CS6007-INFORMATION RETRIEVAL

UNIT -1

Introduction -History of IR- Components of IR – Issues –Open source Search engine Frameworks – The impact of the web on IR – The role of artificial intelligence (AI) in IR – IR Versus Web Search – Components of a Search engine- Characterizing the web.

1.1 INTRODUCTION:

PART-A

1.What is information Retrieval?(NOV/DEC'16)

Information Retrieval is **finding material of an unstructured nature** that satisfies an information need from within large collections. IR locates relevant documents, on the basis of user input such as keywords or example documents.

Information Retrieval System is a system it is a capable of **storing, maintaining from a system and retrieving of information.** This information May any of the form that is **audio, vedio, text.**

- Information Retrieval System is mainly **focus electronic searching and retrieving** of documents.
- Information Retrieval is a activity of **obtaining relevant documents based on user needs** from collection of retrieved documents.

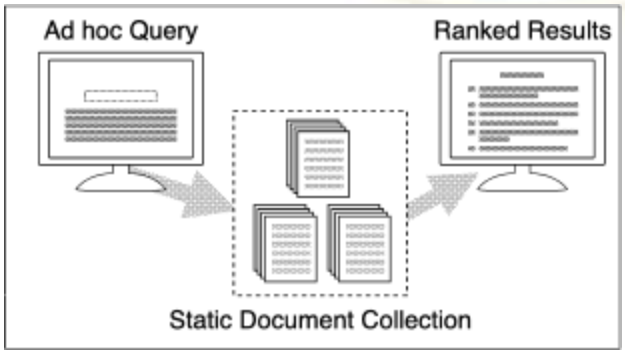
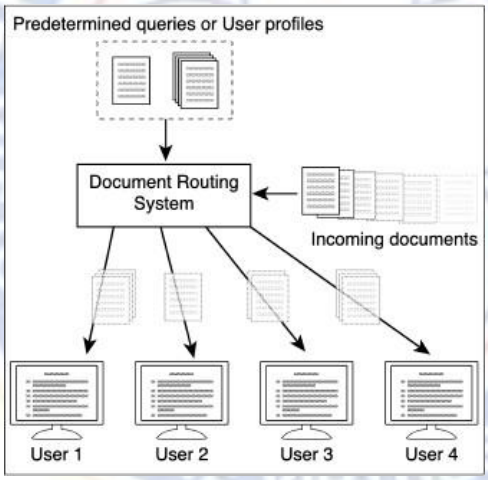


Fig shows basic information retrieval system

- A static, or relatively static, document collection is indexed prior to any user query.
- A query is issued and a set of documents that are deemed relevant to the query are ranked based on their computed similarity to the query and presented to the user query.
- Information Retrieval (IR) is devoted to finding *relevant* documents, not finding simple matches to patterns.

- A related problem is that of document routing or filtering. Here, the queries are static and the document collection constantly changes. An environment where **corporate e-mail is routed based on predefined queries to different parts of the organization (i.e., e-mail about sales is routed to the sales department, marketing e-mail goes to marketing, etc.)** is an example of an application of document routing.
- Figure illustrates document routing



PRECISION AND RECALL:

To measure effectiveness, two ratios are used**:** ***precision*** **and** ***recall.***

Precision:

- **It** is the ratio of the number of relevant documents retrieved to the total number retrieved. Precision provides an indication of the quality of the answer set.
- However, this does not consider the total number of relevant documents. A system might have good precision by retrieving ten documents and finding that nine are relevant (a 0.9 precision), but the total number of relevant documents also matters**.**

Eg:

If there were only nine relevant documents, the system would be a huge success. however if millions of documents were relevant and desired, this would not be a good result set.

Recall:

- Recall considers the total number of relevant documents; it is the ratio of the number of relevant documents retrieved to the total number of documents in the collection that are believed to be relevant.
- Computing the total number of relevant documents is non-trivial.

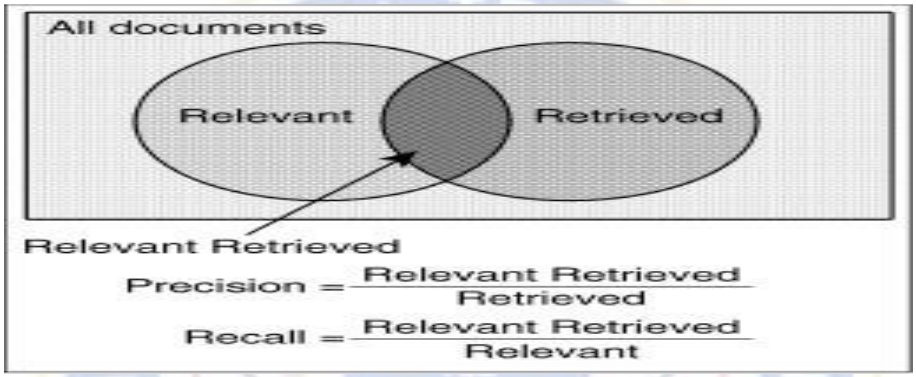


Fig: Precision And Recall

1.2 HISTORY OF IR

PART-B
Appraise the history of information retrieval(APR/MAY'17)

1945
This is very famous (but not a good IMHO) paper by Vannevar Bush.
Bush envisages the memex. It stores the human knowledge in a desktop device.
The essay pays scant attention to information retrieval (bar condemning existing approaches).
1950s
The USSR launches the sputnik.
The US is worried
need to improve the organization of science
need to understand what the Russians are doing
learn Russian
work on machine translation
Boolean Model
1958
Statistical language Properties(Luhn)
Hans Peter Luhn (1896—1964)
Luhn also worked on KWIC (key word in context) indexing.

1960

As computers become more powerful, more and more descriptors could be taken from texts to make them findable in a retrieval systems.
Cyril Cleverdon (1914—1997) developed criteria of precision and recall.
First large scale information systems
online
interactive
distributed
Vector space model

1980 CD ROMS

In the late 80s the CD-ROM appeared to make a dent in online information retrieval.
The CD-ROM fits the standard print publishing model.
Sharing CD-ROMs however, proved a nightmare in libraries.
The 90s were the decade of the Internet.
Online information retrieval become common.
Fuzzy set/logic evidential reasoning

1990 The CD-ROM lost out

Integration of graphics
Start of search engines
Web information retrieval
TREC
Regression, Neural nets, Inference Networks, Intent Semantic Indexing.

- The Page Rank algorithm is a simple algorithm to make such values appear.

1.3 COMPONENTS OF IR BLOCK DIAGRAM.

(Part-B)
Explain in detail about the components of IR. (Nov/Dec'16)

An information retrieval system is an information system, which is used to above items of information that need to be processed, searched, retrieval and disseminated to various user populations.

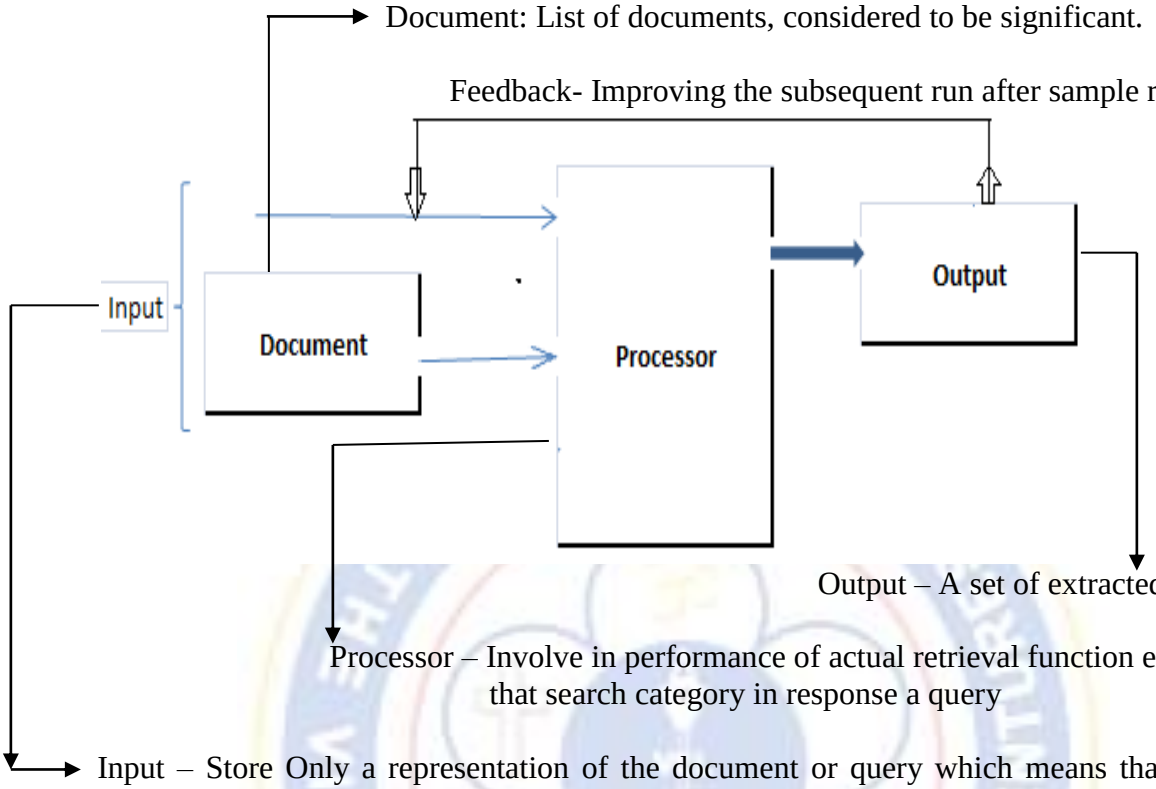
.

IR locates relevant documents, on the basis of user input such as keywords or example documents.

Information retrieval location relevant documents, on the basis of user inputs such as keyword or example documents.

The computer based retrieval systems stores only a representation of the document or query which means that the text of a document is lost once it has been processed for the purpose of

generating its representation.



1.4 ISSUES IN IR

Information retrieval researchers have focused on a few key issues that remain just as important in the era of commercial web search engines working with billions of web pages as they were when tests were done in the 1960s on document col- lections containing about 1.5 megabytes of text. One of these issues is relevance

Challenges in IR

- Scale, distribution of documents
- Controversy over the unit of indexing
  - Hypertext
- High Heterogenity
  - Document structure, size, quality, level of abstraction/specialization.

Big Issue In IR

- **Relevance**

- Relevant document contains the information that a person was looking for when she submitted a query to the search engine. These factors must be taken into account when designing algorithms for comparing text and ranking documents

- **UNIX produces very poor results interims of relevance** This is referred to as the *vocabulary mismatch problem* in information retrieval.
- *Topical relevance* text document is topically relevant to a query if it is on the same topic
- *User relevance.*The person who asked the question

- **Evaluation**

The quality of a document ranking depends on how well it matches a person's expectations, it was necessary early on to develop evaluation measures and experimental procedures for acquiring this data and using it to compare ranking algorithms.

Two of the measures used,

Precision and recall, are still popular

- Precision is a very intuitive measure, and is the proportion of retrieved documents that are relevant.
- Recall is the proportion of relevant documents that are retrieved.

- ***Information needs***

- **The users of a search engine are the ultimate judges of quality.** This has led to numerous studies on how people interact with search engines and, in particular, to the development of techniques to help people express their information needs.

  Eg: Balance for Bank Account

Techniques such as ***query suggestion, query expansion, and relevance feedback u****se interaction and context to refine the initial query in order to produce* better ranked lists.

1.5 OPEN SOURCE SEARCH ENGINE FRAMEWORKS

1)Lucene

2)Indri and

3)Wumpus

chosen because of their popularity & their influence on IR research

1) Apache **Lucene** - a free and open-source IR software library

Originally written in Java by Doug Cutting
Currently hosted by the Apache Foundation
The most successful open-source search engine
Known for its modularity and extensibility Scalable, High-Performance Indexing .

- small RAM requirements -- only 1MB heap
- incremental indexing as fast as batch indexing

•Powerful, Accurate and Efficient Search Algorithms

many powerful query types: phrase queries, wildcard queries, proximity queries, range queries and more
allows simultaneous update and searching
fast, memory-efficient

•**Cross-Platform Solution**

Available in both commercial and Open Source programs ,100%-pure Java

2) Indri

An academic information retrieval system written in C++

a new search engine from the Lemur project

a cooperative effort between the University of Massachusetts and Carnegie Mellon University to build language modeling information retrieval tools

well known for its high retrieval effectiveness

Indri Features

Effective - Best-in-class ad hoc retrieval performance

Flexible - Supports popular structured query operators from INQUERY

Usable
Includes both command line tools and a Java user interface
API can be used from Java, PHP, or C++

Works on Windows, Linux, Solaris and Mac OS X

Powerful
Can be used on a cluster of machines for faster indexing and retrieval
Scales to terabyte-sized collections

3) Wumpus

IR system developed at the University of Waterloo ,http://www.wumpus-search.org/

Scalable and has been used on text collections consisting of many hundreds of gigabytes of text and containing dozens of millions of documents

1.6 THE IMPACT OF THE WEB ON IR

(Part-A)
Outline the impact of the web on Information retrieval (APR/MAY'17)

The worldwide web is developed by Tim Berners Lee in 1990 at CERN to organize research documents available on the internet. It is combined idea of documents available by FTP with the idea of hypertext to link documents.

Query Operations:

Clients to the browser application to send URLs via HTTP to servers requesting a web page.

Web pages constructed using HTML or other markup language and consist of text, graphics, and sound plus embedded files.

Web search also has root in IR. IR helps users of finding information that matches their information needs.

In IR Two types of terms **Objective** and **Non Objective**.

Objective terms are extrinsic to semantic content, and there is generally no disagreement about how to assign them.

Non Objective terms are intended to reflect the information manifested in the document and there is no agreement about the choice or degree of applicability of these terms.

Keyword queries
A query is formulation of a user information need Keyword-based queries are popular
A sequence of single-word queries
Eg: retrieval

- **Boolean queries (using AND, OR, NOT)**

  Document = Logical conjunction of keywords
  Query = Boolean expression of keywords
  $R(D, Q) = D \rightarrow Q$
  e.g. $D = t_1 \wedge t_2 \wedge \ldots \wedge t_n$
  $Q = (t_1 \wedge t_2) \vee (t_3 \wedge \neg t_4)$
  $D \rightarrow Q$, thus $R(D, Q) = 1$.

- **Vector Space queries**

  Vector space = all the keywords encountered
  $<t_1, t_2, t_3, \ldots, t_n>$
  Document
  $D = < a_1, a_2, a_3, \ldots, a_n>$
  a_i = weight of t_i in D
  Query
  $Q = < b_1, b_2, b_3, \ldots, b_n>$
  b_i = weight of t_i in Q
  R(D,Q) = Sim(D,Q)

- **Phrase queries**

  Retrieve documents with a specific phrase (ordered list of contiguous words)
  "Information theory"

- **Proximity queries**

  List of words with specific maximal distance constraints between terms.
  Example: "dogs" and "race" within 4 words match "…dogs will begin the race…"

- **Full document queries**

  Queries should be full document

- **Natural language questions**

  Predictable queries

Retrieval System

.

Finding documents relevant to user queries. One way to find relevant documents on the web is to launch a **web robot**. **Web robot** is also called a wanderer, worm, walker spider etc….These software program receives a query then systematically explore the web to locate documents, evaluate their relevance and return a ranked ordered list of documents.

Selective systematic recall of logically stored information

Collect and organize information in one or more areas

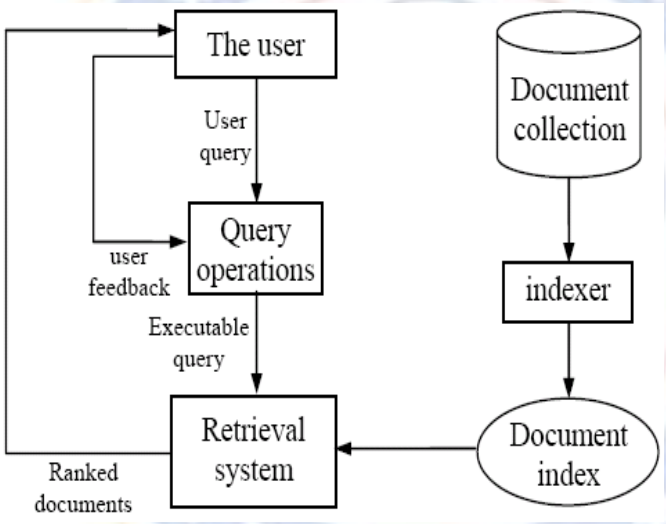
Set of operations, procedures units are indexed the resulting records are stored and displayed and so can be retrieved.

Indexing:

Indexing Processing involves Preprocessing and Storing of information into a repository.

Retrieval/Runtime process involves issuing a query, accessing the index to find the documents relevant the query.

In IR indexing is the process developing a document representation representation by an context descriptors or terms due to the documents.



The Impact of the Web on Information Retrieval

1.7 ROLE OF AI IN INFORMATION RETRIEVAL

PART-A

1. Define supervised learning and unsupervised learning.
2. Define AI in IR.

PART-B

1. Write a short notes on AI in IR (NOV/DEC'16**)**

"[AI] This is the use of computers to carry out tasks requiring reasoning on *world knowledge*, as exemplified by *giving responses* to questions in situation where one is dealing with only *partial knowledge* and with *indirect connectivity*"

- **Natural language processing**

Focussed on the syntactic, semantic and pragmatic analysis of natural language text and discourse.

Ability to analyse syntax and semantics could allow retrieval based on meaning rather than keywords

Methods for determining the sense of an ambiguous word based on context

Methods for identifying specific pieces of information in a document

Methods for answering specific questions.

- **Knowledge representation**
  - Expert systems
  - Ex: Logical formalisms, conceptual graphs, etc
- **Machine learning**

Focussed on the development of computational systems that improve their performance with experience.

Automated classification of examples based on learning concepts from labelled training examples (supervised learning)

Automated method for clustering unlabeled examples into meaningful groups(unsupervised learning)

Text Categorization

Automated hierarchical classification(Yahoo)

Adaptive Spam Filtering

Text Clustering

Clustering of IR query results

Automation formation of hierarchies

Short term: over a single session

Long term: over multiple searches by multiple users

- **Computer Vision**

Ex: OCR

- **Reasoning under uncertainty**

Ex: Dempster-Shafer, Bayesian networks, probability theory, etc

- **Cognitive theory**

Ex: User modelling

1.8 INFORMATION RETRIEVAL AND WEB SEARCH

Part - B
1. Compare information retrieval and web search.(APR/MAY'17)

Parameters	**Information Retrieval**	**Web Search**
Volume	Large	Huge

Data Quality	Clean , no duplicate4s	Noisy, Duplicates
Document	Text	HTML
Technique	Content Based	Link based
Data Change Rate	Infrequent	In flux
Format Diversity	Homogeneous	Heterogeneous
Matching	Small, Exact	Large, Partial Match, Best Match
Data Accessibility	Accessible	Partially Accessible
Parameters	Data Retrieval	Information retrieval
Example	Data Base Query	WWW Search
Inference	Deduction	Induction
Model	Deterministic	Probabilistic

1.9 COMPONENTS OF A SEARCH ENGINE

PART-A

1.What is Peer-to-peer search?(Nov/Dec'17)

PART-B

1.Explain in detail the components of Information Retrieval ans Search Engine(Apr/may'17)

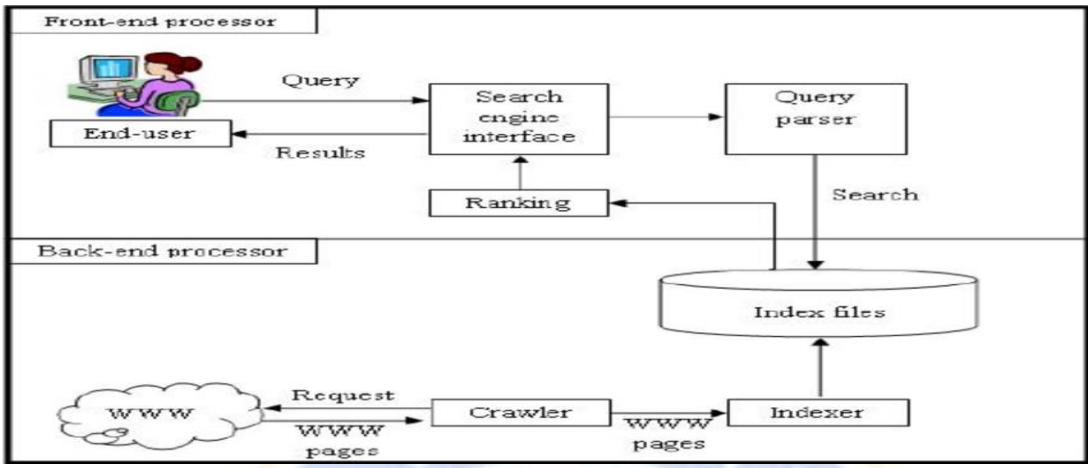
A search engine is the practical application of information retrieval techniques to large-scale text collections. A web search engine is the obvious example

1970s-MEDLINE FOR TEXT SEARCH
1980S-"information retrieval system"
Web search engines, such as Google and Yahoo

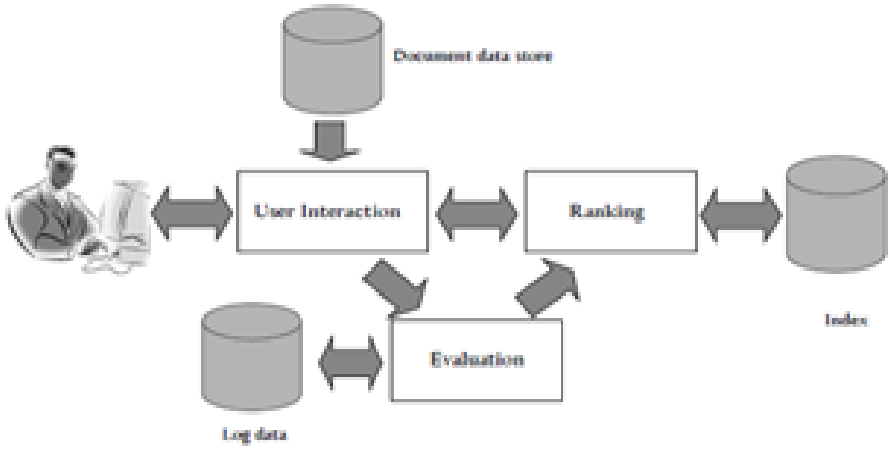
Must be able to capture, or *crawl, many terabytes of data, and then provide sub second* response times to millions of queries submitted every day from around the world **An architecture is designed to ensure that a system will satisfy the application requirements or goals. The two primary goals of a search engine are:**

- **Effectiveness (quality**): We want to be able to retrieve the most relevant set of documents possible for a query.
- **Efficiency (speed**): We want to process queries from users as quickly as possible

Search Engine Architecture



Indexing Process



Query Process

Text Acquisition

- **Crawler**

  The crawler component has the primary responsibility for identifying and acquiring documents for the search engine. There are a number of different types of crawlers, but the most common is the general web crawler.

  A web crawler is designed to follow the links on web pages to discover and download new pages. Although this sounds deceptively simple, there are significant challenges in designing a web crawler that can efficiently handle the huge volume of new pages on the Web, while at the same time ensuring that pages that may have changed since the last time a crawler visited a site are kept "fresh" for the search engine. A web crawler can be restricted to a single site, such as a university, as the basis for site search. **Focused, or topical, web crawlers** use classification techniques to restrict the pages that are visited to those that are likely to be about a specific topic.

  This type of crawler may be used by a **vertical or topical search** application, such as a search engine that provides access to medical information on web pages.
  For **enterprise search**, the crawler is adapted to discover and update all documents and web pages related to a company's operation. An enterprise document crawler follows links to discover both external and internal (i.e., restricted to the corporate intranet) pages, but also must scan both corporate and personal di- rectories to identify email, word processing documents, presentations, database records, and other company information. **Document crawlers** are also used for **desktop search**, although in this case only the user's personal directories need to be scanned.

- **Feeds:**

  Document feeds are a mechanism for accessing a real-time stream of documents. For example, a news feed is a constant stream of news stories and updates. In con- trast to a crawler, which must discover new documents, a search engine acquires new documents from a feed simply by monitoring it.

  Eg: RSS2 is a common standard used for web feeds for content such as **news, blogs, or video.**

  The reader monitors those feeds and provides new content when it arrives. **Radio and television feeds are also used in some search applications, where the "documents" contain automatically segmented audio and video streams**, together with associated text from closed captions or speech recognition.

- **Conversion:**

  The documents found by a crawler or provided by a feed are rarely in plain text. Instead, they come in a variety of formats, such as **HTML, XML, Adobe PDF, Microsoft Word, Microsoft PowerPoin**t and so on, search engines require that these documents be converted into a consistent text plus metadata format.

- **Document data store:**

  The document data store is a database **used to manage large numbers of documents** and the structured data that is associated with them. The document contents are typically stored in compressed form for efficiency. The structured data consists of document metadata and other information extracted from the documents, such as links and anchor text.

- **Text Transformation**

  Parser:

  The parsing component is responsible **for processing the sequence of text tokens in the document to recognize structural elements such as titles, figures, links, and headings**. Tokenizing the text is an important first step in this process. In many cases, tokens are the same as words. Both document and query text must be transformed into tokens in the same manner so that they can be easily compared.

- **Stopping:**

  The stopping component has the simple task of removing common words from the stream of tokens that become index terms.

- **Stemming:**

  Stemming is another word-level transformation. The task of the stemming com- ponent (or stemmer) is to group words that are derived from a common stem. Grouping “fish”, “fishes”, and “fishing” is one example

- **Link extraction and analysis:**

  Links and the corresponding anchor text in web pages can readily be identified and extracted during document parsing. Extraction means that this information is recorded in the document data store, and can be indexed separately from the general text content.
  Eg: Page Rank

- **Information extraction:**

Information extraction is used to identify index terms that are more complex than single words. This may be as simple as words in bold or words in headings, but in general may require significant additional computation.

- **Classifier**

  The classifier component identifies class-related metadata for documents or parts of documents. This covers a range of functions that are often described separately. Classification techniques assign predefined class labels to documents. These labels typically represent topical categories such as “sports”, “politics”, or “business”. Two

- **Index Creation**:

  The task of the document statistics component is simply to gather and record statistical information about words, features, and documents. This information is used by the ranking component to compute scores for documents.

- **Weighting**
  Index term weights reflect the relative importance of words in documents, and are used in computing scores for ranking. The specific form of a weight is determined by the retrieval model.

  The weighting component calculates weights using the document statistics and stores them in lookup tables.

  Weights could be calculated as part of the query process and some types of weights require information about the query, but by doing as much calculation as possible during the indexing process, the efficiency of the query process will be improved.

Desktop search is the personal version of enterprise search, where the information sources are the files stored on an individual computer, including email messages and web pages that have recently been browsed.

Peer-to-peer search involves finding information in networks of nodes or computers without any centralized control. This type of search began as a file sharing tool for music but can be used in any community based on shared interests, or even shared locality in the case of mobile devices.

Ad hoc search Search based on a user query

1.10 CHARACTERIZING THE WEB.

PART-A

1.What are the Performance measure for Search Engine?(Nov/Dec'17)

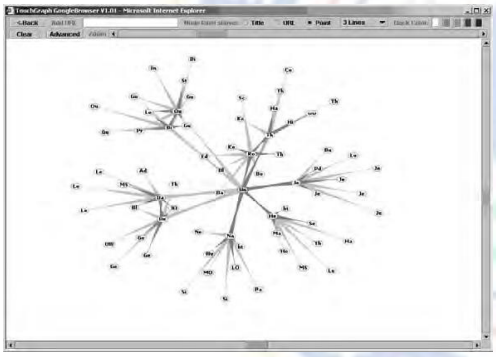
The Web

Many studies investigated the web in specific countries.
Many properties of a subset of the web are valid for the global web.
Still , no understanding of the web and its dynamics.

