Information Retrieval Handbook

Renay Maher

First Edition, 2012

ISBN 978-81-323-3474-3

Published by:
Research World
4735/22 Prakashdeep Bldg,
Ansari Road, Darya Ganj,
Delhi - 110002
Email: info@wtbooks.com

Table of Contents

Chapter 1 - Information Retrieval

Chapter 2 - Discounted Cumulative Gain

Chapter 3 - Web Search Engine

Chapter 4 - Bioinformatic Harvester, Adversarial Information Retrieval and Anchor Text

Chapter 5 - ChemRefer, Compound Term Processing, Document Clustering and Document Retrieval

Chapter 6 - EXCLAIM, Enterprise Search and Expertise Finding

Chapter 7 - Exploratory Search, Faceted Search and Federated Search

Chapter 8 - Human–computer Information Retrieval and Full Text Search

Chapter 9 - Index (search engine)

Chapter 10 - Music Information Retrieval, IR Evaluation, Gain (information retrieval) and Harvester42

Chapter 11 - Nearest Neighbor Search and Multi-Document Summarization

Chapter 12 - Precision and Recall

Chapter 13 - Question Answering

Chapter 14 - Query Expansion, Relevance (information retrieval) and Relevance Feedback

Chapter 15 - Search Engine (computing) and Search Engine Technology

Chapter-1

Information Retrieval

Information retrieval (**IR**) is the area of study concerned with searching for documents, for information within documents, and for metadata about documents, as well as that of searching relational databases and the World Wide Web. There is overlap in the usage of the terms data retrieval, document retrieval, information retrieval, and text retrieval, but each also has its own body of literature, theory, praxis, and technologies. IR is interdisciplinary, based on computer science, mathematics, library science, information science, information architecture, cognitive psychology, linguistics, and statistics.

Automated information retrieval systems are used to reduce what has been called "information overload". Many universities and public libraries use IR systems to provide access to books, journals and other documents. Web search engines are the most visible IR applications.

History

“ But do you know that, although I have kept the diary [on a phonograph] for months past, it never once struck me how I was going to find any particular part of it in case I wanted to look it up? ”

—Dr Seward, Bram Stoker's *Dracula*, 1897

The idea of using computers to search for relevant pieces of information was popularized in the article *As We May Think* by Vannevar Bush in 1945. The first automated information retrieval systems were introduced in the 1950s and 1960s. By 1970 several different techniques had been shown to perform well on small text corpora such as the Cranfield collection (several thousand documents). Large-scale retrieval systems, such as the Lockheed Dialog system, came into use early in the 1970s.

In 1992, the US Department of Defense along with the National Institute of Standards and Technology (NIST), cosponsored the Text Retrieval Conference (TREC) as part of the TIPSTER text program. The aim of this was to look into the information retrieval community by supplying the infrastructure that was needed for evaluation of text retrieval methodologies on a very large text collection. This catalyzed research on methods that scale to huge corpora. The introduction of web search engines has boosted the need for very large scale retrieval systems even further.

The use of digital methods for storing and retrieving information has led to the phenomenon of digital obsolescence, where a digital resource ceases to be readable because the physical media, the reader required to read the media, the hardware, or the software that runs on it, is no longer available. The information is initially easier to retrieve than if it were on paper, but is then effectively lost.

Timeline

- Before the **1900s**

  1880s: Herman Hollerith invents the recording of data on a machine readable medium.
  1890 Hollerith cards, key punches and tabulators used to process the 1890 US Census data.

- **1940s–1950s**

  late 1940s: The US military confronted problems of indexing and retrieval of wartime scientific research documents captured from Germans.
  1945: Vannevar Bush's *As We May Think* appeared in *Atlantic Monthly*.
  1947: Hans Peter Luhn (research engineer at IBM since 1941) began work on a mechanized punch card-based system for searching chemical compounds.
  1950s: Growing concern in the US for a "science gap" with the USSR motivated, encouraged funding and provided a backdrop for mechanized literature searching systems (Allen Kent *et al.*) and the invention of citation indexing (Eugene Garfield).
  1950: The term "information retrieval" appears to have been coined by Calvin Mooers.
  1951: Philip Bagley conducted the earliest experiment in computerized document retrieval in a master thesis at MIT.

1955: Allen Kent joined Case Western Reserve University, and eventually became associate director of the Center for Documentation and Communications Research. That same year, Kent and colleagues published a paper in American Documentation describing the precision and recall measures as well as detailing a proposed "framework" for evaluating an IR system which included statistical sampling methods for determining the number of relevant documents not retrieved.
1958: International Conference on Scientific Information Washington DC included consideration of IR systems as a solution to problems identified. See: *Proceedings of the International Conference on Scientific Information, 1958* (National Academy of Sciences, Washington, DC, 1959)
1959: Hans Peter Luhn published "Auto-encoding of documents for information retrieval."

- **1960s**:

  early 1960s: Gerard Salton began work on IR at Harvard, later moved to Cornell.
  1960: Melvin Earl (Bill) Maron and John Lary Kuhns published "On relevance, probabilistic indexing, and information retrieval" in the Journal of the ACM 7(3):216–244, July 1960.
  1962:

  - Cyril W. Cleverdon published early findings of the Cranfield studies, developing a model for IR system evaluation. See: Cyril W. Cleverdon, "Report on the Testing and Analysis of an Investigation into the Comparative Efficiency of Indexing Systems". Cranfield Collection of Aeronautics, Cranfield, England, 1962.
  - Kent published *Information Analysis and Retrieval*.

  1963:

  - Weinberg report "Science, Government and Information" gave a full articulation of the idea of a "crisis of scientific information." The report was named after Dr. Alvin Weinberg.
  - Joseph Becker and Robert M. Hayes published text on information retrieval. Becker, Joseph; Hayes, Robert Mayo. *Information storage and retrieval: tools, elements, theories*. New York, Wiley (1963).

  1964:

  - Karen Spärck Jones finished her thesis at Cambridge, *Synonymy and Semantic Classification*, and continued work on computational linguistics as it applies to IR.
  - The National Bureau of Standards sponsored a symposium titled "Statistical Association Methods for Mechanized Documentation." Several

highly significant papers, including G. Salton's first published reference (we believe) to the SMART system.

mid-1960s:

- National Library of Medicine developed MEDLARS Medical Literature Analysis and Retrieval System, the first major machine-readable database and batch-retrieval system.
- Project Intrex at MIT.

1965: J. C. R. Licklider published *Libraries of the Future*.
1966: Don Swanson was involved in studies at University of Chicago on Requirements for Future Catalogs.
late 1960s: F. Wilfrid Lancaster completed evaluation studies of the MEDLARS system and published the first edition of his text on information retrieval.
1968:

- Gerard Salton published *Automatic Information Organization and Retrieval*.
- John W. Sammon, Jr.'s RADC Tech report "Some Mathematics of Information Storage and Retrieval..." outlined the vector model.

1969: Sammon's "A nonlinear mapping for data structure analysis" (IEEE Transactions on Computers) was the first proposal for visualization interface to an IR system.

- **1970s**

early 1970s:

- First online systems—NLM's AIM-TWX, MEDLINE; Lockheed's Dialog; SDC's ORBIT.
- Theodor Nelson promoting concept of hypertext, published *Computer Lib/Dream Machines*.

1971: Nicholas Jardine and Cornelis J. van Rijsbergen published "The use of hierarchic clustering in information retrieval", which articulated the "cluster hypothesis." (Information Storage and Retrieval, 7(5), pp. 217–240, December 1971)
1975: Three highly influential publications by Salton fully articulated his vector processing framework and term discrimination model:

- *A Theory of Indexing* (Society for Industrial and Applied Mathematics)
- *A Theory of Term Importance in Automatic Text Analysis* (JASIS v. 26)
- *A Vector Space Model for Automatic Indexing* (CACM 18:11)

1978: The First ACM SIGIR conference.
1979: C. J. van Rijsbergen published *Information Retrieval* (Butterworths). Heavy emphasis on probabilistic models.

- **1980s**

1980: First international ACM SIGIR conference, joint with British Computer Society IR group in Cambridge.
1982: Nicholas J. Belkin, Robert N. Oddy, and Helen M. Brooks proposed the ASK (Anomalous State of Knowledge) viewpoint for information retrieval. This was an important concept, though their automated analysis tool proved ultimately disappointing.
1983: Salton (and Michael J. McGill) published *Introduction to Modern Information Retrieval* (McGraw-Hill), with heavy emphasis on vector models.
1985: Blair and Maron publish: An Evaluation of Retrieval Effectiveness for a Full-Text Document-Retrieval System
mid-1980s: Efforts to develop end-user versions of commercial IR systems.
1985–1993: Key papers on and experimental systems for visualization interfaces. Work by Donald B. Crouch, Robert R. Korfhage, Matthew Chalmers, Anselm Spoerri and others.
1989: First World Wide Web proposals by Tim Berners-Lee at CERN.

- **1990s**

1992: First TREC conference.
1997: Publication of Korfhage's *Information Storage and Retrieval* with emphasis on visualization and multi-reference point systems.
late 1990s: Web search engines implementation of many features formerly found only in experimental IR systems. Search engines become the most common and maybe best instantiation of IR models, research, and implementation.

