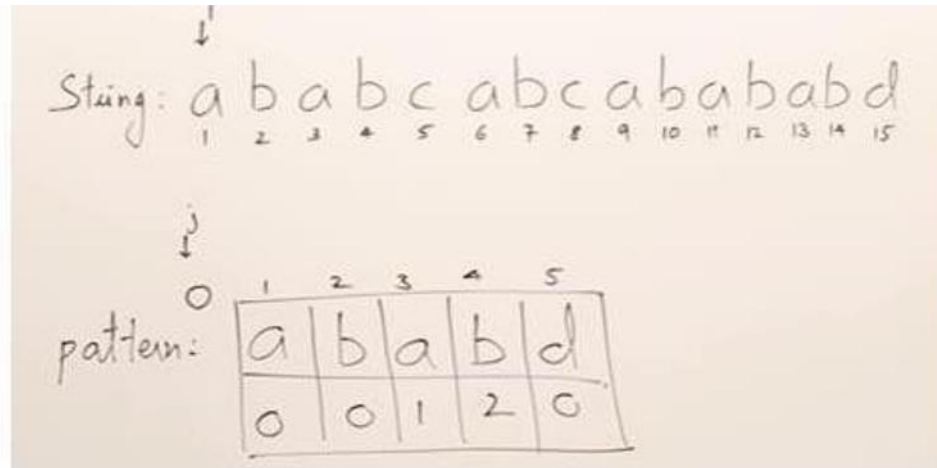
Prefix: a, a b, a b c ,a b c d etc. S Suffix: c bc, a b c, d a b c etc.

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



Example: Given string and pattern is Now constructing table with repeated characters of pattern like following

your roots to success...



Here „a“ is repeated so keep index 1 below it, „b“ is Repeated so keep index 2 below it and re install „0“ s. Now start search process.

Step-1: Initialize string index as I, pattern index as 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.

Step3: Now compare string[i] and pattern [j+1] i.e, band b matching so move both i, j to next position When i=5 and j=4 string [i] = cand pattern [j+1] = d. here not matching then move it to its index location. i.e,2. So now j position is pattern [2].

Now compare string[i]=cand pattern [j+1] = a. Here not matching then move its index location i.e.,0. So now j position is pattern [0]. J is now on 0 positions o we cannot move therefore move now It o next location i.e.6.

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

Step-4: Repeat the process till find a match

Boyer Moore Algorithm:

Step-1 : Construct 'Bad MatchTable'

Step-2 : Comparerightmostcharacterofpatternwithgivenstringbasedonthe'value'ofbadmatchtable

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 l as t value= length of pattern

Constructing Bad Match Table



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



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

$$M = 8 - 3 - 1$$

Constructing Bad Match Table



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

$$M = 8 - 4 - 1$$

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	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