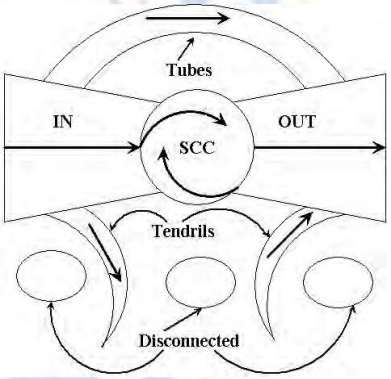
Characteristics:

Structure of the Web

A web crawl is a task performed by special purpose software that surfs the Web, starting from a multitude of web pages and then continuously following the hyperlinks it encountered until the end of the crawl.



Structure of the Web



Bow Tie architecture

Bow-Tie Architecture

The central core of the Web (the knot of the bow-tie) is the strongly connected component (SCC), which means that for any two pages in the SCC, a user can navigate from one of them to the other and back by clicking on links embedded in the pages encountered.

A user browsing a page in the SCC can always reach any other page in the **SCC** by traversing some path of links.

The relative size of the SCC turned out to be 27.5% of the crawled portion of the Web.

The left bow, called **IN**, contains pages that have a directed path of links leading to the SCC and its relative size was 21.5% of the crawled pages. Pages in the left bow might be either new pages that have not yet been linked to, or older web pages that have not become popular enough to become part of the SCC.

The right bow, called **OUT**, contains pages that can be reached from the SCC by following a directed path of links and its relative size was also 21.5% of the crawled pages. Pages in the right bow might be pages in e-commerce sites that have a policy not to link to other sites.

The other components are the **"tendrils"** and the "tubes" that together comprised 21.5% of the crawled portion of the Web, and further "disconnected" components whose total size was about 8% of the crawl.

A web page in Tubes has a directed path from **IN** to **OUT** bypassing the SCC, and a page in .

Tendrils can either be reached from IN or leads into OUT.

The pages in Disconnected are not even weakly connected to the SCC; that is, even if we ignored the fact that hyperlinks only allow forward navigation, allowing them to be traversed backwards as well as forwards, we still could *not* reach reach the SCC from them.

Performance measure of Search Engine

A document is defined to be technically relevant if it fulfils all the conditions posed by the query: all search terms and phrases that suppose to appear in the document do appear, and all terms and phrases that are supposed to be missing from the document, i.e. terms preceded by a minus sign or a NOT operator, do not appear in the document.

\

****

UNIT II

Boolean and vector-space retrieval models- Term weighting – TF-IDF weighting- cosine similarity – Preprocessing – Inverted indices – efficient processing with sparse vectors – Language Model based IR – Probabilistic IR –Latent Semantic Indexing – Relevance feedback and query expansion

2.1 BOOLEAN AND VECTOR-SPACE RETRIEVAL MODELS

Part-B

Explain Vector space retrieval model with an example.(Apr/May'17)

One of the primary goals has been to understand and formalize the processes that underlie a person making the decision that a piece of text is relevant to his information need

Good models should produce outputs that correlate well with human decisions on relevance. To put it another way, ranking algorithms based on good retrieval models will retrieve relevant documents near the top of the ranking (and consequently will have high effectiveness).

Eg::ranking algorithms for general search improved in effectiveness by over 100% in the 1990s, as measured using the TREC test collections.

The Boolean retrieval model was used by the earliest search engines and is still in use today.

It is also known as **exact-match retrieval** since documents are retrieved if they exactly match the query specification, and otherwise are not retrieved.

Al- though this defines a very simple form of ranking, Boolean retrieval is not generally described as a ranking algorithm. This is because the Boolean retrieval model assumes that all documents in the retrieved set are equivalent in terms of relevance.

In addition to the assumption that relevance is binary. The name Boolean comes from the fact that there only two possible outcomes for query evaluation **(TRUE and FALSE**) and because the query is usually specified using operators from Boolean logic *(AND, OR, NOT).*

Advantages to Boolean retrieval

The results of the model are very predictable and easy to explain to users.

The operands of a Boolean query can be any document feature, not just words, so it is straightforward to incorporate metadata such as a document date or document type in the query specification.
Retrieval is usually more efficient than ranked retrieval because documents can be rapidly eliminated from consideration in the scoring process.

Drawback

The effectiveness depends entirely on the user. Because of the lack of a sophisticated ranking algorithm, simple queries will not work well.
All documents containing the specified query words will be retrieved, and this retrieved set will **be presented to the user in some orde**r, such as by publication date, that has little to do with relevance.
It is possible to construct complex Boolean queries that narrow the retrieved set to mostly relevant documents, **but this is a difficult task**

Example of Boolean query formulation, consider the following queries for a search engine that has indexed a collection of news stories. The simple query:

lincoln

would retrieve a large number of documents that mention Lincoln cars and places named Lincoln in addition to stories about President Lincoln. All of these documents would be equivalent in terms of ranking in the Boolean retrieval model, regardless of how many times the word "lincoln" occurs or in what context it occurs. Given this, the user may attempt to narrow the scope of the search with the following query:

president AND lincoln

This query will retrieve a set of documents that contain both words, occurring anywhere in the document. If there are a number of stories involving the management of the Ford t or Company and Lincoln cars, these will be retrieved in the same set as stories about President Lincoln, for example:

Ford Company today announced that Darryl Hazel will succeed Brian Kelley as president of Lincoln

If enough of these types of documents were retrieved, the user may try to eliminate documents about cars by using the NOT operator, as follows:

president AND lincoln AND NOT (automobile OR car)

This would remove any document that contains even a single mention of the words "automobile" or "car" anywhere in the document. The use of the NOT operator, in general, removes too many relevant documents along with non-relevant documents

and is not recommended. For example, one of the top-ranked documents in a web search for "President Lincoln" was a biography containing the sentence:

Lincoln's body departs Washington in a nine-car funeral train.

Using NOT (automobile OR car) in the query would have removed this document. If the retrieved set is still too large, the user may try to further narrow the query by adding in additional words that should occur in biographies:

president AND lincoln AND biography AND life AND birthplace AND gettysburg AND NOT (automobile OR car)

Unfortunately, in a Boolean search engine, putting too many search terms into the query with the AND operator often results in nothing being retrieved. To avoid this, the user may try using an OR instead:

president AND lincoln AND (biography OR life OR birthplace OR gettysburg) AND NOT (automobile OR car)

This will retrieve any document containing the words "president" and "lincoln", along with any one of the words "biography", "life", "birthplace", or "gettysburg" (and does not mention "automobile" or "car").
After all this, we have a query that may do a reasonable job at retrieving a set containing some relevant documents, but we still can't specify which words are more important or that having more of the associated words is better than any one of them. For example, a document containing the following text was retrieved at rank 500 by a web search (which does use measures of word importance):

President's Day - Holiday activities - crafts, mazes, word searches, ... "The Life of Washington" Read the entire book online! Abraham Lincoln Re- search Site ...

Vector space model

The **vector space model** was the basis for most of the research in information retrieval in the 1960s and 1970s, and papers using this model continue to appear at conferences

Advantage
Being a simple and intuitively appealing framework for implementing term weighting, ranking, and relevance feedback.

In this model, documents and queries are assumed to be part of a t-dimensional vector space, where t is the number of index terms (words, stems, phrases, etc.). A document Di is represented by a vector of index terms:

Di = (di1, di2, . . . , dit),

where dij represents the weight of the jth term. A document collection contain- ing n documents can be represented as a matrix of term weights, where each row represents a document and each column describes weights that were assigned to a term for a particular document:

	$Term_1$	$Term_2$	. . .	$Term_t$
Doc_1	d_{11}	d_{12}	. . .	d_{1t}
Doc_2	d_{21}	d_{22}	. . .	d_{2t}
.	.			
Doc_n	d_{n1}	d_{n2}	. . .	d_{nt}

The **term-document matrix** has been rotated so that now the terms are the rows and the documents are the columns. The term weights are simply the count of the terms in the document. Stop words are not indexed in this example, and the words have been stemmed. Document D3, for example, is represented by the vector (1, 1, 0, 2, 0, 1, 0, 1, 0, 0, 1).

Queries are represented the same way as documents. That is, a query Q is represented by a vector of t weights:

Q = (q1, q2, . . . , qt),

where qj is the weight of the jth term in the query. If, for example the query was "tropical fish", then using the vector representation in, the query would be (0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1). One of the appealing aspects of the vector space model is the use of simple diagrams to visualize the documents and queries. Typically, they are shown as points or vectors in a three-dimensional picture, as in

$_1$ Tropical Freshwater Aquarium Fish.

D_2 Tropical Fish, Aquarium Care, Tank Setup.

D_3 Keeping Tropical Fish and Goldfish in Aquariums, and Fish Bowls.

D_4 The Tropical Tank Homepage - Tropical Fish and Aquariums.

Terms	**Documents**			
	D_1	**D $_2$**	**D $_3$**	**D_4**
aquarium	1	1	1	1
bowl	0	0	1	0
care	0	1	0	0
fish	1	1	2	1
freshwater	1	0	0	0
goldfish	0	0	1	0
homepage	0	0	0	1

keep	0	0	1	0
setup	0	1	0	0
tank	0	1	0	1
tropical	1	1	1	2

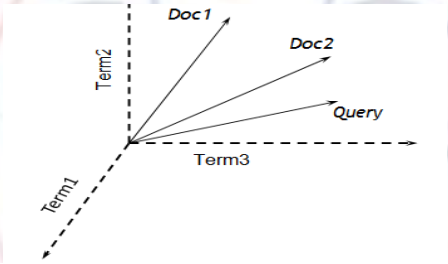
Term-document matrix for a collection of four documents

Documents could be ranked by computing the distance between the points representing the documents and the query.

A **similarity measure is** used (rather than a distance or dissimilarity mea- sure), so that the documents with the highest scores are the most similar to the query. The most successful of these is the cosine correlation similarity measure.

The cosine correlation measures the cosine of the angle between the query and the document vectors.

When the vectors are normalized so that all documents and queries are represented by vectors of equal length, the cosine of the angle between two identical vectors will be 1 (the angle is zero), and for two vectors that do not share any non-zero terms, the cosine will be 0. The cosine measure is de- fined as:



$$\text{Cosine}(D_i, Q) = \frac{\sum_{j=1} d_{ij} \cdot q_i}{\sqrt{\sum_{j=1} d_{ij}^2 \cdot \sum_{j=1} q_j^2}}$$

As an example, consider two documents D1 = (0.5, 0.8, 0.3) and D2 = (0.9, 0.4, 0.2) indexed by three terms, where the numbers represent term weights. Given the query Q = (1.5, 1.0, 0) indexed by the same terms, the cosine mea- sures for the two documents are:

$$\text{Cosine}(D_1, Q) = \frac{(0.5 \times 1.5) + (0.8 \times 1.0)}{(0.5^2 + 0.8^2 + 0.3^2)(1.5^2 + 1.0^2)}$$
$$= \frac{1.55}{\sqrt{(0.98 \times 3.25)}} = 0.87$$

$$\text{Cosine}(D_2, Q) = \frac{(0.9 \times 1.5) + (0.4 \times 1.0)}{(0.9^2 + 0.4^2 + 0.2^2)(1.5^2 + 1.0^2)}$$
$$= \frac{1.75}{\sqrt{(1.01 \times 3.25)}} = 0.97$$

2.2 TERM WEIGHTING

Part-B

Briefly explain weighting and cosine similarity(Nov/Dec'16)

We assign to each term in a document a weight for the term that depends on the number of occurrences of the term in the document.

We would like to computer a score between a query term f and document d based on the weight of t in d . The simplest approach is to assign the weight to the equal to the number of occurrences of term t and document d.

The weighting scheme is referred to as term frequency and is denoted by $tf_{ij.}$

Documents are also treated as a "bag" of words or terms. ν Each document is represented as a vector. ν However, the term weights are no longer 0 or 1. Each term weight is computed based on some variations of TF or TF-IDF scheme.

Term Frequency (TF) Scheme: The weight of a term t i in document dj is the number of times that t i appears in dj , denoted by f ij. Normalization may also be applied.

2.3 TF-IDF TERM WEIGHTING

- The most well known weighting scheme
  - TF: still **term frequency**
  - IDF: **inverse document frequency**.

  N: total number of docs

  df_i: the number of docs that t_i appears.

$$tf_{ij} = \frac{f_{ij}}{\max\{f_{1j}, f_{2j}, \ldots, f_{|V|j}\}}$$

$$idf_i = \log \frac{N}{df_i}$$

- The final TF-IDF term weight is:

$$w_{ij} = tf_{ij} \times idf_i.$$

A term occurring frequently in the document but rarely in the rest of the collection is given high weight.

Many other ways of determining term weights have been proposed.

Experimentally, *tf-idf* has been found to work well.

$$w_{ij} = tf_{ij}\, idf_i = tf_{ij} \log_2 (N/ df_i)$$

Given a document containing terms with given frequencies:

A(3), B(2), C(1)

Assume collection contains 10,000 documents and

document frequencies of these terms are:

A(50), B(1300), C(250)

Then:

A: tf = 3/3; idf = $\log_2$(10000/50) = 7.6; tf-idf = 7.6

B: tf = 2/3; idf = $\log_2$ (10000/1300) = 2.9; tf-idf = 2.0

C: tf = 1/3; idf = $\log_2$ (10000/250) = 5.3; tf-idf = 1.8

- Query vector is typically treated as a document and also tf-idf weighted.
- Alternative is for the user to supply weights for the given query terms.

2.4 COSINE SIMILARITY

- A similarity measure is a function that computes the *degree of similarity* between two vectors.
- Using a similarity measure between the query and each document:
  - It is possible to rank the retrieved documents in the order of presumed relevance.

- It is possible to enforce a certain threshold so that the size of the retrieved set can be controlled.

- Similarity between vectors for the document d_i and query q can be computed as the vector inner product (a.k.a. dot product):

  $\text{sim}(d_j,q) = d_j \bullet q =$

  where w_{ij} is the weight of term i in document j and w_{iq} is the weight of term i in the query

- For binary vectors, the inner product is the number of matched query terms in the document (size of intersection).
- For weighted term vectors, it is the sum of the products of the weights of the matched terms.
- The inner product is unbounded.
- Favors long documents with a large number of unique terms.
- Measures how many terms matched but not how many terms are *not* matched.

Binary:

- D = 1, 1, 1, 0, 1, 1, 0
- Q = 1, 0 , 1, 0, 0, 1, 1

Size of vector = size of vocabulary = 7
0 means corresponding term not found in document or query

sim(D, Q) = 3

\Weighted:

$D_1 = 2T_1 + 3T_2 + 5T_3 \qquad D_2 = 3T_1 + 7T_2 + 1T_3$

$Q = 0T_1 + 0T_2 + 2T_3$

$\text{sim}(D_1 , Q) = 2*0 + 3*0 + 5*2 = 10$

$\text{sim}(D_2 , Q) = 3*0 + 7*0 + 1*2 = 2$

- Cosine similarity measures the cosine of the angle between two vectors.
- Inner product normalized by the vector lengths.
- $\text{CosSim}(\boldsymbol{d}_j, \boldsymbol{q}) = \dfrac{\vec{d}_j \cdot \vec{q}}{\left|\vec{d}_j\right| \cdot \left|\vec{q}\right|} = \dfrac{\sum_{i=1}^{t} (w_{ij} \cdot w_{iq})}{\sqrt{\sum_{i=1}^{t} w_{ij}^2 \cdot \sum_{i=1}^{t} w_{iq}^2}}$

$D_1 = 2T_1 + 3T_2 + 5T_3$ CosSim(D_1 , Q) = 10 / √(4+9+25)(0+0+4) = 0.81

$D_2 = 3T_1 + 7T_2 + 1T_3$ CosSim(D_2 , Q) = 2 / √(9+49+1)(0+0+4) = 0.13

$Q = 0T_1 + 0T_2 + 2T_3$

D_1 is 6 times better than D_2 using cosine similarity but only 5 times better using inner product.

2.5 PREPROCESSING

PART-A

Define Document Preprocessing.(Nov/Dec'16)

- Document preprocessing is a procedure which can be divided mainly into five text operations (or transformations):

(1) **Lexical analysis of the text** with the objective of treating digits, hyphens, punctuation marks, and the case of letters.

(2) **Elimination of stop-words** with the objective of filtering out words with very low discrimination values for retrieval purposes.

Eg: the , an , a ,m as and…..

(3) **Stemming** of the remaining words with the objective of removing affixes (i.e., prefixes and suffixes) and allowing the retrieval of documents containing syntactic variations of query terms (e.g., connect, connecting, connected, etc).

- Techniques used to find out the root/stem of a word. E.g.,
  - user
  - users
  - used
  - using

  engineering
  engineered
  engineer
- stem: use engineer

(4) **Selection of index terms** to determine which words/stems (or groups of words) will be used as an indexing elements. Usually, the decision on whether a particular word will be used as an index term is related to the syntactic nature of the word. In fact , noun words frequently carry more semantics than adjectives, adverbs, and verbs.

(5) **Construction of term categorization** structures such as a thesaurus, or extraction of structure directly represented in the text, for allowing the expansion of the original query with related terms (a usually useful procedure).

Lexical Analysis of the text

Task: convert strings of characters into sequence of words.

- Main task is to deal with spaces, e.g, multiple spaces are treated as one space.
- Digits—ignoring numbers is a common way. Special cases, 1999, 2000 standing for specific years are important. Mixed digits are important, e.g., 510B.C. 16 digits numbers might be credit card #.
- Hyphens: state-of-the art and "state of the art" should be treated as the same.
- Punctuation marks: remove them. Exception: 510B.C
- Lower or upper case of letters: treated as the same.
- Many exceptions: semi-automatic.

Elimination of Stopwords

- words appear too often are not useful for IR.
- Stopwords: words appear more than 80% of the documents in the collection are stopwords and are filtered out as potential index words.

Stemming

- A Stem: the portion of a word which is left after the removal of its affixes (i.e., prefixes or suffixes).
- Example: connect is the stem for {connected, connecting connection, connections}
- Porter algorithm: using suffix list for suffix stripping. S→Φ, sses →ss, etc.

Index terms selection

- Identification of noun groups
- Treat nouns appear closely as a single component, e.g., computer science

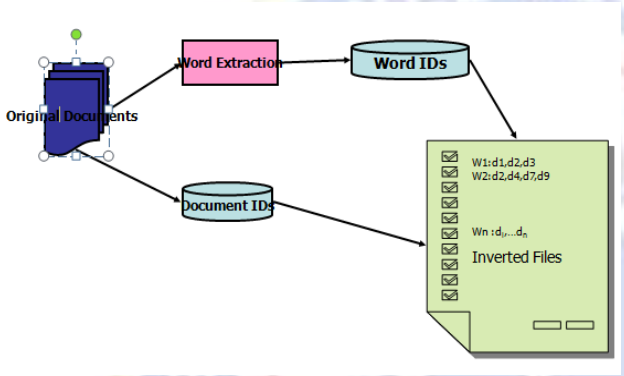
$$P(a \cap b \cap c) = P(a) \cdot P(b) \cdot P(c)$$

where P (a b c) is the joint probability, or the probability that all three words occur in a document, and P (a), P (b), and P (c) are the probabilities of each word occurring in a document. A search engine will always have access to the number of documents that a word occurs in (fa, fb, and fc),7 and the number of documents in the collection (N), so these probabilities can easily be estimated as P (a) = fa/N , P (b) = fb/N , and P (c) = fc/N This gives us fabc = N · fa/N · fb/N · fc/N = (fa · fb · fc)/N 2 where fabc is the estimated size of the result set.

Word(s)	Document Frequency	Estimated Frequency
tropical	120,990	
fish	1,131,855	
aquarium	26,480	
breeding	81,885	
tropical fish	18,472	5,433
tropical aquarium	1,921	127
tropical breeding	5,510	393
fish aquarium	9,722	1,189
fish breeding	36,427	3,677
aquarium breeding	1,848	86
tropical fish aquarium	1,529	6
tropical fish breeding	3,629	18

2.6 INVERTED INDICES

- **Arrangement of data (data structure) to permit fast searching**



Creating Inverted file

- Map the file names to file IDs
- Consider the following Original Documents

D1	The Department of Computer Science was established in 1984.
D2	The Department launched its first BSc(Hons) in Computer Studies in 1987.
D3	followed by the MSc in Computer Science which was started in 1991.
D4	The Department also produced its first PhD graduate in 1994.
D5	Our staff have contributed intellectually and professionally to the advancements in these fields.

D1	The Department of Computer Science was established in 1984.
D2	The Department launched its first BSc(Hons) in Computer Studies in 1987.
D3	followed by the MSc in Computer Science which was started in 1991.
D4	The Department also produced its first PhD graduate in 1994.
D5	Our staff have contributed intellectually and professionally to the advancements in these fields.

After stemming, make lowercase (option), delete numbers (option)

D1	depart comput scienc establish
D2	depart launch bsc hons comput studi
D3	follow msc comput scienc start
D4	depart produc phd graduat
D5	staff contribut intellectu profession advanc field

Creating Inverted file (unsorted)

Words	Documents	Words	Documents
depart	d1,d2,d4	produc	d4
comput	d1,d2,d3	phd	d4
scienc	d1,d3	graduat	d4
establish	d1	staff	d5
launch	d2	contribut	d5
bsc	d2	intellectu	d5
hons	d2	profession	d5
studi	d2	advanc	d5
follow	d3	field	d5
msc	d3		
start	d3		

2.7 EFFICIENT PROCESSING WITH SPARSE VECTORS

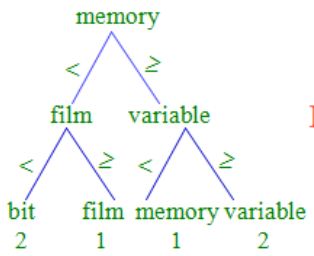
- Vocabulary and therefore dimensionality of vectors can be very large, ~10^4 .
- However, most documents and queries do not contain most words, so vectors are sparse (i.e. most entries are 0).
  Need efficient methods for storing and computing with sparse vectors.

- Store vectors as linked lists of non-zero-weight tokens paired with a weight.
  - Space proportional to number of unique tokens (n) in document.
  - Requires linear search of the list to find (or change) the weight of a specific token.

  Requires quadratic time in worst case to compute vector for a document:

$$\sum_{i=1}^{n} i = \frac{n(n+1)}{2} = O(n^2)$$

- Index tokens in a document in a balanced binary tree or trie with weights stored with tokens at the leaves.



Balanced Binary Tree

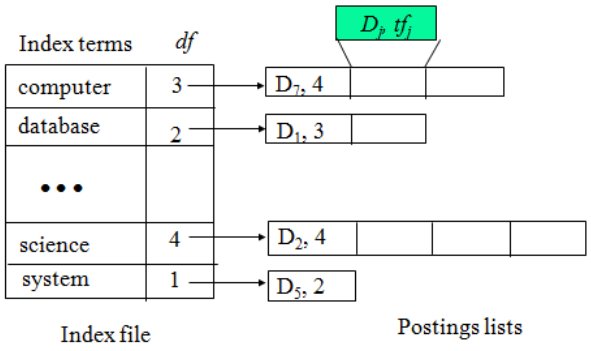
Space overhead for tree structure: ~2n nodes.

O(log *n*) time to find or update weight of a specific token.

O(*n* log *n*) time to construct vector.

Need software package to support such data structures.

Implementation Based on Inverted Files



Algorithm

Create an empty HashMap, H;

For each document, D, (i.e. file in an input directory):

Create a HashMapVector,V, for D;

For each (non-zero) token, T, in V:

If T is not already in H, create an empty

TokenInfo for T and insert it into H;

Create a TokenOccurence for T in D and

add it to the occList in the TokenInfo for T;

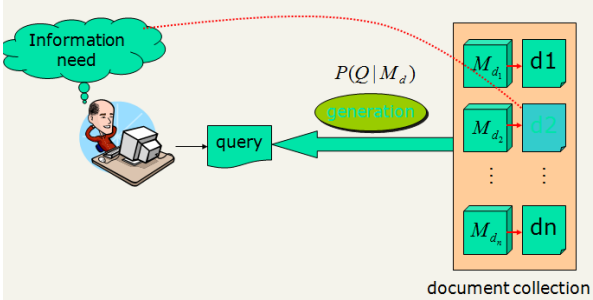
Compute IDF for all tokens in H;

Compute vector lengths for all documents in H;

2.8 LANGUAGE MODEL BASED IR

A common search heuristic is to use words that you expect to find in matching documents as your query – why, I saw Sergey Brin advocating that strategy on late night TV one night in my hotel room, so it must be good!

The LM approach directly exploits that idea!

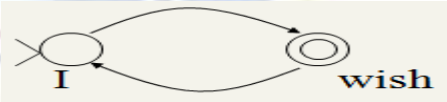


Traditional generative model: generates strings

Finite state machines or regular grammars, etc.

Example:

I wish
I wish I wish
I wish I wish I wish
I wish I wish I wish I wish
……



Models *probability* of generating strings in the language (commonly all strings over alphabet ∑)

Model M	
0.2	the
0.1	a
0.01	man
0.01	woman
0.03	said
0.02	likes
…	

the	man	likes	the	woman
0.2	0.01	0.02	0.2	0.01

multiply

P(s | M) = 0.00000008

A statistical model for generating text

Probability distribution over strings in a given language

2.9 PROBABILISTIC IR

Part-A

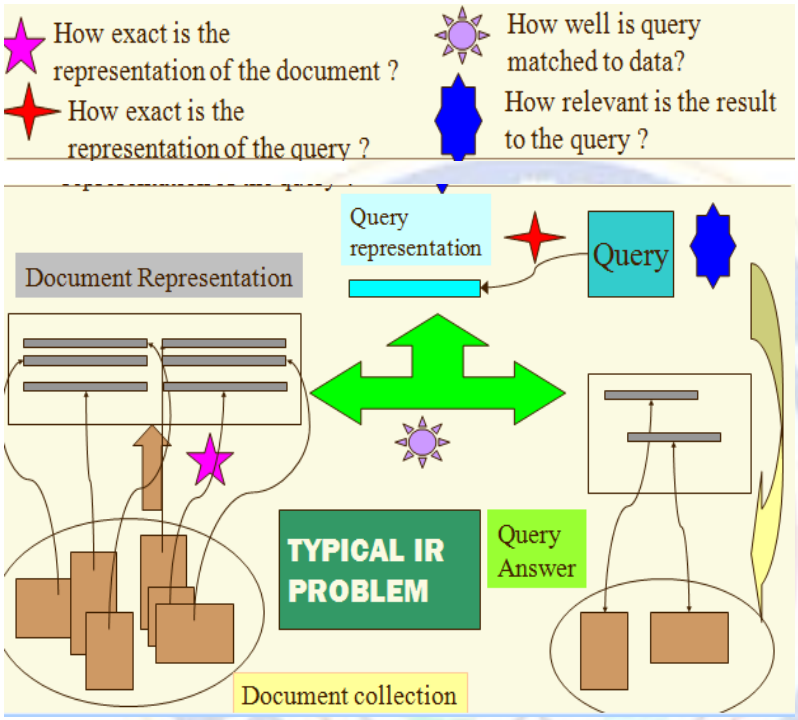
State Bayes Rule (Nov/Dec'17

Part-B

Write short notes on the probabilistic relevance feedback. **(Nov/Dec'17)**

Information Retrieval deals with Uncertain Information.

Probability theory seems to be the most natural way to quantify uncertainty



Probability Ranking Principle

- If a reference retrieval system's **response to each request** is *a ranking of the documents in the collections* in ***order of decreasing probability of usefulness*** to the user who submitted the request ...
- where the probabilities are estimated as accurately a possible on the basis of whatever data made available to the system for this purpose
- then the **overall effectiveness** of the system to its users **will be the best** that is obtainable on the basis of that data
- Probabilistic model
- Information Retrieval deals with Uncertain Information

Let *a, b* be two events

Bayesian formulas

Let *x* be a document in the collection.