Overview

An information retrieval process begins when a user enters a query into the system. Queries are formal statements of information needs, for example search strings in web search engines. In information retrieval a query does not uniquely identify a single object in the collection. Instead, several objects may match the query, perhaps with different degrees of relevancy.

An object is an entity that is represented by information in a database. User queries are matched against the database information. Depending on the application the data objects may be, for example, text documents, images, audio, mind maps or videos. Often the documents themselves are not kept or stored directly in the IR system, but are instead represented in the system by document surrogates or metadata.

Most IR systems compute a numeric score on how well each object in the database match the query, and rank the objects according to this value. The top ranking objects are then shown to the user. The process may then be iterated if the user wishes to refine the query.

Performance measures

Many different measures for evaluating the performance of information retrieval systems have been proposed. The measures require a collection of documents and a query. All common measures described here assume a ground truth notion of relevancy: every document is known to be either relevant or non-relevant to a particular query. In practice queries may be ill-posed and there may be different shades of relevancy.

Precision

Precision is the fraction of the documents retrieved that are relevant to the user's information need.

$$\text{precision} = \frac{|\{\text{relevant documents}\} \cap \{\text{retrieved documents}\}|}{|\{\text{retrieved documents}\}|}$$

In binary classification, precision is analogous to positive predictive value. Precision takes all retrieved documents into account. It can also be evaluated at a given cut-off rank, considering only the topmost results returned by the system. This measure is called *precision at n* or *P@n*.

Note that the meaning and usage of "precision" in the field of Information Retrieval differs from the definition of accuracy and precision within other branches of science and technology.

Recall

Recall is the fraction of the documents that are relevant to the query that are successfully retrieved.

$$\text{recall} = \frac{|\{\text{relevant documents}\} \cap \{\text{retrieved documents}\}|}{|\{\text{relevant documents}\}|}$$

In binary classification, recall is called sensitivity. So it can be looked at as *the probability that a relevant document is retrieved by the query*.

It is trivial to achieve recall of 100% by returning all documents in response to any query. Therefore recall alone is not enough but one needs to measure the number of non-relevant documents also, for example by computing the precision.

Fall-Out

The proportion of non-relevant documents that are retrieved, out of all non-relevant documents available:

$$\text{fall-out} = \frac{|\{\text{non-relevant documents}\} \cap \{\text{retrieved documents}\}|}{|\{\text{non-relevant documents}\}|}$$

In binary classification, fall-out is closely related to specificity (1 − specificity). It can be looked at as *the probability that a non-relevant document is retrieved by the query*.

It is trivial to achieve fall-out of 0% by returning zero documents in response to any query.

F-measure

The weighted harmonic mean of precision and recall, the traditional F-measure or balanced F-score is:

$$F = \frac{2 \cdot \text{precision} \cdot \text{recall}}{(\text{precision} + \text{recall})}.$$

This is also known as the F_1 measure, because recall and precision are evenly weighted.

The general formula for non-negative real β is:

$$F_\beta = \frac{(1+\beta^2) \cdot (\text{precision} \cdot \text{recall})}{(\beta^2 \cdot \text{precision} + \text{recall})}.$$

Two other commonly used F measures are the F_2 measure, which weights recall twice as much as precision, and the $F_{0.5}$ measure, which weights precision twice as much as recall.

The F-measure was derived by van Rijsbergen (1979) so that F_β "measures the effectiveness of retrieval with respect to a user who attaches β times as much importance to recall as precision". It is based on van Rijsbergen's effectiveness measure $E = 1 - (1 / (\alpha / P + (1 - \alpha) / R))$. Their relationship is $F_\beta = 1 - E$ where $\alpha = 1 / (\beta^2 + 1)$.

Average precision

Precision and recall are single-value metrics based on the whole list of documents returned by the system. For systems that return a ranked sequence of documents, it is desirable to also consider the order in which the returned documents are presented. Average precision emphasizes ranking relevant documents higher. It is the average of precisions computed at the point of each of the relevant documents in the ranked sequence:

$$\mathrm{AveP} = \frac{\sum_{r=1}^{N} (P(r) \times \mathrm{rel}(r))}{\text{number of relevant documents}}$$

where *r* is the rank, *N* the number retrieved, *rel()* a binary function on the relevance of a given rank, and *P(r)* precision at a given cut-off rank:

$$\mathrm{P(r)} = \frac{|\{\text{relevant retrieved documents of rank r or less}\}|}{r}$$

This metric is also sometimes referred to geometrically as the area under the Precision-Recall curve.

Note that the denominator (number of relevant documents) is the number of relevant documents in the entire collection, so that the metric reflects performance over all relevant documents, regardless of a retrieval cutoff. See:.

R-Precision

Precision at rank R. Where R is the total number of relevant documents. This measure is highly correlated to Average Precision.

Mean average precision

Mean average precision for a set of queries is the mean of the average precision scores for each query.

$$\mathrm{MAP} = \frac{\sum_{q=1}^{Q} \mathrm{AveP(q)}}{Q}$$

where *Q* is the number of queries.

Discounted cumulative gain

DCG uses a graded relevance scale of documents from the result set to evaluate the usefulness, or gain, of a document based on its position in the result list. The premise of DCG is that highly relevant documents appearing lower in a search result list should be penalized as the graded relevance value is reduced logarithmically proportional to the position of the result.

The DCG accumulated at a particular rank position *p* is defined as:

$$\mathrm{DCG_p} = rel_1 + \sum_{i=2}^{p} \frac{rel_i}{\log_2 i}$$

Since result set may vary in size among different queries or systems, to compare performances the normalised version of DCG uses an ideal DCG. To this end, it sorts documents of a result list by relevance, producing an ideal DCG at position p (*IDCG$_p$*), which normalizes the score:

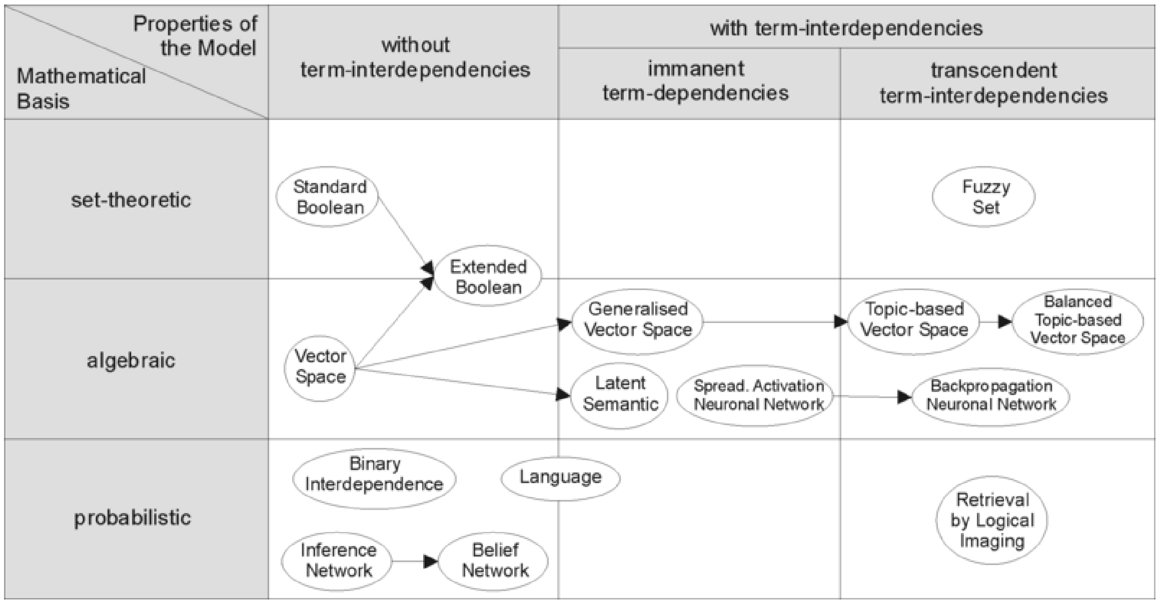
$$\mathrm{nDCG_p} = \frac{DCG_p}{IDCGp}$$

The nDCG values for all queries can be averaged to obtain a measure of the average performance of a ranking algorithm. Note that in a perfect ranking algorithm, the *DCG$_p$* will be the same as the *IDCG$_p$* producing an nDCG of 1.0. All nDCG calculations are then relative values on the interval 0.0 to 1.0 and so are cross-query comparable.

Other Measures

- Mean reciprocal rank
- Spearman's rank correlation coefficient

Model types



Categorization of IR-models (translated from German entry, original source Dominik Kuropka).

For the information retrieval to be efficient, the documents are typically transformed into a suitable representation. There are several representations. The picture on the right illustrates the relationship of some common models. In the picture, the models are categorized according to two dimensions: the mathematical basis and the properties of the model.