W E L C O M E T O T E A M M A S T
T E A M M A S T

Mismatch here so move 8 characters right hand side

Boyer Moore Example

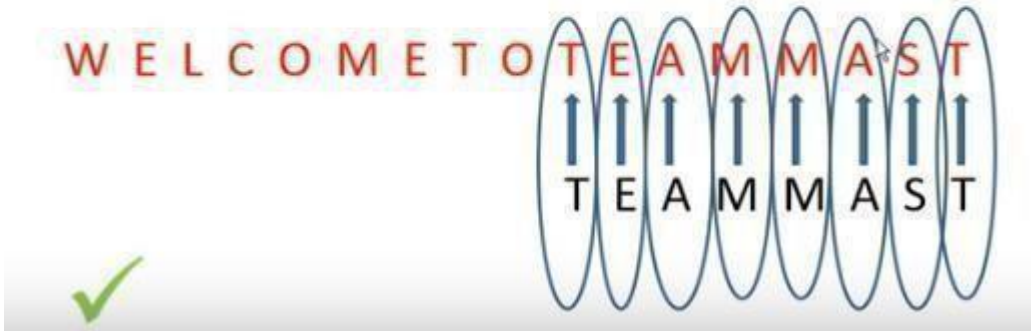
Letter	T	E	A	M	S	*
Value	8	6	2	3	1	8

W E L C O M E T O T E A M M A S T
T E A M M A S T

Mismatch so move 1 character to the right hand side

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



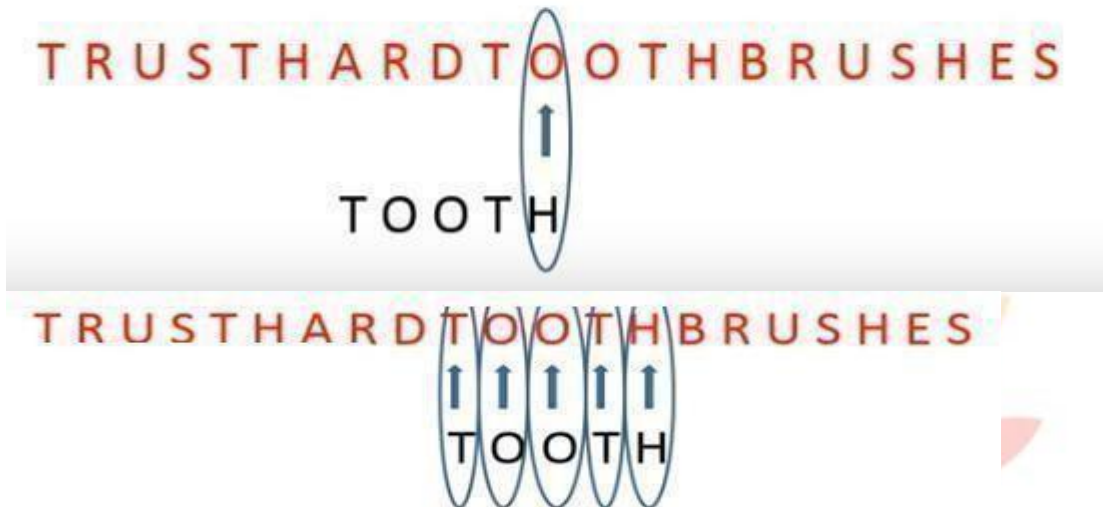
Boyer Moore Example

Letter	T	O	H	*
Value	1	2	5	5



Boyer Moore Example

Letter	T	O	H	*
Value	1	2	5	5



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 offloaded 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 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.

The arithmetic part of the system is focused on the term detector. There have

been three approaches to implementing term detectors:

1. parallel comparators or associative memory,
2. a cellular structure, and
3. 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.

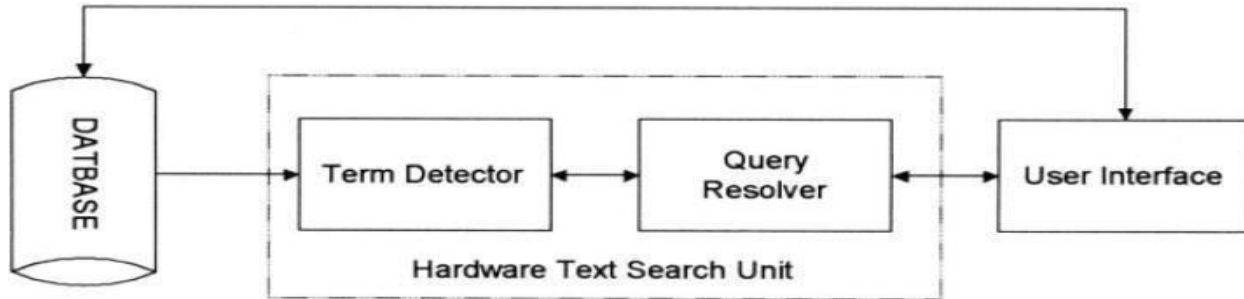


Figure 9.9 Hardware Text Search Unit

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 Speed Text 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.

In parallel 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 accesses 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.

A query processor works two operations in parallel; matching query terms to input text and Bool

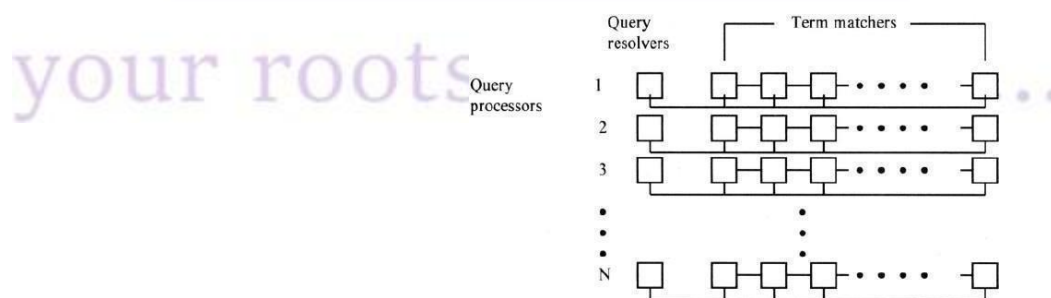


Figure 9.10 GESCAN Text Array Processor

ean 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 chips can be connected in series for longer str

ings. When a word or phrase of the query is matched, a signal is sent to the resolution sub-processor 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 user's computer.