Let *R* represent **relevance** of a document w.r.t. given (fixed)

query and let *NR* represent **non-relevance.**

Need to find p(*R*|*x)* - probability that a retrieved document *x* is **relevant.**

$$p(a|b)p(b) = p(a \cap b) = p(b|a)p(a)$$
$$p(a|b) = \frac{p(b|a)p(a)}{p(b)}$$
$$p(\bar{a}|b)p(b) = p(b|\bar{a})p(\bar{a})$$

Let *x* be a document in the collection.
Let *R* represent **relevance** of a document w.r.t. given (fixed) query and let *NR* represent **non-relevance.**
Need to find p(*R*|*x)* - probability that a retrieved document *x* Is relevant.

$$p(R|x) = \frac{p(x|R)p(R)}{p(x)}$$
$$p(NR|x) = \frac{p(x|NR)p(NR)}{p(x)}$$

p(*R)*,p(*NR*) - prior probability of retrieving a (non) relevant document
p(*x*|*R), p(x*|*NR)* - probability that if a relevant (non-relevant) document is retrieved, it is *x.*
If p(*R*|*x)* > p(*NR*|*x*) then *x* <u>is relevant</u>, otherwise *x* <u>is not relevant</u>

2.10 LATENT SEMANTIC INDEXING

Part-B

Give an example for latent semantic indexing and explain the same. .(Apr/May'17)

Term-document matrices are very large
But the number of topics that people talk about is small (in some sense)
• Clothes, movies, politics, …

Eigenvalues & Eigenvectors

Eigenvectors (for a square m·m matrix S)

$$\mathbf{S}\mathbf{v} = \lambda\mathbf{v}$$

(right) eigenvector $\mathbf{v} \in \mathbb{R}^m \neq \mathbf{0}$ eigenvalue $\lambda \in \mathbb{R}$

Example

$$\begin{pmatrix} 6 & -2 \\ 4 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 2 \end{pmatrix} = \begin{pmatrix} 2 \\ 4 \end{pmatrix} = 2 \begin{pmatrix} 1 \\ 2 \end{pmatrix}$$

$$\mathbf{S}\mathbf{v} = \lambda\mathbf{v} \iff (\mathbf{S} - \lambda\mathbf{I})\,\mathbf{v} = \mathbf{0}$$

only has a non-zero solution if $|\mathbf{S} - \lambda\mathbf{I}| = 0$

This is a mth order equation in λ which can have at most m distinct solutions (roots of the characteristic polynomial) - can be complex even though S is real.

6 CS6007 - IR - Unit 2

$$S = \begin{vmatrix} 30 & 0 & 0 \\ 0 & 20 & 0 \\ 0 & 0 & 1 \end{vmatrix}$$

has eigen 30,20,1 with corresponding eigen vectors.

$$V1 = \begin{vmatrix} 1 \\ 0 \\ 0 \end{vmatrix} \quad V2 = \begin{vmatrix} 0 \\ 1 \\ 0 \end{vmatrix} \quad V3 = \begin{vmatrix} 0 \\ 0 \\ 1 \end{vmatrix}$$

On each eigen vectors S acts as a multiple of the identity matrix, but as a different multiple on each.

2.11 RELEVANCE FEEDBACK AND QUERY EXPANSION

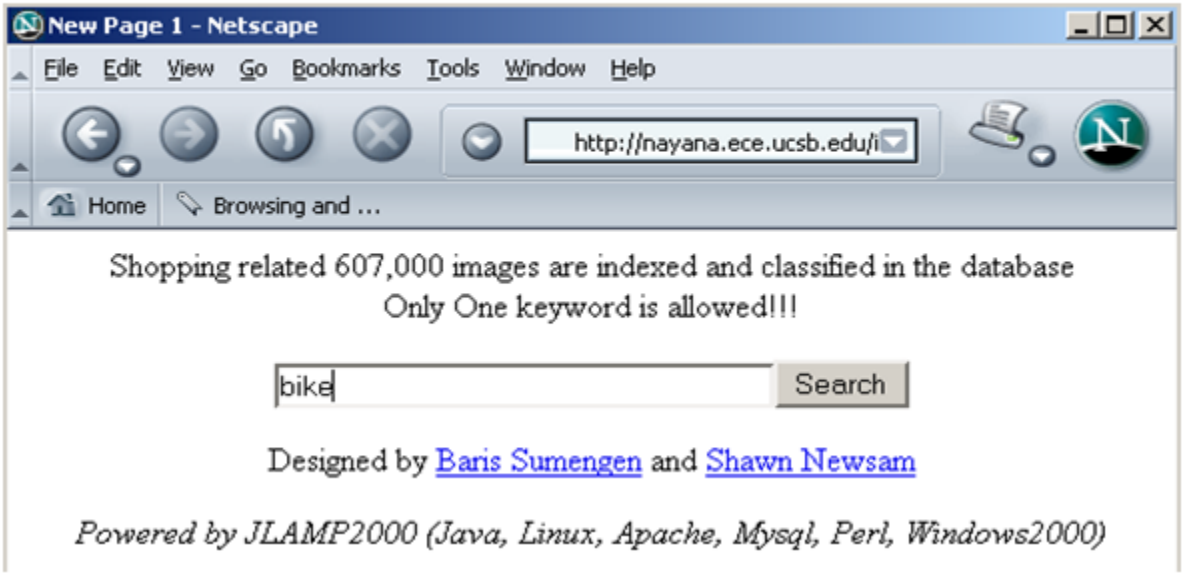
Part-B

Write short notes on query expansion. .(Apr/May'17)
What is Relevance feedback? Explain with an example an algorithm for relevance feedback..(Apr/May'17)(Nov/Dec'16)

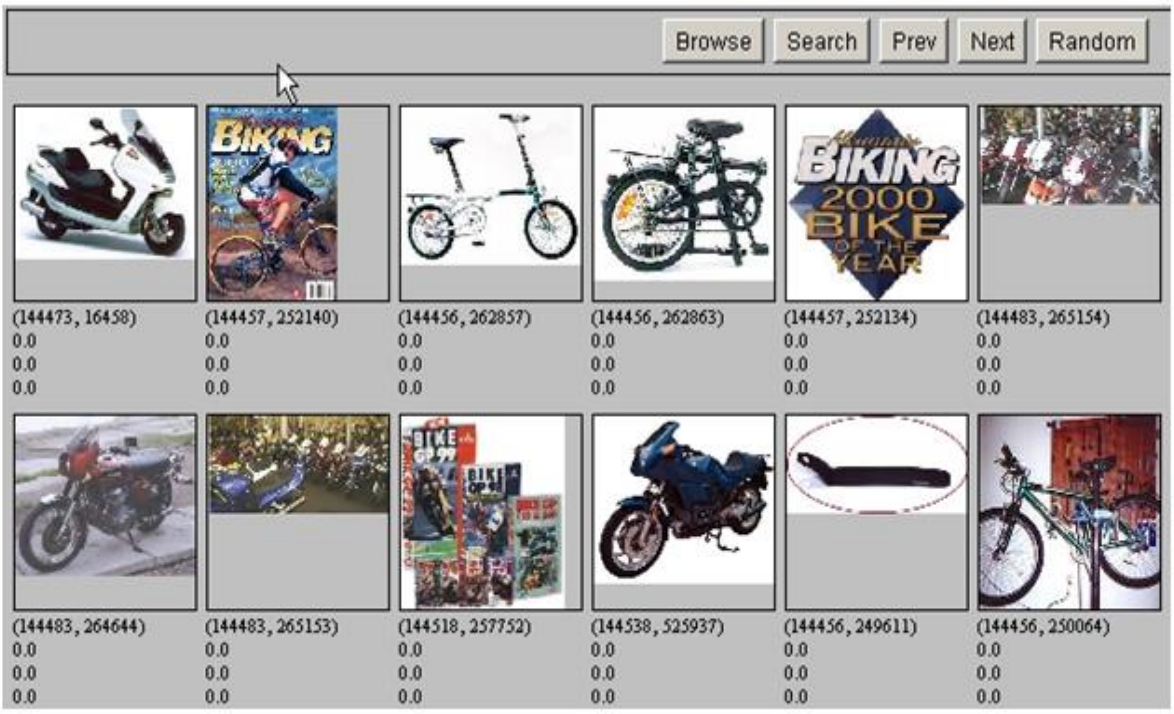
- Relevance feedback: user feedback on relevance of docs in initial set of results
  - User issues a (short, simple) query
  - The user marks some results as relevant or non-relevant.
  - The system computes a better representation of the information need based on feedback.
  - Relevance feedback can go through one or more iterations.
- We will use *ad hoc retrieval* to refer to regular retrieval without relevance feedback.

Relevance Feed back Example:

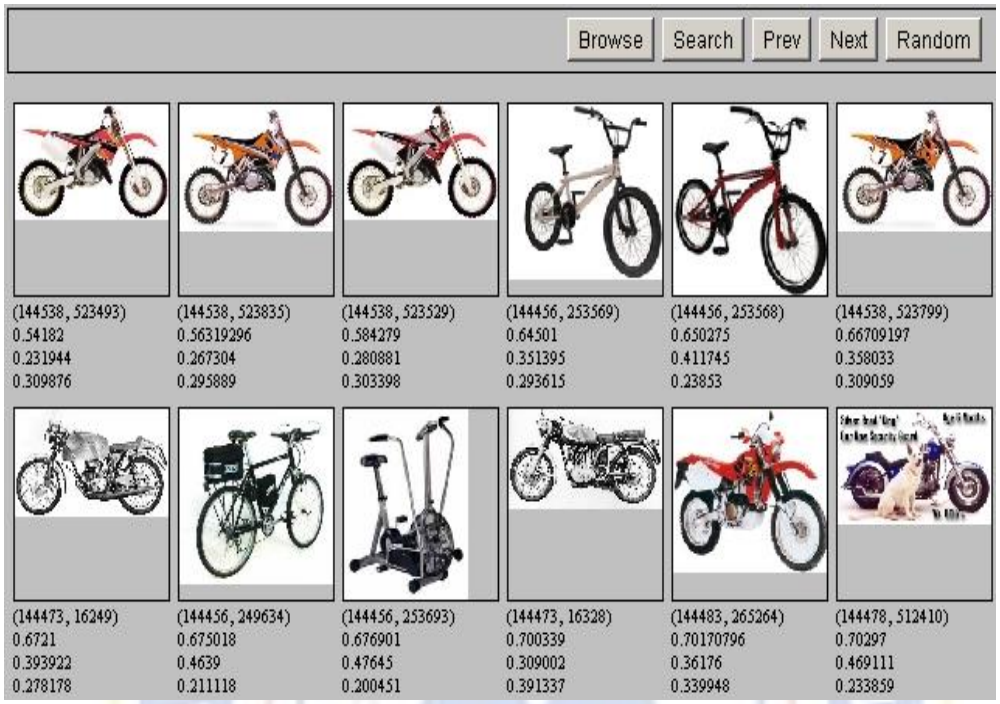
- Image search engine
  http://nayana.ece.ucsb.edu/imsearch/imsearch.html



Results for Initial Query



Results after Relevance feedback



The process of query modification is commonly referred as

R**elevance feedback**, when the user provides information on relevant documents to a query, or

Query expansion, when information related to the query is used to expand it.

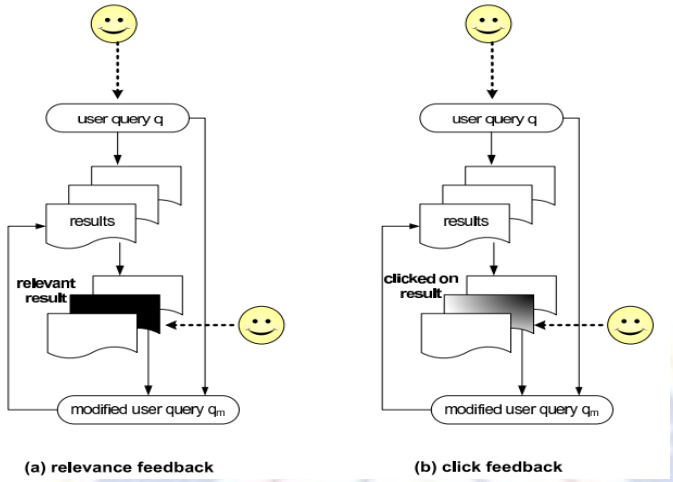
Two basic approaches of feedback methods:

explicit feedback, in which the information for query reformulation is provided directly by the users.

In an explicit relevance feedback cycle, the feedback information is provided directly by the users However, collecting feedback information is expensive and time consuming

In the Web, user clicks on search results constitute a new source of feedback information A click indicate a document that is of interest to the user in the context of the current query

Explicit Feedback



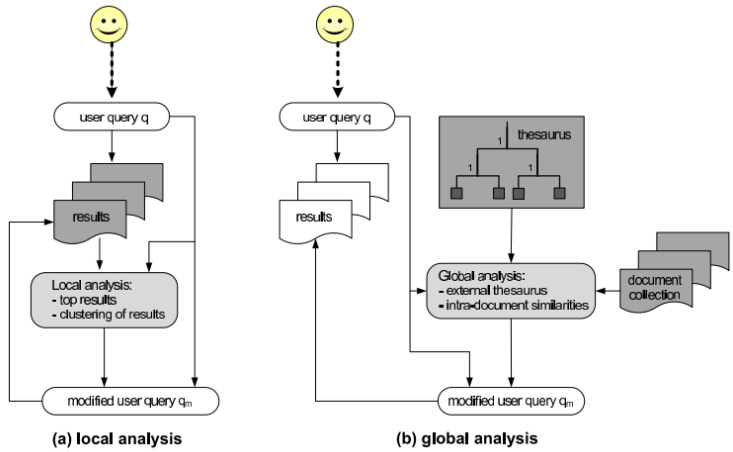
Implicit Feedback, in which the information for query reformulation is implicitly derived by the system.

There are **two basic approaches** for compiling **implicit feedback** information:

local analysis, which derives the feedback information from the top ranked documents in the result set

global analysis, which derives the feedback information from external sources such as a thesaurus

Implicit Feedback



Centroid

- The centroid is the center of mass of a set of points
- Recall that we represent documents as points in a high-dimensional space

Centroid:

$$\vec{\mu}(C) = \frac{1}{|C|}\sum_{\vec{d} \in C} \vec{d}$$

where C is a set of documents.

- The Rocchio algorithm uses the vector space model to pick a relevance feedback query

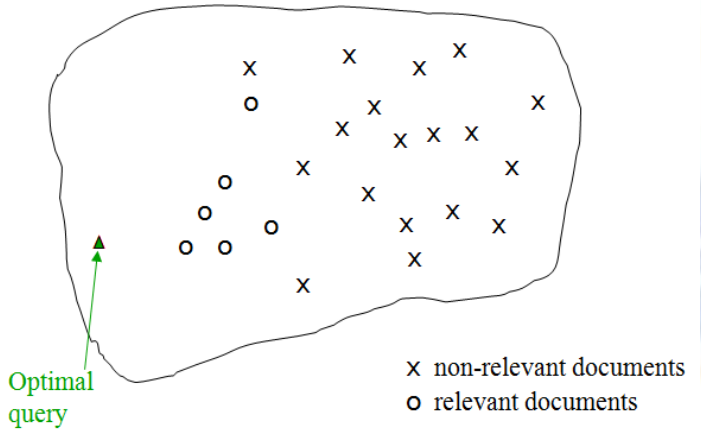
Rocchio seeks the query q_{opt} that maximizes

$$\vec{q}_{opt} = \arg\max_{\vec{q}} [\cos(\vec{q}, \vec{\mu}(C_r)) - \cos(\vec{q}, \vec{\mu}(C_{nr}))]$$

- Tries to separate docs marked relevant and non-relevant

$$\vec{q}_{opt} = \frac{1}{|C_r|}\sum_{\vec{d}_j \in C_r} \vec{d}_j - \frac{1}{|C_{nr}|}\sum_{\vec{d}_j \notin C_r} \vec{d}_j$$

Theoretical Best Query



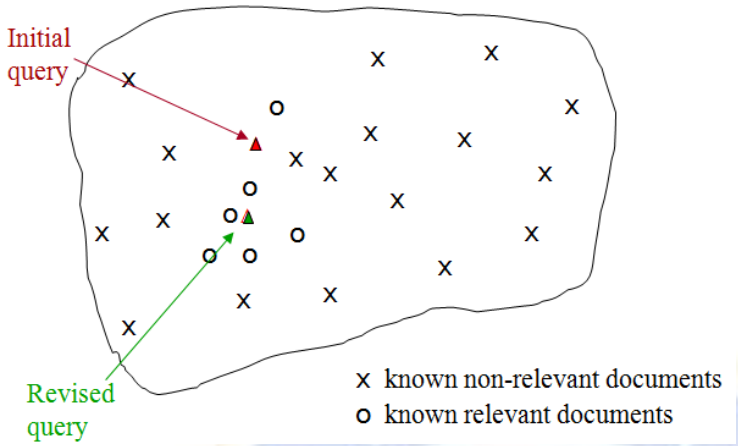
$$\vec{q}_m = \alpha \vec{q}_0 + \beta \frac{1}{|D_r|}\sum_{\vec{d}_j \in D_r} \vec{d}_j - \gamma \frac{1}{|D_{nr}|}\sum_{\vec{d}_j \in D_{nr}} \vec{d}_j$$

Rocchio 1971 Algorithm (SMART)

- D_r = set of known relevant doc vectors
- D_{nr} = set of known irrelevant doc vectors
  - Different from C_r and C_{nr}
- q_m = modified query vector; q_0 = original query vector; α,β,γ: weights (hand-chosen or set empirically)

- New query moves toward relevant documents and away from irrelevant documents

Relevance feedback on initial query



- We can modify the query based on relevance feedback and apply standard vector space model.
- Use only the docs that were marked.Relevance feedback can improve recall and precision
- Relevance feedback is most useful for increasing *recall* in situations where recall is important

Users can be expected to review results and to take time to iterate

Positive feedback is more valuable than negative feedback

Query Expansion

- In relevance feedback, users give additional input (relevant/non-relevant) on documents, which is used to reweight terms in the documents
- In query expansion, users give additional input (good/bad search term) on words or phrases
- For each term, *t*, in a query, expand the query with synonyms and related words of *t* from the thesaurus
  - feline → feline cat
- May weight added terms less than original query terms.
- Generally increases recall
- Widely used in many science/engineering fields
- May significantly decrease precision, particularly with ambiguous terms.

  "interest rate" → "interest rate fascinate evaluate"
- There is a high cost of manually producing a thesaurus

  And for updating it for scientific changes

****

UNIT-III

WEB SEARCH ENGINE – INTRODUCTION AND CRAWLING

Web search overview, web structure, the user, paid placement, search engine optimization/ spam. Web size measurement - search engine optimization/spam – Web Search Architectures - crawling - meta-crawlers- Focused Crawling - web indexes –- Near-duplicate detection - Index Compression - XML retrieval.

3.1. WEB SEARCH OVERVIEW:

- The **IR system contains a collection of documents**, with each document represented by a sequence of tokens. Markup may indicate titles, authors, and other structural elements.
- Many **Web pages are actually *spam*** — malicious pages deliberately posing as something that they are not in order to attract unwarranted attention of a commercial or other nature.
- Other problems derive from the **scale of the Web** — billions of pages scattered among millions of hosts. In order to index these pages, they must be gathered from across the Web by a *crawler* and stored locally by the search engine for processing.
- Because many pages **may change daily or hourly,** this snapshot of the Web must be refreshed on a regular basis. While gathering data the crawler may detect duplicates and near-duplicates of pages, which must be dealt with appropriately.
- Another consideration is the **volume and variety of queries commercial Web search** engines receive, which direct reflect the volume and variety of information on the Web itself. Queries are often short — one or two terms .

3.2. STRUCTURE OF THE WEB

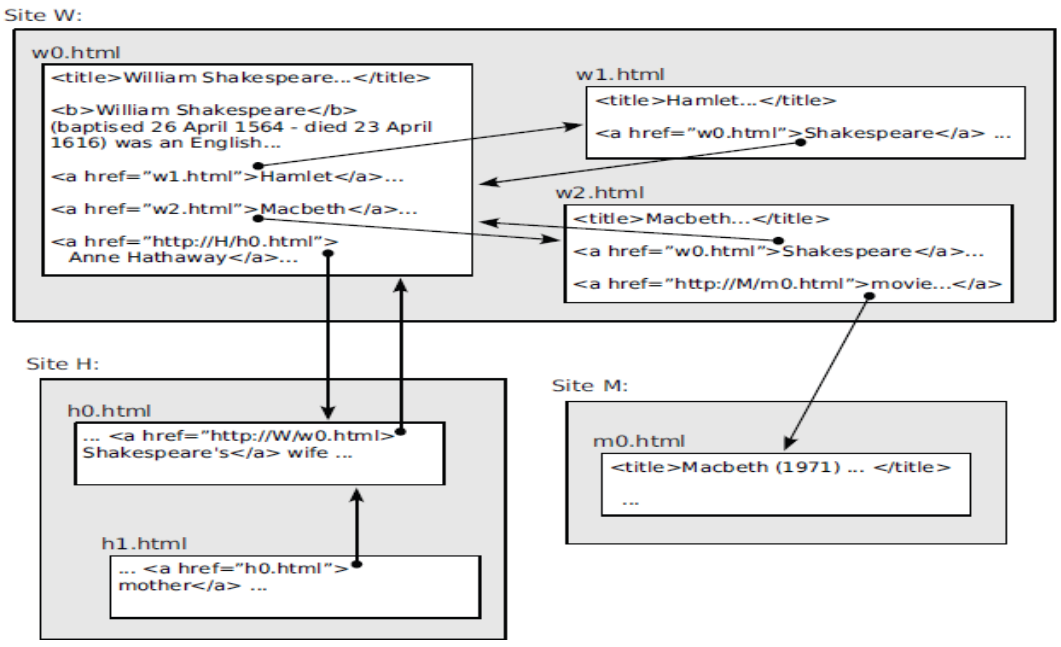


Figure. Structure on the Web. Pages can link to each other (<a href=...>) and can associate each link with anchor text (e.g., "mother").

- Figure provides an example of the most important features related to the structure of the Web. It shows Web pages on three sites: W, M, and H. Site W provides a general encyclopedia including pages on Shakespeare (w0.html) and two of his plays, *Hamlet* (w1.html) and *Macbeth* (w2.html). Site H provides historical information including information on Shakespeare's wife (h0.html) and son (h1.html). Site M provides movie and TV information including a page on the 1971 movie version of *Macbeth* directed by Roman Polanski (m0.html).
- The figure illustrates the link structure existing between these pages and sites. For example, the HTML *anchor* on page w0.html

Anne Hathaway

- Establishes a link between that page and h0.html on site H. The anchor text associated with this link ("Anne Hathaway") provides an indication of the content on the target page.

The Web Graph:

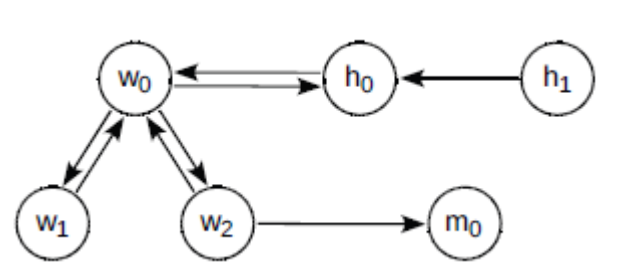


Figure. A Web graph. Each link corresponds to a directed edge in the graph.

- The relationship between sites and pages indicated by these hyperlinks gives rise to what is called a ***Web graph***. When it is viewed as a purely mathematical object, each page forms a node in this graph and each hyperlink forms a directed edge from one node to another. For convenience we have simplified the labels on the pages, with **http://W/w0.html becoming w0,** for example.
- That page are grouped into sites, and that anchor text and images may be associated with each link. Links may reference a labeled position within a page as well as the page as a whole.
- The highly dynamic and fluid nature of the Web must also be remembered**. Pages and links** are continuously added and removed at sites around the world; it is never possible to capture more than a rough approximation of the entire Web graph.

Static and Dynamic Pages:

- It is not uncommon to hear Web pages described as "static" or "dynamic".

- The HTML for a **static Web page** is assumed to be generated in advance of any request, placed on disk, and transferred to the browser or Web crawler on demand. The home page of an organization is a typical example of a static page.
- A **dynamic Web page** is assumed to be generated at the time the request is made, with the contents of the page partially determined by the details of the request. A search engine result page (SERP) is a typical example of a dynamic Web page for which the user's query itself helps to determine the content.

- **A static Web pages** are more important for crawling and indexing than dynamic Web pages, and it is certainly the case that many types of dynamic Web pages are not suitable for crawling and indexing.

- For example, **on-line calendar systems**, which maintain appointments and schedules for people and events, will often serve up dynamically generated pages for dates far into the past and the future.

The Hidden Web:

- Many pages are part of the **so-called "hidden" or "invisible" or "deep" Web**. This hidden Web includes pages that have no links referencing them, those that are protected by passwords, and those that are available only by querying a digital library or database. Although these pages can contain valuable content, they are difficult or impossible for a Web crawler to locate.

- Pages **in *intranets*** represent a special case of the hidden Web. Pages in a given intranet are accessible only within a corporation or similar entity.

3.3. THE USERS:

- **Web queries are short**. Although the exact numbers differ from study to study, these studies consistently reveal that many queries are just one or two terms long, with a mean query length between two and three terms. The topics of these queries range across the full breadth of **human interests, with sex, health, commerce, and entertainment** representing some of the major themes.

- Until quite recently, the query processing strategies of Web search engines actually **discouraged longer queries**.

- These processing strategies **filter the collection against a Boolean conjunction** of the query terms prior to ranking. For a page to be included in the result set, all query terms must be associated with it in some way, either by appearing on the page itself or by appearing in the anchor text of links referencing it.
- As a result, increasing the **length of the query by adding related terms could cause relevant pages to be excluded** if they are missing one or more of the added terms. This strict filtering has been relaxed in recent years.

- For example, synonyms may be accepted in place of exact matches — the term "how to" might be accepted in place of the query term "**FAQ**
- The distribution of **Web queries follows Zipf's law.**

User Intent:

Web queries are classified into three categories reflecting users' apparent intent, as follows:

1) ***Navigational* query:**

The intent behind a *navigational* query is to locate a specific page or site on the Web.

For example, a user intending to locate the home page of the CNN news network might enter the query "CNN".

2) *Informational* query:

A user issuing an ***informational* query** is interested in learning something about a particular topic and has a lesser regard for the source of the information, provided it is reliable.

A user entering the query "president", "OBAMA" might find the information she is seeking on the CNN Web site, on Wikipedia, or elsewhere

3) *Transactional* query:

A user issuing a ***transactional* query** intends to interact with a Web site once she finds it. This interaction may involve activities such as **playing games, purchasing items**, **booking travel, or downloading images, music, and videos**.

Clickthrough Curves:

- The distinction between navigational and informational queries is visible in user behavior. ***clickthroughs*** may be used to infer user intent. A clickthrough is the act of clicking on a search result on a search engine result page. Clickthroughs are often logged by commercial search engines as a method for measuring performance.

3.4 PAID PLACEMENT

- A far more effective and profitable form of advertising for search engines, pioneered by GoTo.com (that was renamed to Overture in 2001 and acquired by Yahoo in 2003), is **called *paid placement* also known as *sponsored search*.**

In this scheme the search engine separates its query results list into two parts:

(i) **An organic list,** which contains the free unbiased results, displayed according to the search engine's ranking algorithm, and

(ii) **A sponsored list,** which is paid for by advertising managed with the aid of an online auction mechanism. This method of payment is called ***pay per click* (PPC),** also known as ***cost per click* (CPC)**, since payment is made by the advertiser each time a user clicks on the link in the sponsored listing.

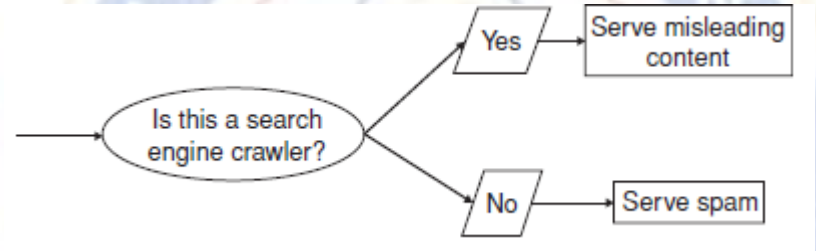
- In the context of CTRs it is worth mentioning that query popularity gives rise to a long tail, more formally known as a ***power law distribution***. A distribution is long tailed when a few data items are very popular and most of the items, comprising the long tail, are rare.

- The main web search engines display sponsored links on a results page in three possible places:
  1) **above the organic list**
  2) **below the organic list**
  3) **In a separate, reduced width, area on the right-hand side of the organic results.**

3.5. SEARCH ENGINE OPTIMIZATION / SPAM:

PART-B
Brief about search engine optimization(Nov/Dec'17)

- In a search engine whose scoring was based on term frequencies, a web page with numerous repetitions of Maui golf spam real estate would rank highly. This led to the first generation of ***spam,*** which is the manipulation of web page content for the purpose of appearing high up in search results for selected keywords.