First dimension: mathematical basis

- *Set-theoretic models represent documents as sets of words or phrases. Similarities are usually derived from set-theoretic operations on those sets. Common models are:*
    - Standard Boolean model
    - Extended Boolean model
    - Fuzzy retrieval
- *Algebraic models* represent documents and queries usually as vectors, matrices, or tuples. The similarity of the query vector and document vector is represented as a scalar value.
    - Vector space model
    - Generalized vector space model
    - (Enhanced) Topic-based Vector Space Model
    - Extended Boolean model
    - Latent semantic indexing aka latent semantic analysis
- *Probabilistic models* treat the process of document retrieval as a probabilistic inference. Similarities are computed as probabilities that a document is relevant for a given query. Probabilistic theorems like the Bayes' theorem are often used in these models.
    - Binary Independence Model
    - Probabilistic relevance model on which is based the okapi (BM25) relevance function
    - Uncertain inference
    - Language models
    - Divergence-from-randomness model
    - Latent Dirichlet allocation
- *Feature-based retrieval models* view documents as vectors of values of *feature functions* (or just *features*) and seek the best way to combine these features into a single relevance score, typically by learning to rank methods. Feature functions are arbitrary functions of document and query, and as such can easily incorporate almost any other retrieval model as just a yet another feature.

Second dimension: properties of the model

- *Models without term-interdependencies* treat different terms/words as independent. This fact is usually represented in vector space models by the orthogonality assumption of term vectors or in probabilistic models by an independency assumption for term variables.
- *Models with immanent term interdependencies* allow a representation of interdependencies between terms. However the degree of the interdependency between two terms is defined by the model itself. It is usually directly or

indirectly derived (e.g. by dimensional reduction) from the co-occurrence of those terms in the whole set of documents.

- *Models with transcendent term interdependencies* allow a representation of interdependencies between terms, but they do not allege how the interdependency between two terms is defined. They relay an external source for the degree of interdependency between two terms. (For example a human or sophisticated algorithms.)

Major figures

- Thomas Bayes
- Claude E. Shannon
- Gerard Salton
- Hans Peter Luhn
- Cyril Cleverdon
- W. Bruce Croft
- Karen Spärck Jones
- Calvin Mooers
- C. J. van Rijsbergen
- Stephen E. Robertson

Chapter-2

Discounted Cumulative Gain

Discounted cumulative gain (**DCG**) is a measure of effectiveness of a Web search engine algorithm or related applications, often used in information retrieval. Using a graded relevance scale of documents in a search engine result set, DCG measures the usefulness, or *gain*, of a document based on its position in the result list. The gain is accumulated from the top of the result list to the bottom with the gain of each result discounted at lower ranks.

Mathematical Details

Two assumptions are made in using DCG and its related measures.

1. Highly relevant documents are more useful when appearing earlier in a search engine result list (have higher ranks)
2. Highly relevant documents are more useful than marginally relevant documents, which are in turn more useful than irrelevant documents.

DCG originates from an earlier, more primitive, measure called Cumulative Gain.

Cumulative Gain

Cumulative Gain (CG) is the predecessor of DCG and does not include the position of a result in the consideration of the usefulness of a result set. In this way, it is the sum of the graded relevance values of all results in a search result list. The CG at a particular rank position *p* is defined as:

$$\mathrm{CG_p} = \sum_{i=1}^{p} rel_i$$

Where rel_i is the graded relevance of the result at position *i*.

The value computed with the CG function is unaffected by changes in the ordering of search results. That is, moving a highly relevant document d_i above a higher ranked, less relevant, document d_j does not change the computed value for CG. Based on the two assumptions made above about the usefulness of search results, DCG is used in place of CG for a more accurate measure.

Discounted Cumulative Gain

The premise of DCG is that highly relevant documents appearing lower in a search result list should be penalized as the graded relevance value is reduced logarithmically proportional to the position of the result. The discounted CG accumulated at a particular rank position *p* is defined as:

$$\mathrm{DCG_p} = rel_1 + \sum_{i=2}^{p} \frac{rel_i}{\log_2 i}$$

There has not been shown any theoretically sound justification for using a logarithmic reduction factor other than the fact that it produces a smooth reduction. An alternative formulation of DCG places stronger emphasis on retrieving relevant documents:

$$\mathrm{DCG_p} = \sum_{i=1}^{p} \frac{2^{rel_i} - 1}{\log_2(1+i)}$$

The function is equivalent to the previous DCG function when the relevance values of documents are binary; $rel_i \in \{0,1\}$.

Normalized DCG

Search result lists vary in length depending on the query. Comparing a search engine's performance from one query to the next cannot be consistently achieved using DCG alone, so the cumulative gain at each position for a chosen value of *p* should be normalized across queries. This is done by sorting documents of a result list by relevance, producing an ideal DCG at position *p*. For a query, the *normalized discounted cumulative gain*, or nDCG, is computed as:

$$\mathrm{nDCG_p} = \frac{DCG_p}{IDCGp}$$

The nDCG values for all queries can be averaged to obtain a measure of the average performance of a search engine's ranking algorithm. Note that in a perfect ranking algorithm, the DCG_p will be the same as the $IDCG_p$ producing an nDCG of 1.0. All nDCG calculations are then relative values on the interval 0.0 to 1.0 and so are cross-query comparable.

The main difficulty encountered in using nDCG is the unavailability of an ideal ordering of results when only partial relevance feedback is available.

Example

Presented with a list of documents in response to a search query, an experiment participant is asked to judge the relevance of each document to the query. Each document is to be judged on a scale of 0-3 with 0 meaning irrelevant, 3 meaning completely relevant, and 1 and 2 meaning "somewhere in between". For the documents ordered by the ranking algorithm as

$D_1, D_2, D_3, D_4, D_5, D_6$

the user provides the following relevance scores:

3,2,3,0,1,2

That is: document 1 has a relevance of 3, document 2 has a relevance of 2, etc. The Cumulative Gain of this search result listing is:

$$\mathrm{CG_p} = \sum_{i=1}^{p} rel_i = 3+2+3+0+1+2 = 11$$

Changing the order of any two documents does not affect the CG measure. If D_3 and D_4 are switched, the CG remains the same, 11. DCG is used to emphasize highly relevant documents appearing early in the result list. Using the logarithmic scale for reduction, the DCG for each result in order is:

i	***rel_i***	**log*i***	$\frac{rel_i}{\log_2 i}$
1	3	N/A	N/A
2	2	1	2
3	3	1.59	1.887
4	0	2.0	0
5	1	2.32	0.431
6	2	2.59	0.772

So the DCG_6 of this ranking is:

$$\mathrm{DCG_6} = rel_1 + \sum_{i=2}^{p} \frac{rel_i}{\log_2 i} = 3 + (2 + 1.887 + 0 + 0.431 + 0.772) = 8.09$$

Now a switch of D_3 and D_4 results in a reduced DCG because a less relevant document is placed higher in the ranking; that is, a more relevant document is discounted more by being placed in a lower rank.

The performance of this query to another is incomparable in this form since the other query may have more results, resulting in a larger overall DCG which may not necessarily be better. In order to compare, the DCG values must be normalized.

To normalize DCG values, an ideal ordering for the given query is needed. For this example, that ordering would be the monotonically decreasing sort of the relevance judgments provided by the experiment participant, which is:

3,3,2,2,1,0

The DCG of this ideal ordering, or *IDCG*, is then:

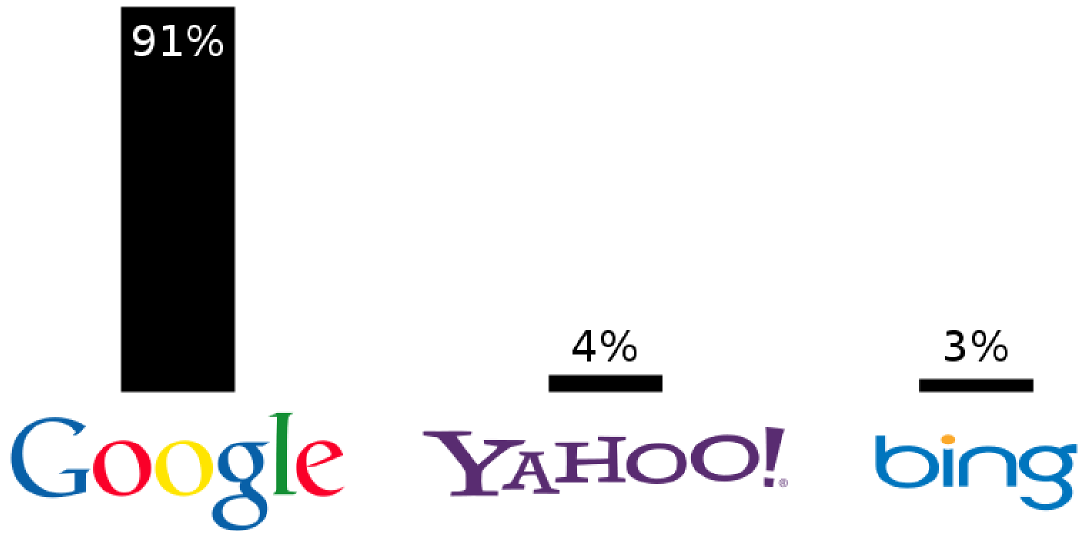
$\mathrm{IDCG}_6 = 8.693$

And so the nDCG for this query is given as:

$$\mathrm{nDCG}_6 = \frac{DCG_6}{IDCG6} = \frac{8.09}{8.693} = 0.9306$$

Chapter-3

Web Search Engine



The three most widely used web search engines and their approximate share as of late 2010

A **web search engine** is designed to search for information on the World Wide Web and FTP servers. The search results are generally presented in a list of results and are often called *hits*. The information may consist of web pages, images, information and other types of files. Some search engines also mine data available in databases or open directories. Unlike web directories, which are maintained by human editors, search engines operate algorithmically or are a mixture of algorithmic and human input.