The text array processor uses these chips in a matrix arrangement as shown in Figure 9.10. Each row of the matrix is a query processor in which the first chip performs the 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 VLSI chip. In the TREC test each chip contained 24 processor 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 completed 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.

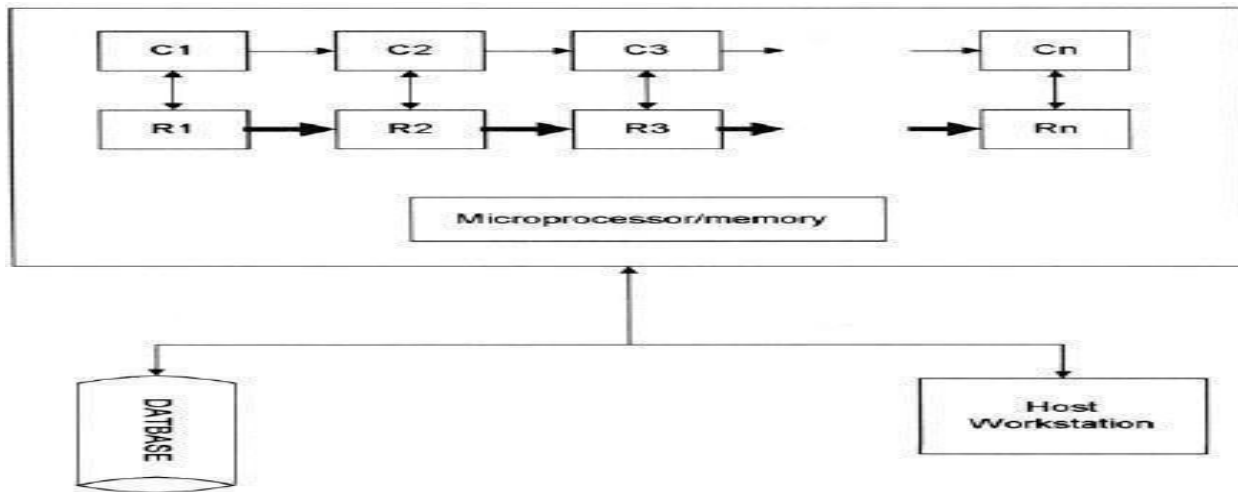


Figure 9.11 Fast Data Finder Architecture

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. These 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 ex: "don't
- cares" Term counting and
- thresholds Fuzzy matching
- Term
- weights Numeric ranges

Multimedia information retrieval (MMIR or MIR)

Data sources included directly perceivable media such as audio, image and video, indirectly perceivable sources such as text, semantic descriptions, bio signals as well as not perceivable sources such as bio information, stock prices, etc.

The methodology of MMIR can be organized in three groups:

1. Methods for the summarization of media content (feature extraction). The result of feature extraction is a description.
2. Methods for the filtering of media descriptions (for example, elimination of redundancy)

3. Methods for the categorization of media descriptions into classes.

Spoken Document Retrieval:

A textual representation of the audio content from a video can be obtained through automatic speech recognition. Information retrieval from speech recognition transcripts has received quite a bit of attention in recent years in the spoken document retrieval track at TREC7, TREC8 and TREC9.

The current „consensus“ from a number of published experiments in this area is that as long as speech recognition has a word error rate better than 35% word error, then information retrieval from the transcripts of spoken documents is only 3-10% worse than information retrieval on perfect text transcriptions of the same documents.

Image Similarity Matching.

Example-

based image retrieval task has been studied for many years. The task requires the image search engine to find the set of images from a given image collection that is similar to the given query image.

Traditional methods for content-based image retrieval are based on a vector model.

These methods represent an image as a set of features and the difference between two images is measured through a (usually Euclidean) distance between their feature vectors. While there have been no large-scale, standardized evaluations of image retrieval systems, most image retrieval systems are based on features such as color, texture, and shape that are extracted from the image pixels.

OCR document retrieval:

A different, textual, representation is derived by reading the text that present in the video images using optical character recognition (OCR). At TREC 5, experiments have shown that information retrieval on documents recognized through OCR with a character error rate of 5% and 20% degrades IR effectiveness by 10% to 50% depending on the metric, when compared to perfect text retrieval.

In contrast, video information retrieval much more complex and combines elements of spoken documents, OCR documents, image similarity as well as the audio and image features.