`Figure. Cloaking as used by spammers.

- To avoid irritating users with these repetitions, sophisticated ***spammers*** resorted to such tricks as rendering these repeated terms in the same color as the background.

- Despite these words being consequently invisible to the human user, a search engine indexer would parse the invisible words out of **the HTML representation** of the web page and index these words as being present in the page.

- In many search engines, it is possible to pay to have one's web page included paid in the search engine's index – a model known as ***paid inclusion***. Different inclusion search engines have different policies on whether to allow paid inclusion, and whether such a payment has any effect on ranking in search results.

***Cloaking*:**

- The spammer's web server returns different pages depending on whether the **http request comes from a web search engine's crawler or from a human user's browser.** The former causes the web page to be indexed by the search engine under misleading keywords.

Doorway page:

- A ***doorway page*** contains text and metadata carefully chosen to rank highly on selected search keywords. When a browser requests the doorway page, it is redirected to a page containing content of a more commercial nature.

Search engine optimizers

Given that spamming is inherently an economically motivated activity, there has sprung search around it an industry of ***search engine optimizers***, or engine optimizers SEOs, to provide consultancy services for clients who seek to have their web pages rank highly on selected keywords.

- Inevitably, the parrying between such SEOs (who gradually infer features of each web search engine's ranking methods) and the web search engines (who adapt in response) is an unending struggle; indeed, the research subarea of *adversarial information retrieval* has sprung up around this battle.
- To combat spammers who manipulate the text of their web pages is the exploitation of the link structure of the Web – a technique known as ***link analysis*.**
- The first web search engine known to apply link analysis on a large scale was Google, although all web search engines currently make use of it (and correspondingly, spammers link spam now invest considerable effort in subverting it – this is known as ***link spam***).

3.6 WEB SIZE MEASUREMENT:

- Even if **we exclude the hidden Web, it is difficult to compute a meaningful estimate for the size of the Web.**
- **Adding or removing a single host can change the number of accessible pages** by an arbitrary amount, depending on the contents of that host.
- Some **hosts contain millions of pages with valuable content**; **others contain millions of pages with little or no useful content .**

Technique for estimating the combined coverage of major search engines:

1) First, a test set of URLs is generated. This step may be achieved by issuing a series of random queries to the engines and selecting a random URL from the results returned by each. We assume that this test set represents a uniform sample of the pages indexed by the engines.
2) Second, each URL from the test set is checked against each engine and the engines that contain it are recorded.

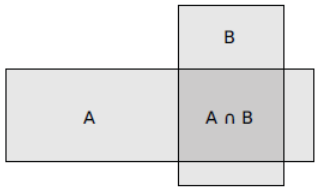


Figure. The collection overlap between search engines A and B can be used to estimate the size of the indexable Web

Given two engines, A and B, the relationship between their collections is illustrated by Figure . Sampling with our test set of URLs allows us to estimate Pr[A ∩ B|A], the probability that a URL is contained in the intersection if it is contained in A:

$$\Pr[A \cap B \mid A] = \frac{\#\text{of tests URLs contained in both A and B}}{\#\text{ of test URls contained in A}}$$

We may estimate Pr[A ∩ B|B] in a similar fashion. If we know the size of A, we may then estimate the size of the intersection as

$$|A \cap B| = |A| \cdot \Pr[A \cap B|A] .$$

and the size of B as

$$|B| = |A \cap B|$$
$$\Pr[A \cap B|B]$$

Thus, the size of the union may be estimated as

$$|A \cup B| = |A| + |B| - |A \cap B| .$$

If sizes for both A and B are available, the size of the intersection may be estimated from both sets, and the average of these estimates may be used to estimate the size of the union

$$|A \cup B| = |A| + |B| - 1/2 \, (|A| \cdot \Pr[A \cap B|A] + |B| \cdot \Pr[A \cap B|B]) .$$

3.7. WEB SEARCH ARCHITECTURE:

PART-B

Elaborate Web Search Architecture(Nov/Dec'16)

- Search engine architecture is used to present high-level descriptions of the important components of the system and the relationships between them.

The two primary goals of a search engine are:

• **Effectiveness (quality):** We want to be able to retrieve the most relevant set of documents possible for a query.
• **Efficiency (speed):** We want to process queries from users as quickly as possible

- The collection of documents we want to search may be changing; making sure that the search engine immediately reacts to changes in documents is both an effectiveness issue and an efficiency issue.

3.7.1. Basic Building Blocks

Search engine components support two major functions

1) ***Indexing process***
2) ***Query process***.

- The indexing process builds the structures that enable searching, and the query process uses those structures and a person's query to produce a ranked list of documents.

1) ***Indexing process:***

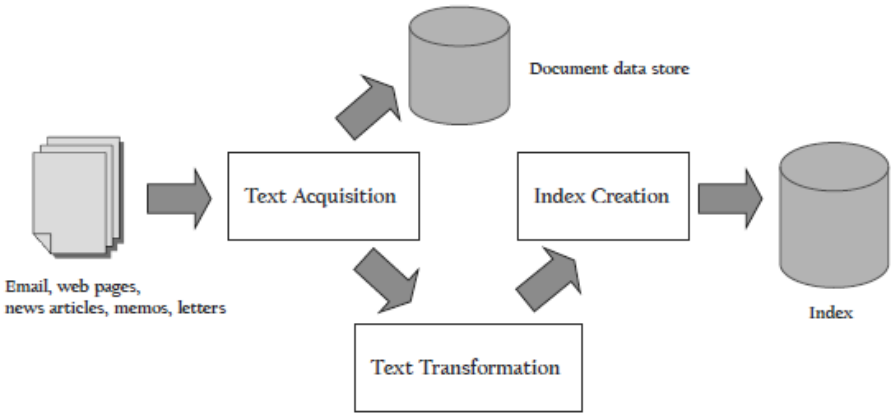


Fig.The indexing process

Figure shows the high-level "building blocks" of the indexing process. These major components are

- *Text acquisition*
- *Text transformation*
- *Index creation.*

Text Acquisition:

i) Crawler:

- In many applications, the *crawler* component has the primary responsibility for identifying and acquiring documents for the search engine. There are a number of different types of crawlers, but the most common is the general web crawler.
- A **web crawler** is designed to follow the links on web pages to discover and download new pages.
- A web crawler can be restricted to a single site, such as a university, as the basis for *site search*. ***Focused,* or *topical***, web crawlers use classification techniques to restrict the pages that are visited to those that are likely to be about a specific topic. This type of

crawler may be used by a ***vertical* or *topical* search** application, such as a search engine that provides access to medical information on web pages.

- For **enterprise search,** the crawler is adapted to discover and update all documents and web pages related to a company's operation.

ii) Feeds

- *Document feeds* are a mechanism for accessing a real-time stream of documents.
- **For example**, a **news feed is a constant stream of news stories and updates**.

iii) Conversion

- The documents found by a crawler or provided by a feed are rarely in plain text.
- Instead, they come in a variety of formats, such as HTML, XML, Adobe PDF, Microsoft Word™, Microsoft PowerPoint®, and so on

Document data store

- The document data store is a database used to manage large numbers of documents and the structured data that is associated with them. The document contents are typically stored in compressed form for efficiency.
- A *relational database system* can be used to store the documents and metadata.

Text Transformation:

i) Parser

- The parsing component is responsible for **processing the sequence of text *tokens*** in the document to recognize structural elements such as titles, figures, links, and headings.
- ***Tokenizing* the text is an important first step in this process.** In many cases, tokens are the same as words. Both document and query text must be transformed into tokens in the same manner so that they can be easily compared.

- Should we treat "apple" the same as "Apple"? Is "on-line" two words or one word? Should the apostrophe in "O'Connor" be treated the same as the one in "owner's"? In some languages, tokenizing gets even more interesting. Chinese, for example, has no obvious word separator like a space in English.

ii) Stopping

- The stopping component has the simple task of removing common words from the stream of tokens that become index terms.

  Examples are "the", "of ", "to", and "for". Because they are so common, removing them can reduce the size of the indexes considerably.

iii) Stemming

- Stemming is another word-level transformation. The task of the stemming component (or *stemmer*) is to group words that are derived from a common *stem*. Grouping "fish", "fishes", and "fishing" is one example.

iv) Link extraction and analysis

- Links and the corresponding anchor text in web pages can readily be identified and extracted during document parsing. Extraction means that this information is recorded in the document data store, and can be indexed separately from the general text content.

link analysis algorithms such as PageRank (Brin & Page, 1998). Link analysis provides the search engine with a rating of the popularity, and to some extent, the *authority* of a page (in other words, how important it is). *Anchor text*, which is the clickable text of a web link, can be used to enhance the text content of a page that the link points to

v) Information extraction

- Information extraction is used to identify index terms that are more complex than single words. This may be as simple as words in bold or words in headings, but in general may require significant additional computation.

Index Creation:

- The task of the document statistics component is simply to gather and record statistical information about words, features, and documents. This information is used by the ranking component to compute scores for documents.
- **Weighting**
- Index term *weights* reflect the relative importance of words in documents, and are used in computing scores for ranking.
- The weighting component calculates weights using the document statistics and stores them in lookup tables.
- One of the most common types used in older retrieval models is known as *tf.idf* weighting.
- The *idf* weight is called inverse document frequency because it gives high weights to terms that occur in very few documents.
- A typical formula for *idf* is log*N*/*n*, where*N* is the total number of documents indexed by the search engine and *n* is the number of documents that contain a particular term.

i) Inversion

- The *inversion* component is the core of the indexing process. Its task is to change the stream of document-term information coming from the text transformation component into term-document information for the creation of inverted indexes.

ii) Index distribution

- The index distribution component distributes indexes across multiple computers and potentially across multiple sites on a network. By distributing the indexes for a subset of the documents **(*document distribution*)**, both indexing and query processing can be done in *parallel*. Distributing the indexes for a subset of terms **(*term distribution*)** can also support parallel processing of queries.
- ***Replication*** is a form of distribution where copies of indexes or parts of indexes are stored in multiple sites so that query processing can be made more efficient by reducing communication delays.

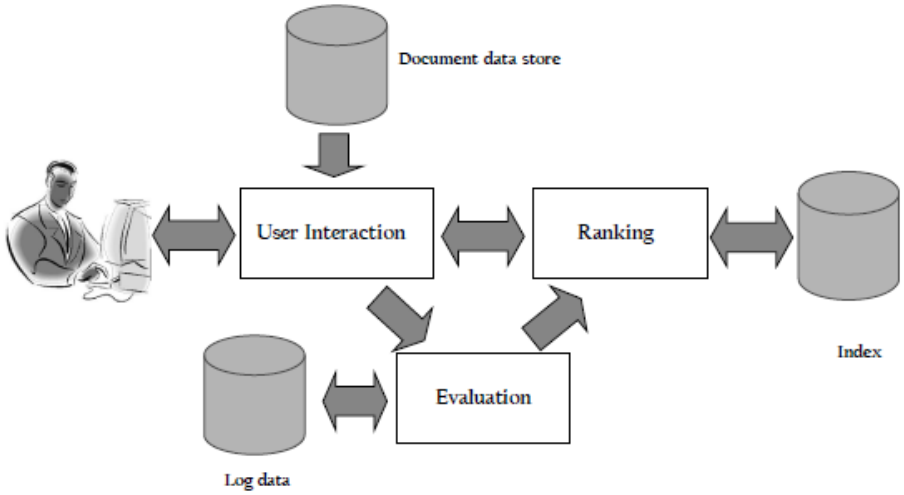


Fig. The query process

User Interaction:

- The query input component provides an interface and a parser for a ***query language***
- SQL query language is not designed for the typical user of a database application (the *end user*), these query languages are not designed for the end users of search applications.
- *Boolean* query languages have a long history in information retrieval. The operators in this language include Boolean AND, OR, and NOT, and some form of *proximity* operator that specifies that words must occur together within a specific distance (usually in terms of word count).
- Other query languages include these and other operators in a probabilistic framework designed to allow specification of features related to both document structure and content.

i) Query transformation

- The query transformation component includes a range of techniques that are designed to improve the initial query, both before and after producing a document ranking. **Tokenizing, stopping, and stemming** must be done on the query text to produce index terms that are comparable to the document terms.
- ***Spell checking*** **and** ***query suggestion*** are query transformation techniques that produce similar output.
- ***Query expansion*** techniques also suggest or add additional terms to the query, but usually based on an analysis of term occurrences in documents.
- ***Relevance feedback*** is a technique that expands queries based on term occurrences in documents that are identified as relevant by the user.

ii) Results output

- The results output component is responsible for constructing the display of ranked documents coming from the ranking component. This may include tasks such as generating ***snippets*** to summarize the retrieved documents, ***highlighting*** important words and passages in documents, clustering the output to identify related groups of documents, and finding appropriate advertising to add to the results display.

- **Ranking:**
- The scoring component, also called ***query processing***, calculates scores for documents using the ranking algorithm, which is based on a retrieval model
- Many different retrieval models and methods of deriving ranking algorithms have been proposed. The basic form of the document score calculated by many of these models is

$$\sum_i q_i.d_i$$

i) Where the summation is over all of the terms in the vocabulary of the collection, *qi* is the query term weight of the *i*th term, and *di* is the document term weight. The term weights depend on the particular retrieval model being used, but are generally similar to *tf.idf* weights.

ii) Performance optimization

- Performance optimization involves the design of ranking algorithms and the associated indexes to decrease response time and increase query throughput. Given
- **For example**, scores can be computed by accessing the index for a query term, computing the contribution for that term to a document's score, adding this contribution to a score accumulator, and then accessing the next index. This is referred to as *term-at-a-time* scoring.

iii) Distribution

- Given some form of index distribution, ranking can also be distributed.
- A ***query broker*** decides how to allocate queries to processors in a network and is responsible for assembling the final ranked list for the query. The operation of the broker depends on the form of index distribution.
- ***Caching*** is another form of distribution here indexes or even ranked document lists from previous queries are left in local memory.

Evaluation:

i) Logging

- **Logs of the users'** queries and their interactions with the search engine are one of the most valuable sources of information for tuning and improving search effectiveness and efficiency.
- **Query logs** can be used for spell checking, query suggestions, query caching, and other tasks, such as helping to match advertising to searches.
- Documents in a result list that are clicked on and browsed tend to be relevant. This means that logs of user clicks on documents (**clickthrough data**) and information such as the ***dwell time*** (time spent looking at a document) can be used to evaluate and train ranking algorithms.

ii) Performance analysis

- The performance analysis component involves **monitoring and improving overall system** performance, in the same way that the ranking analysis component monitors effectiveness.
- A variety of performance measures are used, such as response time and throughput, but the measures used also depend on the application.
- For example, a distributed search application should monitor network usage and efficiency in addition to other measures.

3.8. CRAWLING

PART-A

Define Web Crawling.(Apr/May'17)

What is the purpose of Web Crawler?(Nov/Dec'16)

What are the politeness process in web crawling?(Nov/Dec'17)

PART-B

Explain the working of Web Crawler with an example.(Apr/May'17)

- *Web crawling* is the process by which we gather pages from the Web to index them and support a search engine.
- The objective of crawling is to quickly and efficiently gather as many useful web pages as possible, together with the link structure that interconnects them.

Features of a crawler:

1) **Robustness:**
   The Web contains servers that create *spider traps*, which are generators of web pages that mislead crawlers into getting stuck fetching an infinite number of pages in a particular domain.
2) **Politeness:**
   - Web servers have both implicit and explicit policies regulating the rate at which a crawler can visit them. These politeness policies must be respected.
3) **Distributed:**
   - The crawler should have the ability to execute in a distributed fashion across multiple machines.
4) **Scalable:**
   - The crawler architecture should permit scaling up the crawl rate by adding extra machines and bandwidth.
5) **Performance and efficiency:**
   - The crawl system should make efficient use of various system resources including processor, storage, and network bandwidth.
6) **Quality:**
   - Given that a significant fraction of all web pages are of poor utility for serving user query needs, the crawler should be biased toward fetching "useful" pages first.
7) **Freshness:**
   In many applications, the crawler should operate in continuous mode: It should obtain fresh copies of previously fetched pages.

8) **Extensible:**
   - Crawlers should be designed to be extensible in many ways – to cope with new data formats, new fetch protocols, and so on. This demands that the crawler architecture be modular.

Crawling:

- The basic operation of any hypertext crawler is as follows.
- The crawler begins with one or more URLs that constitute a *seed set*. It picks a URL from this seed set and then fetches the web page at that URL. The fetched page is then parsed, to extract both the text and the links from the page (each of which points to another URL).
- The extracted text is fed to a text indexer. The extracted links (URLs) are then added to a *URL frontier*, which at all times consists of URLs whose corresponding pages have yet to be fetched by the crawler.
- Initially, the URL frontier contains the seed set; as pages are fetched, the corresponding URLs are deleted from the URL frontier. The entire process may be viewed as traversing the web graph. In continuous crawling, the URL of a fetched page is added back to the frontier for fetching again in the future.
- This, seemingly simple, recursive traversal of the web graph is complicated by the many demands on a practical web crawling system: The crawler has to be distributed, scalable, efficient, polite, robust, and extensible while fetching pages of high quality.
- *Mercator* crawler has formed the basis of a number of research and commercial crawlers.

1) The URL frontier, containing URLs yet to be fetched in the current crawl.
2) A *DNS resolution* module determines the web server from which to fetch the page specified by a URL.
3) A fetch module that uses the http protocol to retrieve the web page at a URL.
4) A parsing module that extracts the text and set of links from a fetched web page.
5) A duplicate elimination module that determines whether an extracted link is already in the URL frontier or has recently been fetched.

Crawler architecture:

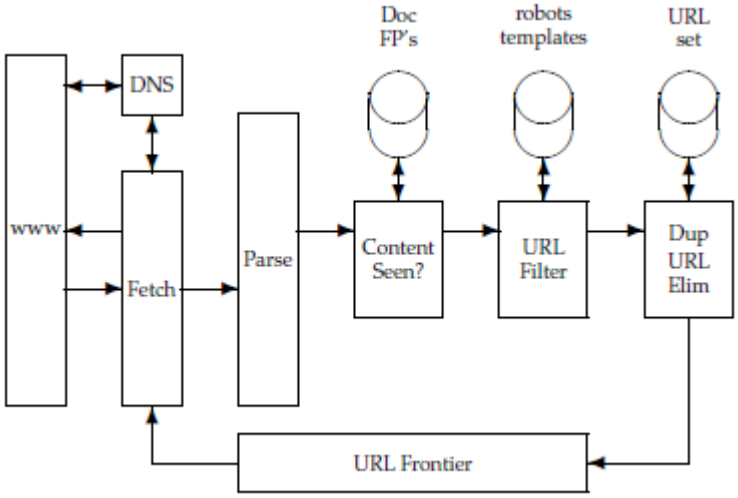


Figure. Basic crawler architecture.

DNS resolution

- Each web server has a unique *IP address*: a sequence of four bytes generally represented as four integers separated by dots; for instance 207.142.131.248 is the numerical IP address associated with the host www.wikipedia.org.

- Given a URL such as www.wikipedia.org in textual form, translating it to an IP address (in this DNS case, 207.142.131.248) is a process known as *DNS resolution* or DNS lookup; here DNS stands for *domain name service*.

- During DNS resolution, the program that wishes to perform this translation contacts a *DNS server* that returns the translated IP address.

The URL frontier

- Two important considerations govern the order in which URLs are returned by the frontier.
- First, high-quality pages that change frequently should be prioritized for frequent crawling. Thus, the priority of a page should be a function of both its change rate and its quality.
- The second consideration is politeness: We must avoid repeated fetch requests to a host within a short time span. The likelihood of this is exacerbated because of a form of locality of reference; many URLs link to other URLs at the same host. As a result, a URL frontier implemented as a simple priority queue might result in a burst of fetch requests to a host. This might occur even if we were to constrain the crawler so that at most one thread could fetch from any single host at any time.
- A common heuristic is to insert a gap between successive fetch requests to a host that is an order of magnitude larger than the time taken for the most recent fetch from that host.
- Figure shows a polite and prioritizing implementation of a URL frontier.Its goals are to ensure that

(i) only one connection is open at a time to any host,
(ii) a waiting time of a few seconds occurs between successive requests to a host
(iii) High-priority pages are crawled preferentially.

- The two major sub modules are a set of *F front queues* in the upper portion of the figure and a set of *B back queues* in the lower part; all of these are FIFO queues.

3.9. META CRAWLERS:

PART-B

Describe Meta and Focused Crawling(Nov/Dec'16)

- **MetaCrawler** is a metasearch engine program that blended web search results from Google, Yahoo!, Bing (formerly Live Search), Ask.com, About.com, MIVA, LookSmart and other popular search engine programs. MetaCrawler also provided users the option to search for images, video, news, business and personal telephone directories, and for a while even audio. MetaCrawler became well known during the late 1990s when the verb "metacrawled" was used by television talk show host Conan O'Brien on TRL. MetaCrawler is a registered trademark of InfoSpace, Inc.
- MetaCrawler formerly fetched results from several other popular search engines, but now, just like Yahoo!, it's just a search engine powered by Bing (although the site never mention anything about that).
- Unlike search engines, meta crawlers don't crawl the web themselves to build listings. Instead, they allow searches to be sent to several search engines all at once. The results are then blended together onto one page. Below are some of the major meta crawlers. Also see the Search Toolbars & Utilities page for meta crawler-style software that you can run from your desktop.

- Dogpile

  Popular meta search site owned by Info Space that sends a search to a customizable list of search engines, directories and specialty search sites, then displays results from each search engine individually. Winner of Best Meta Search Engine award from Search Engine Watch for 2003.
- Vivisimo

  Enter a search term, and Vivismo will not only pull back matching responses from major search engines but also automatically organize the pages into categories. Slick and easy to use. Vivisimo won second place for Best Meta Search Engine in the 2003 Search Engine Watch awards and winner in 2002.

- Kartoo

  If you like the idea of seeing your web results visually, this meta search site shows the results with sites being interconnected by keywords. Honorable mention for Best Meta Search Engine award from Search Engine Watch in 2002.

3.10. FOCUSED CRAWLING

PART-B

Write short notes on the following Focused , Deep Web, Distributed Crawling , Site map .(Nov/Dec'17)

- Some users would like a search engine that focuses on a specific topic of information. **Eg:Movies**.
- If built correctly, this type of ***vertical search*** can provide higher accuracy than general search because of the lack of extraneous information in the document collection.
- The most accurate way to get web pages for this kind of engine would be to crawl a full copy of the Web and then throw out all unrelated pages.
- A less expensive approach is ***focused,* or *topical,* crawling**. A focused crawler attempts to download only those pages that are about a particular topic.
- Focused crawlers rely on the fact that pages about a topic tend to have links to other pages on the same topic. If this were perfectly true, it would be possible to start a crawl at one on-topic page, and then crawl all pages on that topic just by following links from a single root page. In practice, a number of popular pages for a specific topic are typically used as seeds.
- Focused crawlers require some automatic means for determining whether a page is about a particular topic.
- **For example,** Text classifiers are tools that can make this kind of distinction. Once a page is downloaded, the crawler uses the classifier to decide whether the page is on topic. If it is, the page is kept, and links from the page are used to find other related sites. The anchor text in the outgoing links is an important clue of topicality. Also, some pages have more on-topic links than others.

Deep Web

- Not all parts of the Web are easy for a crawler to navigate. Sites that are difficult for a crawler to find are collectively referred to as the ***deep Web* (also called the *hidden Web*).**

- Most sites that are a part of the deep Web fall into three broad categories:

  - ***Private sites*** are intentionally private. They may have no incoming links, or may require you to log in with a valid account before using the rest of the site. These sites generally want to block access from crawlers, although some news publishers may still want their content indexed by major search engines.

- ***Form results*** are sites that can be reached only after entering some data into a form.
  For example, websites selling airline tickets typically ask for trip information on the site's entry page. You are shown flight information only after submitting this trip information. Even though you might want to use a search engine to find flight timetables, most crawlers will not be able to get through this form to get to the timetable information.

3.11. NEAR-DUPLICATES AND SHINGLING

PART-B
Explain in detail about fingerprint algorithm for near duplicate detection(Nov/Dec'17)

- The Web contains multiple copies of the same content. By some estimates, as many as 40% of the pages on the Web are duplicates of other pages. Many of these are legal copies; for instance, certain information repositories are mirrored simply to provide redundancy and access reliability.

- Search engines try to avoid indexing multiple copies of the same content, to keep down storage and processing overheads. The simplest approach to detecting duplicates is to compute, for each web page, **a *fingerprint*** that is a brief (say 64-bit) digest of the characters on that page.

- Then, whenever the **fingerprints of two web pages are equal**, we test whether the pages themselves are equal and if so declare one of them to be a duplicate copy of the other. This simplistic approach fails to capture a crucial and widespread phenomenon on the Web: ***near duplication***.

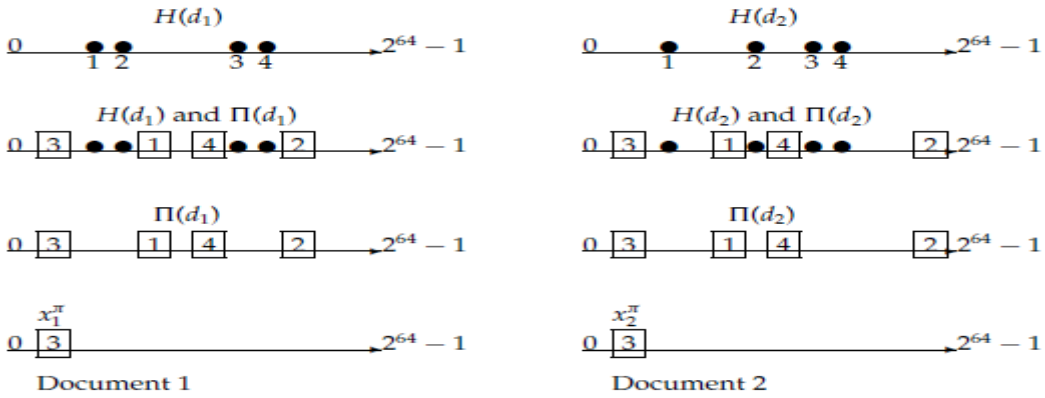


Figure . Illustration of shingle sketches. We see two documents going through four stages of shingle sketch computation. In the first step (*top row*), we apply a 64-bit hash to each shingle from each document to obtain $H(d_1)$ and $H(d_2)$ (*circles*). Next, we apply a random permutation Π to permute $H(d_1)$ and $H(d_2)$, obtaining $\Pi(d_1)$ and $\Pi(d_2)$ (*squares*). The third row shows only $\Pi(d_1)$ and $\Pi(d_2)$; the bottom row shows the minimum values x_1^π and x_2^π for each document.

- We now describe a solution to the problem of detecting near-duplicate web pages. The answer lies in a technique known as ***shingling*.**
- Given a positive integer *k* and a sequence of terms in a document *d*, define the *k* shingles of *d* to be the set of all consecutive sequences of *k* terms in *d*.

- As an example, consider the following text**: flower is a, flower is a flower**

  The 4-shingles for this text (*k* = 4 is a typical value used in the detection of near-duplicate web pages) are a flower is a, flower is a flower, and is a flower is. The first two of these shingles each occur twice in the text. Intuitively, two documents are near duplicates if the sets of shingles generated from them are nearly the same.

- We now make this intuition precise, then develop a method for efficiently computing and comparing the sets of shingles for all web pages.

- Let *S*(*dj*) denote the set of shingles of document *dj* . The Jaccard coefficient measures the degree of overlap between the sets *S*(*d*1) and *S*(*d*2) as |*S*(*d*1) ∩ *S*(*d*2)| / |*S*(*d*1) ∪ *S*(*d*2)|; denote this by *J* (*S*(*d*1), *S*(*d*2)).

- Our test for near duplication between *d*1 and *d*2 is to compute this Jaccard coefficient; if it exceeds a preset threshold (say, 0.9), we declare them near duplicates and eliminate one from indexing. However, this does not appear to have simplified matters: We still have to compute Jaccard coefficients pair wise. To avoid this, we use a form of hashing.