History

Timeline (full list)

Year	**Engine**	**Current status**
1993	W3Catalog	Closed
	Aliweb	Active

	JumpStation	Active
	WebCrawler	Active
1994	Go.com	Active
	Lycos	Active
	AltaVista	Active
	Daum	Active
	Open Text Web Index	Active
1995	Magellan	Active
	Excite	Active
	SAPO	Active
	Yahoo!	Active, Launched as a directory
	Dogpile	Active
1996	Inktomi	Active
	HotBot	Active
	Ask Jeeves	Active
1997	Northern Light	Active
	Yandex	Active
1998	Google	Active
	MSN Search	Active as Bing
	AlltheWeb	Active
	GenieKnows	Active
1999	Naver	Active
	Teoma	Active
	Vivisimo	Active
2000	Baidu	Active
	Exalead	Active
2002	Inktomi	Acquired by Yahoo!
2003	Info.com	Active
2004	Yahoo! Search	Active, Launched own web search
	A9.com	Closed
	Sogou	Active
2005	Ask.com	Active
	GoodSearch	Active
	SearchMe	Active
	Quaero	Active
	Ask.com	Active
	Live Search	Active as Bing, Launched as rebranded MSN Search
	ChaCha	Active

	Guruji.com	Active
2007	Sproose	Closed
	Blackle.com	Active
2008	Powerset	Acquired by Microsoft
	Picollator	Closed
	Viewzi	Closed
	Boogami	Active
	LeapFish	Active
	Forestle	Active
	VADLO	Active
	Duck Duck Go	Active
2009	Bing	Active, Launched as rebranded Live Search
	Yebol	Active
	Mugurdy	Closed due to a lack of funding
	Goby	Active
2010	Yandex	Active, Launched global (English) search
	Cuil	Closed
	Blekko	Active
	Viewzi	Closed due to a lack of funding
	Yummly	Active
2011	Exalead	Acquired by Dassault Systèmes
	Yebol	Domain name expired

During the early development of the web, there was a list of webservers edited by Tim Berners-Lee and hosted on the CERN webserver. One historical snapshot from 1992 remains. As more webservers went online the central list could not keep up. On the NCSA site new servers were announced under the title "What's New!"

The very first tool used for searching on the Internet was Archie. The name stands for "archive" without the "v". It was created in 1990 by Alan Emtage, Bill Heelan and J. Peter Deutsch, computer science students at McGill University in Montreal. The program downloaded the directory listings of all the files located on public anonymous FTP (File Transfer Protocol) sites, creating a searchable database of file names; however, Archie did not index the contents of these sites since the amount of data was so limited it could be readily searched manually.

The rise of Gopher (created in 1991 by Mark McCahill at the University of Minnesota) led to two new search programs, Veronica and Jughead. Like Archie, they searched the file names and titles stored in Gopher index systems. Veronica (*V*ery *E*asy *R*odent-*O*riented *N*et-wide *I*ndex to *C*omputerized *A*rchives) provided a keyword search of most

Gopher menu titles in the entire Gopher listings. Jughead (*J*onzy's *U*niversal *G*opher *H*ierarchy *E*xcavation *A*nd *D*isplay) was a tool for obtaining menu information from specific Gopher servers. While the name of the search engine "Archie" was not a reference to the Archie comic book series, "Veronica" and "Jughead" are characters in the series, thus referencing their predecessor.

In the summer of 1993, no search engine existed yet for the web, though numerous specialized catalogues were maintained by hand. Oscar Nierstrasz at the University of Geneva wrote a series of Perl scripts that would periodically mirror these pages and rewrite them into a standard format which formed the basis for W3Catalog, the web's first primitive search engine, released on September 2, 1993.

In June 1993, Matthew Gray, then at MIT, produced what was probably the first web robot, the Perl-based World Wide Web Wanderer, and used it to generate an index called 'Wandex'. The purpose of the Wanderer was to measure the size of the World Wide Web, which it did until late 1995. The web's second search engine Aliweb appeared in November 1993. Aliweb did not use a web robot, but instead depended on being notified by website administrators of the existence at each site of an index file in a particular format.

JumpStation (released in December 1993) used a web robot to find web pages and to build its index, and used a web form as the interface to its query program. It was thus the first WWW resource-discovery tool to combine the three essential features of a web search engine (crawling, indexing, and searching) as described below. Because of the limited resources available on the platform on which it ran, its indexing and hence searching were limited to the titles and headings found in the web pages the crawler encountered.

One of the first "full text" crawler-based search engines was WebCrawler, which came out in 1994. Unlike its predecessors, it let users search for any word in any webpage, which has become the standard for all major search engines since. It was also the first one to be widely known by the public. Also in 1994, Lycos (which started at Carnegie Mellon University) was launched and became a major commercial endeavor.

Soon after, many search engines appeared and vied for popularity. These included Magellan (search engine), Excite, Infoseek, Inktomi, Northern Light, and AltaVista. Yahoo! was among the most popular ways for people to find web pages of interest, but its search function operated on its web directory, rather than full-text copies of web pages. Information seekers could also browse the directory instead of doing a keyword-based search.

In 1996, Netscape was looking to give a single search engine an exclusive deal to be the featured search engine on Netscape's web browser. There was so much interest that instead a deal was struck with Netscape by five of the major search engines, where for $5Million per year each search engine would be in a rotation on the Netscape search engine page. The five engines were Yahoo!, Magellan, Lycos, Infoseek, and Excite.

Search engines were also known as some of the brightest stars in the Internet investing frenzy that occurred in the late 1990s. Several companies entered the market spectacularly, receiving record gains during their initial public offerings. Some have taken down their public search engine, and are marketing enterprise-only editions, such as Northern Light. Many search engine companies were caught up in the dot-com bubble, a speculation-driven market boom that peaked in 1999 and ended in 2001.

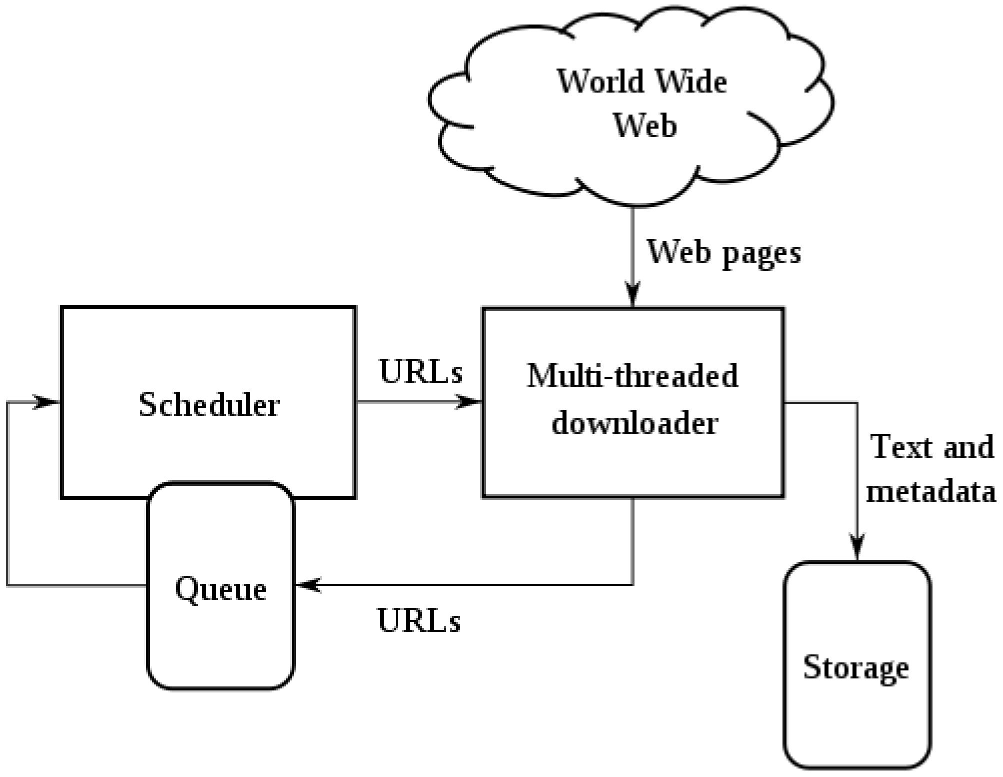
Around 2000, Google's search engine rose to prominence. The company achieved better results for many searches with an innovation called PageRank. This iterative algorithm ranks web pages based on the number and PageRank of other web sites and pages that link there, on the premise that good or desirable pages are linked to more than others. Google also maintained a minimalist interface to its search engine. In contrast, many of its competitors embedded a search engine in a web portal.

By 2000, Yahoo was providing search services based on Inktomi's search engine. Yahoo! acquired Inktomi in 2002, and Overture (which owned AlltheWeb and AltaVista) in 2003. Yahoo! switched to Google's search engine until 2004, when it launched its own search engine based on the combined technologies of its acquisitions.

Microsoft first launched MSN Search in the fall of 1998 using search results from Inktomi. In early 1999 the site began to display listings from Looksmart blended with results from Inktomi except for a short time in 1999 when results from AltaVista were used instead. In 2004, Microsoft began a transition to its own search technology, powered by its own web crawler (called msnbot).