In this paper we will examine the effects of multimodal information retrieval from video documents. There are only a few areas of audio analysis that we examined was automatic speech recognition. While analyzing the video imagery, we considered the color similarity of images, and the presence of faces and text that was readable on the screen. We explored these dimensions of audio analysis and image analysis separately and in combination in our video retrieval experiments. We will present experiments with each different type of extracted metadata performed separately and also combined together in the context of the TREC Video Retrieval evaluation performed by the National Institute of Standards and Technology.

Methods for Extracting Textual Metadata

Speech Recognition

The audio processing component of four video retrieval systems splits the audio track from the MPEG-1 encoded video file, and decodes the audio and down samples it to 16kHz, 16bit samples.

These samples are then passed to a speech recognizer. The speech recognition system we used for these experiments is a state-of-the-art large vocabulary, speaker independent speech recognizer [9]. For the purposes of this evaluation, a 64000-word language model derived from a large corpus of broadcast news transcripts was used. Previous experiments had shown the word error rate on this type of mixed documentary-styled data with frequent overlap of music and speech to be just over 30%.

Video OCR A different, textual, representation is derived by reading the text that present in the video images using optical character recognition (OCR). OCR technology has been commercially available for many years. However, reading the text present in the video stream requires a number of processing steps in addition to the actual character recognition. Our video optical character recognition system [5] uses the following approach to identify and recognize captioned text that appears on the video. Given the number of frames contained in typical broadcast news, it is not computationally feasible to process each and every video frame for text. For this reason a rough or quick text region detection is performed first. Then the text must be extracted from the image, and converted into a binary black and white representation, since the commercially available OCR engines do not recognize colored text on a variably colored background.

Unlike text printed on white paper, the background of the image tends to become complex, with the character hue and brightness very near the background values.

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 input to cost-benefit analysis of an information system

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

Measures Used in System Evaluations

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 it is based upon an individual user's judgment

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.

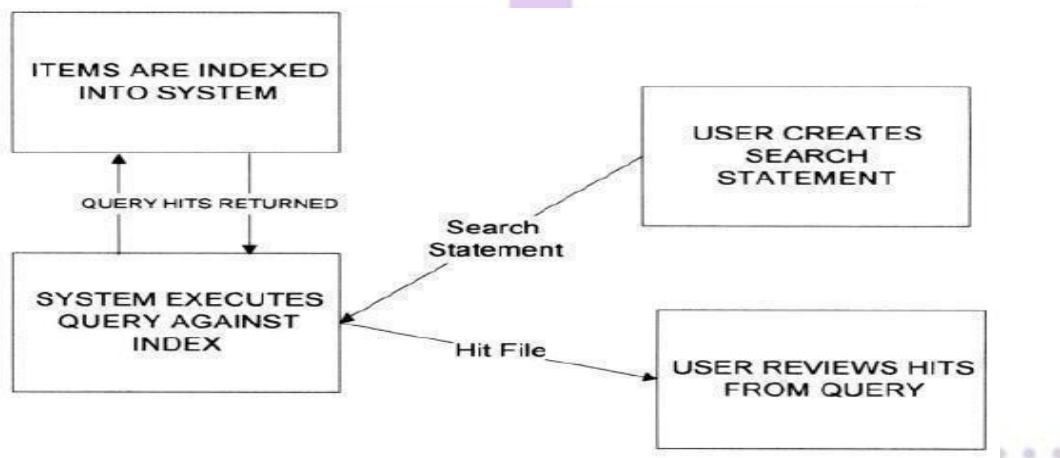


Figure 11.1 Identifying Relevant Items

$$\text{Fallout} = \frac{\text{Number_Retrieved_Nonrelevant}}{\text{Number_Total_Nonrelevant}}$$

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" (U

RR) metric. URR is used to compare more than two or more algorithms or systems.

your roots to success...

It measures the number of relevant items that are retrieved by one algorithm that are not retrieved by the others:

$$\text{Unique_Relevance_Recall} = \frac{\text{Number_unique_relevant}}{\text{Number_relevant}}$$