- First, we map every shingle into a hash value over a large space, say 64 bits. For *j* = 1, 2, let *H*(*dj*) be the corresponding set of 64-bit hash values derived from *S*(*dj*). We now invoke the following trick to detect document pairs whose sets *H*() have large Jaccard overlaps.

S_{j1}	S_{j2}
0	1
1	0
1	1
0	0
1	1
0	1

Fig.Two sets Sj1 and Sj2: their Jaccard Coefficient is 2/5.

- Let π be a random permutation from the 64-bit integers to the 64-bit integers. Denote by $\Pi(d_j)$ the set of permuted hash values in *H*(*dj*); thus for each $h \in H(dj)$, there is a corresponding value $\pi(h) \in \Pi(d_j)$ .Let x_j^{π} be the smallest integer in $\Pi(d_j)$.
-

3.11.INDEX COMPRESSION

PART-B

Explain index compression with an example(Apr/May'17)

Benefits of compression:

1) Need less disk space.
2) Increased use of caching. Search systems use some parts of the dictionary and the index much more than others.
   For example, if we cache the postings list of a frequently used query term *t*, then the computations necessary for responding to the one-term query *t* can be entirely done in memory.
   With compression, we can fit a lot more information into main memory. Instead of having to expend a disk seek when processing a query with *t*, we instead access its postings list in memory and decompress it.
   The second more subtle advantage of compression is faster transfer of data from disk to memory.

Heaps' law: Estimating the number of terms:

- A better way of getting a handle on *M* is *Heaps' law*, which estimates vocabulary size as a function of collection size:

$$M = kT^b$$

- Where *T* is the number of tokens in the collection. Typical values for the parameters *k* and *b* are: $30 \leq k \leq 100$ and $b \approx 0.5$. The motivation for Heaps' law is that the simplest possible relationship between collection size and vocabulary size is linear in log–log space .

Zipf's law: Modeling the distribution of terms:

- A commonly used model of the distribution of terms in a collection is *Zipf's law*. It states that, if t_1 is the most common term in the collection, t_2 is the next most common, and so on, then the collection frequency cf_i of the i^{th} most common term is proportional to $1/i$:

$$cf_i \propto \frac{1}{i}.$$

Dictionary compression:

- One of the primary factors in determining the response time of an IR system is the number of disk seeks necessary to process a query. If parts of the dictionary are on disk, then many more disk seeks are necessary in query evaluation.

- Thus, the main goal of compressing the dictionary is to fit it in main memory, or at least a large portion of it, to support high query throughput.

- For example, an enterprise search server for a large corporation may have to index a multi terabyte collection with a comparatively large vocabulary because of the presence of documents in many different languages.

Dictionary as a string:

- The simplest data structure for the dictionary is to sort the vocabulary lexicographically and store it in an array of fixed-width entries as shown in
- One source of redundancy in the dictionary we have not exploited yet is the fact that consecutive entries in an alphabetically sorted list share common prefixes. This observation leads to *front coding*.

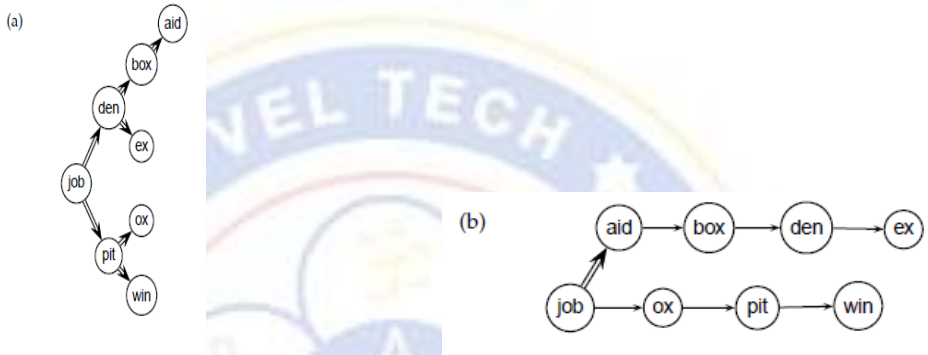


Figure. Search of the uncompressed dictionary (a) and a dictionary compressed by blocking with $k = 4$ (b).

Variable byte codes:

- *Variable byte (VB) encoding* uses an integral number of bytes to encode a gap. The last 7 bits of a byte are "payload" and encode part of the gap. The first bit of the byte is a *continuation bit*.

VB encoding. Gaps are encoded using an integral number of bytes. The first bit, the continuation bit, of each byte indicates whether the code ends with this byte (1) or not (0).

number	unary code	length	offset	γ code
0	0			
1	10	0		0
2	110	10	0	10,0
3	1110	10	1	10,1
4	11110	110	00	110,00
9	1111111110	1110	001	1110,001
13		1110	101	1110,101
24		11110	1000	11110,1000
511		111111110	11111111	111111110,11111111
1025		11111111110	0000000001	11111111110,0000000001

Some examples of unary and γ codes. Unary codes are only shown for the smaller numbers. Commas in γ codes are for readability only and are not part of the actual codes.

- The idea of VB encoding can also be applied to larger or smaller units than bytes: 32-bit words, 16-bit words, and 4-bit words or *nibbles*. Larger words further decrease the amount of bit manipulation necessary at the cost of less effective (or no) compression.

***g* codes:**

- VB codes use an adaptive number of *bytes* depending on the size of the gap. Bit-level codes adapt the length of the code on the finer grained *bit* level. The simplest bit-level code is *unary code*. The unary code of *n* is a string of *n* 1s followed by a 0 (see the first two columns of Table). Obviously, this is not a very efficient code, but it will come in handy in a moment.

3.12. XML RETRIEVAL

PART-A
What are the requirements of XML information retrieval system.(Nov/Dec'16)
PART-B
Explain with an example the framework of a XML retrieval system.(Apr/May'17)

Basic XML concepts:

- An XML document is an ordered, labeled tree.
- Each node of the tree is an XML element, written with an opening and closing XML tag (e.g. <title…>, </title…>)

- An element can have one or more XML attributes (e.g. number)
- Attributes can have values (e.g. vii)
- Attributes can have child elements (e.g. title, verse)

```
<play>
<author>Shakespeare</author>
<title>Macbeth</title>
<act number="I">
<scene  number=""vii">
<title>Macbeth's castle</title>
<verse>Will I with wine
</verse>
</scene>
</act>
</play>
```

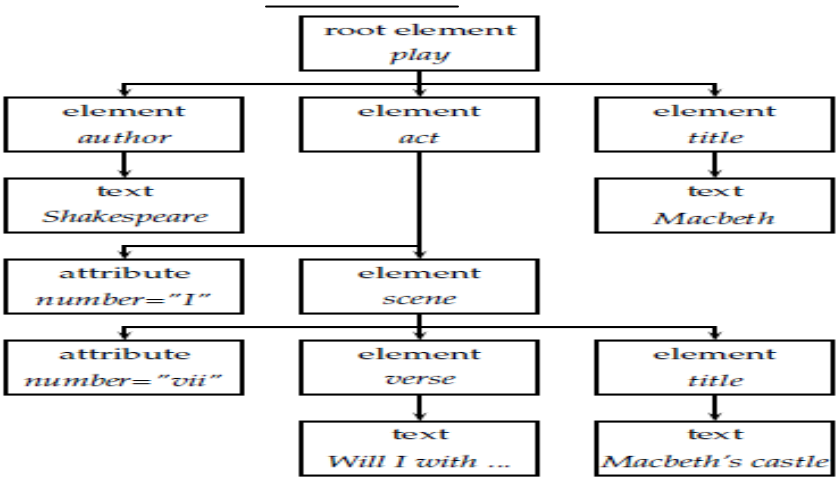


Figure . The XML document as a simplified DOM object.

- The *leaf nodes* of the tree consist of text, e.g., Shakespeare, Macbeth, and Macbeth's castle. The tree's *internal nodes* encode either the structure of the document (*title*, *act*, and *scene*) or metadata functions (*author*).
- The standard for accessing and processing XML documents is the XML Document Object Model or *DOM*. The DOM represents elements, attributes and text within elements as nodes in a tree.

```
//article
[.//yr = 2001 or .//yr = 2002]
//section
[about(.,summer holidays)]
```

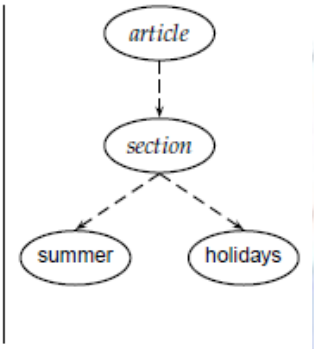
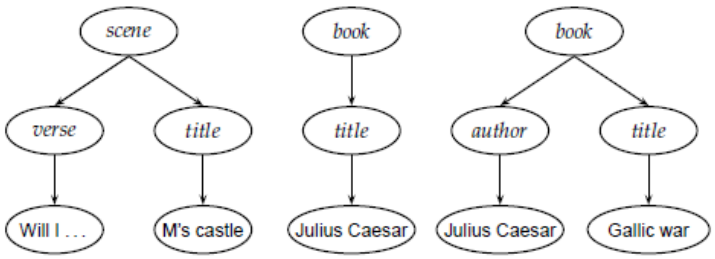


Figure . An XML query in NEXI format and its partial representation as a tree. can process an XML document by starting at the root element and then descending down the tree from parents to children.



. d_1 q_1 q_2

Figure. Tree representation of XML documents and queries.

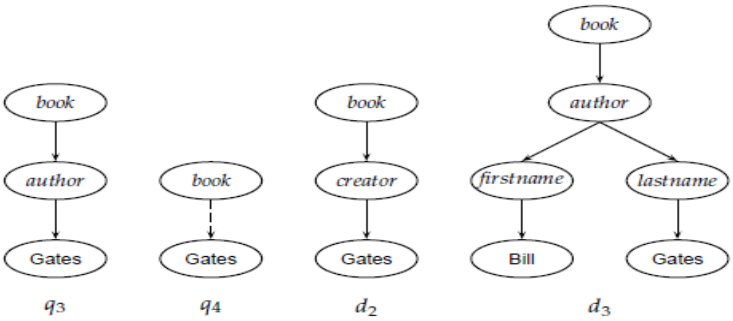


Figure . Schema heterogeneity: intervening nodes and mismatched names.

****

UNIT-IV

WEB SEARCH – LINK ANALYSIS AND SPECIALIZED SEARCH

Link Analysis–hubs and authorities– Page Rank and HITS algorithms -Searching and Ranking – Relevance Scoring and ranking for Web – Similarity – Hadoop & Map Reduce – Evaluation – Personalized search – Collaborative filtering and content-based recommendation of documents and products – handling "invisible" Web – Snippet generation, Summarization, Question Answering, Cross-Lingual Retrieval.

4.1 LINK ANALYSIS

Links connecting pages are a key component of the Web.

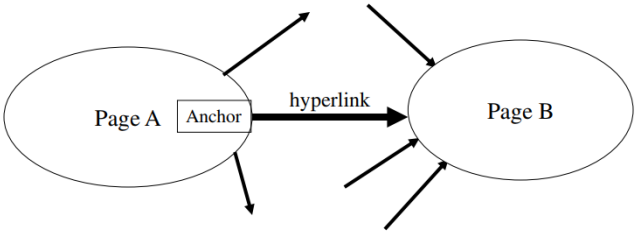
Links are a powerful navigational aid for people browsing the Web, but they also help search engines understand the relationships between the pages.

These detected relationships help search engines rank web pages more effectively. It should be remembered, however, that many document collections used in search applications such as desktop or enterprise search either do not have links or have very little link structure.

4.1.1 Anchor Text

Anchor text has two properties that make it particularly useful for ranking web pages. It tends to be very short, perhaps two or three words, and those words often succinctly describe the topic of the linked page. For instance, links to www.ebay.com are highly likely to contain the word "eBay" in the anchor text

Anchor text is usually written by people who are not the authors of the destination page.



Assumption 1: A hyperlink between pages denotes author perceived relevance (quality signal)

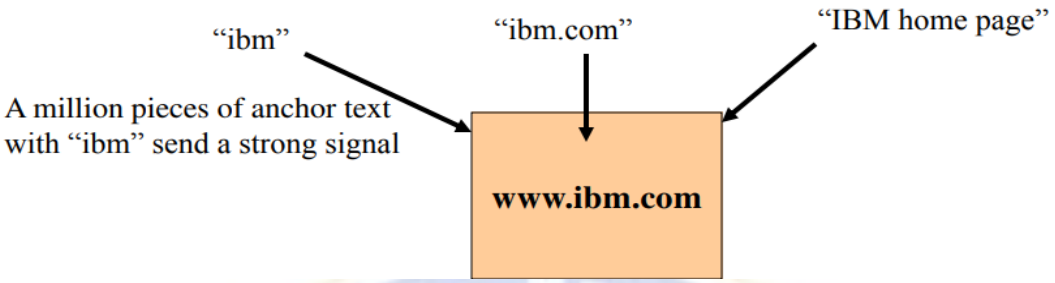
Assumption 2: The anchor text of the hyperlink describes the target page (textual context)

For ibm how to distinguish between:

IBM home page (mostly graphical page)

IBM copy right page (high term frequency . for IBM)

Rival Copyright page (arbitrarily high frequency)



Can sometimes have unexpected effects, e.g., spam, **miserable failure**
Can score anchor text with weight depending on the authority of the anchor page's website
E.g., if we were to assume that content from cnn.com or yahoo.com is authoritative, then trust (more) the anchor text from them
Increase the weight of off-site anchors (non-nepotistic scoring)

4.2 HUBS AND AUTHORITIES

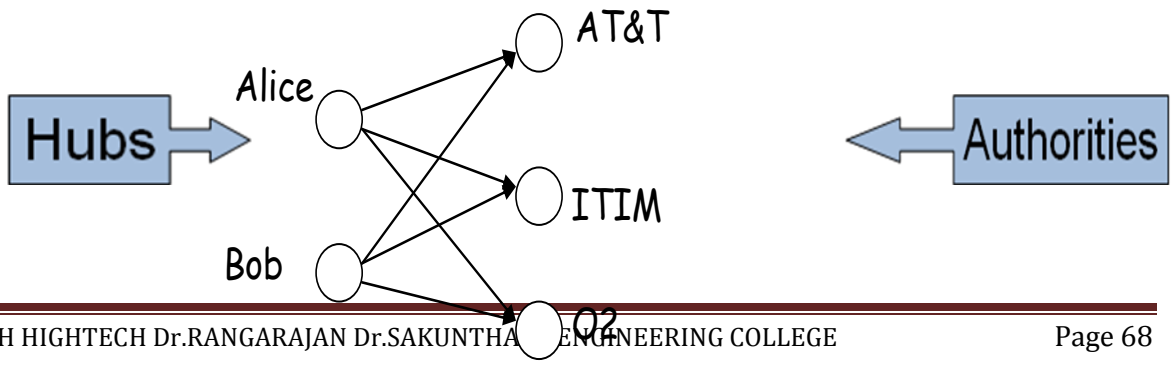
PART-A
Define Authorities(Nov/Dec'16)

Good hub page for a topic *points* to many authoritative pages for that topic.

A good authority page for a topic is *pointed* to by many good hubs for that topic.

Circular definition - will turn this into an iterative computation.

Hubs and Authorities



High-level scheme

Extract from the web a base set of pages that *could* be good hubs or authorities.
From these, identify a small set of top hub and authority pages;

- iterative algorithm.

Base set

Given text query (say ***browser***), use a text index to get all pages containing ***browser.***

→ Call this the root set of pages.

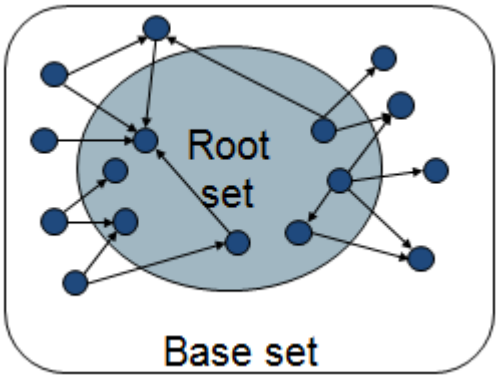
Add in any page that either

→ points to a page in the root set, or

→ is pointed to by a page in the root set.

Call this the base set.

Visualization



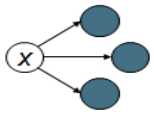
Get in-links (and out-links) from a *connectivity server*

Distilling hubs and authorities

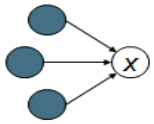
Compute, for each page *x* in the base set, a hub score *h(x)* and an authority score *a(x).*
Initialize: for all *x*, *h(x)←1; a(x) ←1*;
Iteratively update all *h(x), a(x)*;
After iterations

Output pages with highest *h()* scores as top hubs
Highest *a()* scores as top authorities.

$$h(x) \leftarrow \sum_{x\alpha\ y} a(y)$$

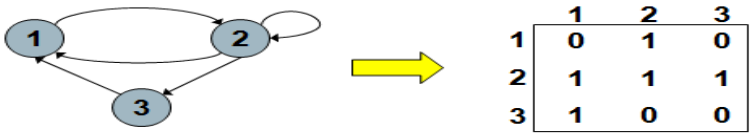


$$a(x) \leftarrow \sum_{y\alpha\ x} h(y)$$



Convergence

- $n \times n$ adjacency matrix **A**:
  - each of the n pages in the base set has a row and column in the matrix.
  - Entry $A_{ij} = 1$ if page i links to page j, else = 0.



	1	2	3
1	0	1	0
2	1	1	1
3	1	0	0

4.3 PAGERANK AND HITS ALGORITHM

PART-B

Compare HITS and Page Rank in detail(Nov/Dec'16)

PART-B

Brief about HITS algorithm

Write short notes on top specific page rank Computation(Apr/may'17)

Page Rank is a method for rating the importance of web pages objectively and mechanically using the link structure of the web.

• Page Rank is an algorithm used by Google Search to rank websites in their search engine results. Page Rank was named after Larry Page, one of the founders of Google. Page Rank is a way of measuring the importance of website pages. According to Google:

• Page Rank works by counting the number and quality of links to a page to determine a rough estimate of how important the website is. The underlying assumption is that more important websites are likely to receive more links from other websites.

• Searching with Page Rank : Two search Engines:

a. Title – based search engine
b. Full text search engine

a. Title – based search engine

It searches only the "Title". Finds all the web pages whose titles contain all the query words.
Sorts the results by page Rank.
Very simple and cheap to implement.
Title match ensure high precision and page rank ensure high quality.

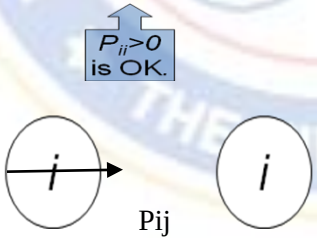
b. Full text search engine Also called Google. It examines allAlso called Google. It examines all the words in every stored document and also performs Page Rank.
More precise but more complicated.

Citation Analysis

Citation frequency
Bibliographic coupling frequency
Articles that co-cite the same articles are related
Citation indexing
Who is this author cited by? (Garfield 1972)
Pagerank preview: Pinsker and Narin '60s
Asked: which journals are authoritative?

Markov chains

A Markov chain consists of n states, plus an $n \times n$ transition probability matrix **P**.
At each step, we are in one of the states.
For $1 \le i,j \le n$, the matrix entry P_{ij} tells us the probability of j being the next state, given we are currently in state i.

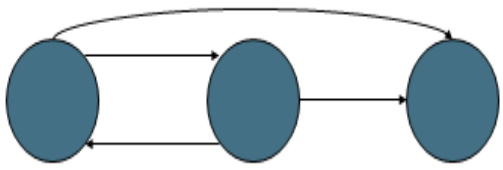


Pij

Clearly, for all i,

Markov chains are abstractions of random walks.

Exercise: represent the teleporting random walk from 3 slides ago as a Markov chain, for this case:



Ergodic Markov chains
For any *ergodic* Markov chain, there is a unique long-term visit rate for each state.

- *Steady-state probability distribution*.
- Over a long time-period, we visit each state in proportion to this rate.
- It doesn't matter where we start.

HITS ALGORITHMS(Hyperlink-Induced Topic Search)

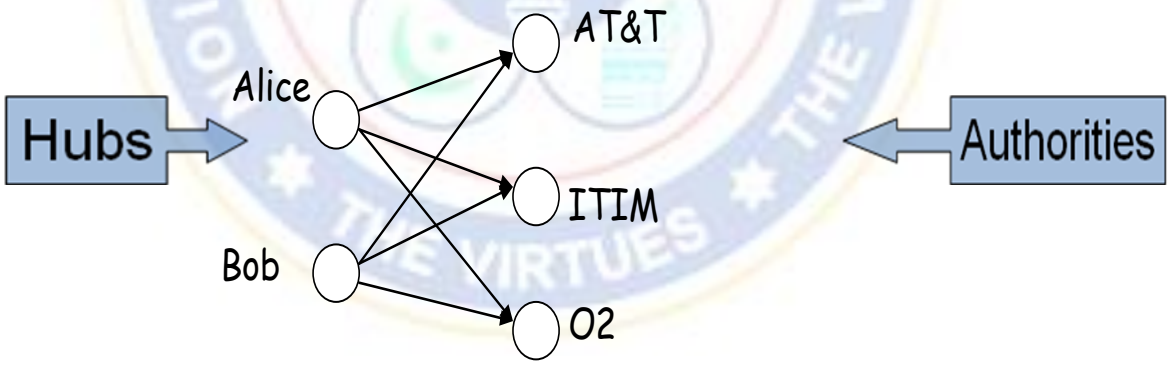
- Page rank and HITS are two solutions to the same problem.

1. In the page rank model the value of the link depends on the link into S.
2. In the HITS model, it depends on the value of the other links out of S.

- The algorithm performs a series of iterations, each consisting of two basic steps:

Authority Update: Update each node's Authority score to be equal to the sum of the Hub Scores of each node that points to it. That is, a node is given a high authority score by being linked from pages that are recognized as Hubs for information.

Hub Update: Update each node's Hub Score to be equal to the sum of the Authority Scores of each node that it points to. That is, a node is given a high hub score by linking to nodes that are considered to be authorities on the subject.



4.4 SEARCHING AND RANKING

Web Query languages

- Web query languages require knowledge of the web site and the language syntax. They are hard to use.

- Query is based on content of each page. The power of the web resides in its capability of redirecting the information flow via hyperlinks, so it should appear natural that in order to evaluate the information content of a web object, the web structure has to be carefully analyzed.
- Recent experiments seem to confirm that hyperlinks can be very valuable in locating or organizing information. They have been used:

To improve an initial ranking of documents.
To compute an estimate of a web pages popularity.
To find the most important hubs and authorities for a given topic.

Web Agents

- Web agents are complex software systems that operate in the World Wide Web, the internet and related corporate, government or military intranets. They are designed to perform a variety of tasks from caching and routing to searching categorizing and filtering.

- The web agents reads the request, talks to the server and sends the results back to the users web browser. A web agent can, for instance request a login web page, enter appropriate login parameters, post the login request and when done return the resulting web page to the caller.

- Agent might moves to one system to another to access remote resources and/or meets other agents. Web agents perform variety of tasks like routing, searching, categorizing and caching

4.5 RELEVANCE SCORING AND RANKING FOR WEB

Ranking of the documents on the basis of estimated relevance to the query is critical
Relevance ranking is based on factors such as

Term frequency
Frequency occurrences of query keywords in documents

Inverse document frequency
How many documents the query keyword occurs in.
Fewer-> give more importance to documents.

Relevance Ranking using Terms

TF-IDF

- The most well known weighting scheme
  - TF: still **term frequency**
  - IDF: **inverse document frequency**.

  N: total number of docs

  df_i: the number of docs that t_i appears.
- The final TF-IDF term weight is:

$$tf_{ij} = \frac{f_{ij}}{\max\{f_{1j}, f_{2j}, \ldots, f_{|V|j}\}}$$

$$idf_i = \log \frac{N}{df_i}$$

$$w_{ij} = tf_{ij} \times idf_i.$$

A term occurring frequently in the document but rarely in the rest of the collection is given high weight.

Many other ways of determining term weights have been proposed. Experimentally, *tf-idf* has been found to work well.

$$w_{ij} = tf_{ij}\, idf_i = tf_{ij} \log_2 (N/ df_i)$$

Given a document containing terms with given frequencies:

A(3), B(2), C(1)

Assume collection contains 10,000 documents and document frequencies of these terms are:

A(50), B(1300), C(250)

Then:

A: tf = 3/3; idf = $\log_2$(10000/50) = 7.6; tf-idf = 7.6

B: tf = 2/3; idf = $\log_2$ (10000/1300) = 2.9; tf-idf = 2.0

C: tf = 1/3; idf = $\log_2$ (10000/250) = 5.3; tf-idf = 1.8

- Query vector is typically treated as a document and also tf-idf weighted.
- Alternative is for the user to supply weights for the given query terms.

Relevance using Hyperlinks

When using keyword queries on the web, the number of documents is enormous.

Using term frequencies makes "spamming" easy.

Eg: Travel agent may add many occurrences of the word.

Most of the people are looking for pages from popular sites.

Refinement

When computing prestige based on links to a site , give more weightage to links from sites that themselves have higher prestige.

Connections to social networking theories that ranked prestige of people.

Eg: President of US

Hub and Authority base ranking

A hub is a page that stores many pages (on a topic)

An authority is a page that contains actual information on a topic.

Each page gets a hub prestige based on prestige of authorities that it points to

Each page gets a authority prestige based on prestige of hubs that it points to it.

Gain prestige definitions are cyclic and can be got by solving linear equations

Use authority prestige when ranking answers to a query.

4.6 SIMILARITY

A similarity measure is a function that computes the *degree of similarity* between two vectors.

Using a similarity measure between the query and each document:

It is possible to rank the retrieved documents in the order of presumed relevance.

It is possible to enforce a certain threshold so that the size of the retrieved set can be controlled.

Similarity between vectors for the document d_i and query q can be computed as the vector inner product (a.k.a. dot product):

$$\text{sim}(d_j,q) = d_j \bullet q$$

where w_{ij} is the weight of term i in document j and w_{iq} is the weight of term i in the query

For binary vectors, the inner product is the number of matched query terms in the document (size of intersection).

For weighted term vectors, it is the sum of the products of the weights of the matched terms.

The inner product is unbounded.

Favors long documents with a large number of unique terms.

Measures how many terms matched but not how many terms are *not* matched.

\Weighted:

$D_1 = 2T_1 + 3T_2 + 5T_3 \qquad D_2 = 3T_1 + 7T_2 + 1T_3$

$Q = 0T_1 + 0T_2 + 2T_3$

$\text{sim}(D_1 , Q) = 2*0 + 3*0 + 5*2 = 10$

$\text{sim}(D_2 , Q) = 3*0 + 7*0 + 1*2 = 2$

Cosine similarity measures the cosine of the angle between two vectors.

Inner product normalized by the vector lengths.

CosSim($\boldsymbol{d_j}$, $\boldsymbol{q}$) =

$$\frac{\vec{d_j} \cdot \vec{q}}{\left|\vec{d_j}\right| \cdot |\vec{q}|} = \frac{\sum_{i=1}^{t} (w_{ij} \cdot w_{iq})}{\sqrt{\sum_{i=1}^{t} w_{ij}^2 \cdot \sum_{i=1}^{t} w_{iq}^2}}$$

$D_1 = 2T_1 + 3T_2 + 5T_3$ CosSim(D_1 , Q) = 10 / √(4+9+25)(0+0+4) = 0.81

$D_2 = 3T_1 + 7T_2 + 1T_3$ CosSim(D_2 , Q) = 2 / √(9+49+1)(0+0+4) = 0.13

$Q = 0T_1 + 0T_2 + 2T_3$

D_1 is 6 times better than D_2 using cosine similarity but only 5 times better using inner product.