Microsoft's rebranded search engine, Bing, was launched on June 1, 2009. On July 29, 2009, Yahoo! and Microsoft finalized a deal in which Yahoo! Search would be powered by Microsoft Bing technology.

How web search engines work



High-level architecture of a standard Web crawler

A search engine operates, in the following order

1. Web crawling
2. Indexing
3. Searching

Web search engines work by storing information about many web pages, which they retrieve from the html itself. These pages are retrieved by a Web crawler (sometimes also known as a spider) — an automated Web browser which follows every link on the site. Exclusions can be made by the use of robots.txt. The contents of each page are then analyzed to determine how it should be indexed (for example, words are extracted from the titles, headings, or special fields called meta tags). Data about web pages are stored in an index database for use in later queries. A query can be a single word. The purpose of an index is to allow information to be found as quickly as possible. Some search engines, such as Google, store all or part of the source page (referred to as a cache) as well as information about the web pages, whereas others, such as AltaVista, store every word of every page they find. This cached page always holds the actual search text since it is the one that was actually indexed, so it can be very useful when the content of the current page has been updated and the search terms are no longer in it. This problem might be considered to be a mild form of linkrot, and Google's handling of it increases usability by satisfying user expectations that the search terms will be on the returned webpage. This satisfies the principle of least astonishment since the user normally expects the search terms to be on the returned pages. Increased search relevance makes these cached pages very useful, even beyond the fact that they may contain data that may no longer be available elsewhere.

When a user enters a query into a search engine (typically by using key words), the engine examines its index and provides a listing of best-matching web pages according to its criteria, usually with a short summary containing the document's title and sometimes parts of the text. The index is built from the information stored with the data and the method by which the information is indexed. Unfortunately, there are currently no known public search engines that allow documents to be searched by date. Most search engines support the use of the boolean operators AND, OR and NOT to further specify the search query. Boolean operators are for literal searches that allow the user to refine and extend the terms of the search. The engine looks for the words or phrases exactly as entered. Some search engines provide an advanced feature called proximity search which allows users to define the distance between keywords. There is also concept-based searching where the research involves using statistical analysis on pages containing the words or phrases you search for. As well, natural language queries allow the user to type a question in the same form one would ask it to a human. A site like this would be ask.com.

The usefulness of a search engine depends on the relevance of the **result set** it gives back. While there may be millions of web pages that include a particular word or phrase, some pages may be more relevant, popular, or authoritative than others. Most search engines employ methods to rank the results to provide the "best" results first. How a search engine decides which pages are the best matches, and what order the results should be shown in, varies widely from one engine to another. The methods also change over time as Internet usage changes and new techniques evolve. There are two main types of search engine that have evolved: one is a system of predefined and hierarchically ordered keywords that humans have programmed extensively. The other is a system that generates an "inverted index" by analyzing texts it locates. This second form relies much more heavily on the computer itself to do the bulk of the work.

Most Web search engines are commercial ventures supported by advertising revenue and, as a result, some employ the practice of allowing advertisers to pay money to have their listings ranked higher in search results. Those search engines which do not accept money for their search engine results make money by running search related ads alongside the regular search engine results. The search engines make money every time someone clicks on one of these ads.

Market share and wars

According to Net Marketshare. In December 2010, rankings the market share of web search engine, showed Google is 84.65%, Yahoo is 6.69%, Baidu is 3.39%, Bing is 3.29% and other is 1.98%. The Google's worldwide market share peaked at 86.3% in April, 2010.

In the United States, Google held a 63.2% market share in May 2009, according to Nielsen NetRatings. In the People's Republic of China, Baidu held a 61.6% market share for web search in July 2009.

Search engine bias

Although search engines are programmed to rank websites based on their popularity and relevancy, empirical studies indicate various political, economic, and social biases in the information they provide. These biases could be a direct result of economic and commercial processes (e.g., companies that advertise with a search engine can become also more popular in its organic search results), and political processes (e.g., the removal of search results in order to comply with local laws). Google Bombing is one example of an attempt to manipulate search results for political, social or commercial reasons.

Chapter-4

Bioinformatic Harvester, Adversarial Information Retrieval and Anchor Text

Bioinformatic Harvester

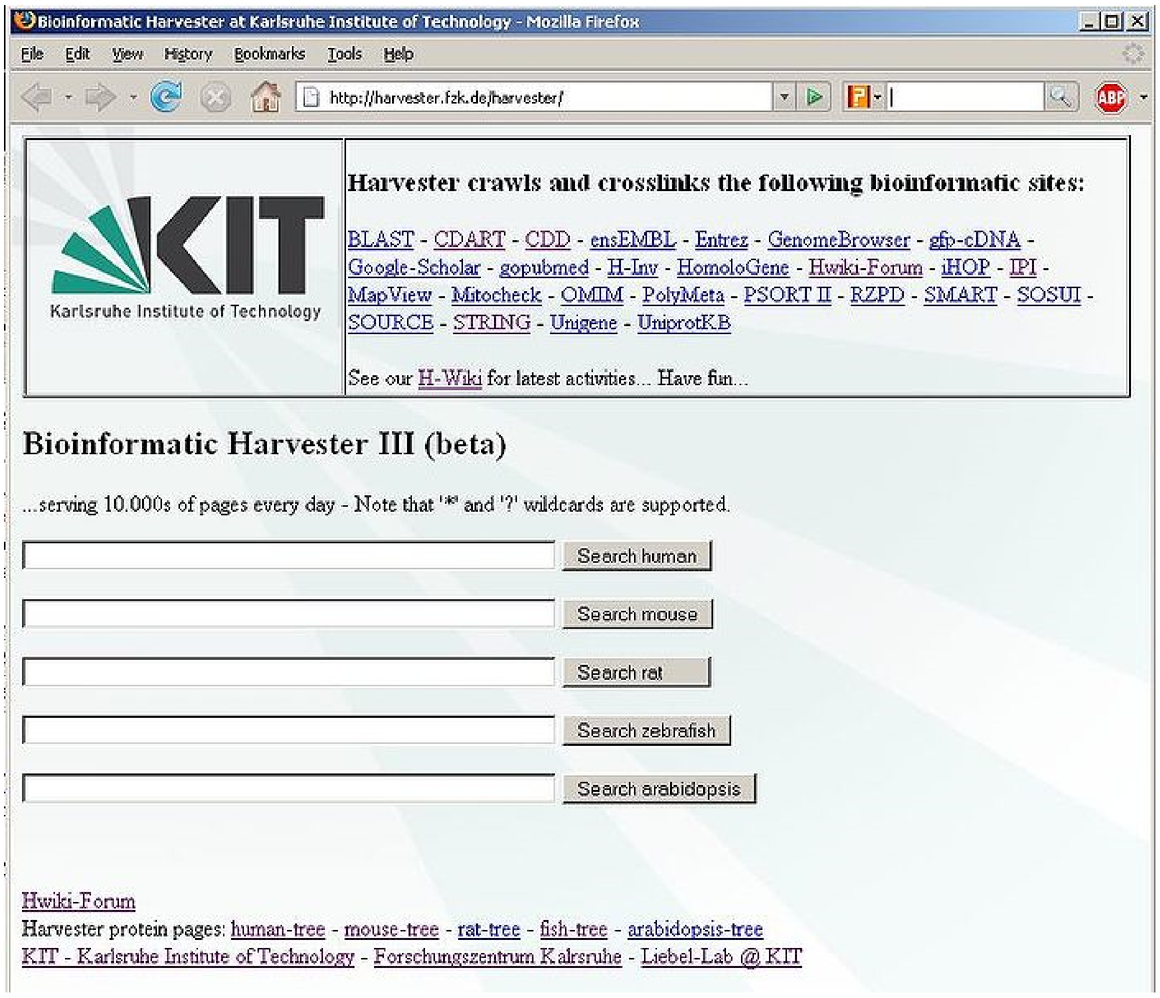
The **Bioinformatic Harvester** is a bioinformatic meta search engine at KIT Karlsruhe Institute of Technology for genes and protein-associated information. Harvester currently works for human, mouse, rat, zebrafish, drosophila and arabidopsis thaliana based information. Harvester cross-links >28 popular bioinformatic resources and allows cross searches. A ranking system similar to Google pagerank sorts the search results and displays the more relevant information. Harvester serves 10.000s of pages every day to scientists and physicians.

Bioinformatic Harvester	
Developer(s)	Urban Liebel, Björn Kindler
Stable release	4 / March 10, 2010; 12 months ago
Operating system	Web based
Type	Bioinformatics tool
License	Public Domain

How Harvester works

Harvester collects information from protein and gene databases along with information from so called "prediction servers." Prediction server e.g. provide online sequence analysis for a single protein. Harvesters search index is based on the IPI and UniProt protein information collection. The collections consists of:

- ~68.000 human, ~53.000 mouse, ~42.000 rat, ~51.000 zebrafish, ~35.000 arabidopsis and ~33.000 drosophila protein pages, which are curated and updated on a regular basis.