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

Figure 11.2a Number Relevant Items Algorithm I

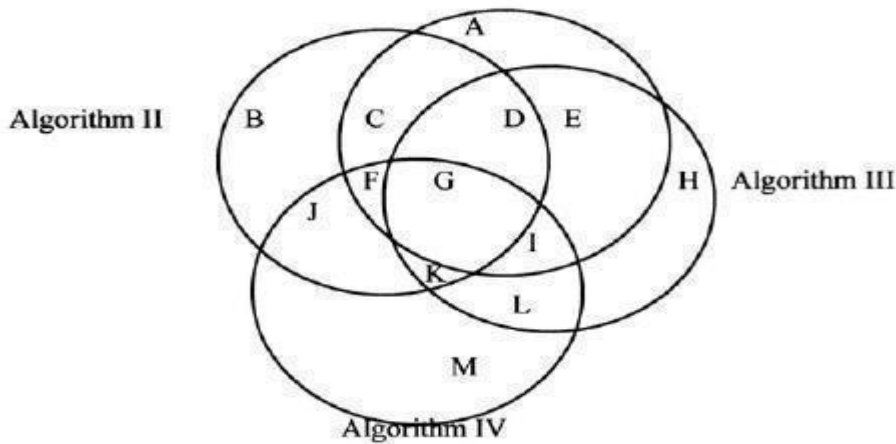


Figure 11.2b Four Algorithms With Overlap of Relevant Retrieved

your roots to success...

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)

$$TNRR = A + B + C + \dots + M = 985$$

$$TURR = A + B + C + E + H + J + L + M = 61$$

Activate Window
Go to PC settings to activate Windows

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

your roots to success...

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):

$$U = \alpha * (\text{Relevant_Retrieved}) + \beta * (\text{Non-Relevant_Not Retrieved}) - \delta * (\text{Non-Relevant_Retrieved}) - \gamma * (\text{Relevant_Not Retrieved})$$

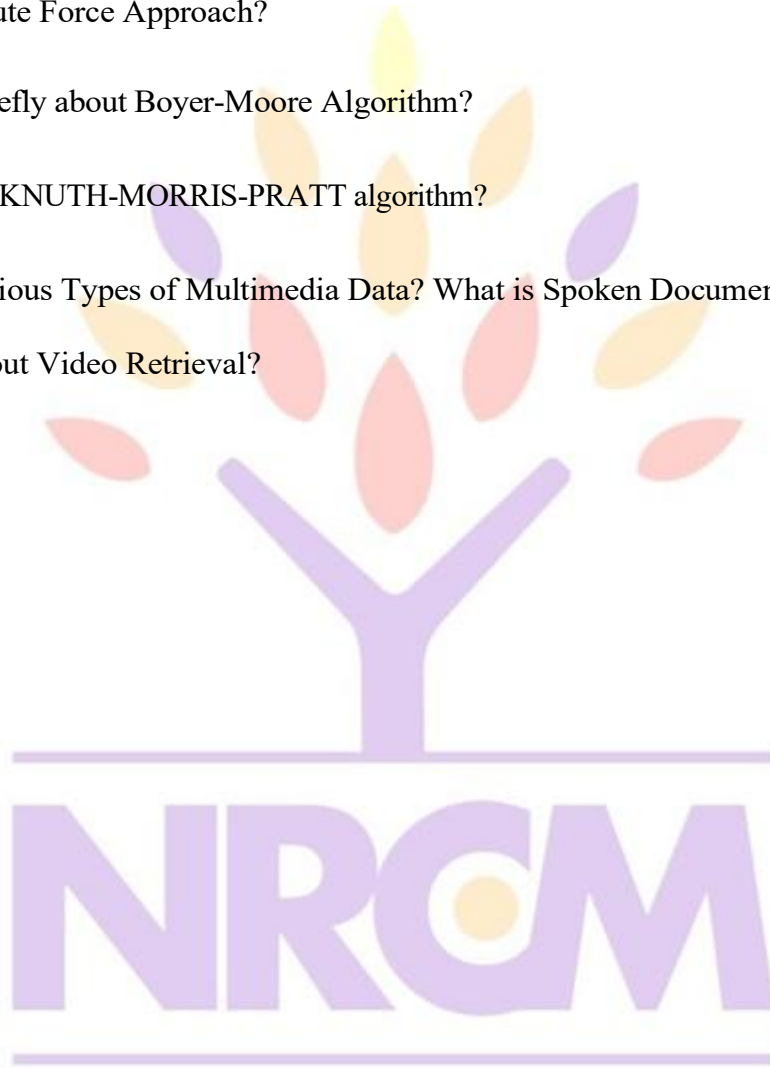
where α and β are positive weighting factors the user places on retrieving relevant items and not retrieving non-relevant items while δ and γ are factors associated with the negative weight of not retrieving relevant items or retrieving non-relevant items. This formula can be simplified to account only 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

your roots to success...

Questions:

1. What is Text Search? Explain Text Streaming Architecture.
2. What is Brute Force Approach?
3. Discuss briefly about Boyer-Moore Algorithm?
4. Explain the KNUTH-MORRIS-PRATT algorithm?
5. Explain various Types of Multimedia Data? What is Spoken Document Retrieval?
6. Explain about Video Retrieval?



your roots to success...