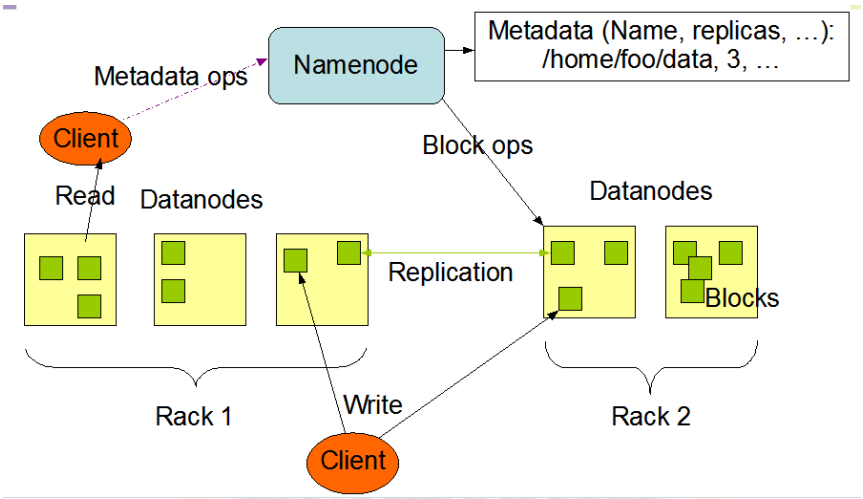
4.7 HADOOP AND MAP REDUCE

Hadoop provides a reliable shared storage and analysis system for large scale data processing.

Storage provides HDFS (Distributed file system)

Analysis provides by Map reduce. (Distributed Data Processing model)

HDFS Architecture



Name Node:

Stores all metadata: File name, locations of each block on data nodes, file attributes etc…

Data Node

Stores file contents as blocks

Different blocks of the same file are stored on different data nodes.

Data nodes exchange heartbeats with name node.

If no heartbeat received within a certain time period , data node assumed to be lost.

Losing name node is equivalent to losing all files on the system.

Hadoop provides two options

Backup files that make up the persistent state of the file system.

Run a Secondary Name Node.

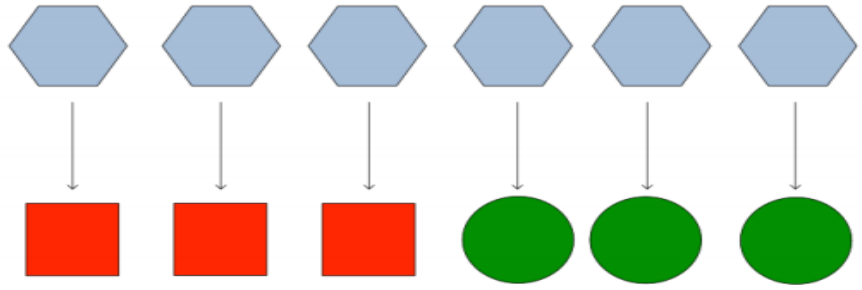
Map Reduce

Map reduce is a method for distributing a task across multiple nodes.

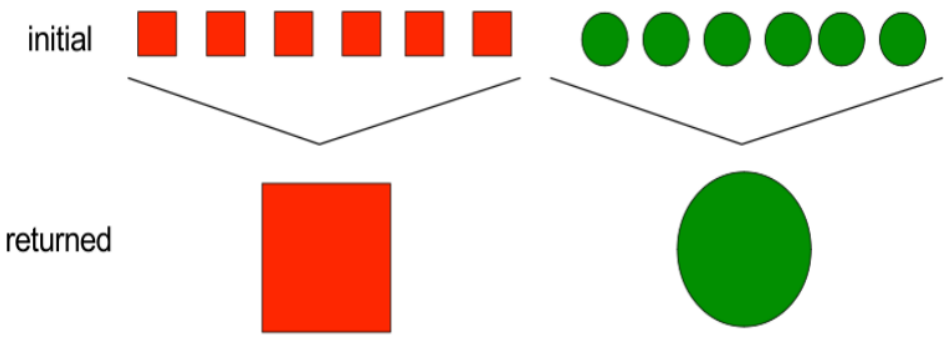
Each node processes data stored on that node

Consist of two Phase:1.Map 2.Reduce.

Map Reduce Process



Reduce Process:



4.8 EVALUATION

TREC Collection

TREC is a workshop series that provides the infrastructure for large-scale testing of retrieval technology.

The Text Retrieval Conference co-sponsored by the National Institute of Standards and Technology and U.S Department of Defense, was started in 1992 as part of TIPSTER Text program.

TREC workshop series has the following goals:

- To encourage research in information retrieval based on large test collections.

- To increase communication among industry, academic and government by creating an open forum for the exchange of research ideas

- To speed the transfer of technology from research labs into commercial products by demonstrating substantial improvements in retrieval methodologies on real-world problems

- To increase the availability of appropriate evaluation techniques Set of tracks in a particular TREC depends on:

  Interest of participants
  Appropriateness of task of TREC
  Need of sponsors
  Resource constraints

Evaluation measures at the TREC conference

- Summary table statistics – Single value measure can also be stored in a table to provide a statistical summary regarding the set of all the queries in a retrieval task.

- Recall-precision average – It consists of a table or graph with average precision at 11 standard recall levels.

- Document level average- Average precision is computed at specified document cutoff values.

- Average precision histogram- It consists of a graph which includes a single measure for each separate topic.

The CACM and ISI Collection

It is small collections about computer science literature. It is text of 3204 documents. The documents in the CACM test collection consists of all articles published in the communication of the ACM.

CACM collection also includes information on structured subfields as follows:

- Word stems from the title and abstract sections.
- Categories
- Direct reference between connections
- Bibliographic coupling connections.
- Number of co-citations for each pair of articles.

- Author names.
- Date information.

4.9 PERSONALIZED SEARCH

PART-B
Brief about Personalized search.(Nov/Dec'17)

In order to personalize search, we need to combine at least two different computational techniques - contextualization and individualization

Contextualization - "the interrelated conditions that occur within an activity..includes factors like the nature of information available, the information currently being examined, and the applications in use"

Individualization - "the totality of characteristics that distinguishes an individual.. Uses the user's goals, prior and tacit knowledge, past information-seeking behaviors"

Main ways to personalize a search are "**query augmentation**" and "**result processing**"

Query augmentation - when a user enters a query, the query can be compared against the contextual information available to determine if the query can be refined to include other terms

Query augmentation can also be done by computing the similarity between the query term and the user model - if the query is on a topic the user has previously seen, the system can reinforce the query with similar terms

This more concise query is then shown to the user and "submitted to a search engine for processing"

- Once the query has been augmented and processed by the search engine, the results can be "**individualized**"
- The results being individualized - this means that the information is filtered based upon information in the user's model and/or context
- The user model "can re-rank search results based upon the similarity of the content of the pages in the results and the user's profile"

- Another processing method is to re-rank the results based upon the "frequency, recency, or duration of usage..providing users with the ability to identify the most popular, faddish and time-consuming pages they've seen"

"Have Seen, Have Not seen" - this features allows new information to be identified and return to information already seen

4.10 COLLABORATION FILTERING

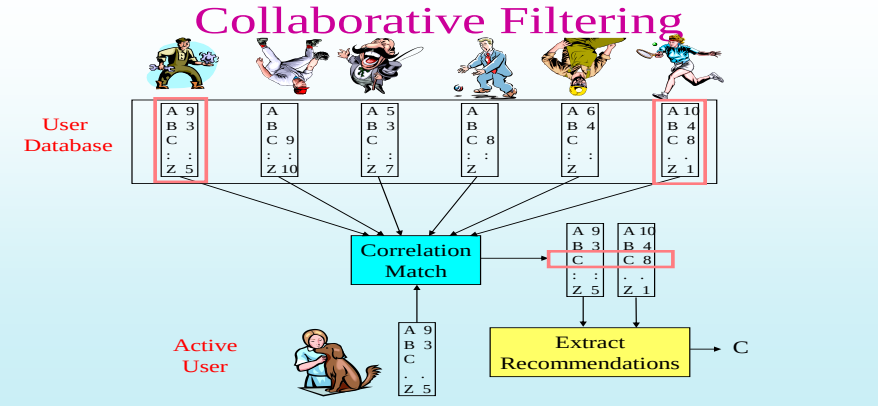
PART-A

Define user based collaborative filtering(Nov/Dec'16)

PART-B

Explain in detail the collaborative filtering using clustering technique(Nov/Dec'17)

Explain in detail Colllaborative filtering and content based recommendation system with an example(Apr/may'17)



Weight all users with respect to similarity with the active user.

Select a subset of the users (*neighbors*) to use as predictors.

Normalize ratings and compute a prediction from a weighted combination of the selected neighbors' ratings.

Present items with highest predicted ratings as recommendations.

Neighbor Selection

For a given active user, *a*, select correlated users to serve as source of predictions.

Standard approach is to use the most similar *n* users, *u*, based on similarity weights, $w_{a,u}$

Alternate approach is to include all users whose similarity weight is above a given threshold.

Rating Prediction

Predict a rating, $p_{a,i}$, for each item *i*, for active user, *a*, by using the *n* selected neighbor users, $u \in \{1,2,\ldots n\}$.

To account for users different ratings levels, base predictions on *differences* from a user's *average* rating.

Weight users' ratings contribution by their similarity to the active user.

$$p_{a,i} = \bar{r}_a + \frac{\sum_{u=1}^{n} w_{a,u}(r_{u,i} - \bar{r}_u)}{\sum_{u=1}^{n} | w_{a,u} |}$$

Similarity Weighting

•Typically use Pearson correlation coefficient between ratings for active user, *a*, and another user, *u*.

$$c_{a,u} = \frac{\text{covar}(r_a, r_u)}{\sigma_{r_a}\sigma_{r_u}}$$

r_a and r_u are the ratings vectors for the *m* items rated by

both *a* and *u*

$$\text{covar}(r_a, r_u) = \frac{\sum_{i=1}^{m}(r_{a,i} - \bar{r}_a)(r_{u,i} - \bar{r}_u)}{m}$$

$r_{i,j}$ is user *i*'s rating for item *j*

$$\bar{r}_x = \frac{\sum_{i=1}^{m} r_{x,i}}{m}$$

$$\sigma_{r_x} = \sqrt{\frac{\sum_{i=1}^{m}(r_{x,i} - \bar{r}_x)^2}{m}}$$

Covariance and Standard Deviation

- Covariance:

$$\bar{r}_x = \frac{\sum_{i=1}^{m} r_{x,i}}{m}$$

•Standard Deviation:

$$\sigma_{r_x} = \sqrt{\frac{\sum_{i=1}^{m} (r_{x,i} - \bar{r}_x)^2}{m}}$$

$$\text{covar}(r_a, r_u) = \frac{\sum_{i=1}^{m} (r_{a,i} - \bar{r}_a)(r_{u,i} - \bar{r}_u)}{m}$$

Significance Weighting

Important not to trust correlations based on very few co-rated items.

$$w_{a,u} = s_{a,u} c_{a,u}$$

Include *significance weights*, $s_{a,u}$, based on number of co-rated items, m.

$$s_{a,u} = \begin{cases} 1 \text{ if } m > 50 \\ \frac{m}{50} \text{ if } m \leq 50 \end{cases}$$

Problems with Collaborative Filtering

Cold Start: There needs to be enough other users already in the system to find a match.

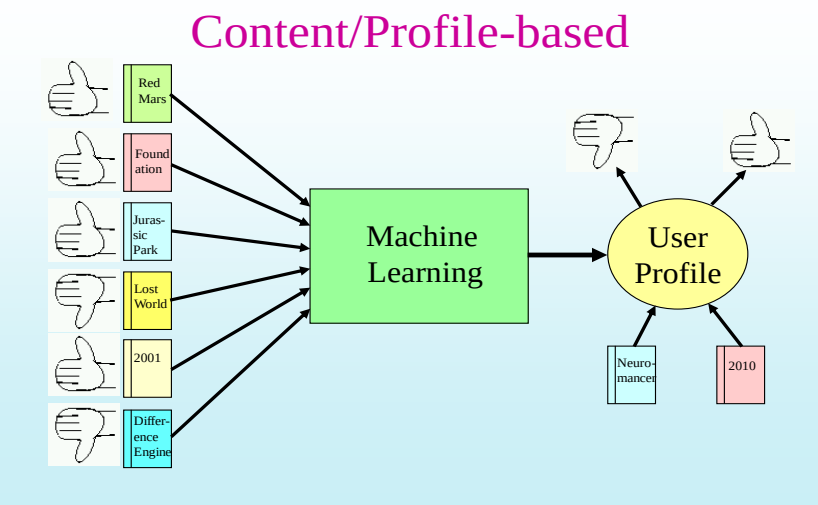
Sparsity: If there are many items to be recommended, even if there are many users, the user/ratings matrix is sparse, and it is hard to find users that have rated the same items.

First Rater: Cannot recommend an item that has not been previously rated.

- New items
- Esoteric items

Popularity Bias: Cannot recommend items to someone with unique tastes.

Content-Based Recommending



•Recommendations are based on information on the content of items rather than on other users' opinions.

•Uses machine learning algorithms to induce a profile of the users preferences from examples based on a featural description of content.

•Lots of systems

Advantages of Content-Based Approach

•No need for data on other users.

– No cold-start or sparsity problems.

•Able to recommend to users with unique tastes.

•Able to recommend new and unpopular items

– No first-rater problem.

• Can provide explanations of recommended items by listing content-features that caused an item to be recommended.

•Well-known technology The entire field of Classification Learning is at (y)our disposal!

Disadvantages of Content-Based Method

•Requires content that can be encoded as meaningful features.

•Users' tastes must be represented as a learnable function of these content features.

•Unable to exploit quality judgments of other users.

– Unless these are somehow included in the content features.

Combining Content and Collaboration

•Content-based and collaborative methods have complementary strengths and weaknesses.

•Combine methods to obtain the best of both.

•Various hybrid approaches:

– Apply both methods and combine recommendations.

– Use collaborative data as content.

– Use content-based predictor as another collaborator.

– Use content-based predictor to complete collaborative data.

Eg:Movie Domain

•Crawled Internet Movie Database (*IMDb*)

– Extracted content for titles in *EachMovie.*

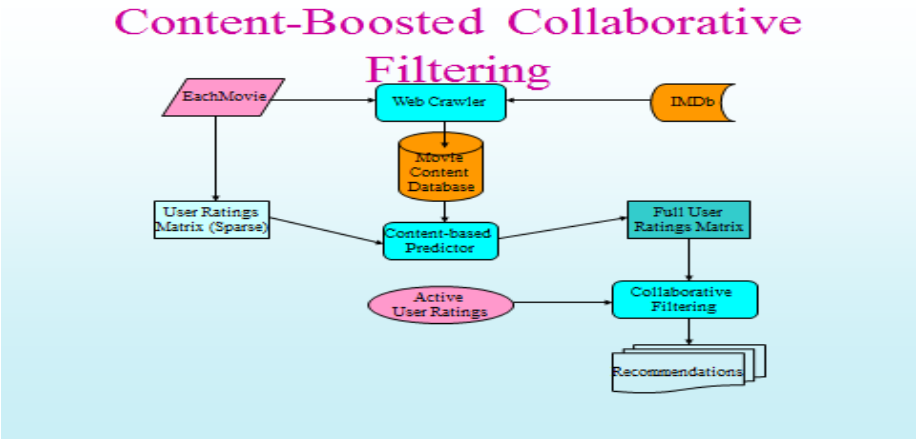
•Basic movie information:

– Title, Director, Cast, Genre, etc.

•Popular opinions:

– User comments, Newspaper and Newsgroup reviews, etc.

Content-Boosted Collaborative Filtering



4.11 HANDLING INVISIBLE WEB

Web sites that are hidden or are unable to be found or cataloged by regular search engines
200,000+ Web sites
550 billion individual documents compared to the three billion of the surface Web
Contains 7,500 terabytes of information compared to nineteen terabytes in the surface Web
Total *quality content* is 1,000 to 2,000 times greater than that of the surface Web.
Sixty of the largest sites collectively contain over 750 terabytes of information — They exceed the size of the surface Web forty times.
Fastest growing category of new information on the Internet
Fifty percent greater monthly traffic than surface sites
More highly linked to than surface sites
Narrower, with deeper content, than conventional surface sites
More than half of the content resides in topic-specific databases
Content is highly relevant to every information need, market, and domain.
Not well known to the Internet-searching public
Usually carried out using a "directory" or "search engine"
Fast and efficient
Misses most of what is out there
70% of searchers start from three sites (Nielson, 2003): Google,Yahoo, and MSN.

Searching Tools

Directories
Search engines
1.Searchable databases:
Typing is required.

Pages are not available until asked for (e.g., Library of Congress).
Pages are not static but dynamic (may not exist until requested).
Search engines can't handle "dynamic pages."
Search engines can't handle "input boxes."

2.password or login required:
(Spiders do not know passwords or login IDs.)

3. *Non-HTML pages:*
PDF, Word, Shockwave, Flash...

4. *Script-based (computer generated) pages:*
- Create all or part of a Web page
- Contain "?" in URL
-

4.12 SNIPPET GENERATION, SUMMARIZATION,QUESTION ANSWERING

PART-A

What is snippet generation?(Nov/Dec'16)

4.12.1 Snippet Generation

A snippet is a short summary of the document, which is designed so as to allow the user to decide its relevance. Snippet is Query-dependent summary.

Snippet consists of the document title and a short summary, which is automatically extracted.

Snippet generation steps

1.Rank each sentence in a document using a significant factor.

2.Select the top sentences for the summary.

Sentence selection:

Significance factor for a sentence is calculated based on the occurrence of significance words.

If fd,w is the frequency of word w in document d, then w is a significant word if it is not a stopword and

$F_{d,w}$>={ 7-0.1*25-sd, if sd<25

{ 7 25<=sd<=40

{ 7+0.1*(sd-40), otherwise

w w w w w w w w w w

(Initial sentence)

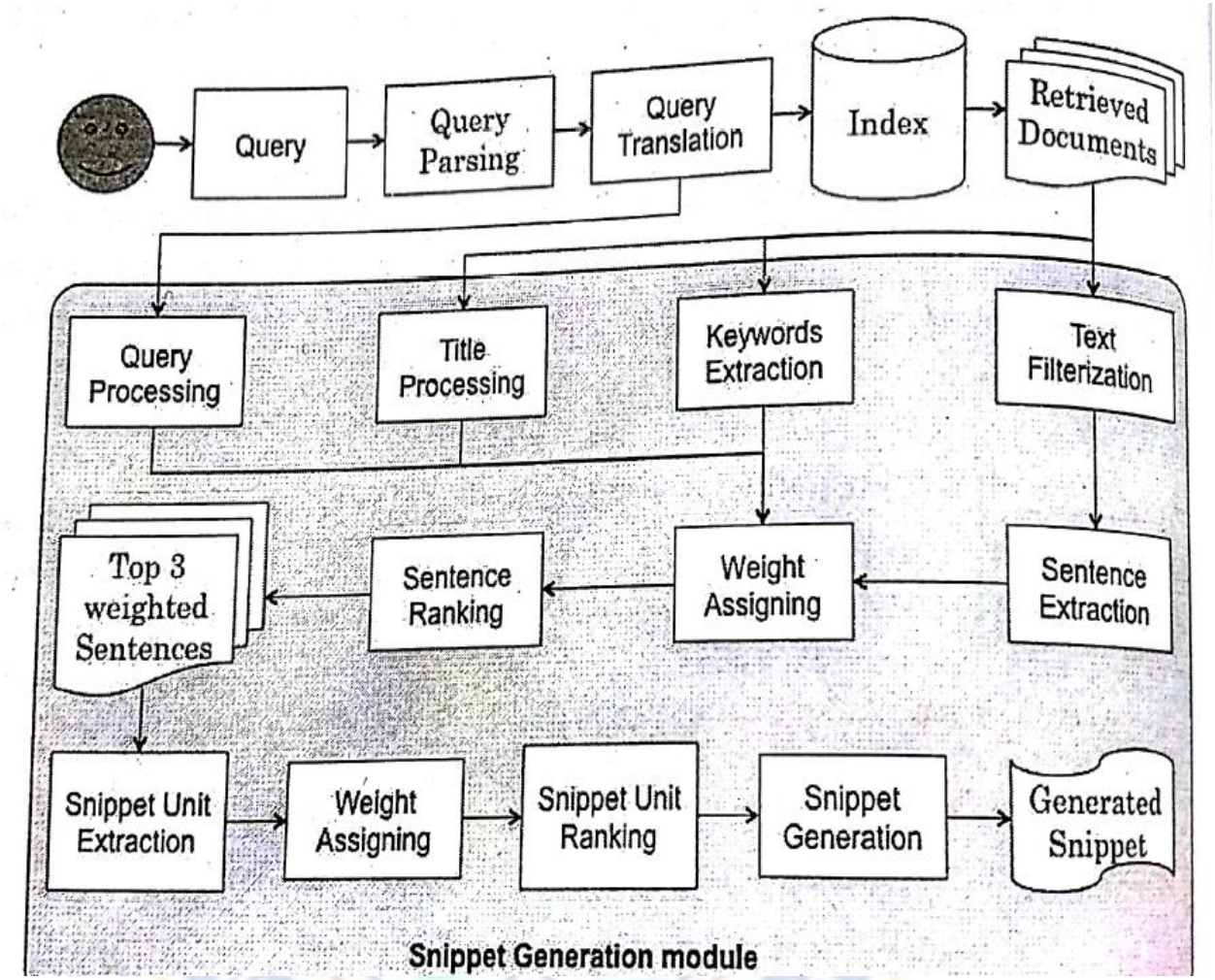
W w s w s s w w s w

(identify significant words)

W w [s w s s w w s] w

(Text span bracketed by significant words)

Significant factor = $4^2/7=2.3$



Snippet Generation module

Key Terms Extraction

Key term Extraction module has three sub modules like Query Term extraction, Title Words Extraction and Meta Keywords Extraction.

Query term Extraction module gets parsed and translated query . Now its extracts all the query terms from the query with their Boolean relations (AND/NOT) .

Sentence Extraction

This will take parsed text of the documents as inputs , filter the input parsed text and extracts all the sentence from parsed the text. Two models Text filterization and Sentence Extract

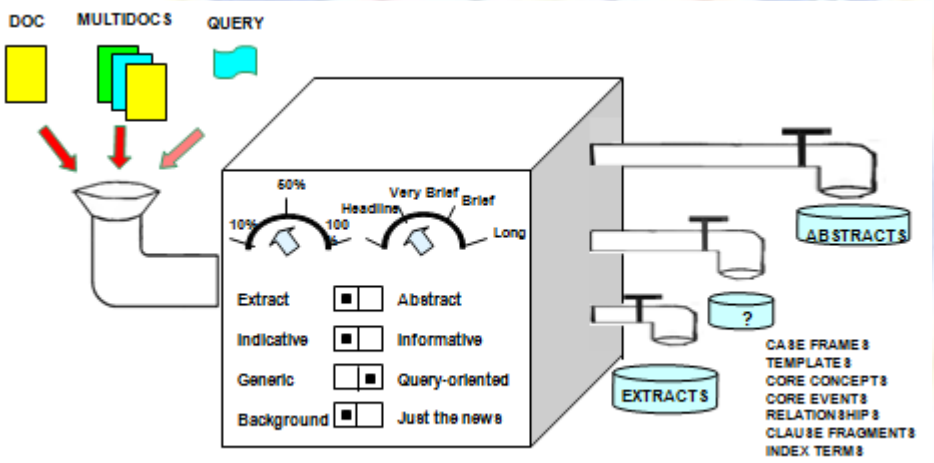
4.12.2 SUMMARIZATION

A summary is a text that is produced from one or more texts and contains a significant portion of the information in the original text is no longer than half of a text.

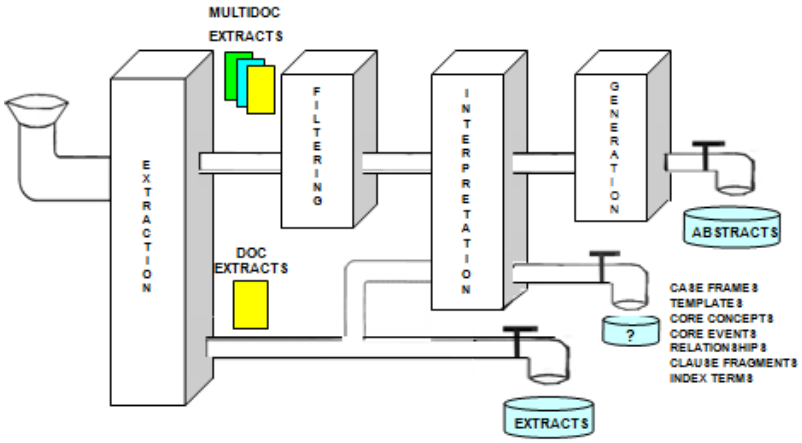
Generes of the Summary

- Indicative *vs*. informative
  ...used for quick categorization vs. content processing.
- Extract *vs*. abstract
  ...lists fragments of text vs. re-phrases content coherently.
- Generic *vs*. query-oriented
  ...provides author's view vs. reflects user's interest.
- Background *vs*. just-the-news
  ...assumes reader's prior knowledge is poor vs. up-to-date.
- Single-document *vs*. multi-document source
  ...based on one text vs. fuses together many texts.

Summarization Machine



Modules Of Summarization Machine



24

4.12.3 QUESTION ANSWERING

PART-B

Explain in detail about Community based Question Answering system.(Nov/Dec'17)

The main aim of QA is to present the user with a short answer to a question rather than a list of possibly relevant documents.

As it become more and more difficult to find answers on the WWW using standard search engines, question answering technology will become increasingly important.

4.13 CROSS LINGUAL

PART-B

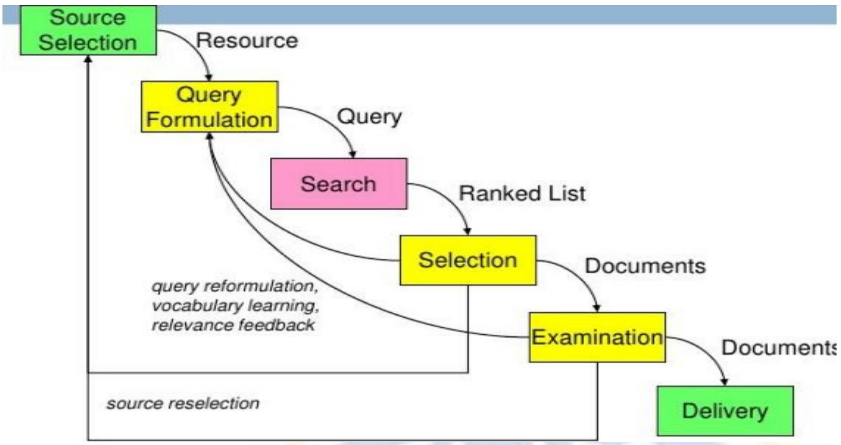
Explain in detail about the working of Naïve Bayesian classifier with an example.(Nov/Dec'16)

Cross-Lingual retrieval refers to the retrieval of documents that are in a language different from the one in which the query is expressed.

This allows users to search document collections in multiple languages and retrieve relevance information in a form that is useful to them, even when they have little or no linguistic competence in the target languages.

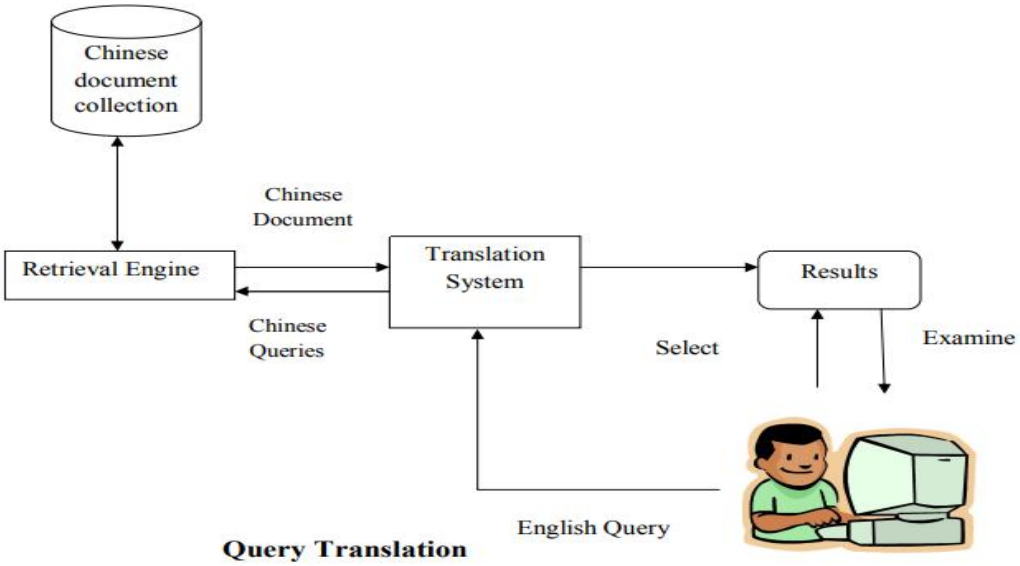
Cross lingual information retrieval is important for countries like India where very large fraction of people are not conversant with English and thus don't have access to the

vast store f information on the web.