Bioinformatic Harvester at Karlsruhe Institute of Technology - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://harvester.fzk.de/harvester/
Karlsruhe Institute of Technology
Search human
Search mouse
Search rat
Search zebrafish
Search arabidopsis

Harvester crawls and crosslinks the following bioinformatic sites:

BLAST - CDART - CDD - ensEMBL - Entrez - GenomeBrowser - gfp-cDNA - Google-Scholar - gopubmed - H-Inv - HomoloGene - Hwiki-Forum - iHOP - IPI - MapView - Mitocheck - OMIM - PolyMeta - PSORT II - RZPD - SMART - SOSUI - SOURCE - STRING - Unigene - UniprotKB

See our H-Wiki for latest activities... Have fun...

Bioinformatic Harvester III (beta)

...serving 10.000s of pages every day - Note that '*' and '?' wildcards are supported.

Hwiki-Forum
Harvester protein pages: human-tree - mouse-tree - rat-tree - fish-tree - arabidopsis-tree
KIT - Karlsruhe Institute of Technology - Forschungszentrum Kalrsruhe - Liebel-Lab @ KIT

A screenshot of the Harvester search engine

Harvester collects several types of information

Text based information

...from the following databases:

- UniProt, world largest protein database
- SOURCE, convenient gene information overview
- Simple Modular Architecture Research Tool (SMART),
- SOSUI, predicts transmembrane domains
- PSORT, predicts protein localisation
- HomoloGene, compares proteins from different species
- gfp-cdna, protein localisation with fluorescence microscopy
- International Protein Index (IPI).

Databases rich in graphical elements

...are not collected, but crosslinked via iframes. Iframes are transparent windows within a HTML pages. The iframe windows allows up-to-date viewing of the "iframed," linked databases. Several such iframes are combined on a Harvester protein page. This method allows convenient comparison of information from several databases.

- NCBI-BLAST, an algorithm for comparing biological sequences from the NCBI.
- Ensembl, automatic gene annotation by the EMBL-EBI and Sanger Institute
- FlyBase is a database of model organism *Drosophila melanogaster*.
- GoPubMed is a knowledge-based search engine for biomedical texts.
- iHOP, information hyperlinked over proteins via gene/protein synonyms
- Mendelian Inheritance in Man project catalogues all the known diseases.
- RZPD, German resources Center for genome research in Berlin/Heidelberg.
- STRING, Search Tool for the Retrieval of Interacting Genes/Proteins, developed by EMBL, SIB and UZH.
- Zebrafish Information Network.
- LOCATE subcellular localization database (mouse).

"linkouts"

- Genome browser, working draft assemblies for genomes UCSC
- Google Scholar
- Mitocheck
- PolyMeta, meta search engine for Google, Yahoo, MSN, Ask, Exalead, AllTheWeb, GigaBlast

What one can find

Harvester allows a combination of different search terms and single words.

Search Examples:

- Gene-name: "golga3"
- Gene-alias: "ADAP-S ADAS ADHAPS ADPS" (one gene name is sufficient)
- Gene-Ontologies: "Enzyme linked receptor protein signaling pathway"
- Unigene-Cluster: "Hs.449360"

- Go-annotation: "intra-Golgi transport"
- Molecular function: "protein kinase binding"
- Protein: "Q9NPD3"
- Protein domain: "SH2 sar"
- Protein Localisation: "endoplasmic reticulum"

- Chromosome: "2q31"
- Disease relevant: use the word "diseaselink"

- Combinations: "golgi diseaselink" (finds all golgi proteins associated with a disease)
- mRNA: "AL136897"

- Word: "Cancer"
- Comment: "highly expressed in heart"
- Author: "Merkel, Schmidt"
- Publication or project: "cDNA sequencing project"

Adversarial information retrieval

Adversarial information retrieval (adversarial IR) is a topic in information retrieval related to strategies for working with a data source where some portion of it has been manipulated maliciously. Tasks can include gathering, indexing, filtering, retrieving and ranking information from such a data source. Adversarial IR includes the study of methods to detect, isolate, and defeat such manipulation.

On the Web, the predominant form of such manipulation is search engine spamming (also known as spamdexing), including techniques that are employed to disrupt the activity of web search engines, usually for financial gain. Examples of spamdexing are link-bombing, comment or referrer spam, spam blogs (splogs), malicious tagging, reverse engineering of ranking algorithms, advertisement blocking, and web content filtering .

Activities intended to poison the supply of useful data make search engines less useful for users. If search engines are more exclusionary they risk becoming more like directories and less dynamic.

Topics

Topics related to Web spam (spamdexing):

- Link spam
- Keyword spamming
- Cloaking
- Malicious tagging
- Spam related to blogs, including comment spam, splogs, and ping spam

Other topics:

- Click fraud detection
- Reverse engineering of a search engine's ranking algorithm
- Web content filtering
- Advertisement blocking

- Stealth crawling
- Malicious tagging or voting in social networks

History

The term "adversarial information retrieval" was first coined in 2000 by Andrei Broder (then Chief Scientist at Alta Vista) during the Web plenary session at the TREC-9 conference.

Anchor text

The **anchor text**, **link label** or **link title** is the visible, clickable text in a hyperlink. The words contained in the anchor text can determine the ranking that the page will receive by search engines. Since 1998, some web browsers have added the ability to show a tooltip for a hyperlink before it is selected. Not all links have anchor texts because it may be obvious where the link will lead due to the context in which it is used. Anchor texts normally remain below 60 characters. Different browsers will display anchor texts differently.

Common misunderstanding of the concept

This proper method of linking is beneficial to users and webmasters as anchor text holds significant weight in search engine rankings. The limit of the concept is building sentences only composed with linked words.

Search engine algorithms

Anchor text is weighted (ranked) highly in search engine algorithms, because the linked text is usually relevant to the landing page. The objective of search engines is to provide highly relevant search results; this is where anchor text helps, as the tendency was, more often than not, to hyperlink words relevant to the landing page.

Webmasters may use anchor text to procure high results in search engine results pages. Google's Webmaster Tools facilitate this optimization by letting website owners view the most common words in anchor text linking to their site. In the past, Google bombing was possible through anchor text manipulation; however, in January, 2007, Google announced it had updated its algorithm to minimize the impact of Google bombs.

Chapter-5

ChemRefer, Compound Term Processing, Document Clustering and Document Retrieval

ChemRefer

Chemrefer

Commercial?	Yes
Type of site	Search engine
Registration	Not Applicable
Available language(s)	English
Owner	ChemRefer Limited
Created by	William James Griffiths
Launched	2006
Current status	Active

ChemRefer is a service that allows searching of freely-available and full-text chemical and pharmaceutical literature that is published by authoritative sources.

Features include basic and advanced search options, mouseover detailed view, an integrated chemical structure drawing and search tool, downloadable toolbar, customized RSS feeds, and newsletter.

ChemRefer is primarily of use to readers who do not have subscriptions for accessing restricted chemical literature, and to publishers who offer either open access or hybrid open access journals and seek to attract further subscriptions by publicly releasing part of their archive.

Compound term processing

Compound term processing is the name that is used for a category of techniques in Information retrieval applications that performs matching on the basis of compound terms. Compound terms are built by combining two (or more) simple terms, for example "triple" is a single word term but "triple heart bypass" is a compound term.

In August 2003 Concept Searching Limited introduced the idea of using statistical Compound Term Processing via an article published in INFORMATION MANAGEMENT AND TECHNOLOGY (VOL 36 PART 4). A British Library Direct catalogue entry can be found here: .

The complete original article can also be downloaded from here: .

Further discussion of Compound Term Processing can be found here: . CLAMOUR is a European collaborative project which aims to find a better way to classify when collecting and disseminating industrial information & statistics. In contrast to the techniques discussed by Concept Searching Limited, CLAMOUR appears to be primarily a linguistic approach, rather than one based on statistical modelling. The final project report (dated March 2002) can be found here:

Compound Term Processing is important because it allows search (and other Information Retrieval) applications to perform their matching on the basis of multi-word concepts rather than single words in isolation which can be highly ambiguous.

Most search engines simply look for documents that contain the words that the user enters into the search box (aka "keyword search" engines). Boolean search engines add a degree of sophistication by allowing the user to specify additional requirements but most users struggle to comprehend and use the necessary syntax (e.g. Tiger NEAR Woods AND (golf OR golfing) NOT Volkswagen). Phrase search is easier to understand but can lead to many useful documents being missed if they do not contain the exact phrase specified.

Techniques for probabilistic weighting of single word terms dates back to at least 1976 and the landmark publication by Stephen E. Robertson and Karen Spärck Jones: Relevance weighting of search terms originally published in the Journal of the American Society for Information Science. Robertson has stated that the assumption of word independence is not justified and exists simply as a matter of mathematical convenience. The objection to assumptions about term independence are not new, dating back to at least 1964 when H. H. Williams expressed it this way: "The assumption of independence of words in a document is usually made as a matter of mathematical convenience".

Compound term processing is a new approach to an old problem: how to improve the relevance of search results without missing anything important whilst maintaining ease of use. By forming compound (i.e. multi-word) terms and placing these in the search engine's index the search can be performed with a higher degree of accuracy because the ambiguity inherent in single words is no longer a problem. A search for *survival rates following a triple heart bypass in elderly people* will locate documents about this topic even if this precise phrase is not contained in any document. A concept search using "Compound Term Processing" can extract the key concepts automatically (in this case "survival rates", "triple heart bypass" and "elderly people") and use these to select the most relevant documents.