Two methods are used to solve this problem:

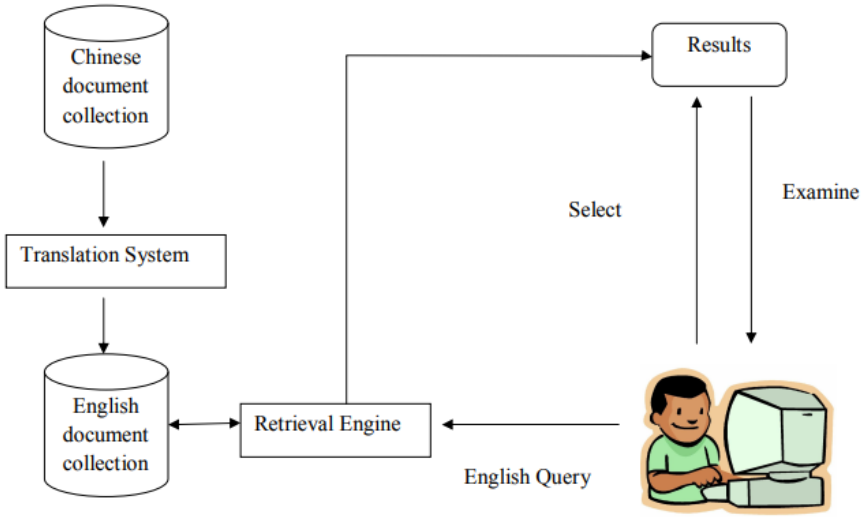
- Query translation:
- Translate English query into Chinese query
- Search Chinese document collection
- Translate retrieved results back into English
- Query translation is easy.
- Translation of documents must be performed at query time.



Query Translation

Document translation:

- Translate entire document collection into English
- Search collection in English
- Documents can be translated and stored offline. Automatic translation can be slow



Document Translation

****

UNIT-V

DOCUMENT TEXT MINING Information filtering; organization and relevance feedback – Text Mining -Text classification and clustering – Categorization algorithms: naive Bayes; decision trees; and nearest neighbor – Clustering algorithms: agglomerative clustering; k-means; expectation maximization (EM).

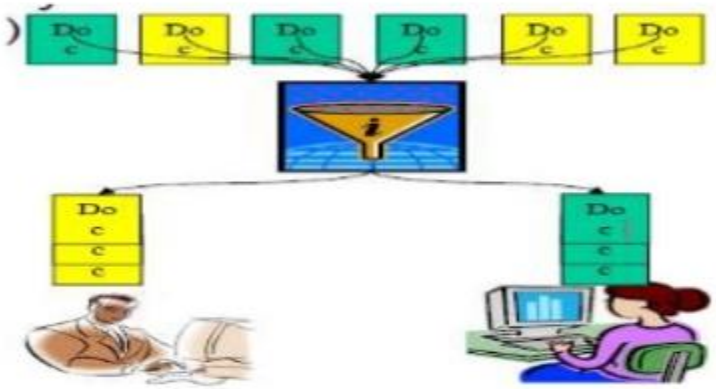
5.1 INFORMATION FILTERING; ORGANIZATION AND RELEVANCE FEEDBACK

PART-A

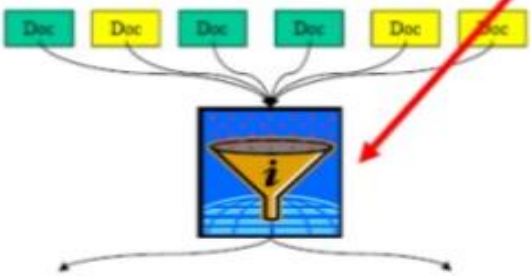
Differentiate between information filtering and information retrieval(Nov/Dec'17)

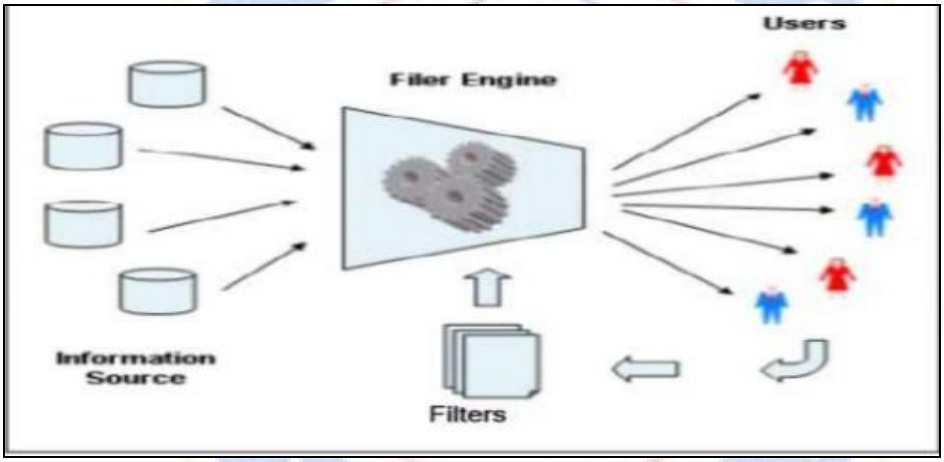
What are the characteristics of information filtering.(Nov/Dec'16)

- Information Filtering is the process of monitoring large amounts of dynamically generated information and pushing to a user the subset of information likely to be of her/his interest (based on her/his information needs).



An IFS needs an *information filter* that, when applied to an information item, evaluates whether the item is of interest or not to the considered user.



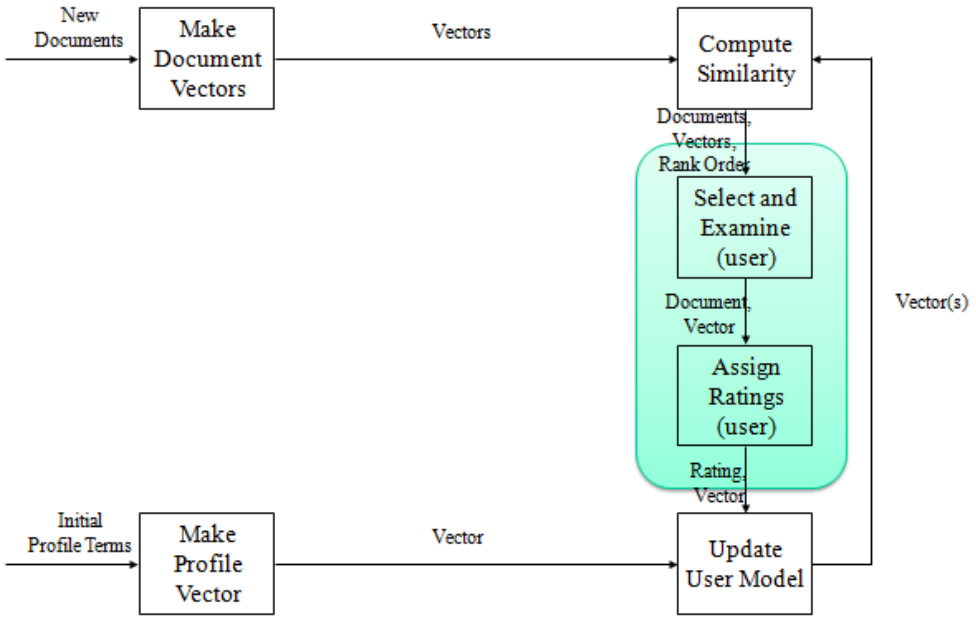


Three main categories of information filtering systems may be identified, which differ on how the information filter is defined and acquired:

- Content-based filtering — Objects to be filtered: generally texts; Filter engine based on *content analysis*
- Collaborative filtering — Objects to be filtered: products/goods; Filter engine based on *usage analysis*
- Hybrid Filtering — Combination of the two previous approaches

- Systems for recommending items (e.g. books, movies, CD's, web pages, newsgroup messages) to users based on examples of their preferences.
- Many on-line stores provide recommendations (e.g. Amazon, CDNow).
- Recommenders have been shown to substantially increase sales at on-line stores.
- There are two basic approaches to recommending:
  - Collaborative Filtering (a.k.a. social filtering)
  - Content-based

Relevance Feedback for Filtering



- Relevance feedback: user feedback on relevance of docs in initial set of results
  User issues a (short, simple) query

  The user marks some results as relevant or non-relevant.

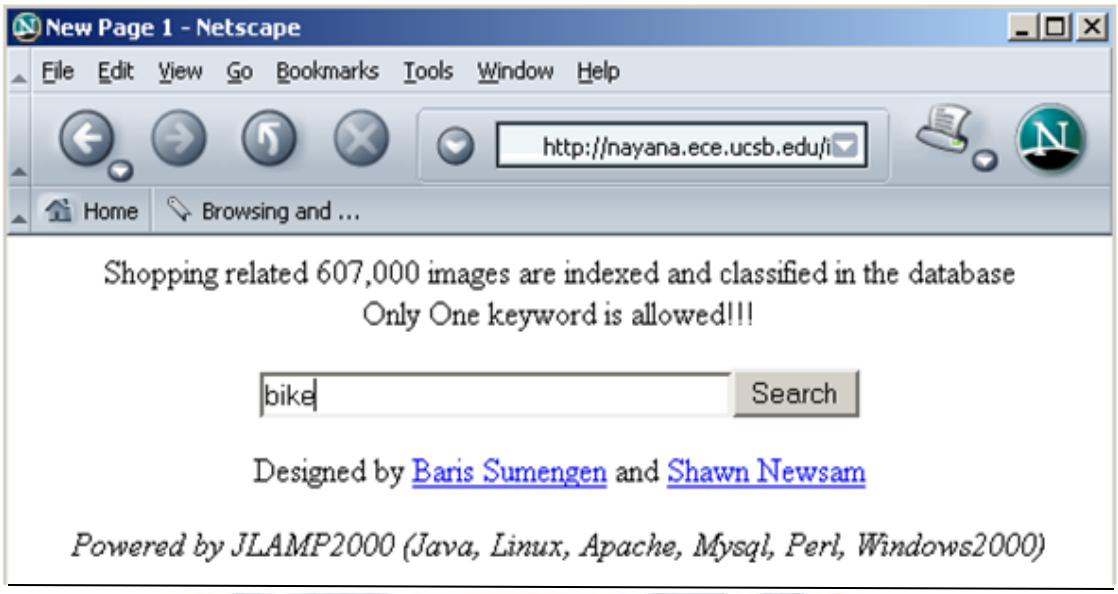
The system computes a better representation of the information need based on feedback.

Relevance feedback can go through one or more iterations.

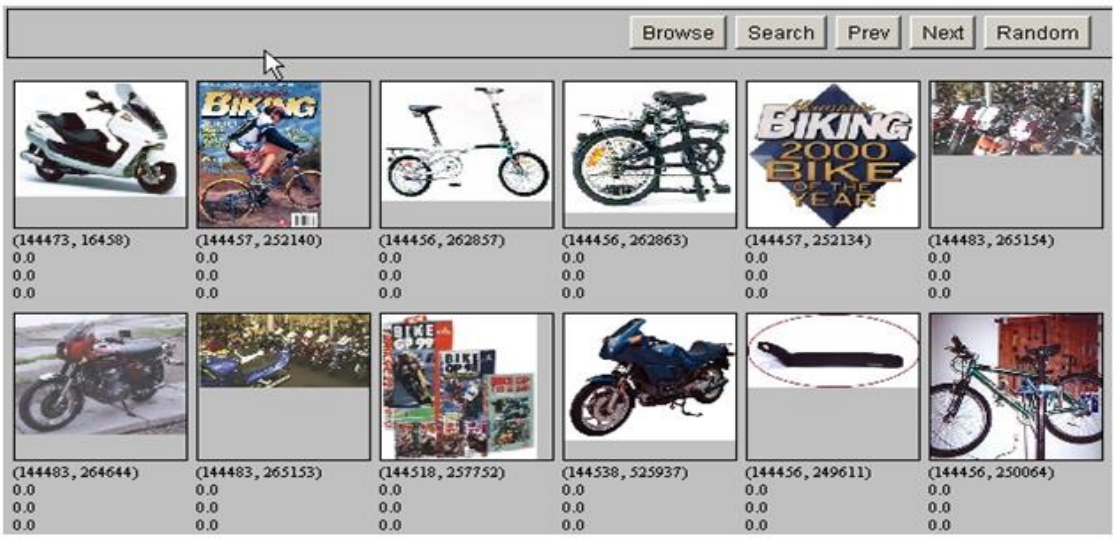
We will use *ad hoc retrieval* to refer to regular retrieval without relevance feedback.

Relevance Feed back Example:

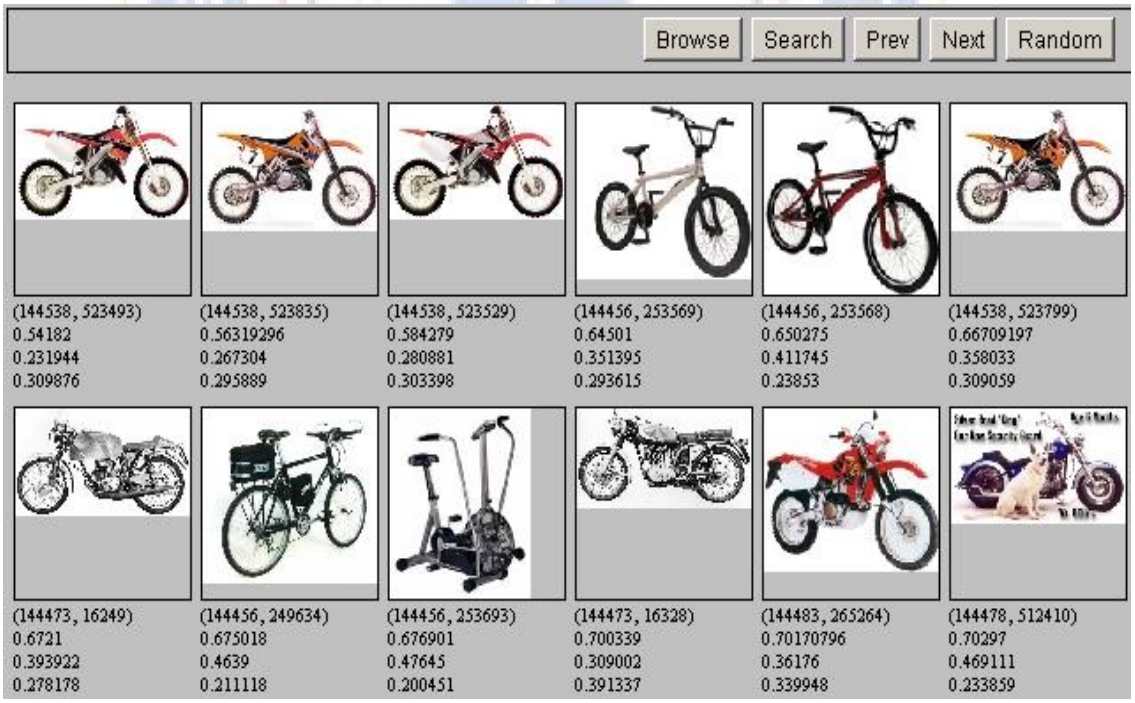
- Image search engine
  http://nayana.ece.ucsb.edu/imsearch/imsearch.html



Results for Initial Query



Results after Relevance feedback



The process of query modification is commonly referred as

R**elevance feedback**, when the user provides information on relevant documents to a query, or

Query expansion, when information related to the query is used to expand it.

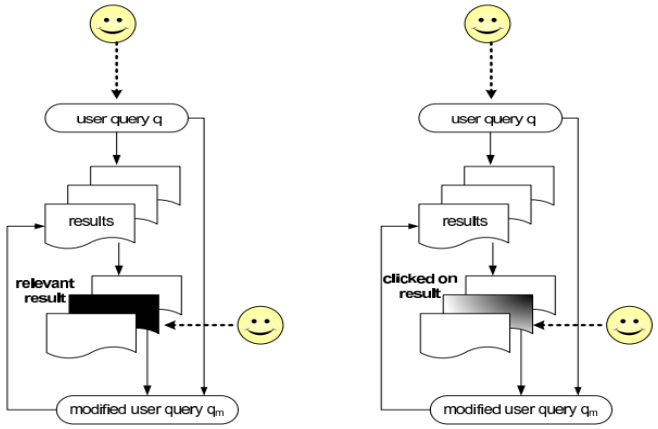
Two basic approaches of feedback methods:

explicit feedback, in which the information for query reformulation is provided directly by the users.

In an explicit relevance feedback cycle, the feedback information is provided directly by the users However, collecting feedback information is expensive and time consuming

In the Web, user clicks on search results constitute a new source of feedback information A click indicate a document that is of interest to the user in the context of the current query

Explicit Feedback



(a) relevance feedback (b) click feedback

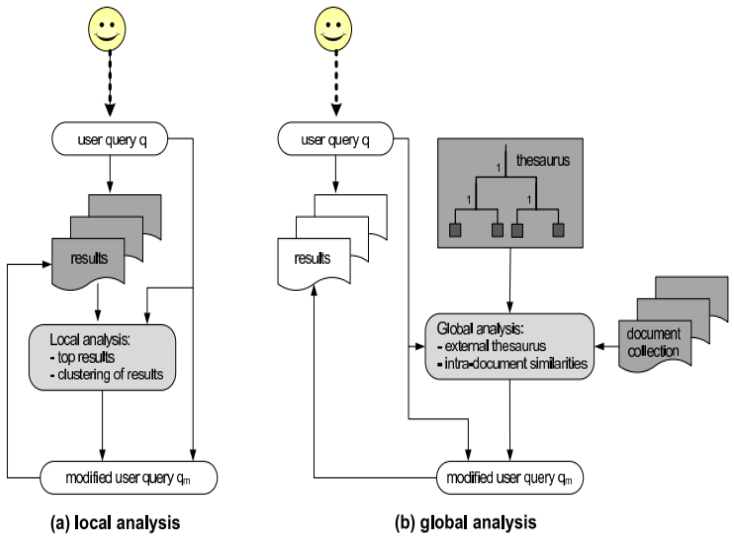
Implicit Feedback, in which the information for query reformulation is implicitly derived by the system.

There are **two basic approaches** for compiling **implicit feedback** information:

local analysis, which derives the feedback information from the top ranked documents in the result set

global analysis, which derives the feedback information from external sources such as a thesaurus

Implicit Feedback



Centroid

- The centroid is the center of mass of a set of points
- Recall that we represent documents as points in a high-dimensional space

Centroid:

$$\vec{\mu}(C) = \frac{1}{|C|}\sum_{d \in C}\vec{d}$$

where C is a set of documents.

The Rocchio algorithm uses the vector space model to pick a relevance feedback query

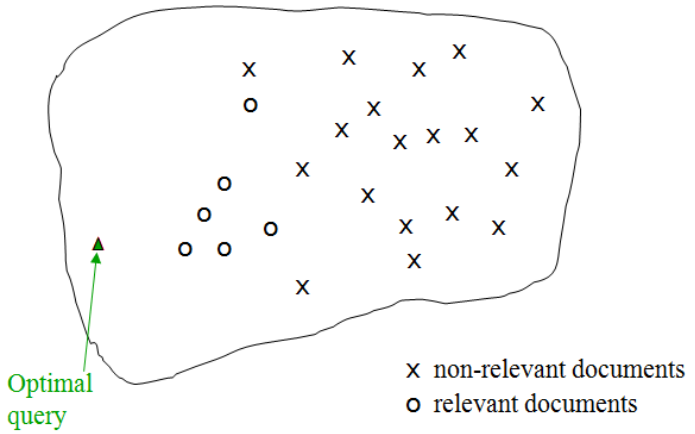
Rocchio seeks the query q_{opt} that maximizes

$$\vec{q}_{opt} = \arg\max_{\vec{q}} [\cos(\vec{q}, \vec{\mu}(C_r)) - \cos(\vec{q}, \vec{\mu}(C_{nr}))]$$

- Tries to separate docs marked relevant and non-relevant

$$\vec{q}_{opt} = \frac{1}{|C_r|}\sum_{\vec{d}_j \in C_r}\vec{d}_j - \frac{1}{|C_{nr}|}\sum_{\vec{d}_j \notin C_r}\vec{d}_j$$

Theoretical Best Query

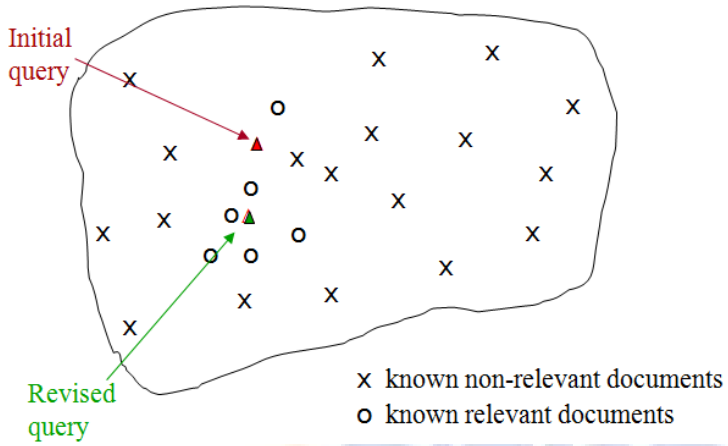


$$\vec{q}_m = \alpha\vec{q}_0 + \beta\frac{1}{|D_r|}\sum_{\vec{d}_j \in D_r}\vec{d}_j - \gamma\frac{1}{|D_{nr}|}\sum_{\vec{d}_j \in D_{nr}}\vec{d}_j$$

Rocchio 1971 Algorithm (SMART)

- D_r = set of known relevant doc vectors
- D_{nr} = set of known irrelevant doc vectors
  Different from C_r and C_{nr}
- q_m = modified query vector; q_0 = original query vector; α,β,γ: weights (hand-chosen or set empirically)
- New query moves toward relevant documents and away from irrelevant documents

Relevance feedback on initial query



- We can modify the query based on relevance feedback and apply standard vector space model.
- Use only the docs that were marked.
- Relevance feedback can improve recall and precision
- Relevance feedback is most useful for increasing *recall* in situations where recall is important

  Users can be expected to review results and to take time to iterate

  Positive feedback is more valuable than negative feedback

Query Expansion

- In relevance feedback, users give additional input (relevant/non-relevant) on documents, which is used to reweight terms in the documents
- In query expansion, users give additional input (good/bad search term) on words or phrases
- For each term, *t*, in a query, expand the query with synonyms and related words of *t* from the thesaurus

  feline → feline cat
- May weight added terms less than original query terms.
- Generally increases recall

- Widely used in many science/engineering fields
- May significantly decrease precision, particularly with ambiguous terms.
  "interest rate" → "interest rate fascinate evaluate"
- There is a high cost of manually producing a thesaurus

  And for updating it for scientific changes

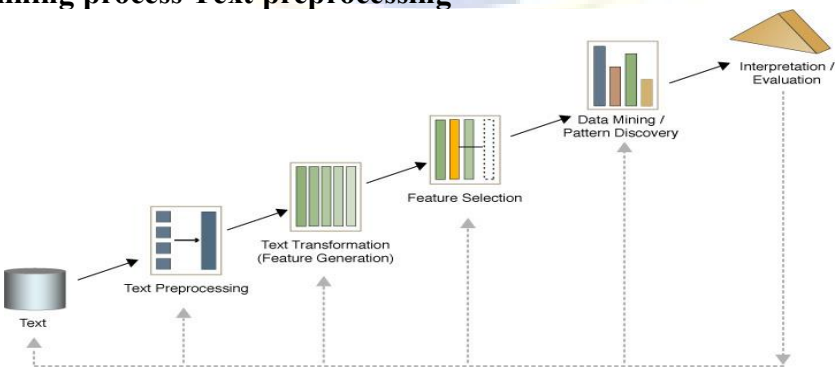
5.2 TEXT MINING

Another way to view text data mining is as a process of exploratory data analysis that leads to heretofore unknown information, or to answers for questions for which the answer is not currently known."

Text Mining Methods

- Information Retrieval
  - Indexing and retrieval of textual documents
- Information Extraction
  - Extraction of partial knowledge in the text
- Web Mining
  - Indexing and retrieval of textual documents and extraction of partial knowledge using the web
- Clustering
  - Generating collections of similar text documents

- **Text mining process Text preprocessing**



Syntactic/Semantic text analysis

- Part Of Speech (pos) tagging
  - Find the corresponding pos for each word

  e.g., John *(noun*) gave *(verb)* the *(det)* ball *(noun)*
- Word sense disambiguation
  - Context based or proximity based
  - Very accurate
- Parsing
  - Generates a parse tree (graph) for each sentence
  - Each sentence is a stand alone graph

Features Generation

Bag of words

- Text document is represented by the words it contains (and their occurrences)
  - e.g., "Lord of the rings" → {"the", "Lord", "rings", "of"}
  - Highly efficient
  - Makes learning far simpler and easier
  - Order of words is not that important for certain applications
- Stemming: identifies a word by its root
  - Reduce dimensionality
  - e.g., flying, flew → fly
  - Use Porter Algorithm
- Stop words: The most common words are unlikely to help text mining
  - e.g., "the", "a", "an", "you" …

Features Selection

Simple counting

Statistics

- Reduce dimensionality
- Learners have difficulty addressing tasks with high dimensionality
- Irrelevant features
- Not all features help!
- e.g., the existence of a noun in a news article is unlikely to help classify it as "politics" or "sport"
- Use Weightening

Text/Data Mining

Classification- Supervised learning

Clustering- Unsupervised learning

- **Given:** a collection of labeled records *(training set)*
  - Each record contains a set of features *(attributes),* and the true class *(label)*
- **Find:** a model for the class as a function of the values of the features
- **Goal:** previously unseen records should be assigned a class as accurately as possible

- A test set is used to determine the accuracy of the model. Usually, the given data set is divided into training and test sets, with training set used to build the model and test set used to validate it

- **Analyzing results**

Correct classification: The known label of test sample is identical with the *class result* from the classification model

Accuracy ratio: the percentage of test set samples that are correctly classified by the model

A *distance measure* between classes can be used

- e.g., classifying "football" document as a "basketball" document is not as bad as classifying it as "crime".

Text Mining Application

Email: Spam filtering

News Feeds: Discover what is interesting

Medical: Identify relationships and link information from different medical fields

Marketing: Discover distinct groups of potential buyers and make suggestions for other products

Industry: Identifying groups of competitors web pages

Job Seeking: Identify parameters in searching for jobs

5.3 TEXT CLASSIFICATION AND CLUSTERING

Text classification and Naive Bayes

PART-A

What are the desirable properties of a clustering algorithm(Nov/Dec'16'17)

A class is a more general subject area like *China* or *coffee*. Such more general classes are usually referred to as *topics*, and the classifica- tion task is then called *text classification*, *text categorization*, *topic classification*, or *topic spotting*.

In text classification, we are given a description $d \in X$ of a document, where X is the *document space*; and a fixed set of *classes* $C = \{c_1, c_2, \ldots, c_J\}$. Classes are also called *categories* or *labels*. Typically, the document space X is some type of high-dimensional space, and the classes are human defined for the needs of an application, as in the examples *China* and *documents that talk about*

multicore computer chips above. We are given a *training set* D of labeled documents (d, c),where $(d, c) \in X \times C$. For example:

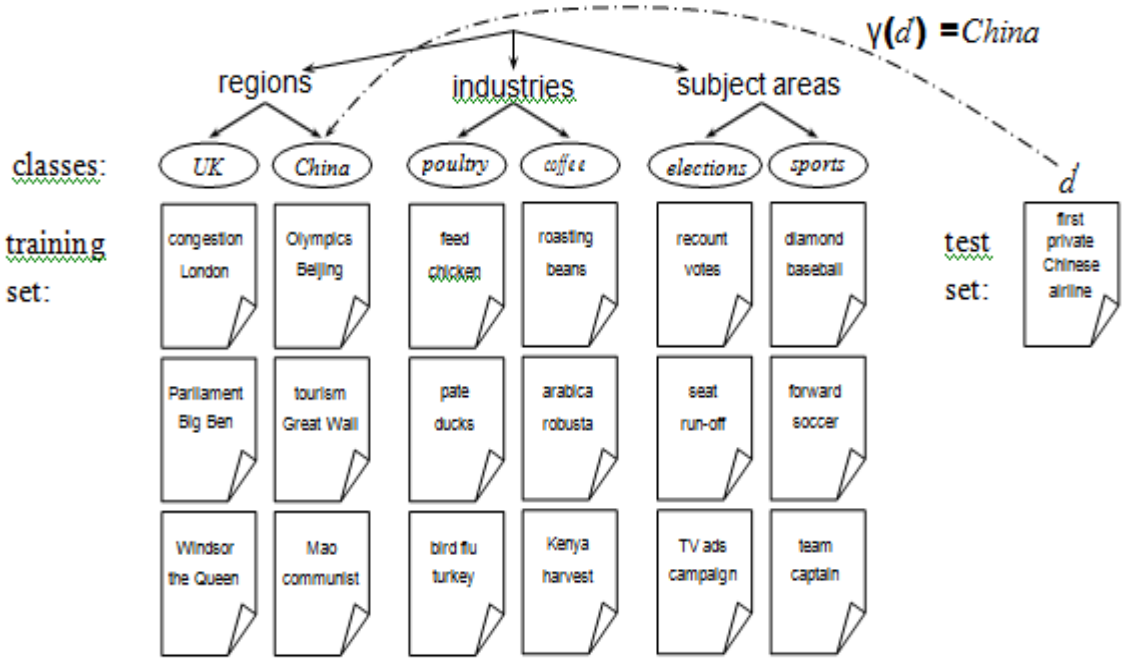
(d, c) = (Beijing joins the World Trade Organization, *China*)

for the one-sentence document *Beijing joins the World Trade Organization* and the class (or label) *China*.

Using a *learning method* or *learning algorithm*, we then wish to learn a clas- sifier or *classification function* γ that maps documents to classes:

$$\gamma : X \rightarrow C$$

This type of learning is called *supervised learning* because a supervisor (the human who defines the classes and labels training documents) serves as a teacher directing the learning process. We denote the supervised learning method by and write $\Gamma(D) = \gamma$. The learning method Γ takes the training set D as input and returns the learned classification function γ.



5.4 TEXT CATEGORIZATION

PART-B

Explain in detail the multiple –Bernoulli and multinomial model(Nov/Dec'17)

Discuss in detail about the working of naïve Bayesian and its limitation in web search. (Nov/Dec'16)

State bayes Theorem, Explain Naïve bayes classification with an example(Apr/may'17)

Assign labels to each document or web-page:

- Labels are most often topics such as Yahoo-categories
  - *e.g., "finance," "sports," "news>world>asia>business"*
- Labels may be genres
  - *e.g., "editorials" "movie-reviews" "news"*
- Labels may be opinion
  - *e.g., "like", "hate", "neutral"*
- Labels may be domain-specific binary
  - *e.g., "interesting-to-me" : "not-interesting-to-me"*
  - *e.g., "spam" : "not-spam"*
  - *e.g., "contains adult language" :"doesn't"*
- Given:
  - A description of an instance, $x \in X$, where X is the *instance language* or *instance space*.
    - Issue: how to represent text documents.
  - A fixed set of categories:
    $C = \{c_1, c_2, \ldots, c_n\}$
- Determine:
  - The category of x: $c(x) \in C$, where $c(x)$ is a *categorization function* whose domain is X and whose range is C.
    - We want to know how to build categorization functions ("classifiers").

5.4.1 Naive Bayes text classification

The first supervised learning method we introduce is the *multinomial Naïve Bayes* or *multinomial NB* model, a probabilistic learning method. The proba bility of a document *d* being in class *c* is computed as

where $P(t_k\ c)$ is the conditional probability of term t_k occurring in a document of class *c*.

We interpret $P(t_k\ c)$ as a measure of how much evidence t_k contributes that *c* is the correct class.

$P(c)$ is the prior probability of a document occurring in class *c*. If a document's terms do not provide clear evidence for one class versus another,

we choose the one that has a higher prior probability. $t_1, t_2, \ldots, t_{n_d}$ are the tokens in *d* that are part of the vocabulary we use for classification and n_d is the number of such tokens in *d*.

Naive Bayes algorithm (multinomial model): Training and testing

```
TRAINMULTINOMIALNB(C, D)
1  V ← EXTRACTVOCABULARY(D)
2  N ← COUNTDOCS(D)
3  for each c ∈ C
4  do N_c ← COUNTDOCSINCLASS(D, c)
5     prior[c] ← N_c/N
6     text_c ←
      CONCATENATETEXTOFALLDOCSINCLASS(D, c)
7     for each t ∈ V
8     do T_ct ← COUNTTOKENSOFTERM(text_c, t)
9     for each t ∈ V
10    do condprob[t][c] ←
11 return V, prior, condprob
```

```
APPLYMULTINOMIALNB(C, V, prior, cond prob, d)
1  W ← EXTRACTTOKENSFROMDOC(V, d)
2  for each c ∈ C
3  do score[c] ← log prior[c]
4     for each t ∈ W
5     do score[c] += log condprob[t][c]
6  return arg max_{c∈C} score[c]
```

5.4.2 The Bernoulli model

There are two different ways we can set up an NB classifier. The model we introduced in the previous section is the multinomial model. It generates one term from the vocabulary in each position of the document.

An alternative to the multinomial model is the *multivariate Bernoulli model*. or *Bernoulli model*. It is equivalent to the binary independence model, which generates an indicator for each term of the vocabulary, either 1 indicating presence of the term in the document or 0 indi- cating absence.

Applying the Bernoulli model to the example in we

have the same estimates for the priors as before: $\hat{P}(c) = 3/4$, $\hat{P}(\bar{c}) = 1/4$. The conditional probabilities are:

(Chinese|c) = (3 + 1)/(3 + 2) = 4/5

P^(Japan|c) = P^(Tokyo|c) = (0 + 1)/(3 + 2) = 1/5

P^(Beijing|c) = P^(Macao|c) = P^(Shanghai|c)=(1 + 1)/(3 + 2) = 2/5

P^(Chinese|c) = (1 + 1)/(1 +-2) = 2/3

P^(Japan|c) = P^(Tokyo|c) = - (1 + 1)/(1 + 2) = 2/3

P^(Beijing|c) = P^(Macao|c) = P^(Shanghai|c) = (0 + 1)/(1 + 2) = 1/3

The denominators are (3 + 2) and (1 + 2) because there are three documents in *c* and one document in $\bar{c}$ and because the constant *B* in

The scores of the test document for the two classes are

P^(c|d5) ∝ P^(c) · P^(Chinese|c) · P^(Japan|c) · P^(Tokyo|c)

· (1 − P^(Beijing|c)) · (1 − P^(Shanghai|c)) · (1 − P^(Macao|c))
=3/4 · 4/5 · 1/5 · 1/5 · (1−2/5) · (1−2/5) · (1−2/5)

· ≈ 0.005

· and, analogously,

P^(c|d5) _ ∝ 1/4 · 2/3 · 2/3 · 2/3 · (1−1/3) · (1−1/3) · (1−1/3)

· ≈ 0.022

· Thus, the classifier assigns the test document to *c* = *not-China*. When looking only at binary occurrence and not at term frequency, Japan and Tokyo are indicators

for c (2/3 > 1/5) and the conditional probabilities of Chinese for c and c are not different enough (4/5 vs. 2/3) to affect the classification decision.

Properties of Naive Bayes

$$c_{map} = \arg\max_{c \in \mathbf{C}} P(c|d)$$
$$= \arg\max_{c \in \mathbf{C}} \frac{P(d|c)P(c)}{P(d)}$$
$$= \arg\max_{c \in \mathbf{C}} P(d|c)P(c),$$

$P(d)$ is the same for all classes and does not affect the argmax.

To generate a document, we first choose class c with probability $P(c)$

The two models differ in the formalization of the second step, the generation of the document given the class, corresponding to the conditional distribution $P(d|c)$:

Multinomial $P(d|c) = P((t_1, \ldots, t_k, \ldots, t_{n_d})|c)$

Bernoulli $P(d|c) = P((e_1, \ldots, e_i, \ldots, e_M)|c),$

where $t_1, \ldots, t_{n_d}$ is the sequence of terms as it occurs in d (minus terms that were excluded from the vocabulary) and $e_1, \ldots, e_i, \ldots, e_M$ is a binary vector of dimensionality M that indicates for each term whether it occurs in d or not.

5.4.3 k Nearest Neighbor Classification

To classify document d into class c

- Define k-neighborhood N as k nearest neighbors of d
- Count number of documents i in N that belong to c
- Estimate P(c|d) as i/k
- Choose as class argmax_c P(c|d) [= majority class]
- Learning is just storing the representations of the training examples in D.

· Testing instance x:

Compute similarity between x and all examples in D.

- Assign x the category of the most similar example in D.
- Does not explicitly compute a generalization or category prototypes.
- Also called:
- Case-based learning

- Memory-based learning
- Lazy learning

Using only the closest example to determine the categorization is subject to errors due to:

A single atypical example.

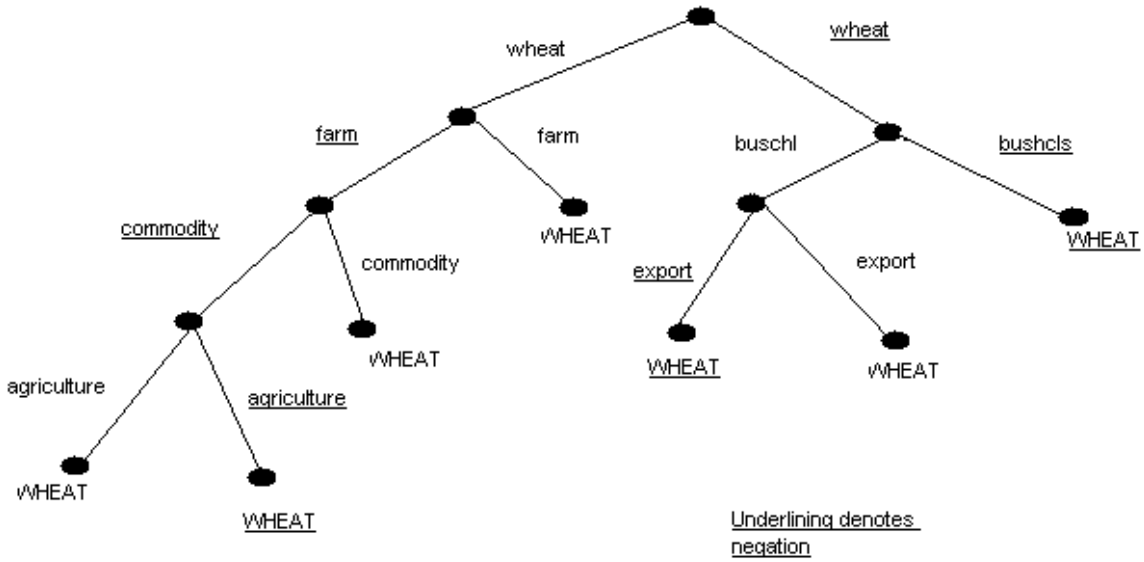
Noise (i.e. error) in the category label of a single training example.

· More robust alternative is to find the *k* most-similar examples and return the majority category of these *k* examples.

· Value of *k* is typically odd to avoid ties; 3 and 5 are most common.

Decision Tree Classification

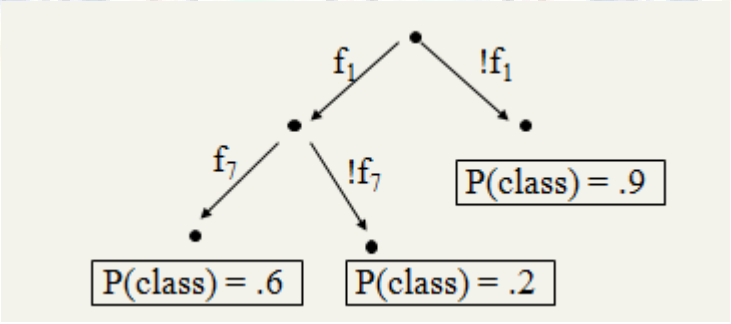
- Tree with internal nodes labeled by terms
- Branches are labeled by tests on the weight that the term has
- Leaves are labeled by categories
- Classifier categorizes document by descending tree following tests to leaf
- The label of the leaf node is then assigned to the document
- Most decision trees are binary trees (never disadvantageous; may require extra internal nodes)
- DT make good use of a few high-leverage features



Learn a sequence of tests on features, typically using top-down, greedy search

At each stage choose the unused feature with highest Information Gain (feature/class MI)

Binary (yes/no) or continuous decisions



- Fully grown trees tend to have decision rules that are overly specific and are therefore unable to categorize documents well
  - Therefore, pruning or early stopping methods for Decision Trees are normally a standard part of classification packages
- Use of small number of features is potentially bad in text cat, but in practice decision trees do well for some text classification tasks

- Decision trees are very easily interpreted by humans – much more easily than probabilistic methods like Naive Bayes
- Decision Trees are normally regarded as a symbolic machine learning algorithm, though they can be used probabilistically

5.6 CLUSTERING

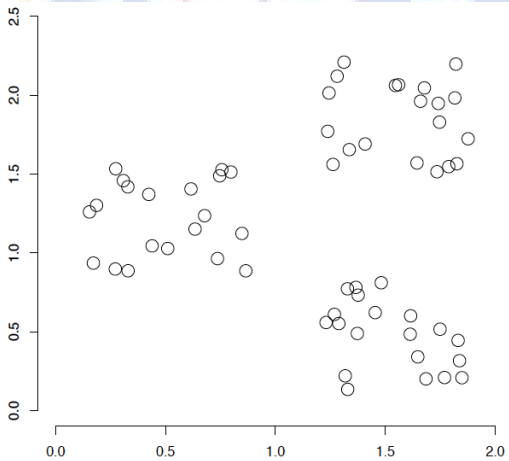
PART-A
Differentiate supervised learning and unsupervised learning(Nov/Dec'17)
PART-B
Explain the process of choosing K in Nearest neighbor clustering. (Nov/Dec'17)

- Clustering: the process of grouping a set of objects into classes of similar objects
  Documents within a cluster should be similar.

  Documents from different clusters should be dissimilar.

- The commonest form of *unsupervised learning*

  Unsupervised learning = learning from raw data, as opposed to supervised data where a classification of examples is given
  A common and important task that finds many applications in IR and other places.

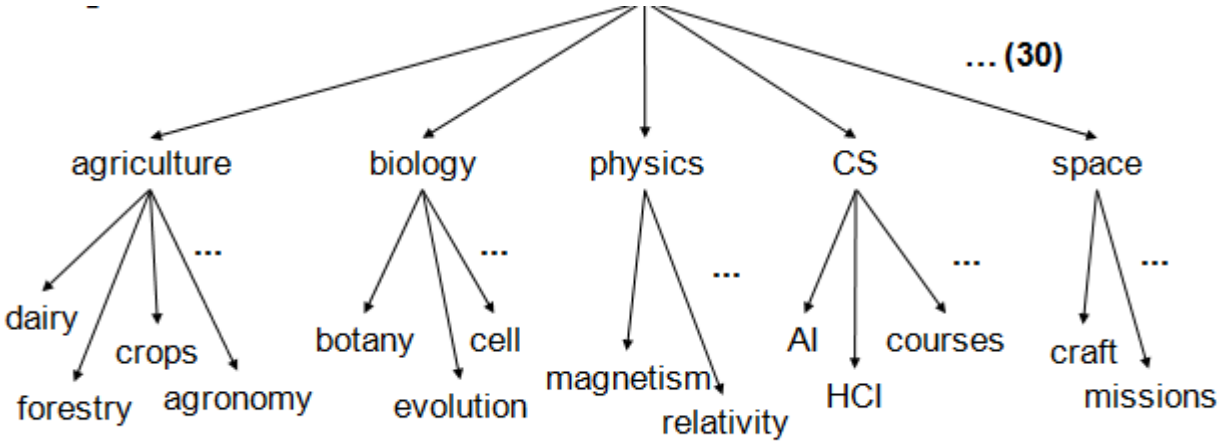


Whole corpus analysis/navigation
Better user interface: search without typing
For improving recall in search applications
Better search results (like pseudo RF)
For better navigation of search results
Effective "user recall" will be higher

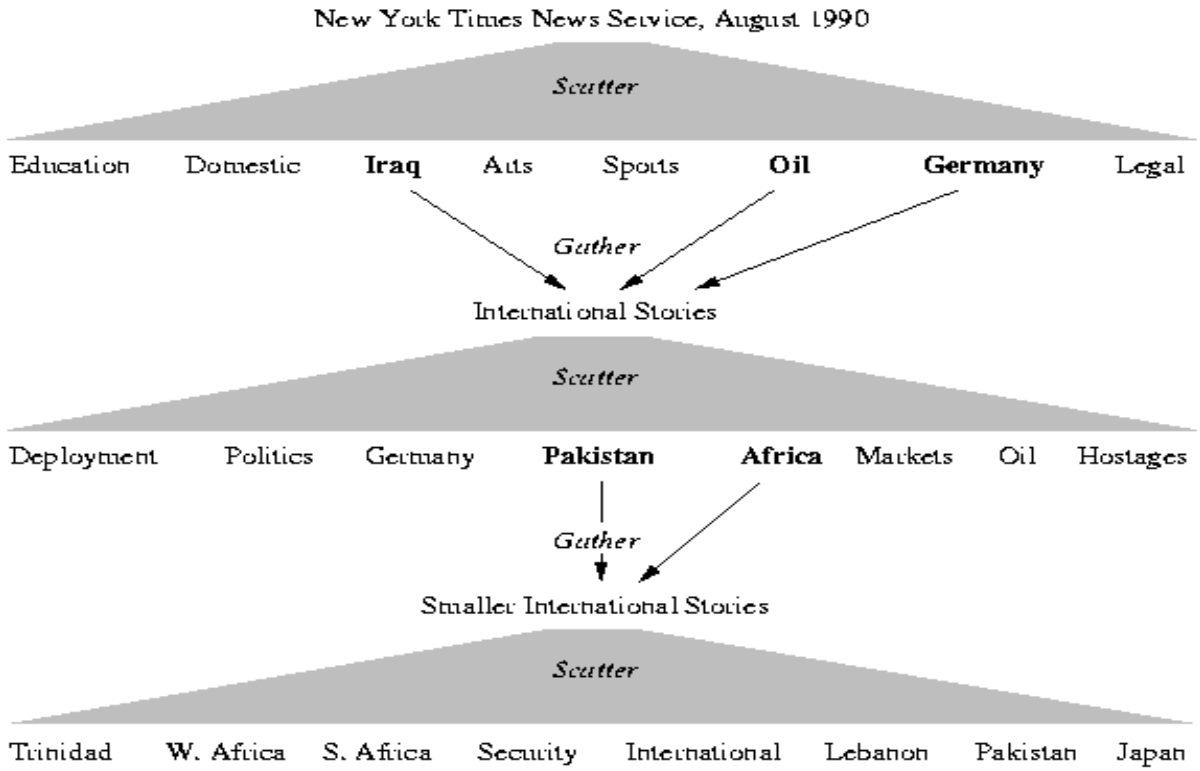
For speeding up vector space retrieval

Cluster-based retrieval gives faster search

www.yahoo.com/Science



Scatter/Gather: Cutting, Karger, and Pedersen



5.6.1 K-MEANS

Assumes documents are real-valued vectors.
Clusters based on *centroids* (aka the *center of gravity* or mean) of points in a cluster, *c*:

$$\vec{\mu}(c) = \frac{1}{|c|}\sum_{\vec{x}\in c}\vec{x}$$

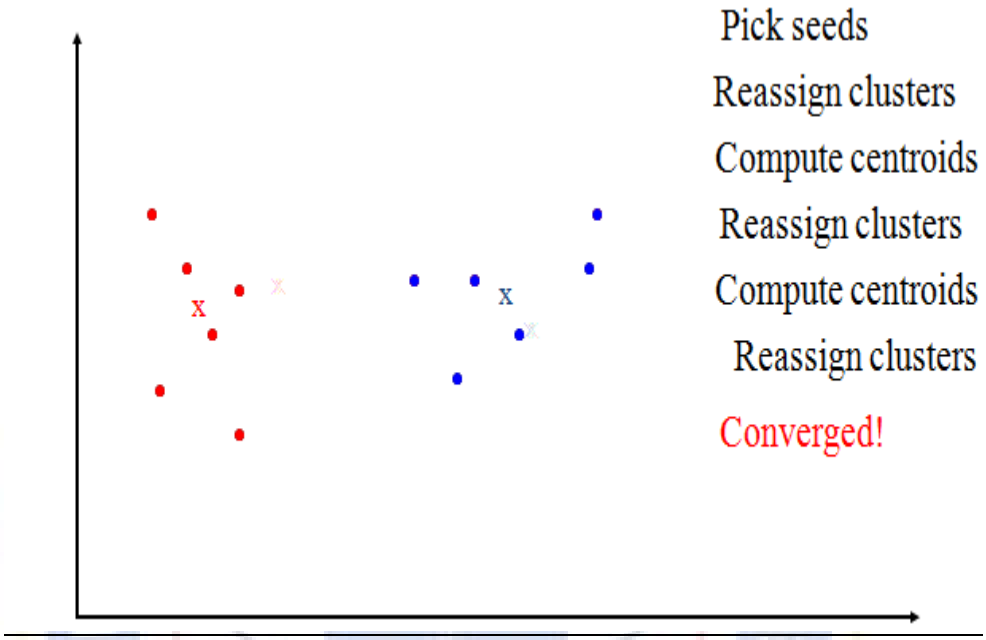
Reassignment of instances to clusters is based on distance to the current cluster centroids.
(Or one can equivalently phrase it in terms of similarities)

Select *K* random docs $\{s_1, s_2, \ldots s_K\}$ as seeds.

Until clustering *converges* (or other stopping criterion):
For each doc d_i:
Assign d_i to the cluster c_j such that $dist(x_i, s_j)$ is minimal.
(*Next, update the seeds to the centroid of each cluster*)

For each cluster c_j
$s_j = \mu(c_j)$

K Means Example (*K*=2)



5.6.2 AGGLOMERATIVE CLUSTERING

Starts with each doc in a separate cluster
then repeatedly joins the *closest pair* of clusters, until there is only one cluster.

The history of merging forms a binary tree or hierarchy.

Many variants to defining closest pair of clusters

Single-link

Similarity of the ***most*** **cosine-similar (single-link)**

Complete-link

Similarity of the "furthest" points, the *least* cosine-similar

Centroid
Clusters whose centroids (centers of gravity) are the most cosine-similar

Average-link
Average cosine between pairs of elements

Single Link Agglomerative Clustering

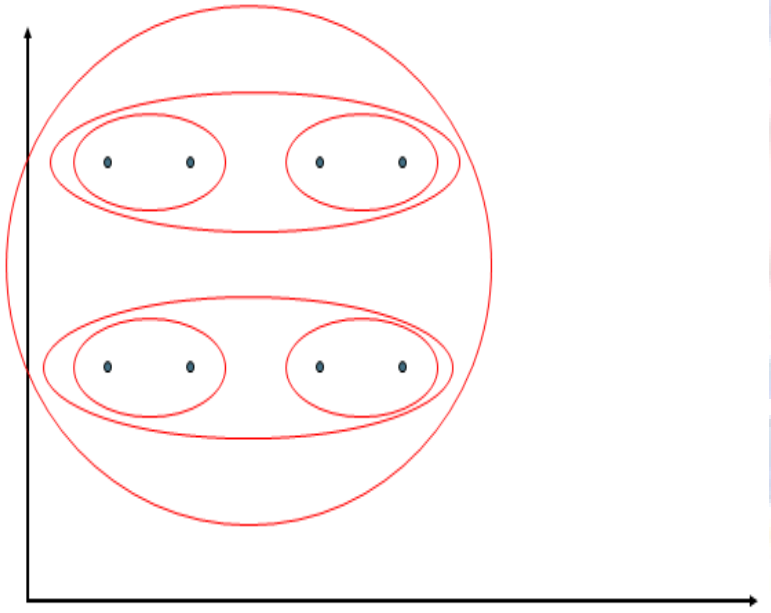
- Use maximum similarity of pairs:

$$sim(c_i, c_j) = \max_{x \in c_i, y \in c_j} sim(x, y)$$

- Can result in "straggly" (long and thin) clusters due to chaining effect.
- After merging c_i and c_j, the similarity of the resulting cluster to another cluster, c_k, is:

$$sim((c_i \cup c_j), c_k) = \max(sim(c_i, c_k), sim(c_j, c_k))$$

Single Link:



Complete Link:

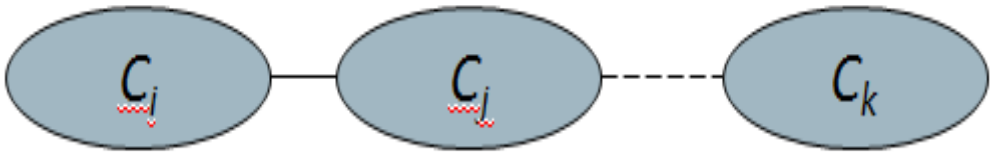
- Use minimum similarity of pairs:

$$sim(c_i, c_j) = \min_{x \in c_i, y \in c_j} sim(x, y)$$

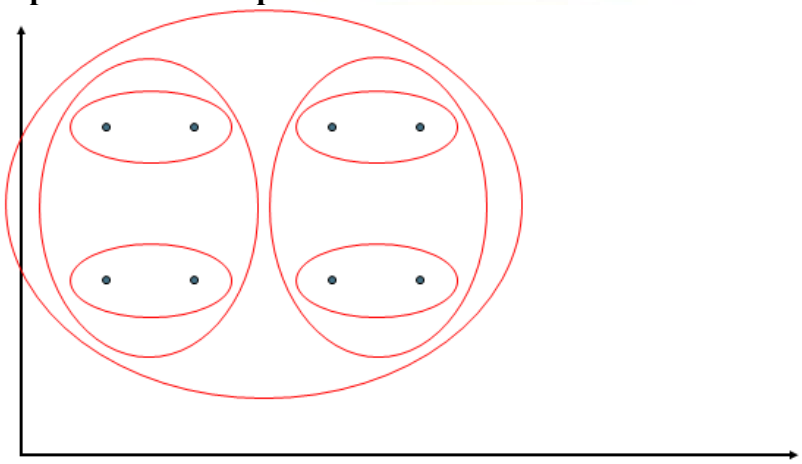
- Makes "tighter," spherical clusters that are typically preferable.

After merging c_i and c_j, the similarity of the resulting cluster to another cluster, c_k, **is:**

$$sim((c_i \cup c_j), c_k) = \min(sim(c_i, c_k), sim(c_j, c_k))$$



Complete Link Example



Internal criterion: A good clustering will produce high quality clusters in which:

The intra-class (that is, intra-cluster) similarity is high
The inter-class similarity is low
The measured quality of a clustering depends on both the document representation and the similarity measure used

5.7 EXPECTATION MAXIMIZATION (EM)

PART-B
Brief about Expectation maximization algorithm(Nov/Dec'17)
Give an account of the Expaectation maximization problem. (Nov/Dec'16)

- Iterative method for learning probabilistic categorization model from unsupervised data.
- Initially assume random assignment of examples to categories.

- Learn an initial probabilistic model by estimating model parameters θ from this randomly labeled data.

- Iterate following two steps until convergence:
  Expectation (E-step): Compute P(c_i | E) for each example given the current model, and probabilistically re-label the examples based on these posterior probability estimates.
  Maximization (M-step): Re-estimate the model parameters, θ, from the probabilistically re-labeled data.

EM Experiment :

Semi-supervised: some labeled and unlabeled data

Take a completely labeled corpus D, and randomly select a subset as D_K.
Also use the set of unlabeled documents in the EM procedure.
Correct classification of a document

Concealed class label = class with largest probability

Accuracy with unlabeled documents > accuracy without unlabeled documents
Keeping labeled set of same size

EM beats naïve Bayes with same size of labeled document set

Largest boost for small size of labeled set

Comparable or poorer performance of EM for large labeled sets.

****