In 2004 Anna Lynn Patterson filed a number of patents on the subject of "Phrase based indexing and retrieval" and to which Google subsequently acquired the rights. A full discussion of the patents can be found here: Webmaster Woman. The patents themselves can be found online, for example: .

Statistical Compound Term Processing is more adaptive than the "phrase based indexing and retrieval" detailed by Anna Lynn Patterson in her patent applications. The "phrase based indexing" is targeted at searching the World Wide Web where an extensive statistical knowledge of common searches can be used to identify candidate phrases. Statistical Compound Term Processing is more suited to Enterprise Search applications where such a priori knowledge is not available.

Statistical Compound Term Processing is also more adaptive than the linguistic approach taken by the CLAMOUR project which considers the syntactic properties of the terms (part of speech, gender, number) and their combination. CLAMOUR is highly language dependent, whereas the statistical approach is language independent.

Document clustering

Document clustering (also referred to as **Text clustering**) is closely related to the concept of data clustering. Document clustering is a more specific technique for unsupervised document organization, automatic topic extraction and fast information retrieval or filtering.

A web search engine often returns thousands of pages in response to a broad query, making it difficult for users to browse or to identify relevant information. Clustering methods can be used to automatically group the retrieved documents into a list of meaningful categories, as is achieved by Enterprise Search engines such as Northern Light and Vivisimo or open source software such as Carrot2.
Example:
FirstGov.gov, the official Web portal for the U.S. government, uses document clustering to automatically organize its search results into categories. For example, if a user submits “immigration”, next to their list of results they will see categories for “Immigration Reform”, “Citizenship and Immigration Services”, “Employment”, “Department of

Homeland Security”, and more. Perform Probabilistic Latent Semantic Analysis (PLSA) can also be conducted to perform document clustering.

Document clustering involves the use of descriptors and descriptor extraction. Descriptors are sets of words that describe the contents within the cluster. Document clustering is generally considered to be a centralized process. Examples of document clustering include web document clustering for search users.

Document retrieval

Document retrieval is defined as the matching of some stated user query against a set of free-text records. These records could be any type of mainly unstructured text, such as newspaper articles, real estate records or paragraphs in a manual. User queries can range from multi-sentence full descriptions of an information need to a few words.

Document retrieval is sometimes referred to as, or as a branch of, **Text Retrieval**. Text retrieval is a branch of information retrieval where the information is stored primarily in the form of text. Text databases became decentralized thanks to the personal computer and the CD-ROM. Text retrieval is a critical area of study today, since it is the fundamental basis of all internet search engines.

Description

Document retrieval systems find information to given criteria by matching text records (*documents*) against user queries, as opposed to expert systems that answer questions by inferring over a logical knowledge database. A document retrieval system consists of a database of documents, a classification algorithm to build a full text index, and a user interface to access the database.

A document retrieval system has two main tasks:

1. Find relevant documents to user queries
2. Evaluate the matching results and sort them according to relevance, using algorithms such as PageRank.

Internet search engines are classical applications of document retrieval. The vast majority of retrieval systems currently in use range from simple Boolean systems through to systems using statistical or natural language processing techniques.

Variations

There are two main classes of indexing schemata for document retrieval systems: *form based* (or *word based*), and *content based* indexing. The document classification scheme (or indexing algorithm) in use determines the nature of the document retrieval system.

Form based document retrieval addresses the exact syntactic properties of a text, comparable to substring matching in string searches. The text is generally unstructured and not necessarily in a natural language, the system could for example be used to process large sets of chemical representations in molecular biology. A suffix tree algorithm is an example for form based indexing.

The content based approach exploits semantic connections between documents and parts thereof, and semantic connections between queries and documents. Most content based document retrieval systems use an inverted index algorithm.

Example: PubMed

The PubMed form interface features the "related articles" search which works through a comparison of words from the documents' title, abstract, and MeSH terms using a word-weighted algorithm.

Chapter-6

EXCLAIM, Enterprise Search and Expertise Finding

EXCLAIM

The **EXtensible Cross-Linguistic Automatic Information Machine (EXCLAIM)** is an integrated tool for cross-language information retrieval (CLIR), created at the University of California, Santa Cruz in early 2006. It is currently in a beta stage of development, with some support for more than a dozen languages. The lead developers are Justin Nuger and Jesse Saba Kirchner.

Early work on CLIR depended on manually constructed parallel corpora for each pair of languages. This method is labor-intensive compared to parallel corpora created automatically. A more efficient way of finding data to train a CLIR system is to use matching pages on the web which are written in different languages.

EXCLAIM capitalizes on the idea of latent parallel corpora on the web by automating the alignment of such corpora in various domains. The role of EXCLAIM is to use semantics and linguistic analytic tools to align the information so that they can be treated as parallel corpora. EXCLAIM is also extensible to incorporate information from many other sources, such as the Chinese Community Health Resource Center (CCHRC).

One of the main goals of the EXCLAIM project is to provide the kind of computational tools and CLIR tools for minority languages and endangered languages which are often available only for powerful or prosperous majority languages.

Current status

EXCLAIM is in a beta state, with varying degrees of functionality for different languages. Support for CLIR using the dataset and the most current version of EXCLAIM (v.0.5), including full UTF-8 support and Porter stemming for the English component, is available for the following twenty-three languages:

Albanian

Amharic
Bengali
Gothic
Greek
Icelandic
Indonesian
Irish
Javanese
Latvian
Malagasy
Mandarin Chinese
Nahuatl
Navajo
Quechua
Sardinian
Swahili
Tagalog
Tibetan
Turkish
Welsh
Wolof
Yiddish
Dutch
Spanish

Significant developments in the most recent version of EXCLAIM include support for Mandarin Chinese. By developing support for this language, EXCLAIM has added solutions to segmentation and encoding problems which will allow the system to be extended to many other languages written with non-European orthographic conventions. This support is supplied through the Trimming And Reformatting Modular System (TARMS) toolkit.

Future versions of EXCLAIM will extend the system to additional languages.

The EXCLAIM development plan calls for an integrated CLIR instrument usable searching from English for information in any of the supported languages, or searching from any of the supported languages for information in English when EXCLAIM 1.0 is released. Future versions will allow searching from any supported language into any other, and searching from and into multiple languages.

Enterprise search

Enterprise search is the practice of making content from multiple enterprise-type sources, such as databases and intranets, searchable to a defined audience.

Enterprise search summary

"Enterprise Search" is used to describe the software of search information within an enterprise (though the search function and its results may still be public). Enterprise search can be contrasted with web search, which applies search technology to documents on the open web, and desktop search, which applies search technology to the content on a single computer.

Enterprise search systems index data and documents from a variety of sources such as: file systems, intranets, document management systems, e-mail, and databases. Many enterprise search systems integrate structured and unstructured data in their collections. Enterprise search systems also use access controls to enforce a security policy on their users.

Components of an enterprise search system

In an enterprise search systems, content goes through various phases from source repository to search results:

Content ingestion

Content ingestion (or "content collection") is us either a push or pull model. In the push model, a source system is integrated with the search engine in such a way that it connects to it and pushes new content directly to its APIs. This model is used when realtime indexing is important. In the pull model, the software gathers content from sources using a connector such as a web crawler or a database connector. The connector typically polls the source with certain intervals to look for new, updated or deleted content.

Content processing and analysis

Content from different sources may have many different formats or document types, such as XML, HTML, Office document formats or plain text. The content processing phase processes the incoming documents to plain text using document filters. It is also often necessary to normalize content in various ways to improve recall or precision. These may include stemming, lemmatization, synonym expansion, entity extraction, part of speech tagging.

As part of processing and analysis, tokenization is applied to split the content into tokens which is the basic matching unit. It is also common to normalize tokens to lower case to provide case-insensitive search, as well as to normalize accents to provide better recall.

Indexing

The resulting text is stored in an index, which is optimized for quick lookups without storing the full text of the document. The index may contain the dictionary of all unique words in the corpus as well as information about ranking and term frequency.

Query parsing

Using a web page, the user issues a query to the system. The query consists of any terms the user enters as well as navigational actions such as faceting and paging information.

Matching

The processed query is then compared to the stored index, and the search system returns results (or "hits") referencing source documents that match. Some systems are able to present the document as it was indexed.

Differences from web search

Beyond the difference in the kinds of materials being indexed, enterprise search systems also typically include functionality that is not associated with the mainstream web search engines. These include:

- Adapters to index content from a variety of repositories, such as databases and content management systems.
- Federated search, which consists of

1. transforming a query and broadcasting it to a group of disparate databases or external content sources with the appropriate syntax,
2. merging the results collected from the databases,
3. presenting them in a succinct and unified format with minimal duplication, and
4. providing a means, performed either automatically or by the portal user, to sort the merged result set.

- Enterprise bookmarking, collaborative tagging systems for capturing knowledge about structured and semi-structured enterprise data.
- Entity extraction that seeks to locate and classify elements in text into predefined categories such as the names of persons, organizations, locations, expressions of times, quantities, monetary values, percentages, etc.
- Faceted search, a technique for accessing a collection of information represented using a faceted classification, allowing users to explore by filtering available information.
- Access control, usually in the form of an Access control list (ACL), is often required to restrict access to documents based on individual user identities. There are many types of access control mechanisms for different content sources

making this a complex task to address comprehensively in an enterprise search environment.
- Text clustering, which groups the top several hundred search results into topics that are computed on the fly from the search-results descriptions, typically titles, excerpts (snippets), and meta-data. This technique lets users navigate the content by topic rather than by the meta-data that is used in faceting. Clustering compensates for the problem of incompatible meta-data across multiple enterprise repositories, which hinders the usefulness of faceting.
- User interfaces, which in web search are deliberately kept simple in order not to distract the user from clicking on ads, which generates the revenue. Although the business model for enterprise search could include showing ads, in practice this is not done. To enhance end user productivity, enterprise vendors continually experiment with rich UI functionality which occupies significant screen space, which would be problematic for web search.

Relevance factors for enterprise search

The factors that determine the relevance of search results within the context of an enterprise overlap with but are different from those that apply to web search. In general, enterprise search engines cannot take advantage of the rich link structure as is found on the web's hypertext content, however, a new breed of Enterprise search engines based on a bottom-up Web 2.0 technology are providing both a contributory approach and hyperlinking within the enterprise. Algorithms like PageRank exploit hyperlink structure to assign authority to documents, and then use that authority as a query-independent relevance factor. In contrast, enterprises typically have to use other query-independent factors, such as a document's recency or popularity, along with query-dependent factors traditionally associated with information retrieval algorithms. Also, the rich functionality of enterprise search UIs, such as clustering and faceting, diminish reliance on ranking as the means to direct the user's attention.

Search Relevance Testing options

Search application relevance can be determined by following relevance testing options:

- Empirical Testing
- A/B Testing
- Log Analysis on a Beta Production Site
- Online Ratings.

Expertise finding

a. Expert opinion or knowledge, often obtained through the action of submitting a matter to, and its consideration by, experts; an expert's appraisal, valuation, or report. b. The quality or state of being expert; skill or expertness in a particular branch of study or sport. *Oxford English Dictionary*, second edition, 1989.

One could further say that expertise is the quality exhibited by people whom we believe demonstrate an above-average ability to perform a non-trivial task. This can be restated by saying that when one is not an expert in a given field, the perception of expertise is often granted to a person based on what *other* people say is demonstrated by the presumed expert. That people often accept such claims *prima facie* is somewhat understandable, if ill-advised, and is just one of the many problems associated with assessing and quantifying human expertise. It can be argued that this behavior is partly due to a paucity of tools, metrics and software that focuses on characterizing whatever expertise they demonstrate in areas other than sports, the one domain where excellent tools are available.

Locating and assessing expertise, and why it matters

It can be argued that human expertise is the most valuable resource in the universe, more valuable than capital, means of production or intellectual property. Why? Contrary to expertise, all other aspects of capitalism are now relatively generic: access to capital is global, as is access to means of production for many areas of manufacturing. Intellectual property can be similarly licensed. Furthermore, expertise finding is also a key aspect of institutional memory, as without its experts an institution is effectively decapitated. However, finding and "licensing" expertise, the key to the effective use of these resources, remain much harder, starting with the very first step: finding expertise that you can trust.

Until very recently, finding expertise required a mix of individual, social and collaborative practices, a haphazard process at best. Mostly, it involved contacting individuals one trusts and asking them for referrals, while hoping that one's judgment about those individuals is justified and that their answers are thoughtful.

In the last fifteen years, a class of knowledge management software has emerged to facilitate and improve the quality of expertise finding, termed "expertise locating systems". These software range from social networking systems to knowledge bases. Some software, like those in the social networking realm, rely on users to connect each other, thus using social filtering to act as "recommender systems".

At the other end of the spectrum are specialized knowledge bases that rely on experts to populate a specialized type of database with their self-determined areas of expertise and contributions, and do not rely on user recommendations. Hybrids that feature expert-populated content in conjunction with user recommendations also exist, and are arguably more valuable for doing so (e.g., LinkedIn).

Still other expertise knowledge bases rely strictly on external manifestations of expertise, herein termed "gated objects", e.g., citation impacts for scientific papers or data mining approaches wherein many of the work products of an expert are collated. Such systems are more likely to be free of user-introduced biases (e.g., ResearchScorecard), though the use of computational methods can introduce other biases.

Examples of the systems outlined above are listed in Table 1.

Table 1: A classification of expertise location systems

Type	Application domain	Data source	Examples
Social networking	Professional networking	User-generated	• LinkedIn
Scientific literature	Identifying publications with strongest research impact	Third-party generated	• Science Citation Index (Thomson Reuters)
Knowledge base	Private expertise database	User-generated	• MITRE Expert Finder (MITRE Corporation) • MIT ExpertFinder (ref. 3) • MindServer Expertise (Recommind, Inc.) • Tacit Software (Oracle Corporation)
Knowledge base	Publicly-accessible expertise database	User-generated	• Community of Science Expertise • ResearcherID (Thomson Reuters) • SciLink.com (SciLink Inc.)
Knowledge base	Private expertise database	Third party-generated	• MITRE Expert Finder (MITRE Corporation) • MIT ExpertFinder (ref. 3) • MindServer Expertise (Recommind, Inc.) • Tacit Software
Knowledge base	Publicly-accessible expertise database	Third party-generated	• ResearchScorecard (ResearchScorecard Inc.) • authoratory.com • BiomedExperts (Collexis Holdings Inc.) • KnowledgeMesh (Hershey Center for Applied Research) • Community Academic Profiles (Stanford School of Medicine) • ResearchCrossroads.org (Innolyst,

			Inc.)
Blog search engines		Third party-generated	• Technorati

Technical problems associated with expertise finding

A number of interesting problems follow from the use of expertise finding systems:

- The matching of questions from non-expert to the database of existing expertise is inherently difficult, especially when the database does not store the requisite expertise. This problem grows even more acute with increasing ignorance on the part of the non-expert due to typical search problems involving use of keywords to search unstructured data that are not semantically normalized, as well as variability in how well an expert has set up their descriptive content pages. Improved question matching is one reason why third-party semantically normalized systems such as ResearchScorecard and BiomedExperts should be able to provide better answers to queries from non-expert users.
- Avoiding expert-fatigue due to too many questions/requests from users of the system (ref. 1).
- Finding ways to avoid “gaming” of the system to reap unjustified expertise credibility.

Beyond expertise finding: Expertise ranking

Means of classifying and ranking expertise (and therefore experts) become essential if the number of experts returned by a query is greater than a handful. This raises the following social problems associated with such systems:

- How can expertise be assessed objectively? Is that even possible?
- What are the consequences of relying on unstructured social assessments of expertise, such as user recommendations?
- How does one distinguish *authoritativeness* as a proxy metric of expertise from simple *popularity*, which is often a function of one's ability to express oneself coupled with a good social sense?
- What are the potential consequences of the social or professional stigma associated with the use of an authority ranking, such as used in Technorati and ResearchScorecard)?

Sources of data for assessing expertise

Many types of data sources have been used to infer expertise. They can be broadly categorized based on whether they measure "raw" contributions provided by the expert, or whether some sort of filter is applied to these contributions.

Unfiltered data sources that have been used to assess expertise, in no particular ranking order:

- user recommendations
- help desk tickets: what the problem was and who fixed it
- e-mail traffic between users
- documents, whether private or on the web, particularly publications
- user-maintained web pages
- reports (technical, marketing, etc)

Filtered data sources, that is, contributions that require approval by third-parties (grant committees, referees, patent office, etc) are particularly valuable for measuring expertise in a way that minimizes biases that follow from popularity or other social factors:

- patents, particularly if issued
- scientific publications
- issued grants (failed grant proposals are rarely know beyond the authors)
- clinical trials
- product launches
- pharmaceutical drugs

Approaches for creating expertise content

- Manual, either by experts themselves (e.g., LinkedIn) or by a curator
- Automated, e.g., using software agents (e.g., MIT's ExpertFinder and the ExpertFinder initiative) or a combination of agents and human curation (e.g., ResearchScorecard)

Interesting expertise systems over the years

In no particular order...

- Autonomy's IDOL
- AskMe
- Tacit Knowledge Systems' ActiveNet
- Triviumsoft's SEE-K
- MIT's ExpertFinder (ref 3)
- MITRE's (ref 1) Expert Finder
- MITRE's XpertNet
- Dataware II Knowledge Directory
- Thomson's tool
- Hewlett-Packard's CONNEX
- Microsoft's SPUD project

Chapter-7

Exploratory Search, Faceted Search and Federated Search

Exploratory search

Exploratory search is a specialization of information exploration which represents the activities carried out by searchers who are either:

- a) unfamiliar with the domain of their goal (ie need to learn about the topic in order to understand how to achieve their goal)
- b) unsure about the ways to achieve their goals (either the technology or the process)
- c) or even unsure about their goals in the first place.

Consequently, exploratory search covers a broader class of activities than typical information retrieval, such as investigating, evaluating, comparing, and synthesizing, where new information is sought in a defined conceptual area; exploratory data analysis is another example of an information exploration activity. Typically, therefore, such users generally combine querying and browsing strategies to foster learning and investigation.

History

Exploratory search is a topic that has grown from the fields of information retrieval and information seeking but has become more concerned with alternatives to the kind of search that has received the majority of focus (returning the most relevant documents to a Google-like keyword search). The research is motivated by questions like "what if the user doesn't know which keywords to use?" or "what if the user isn't looking for a single answer?" Consequently, research has begin to focus on defining the broader set of *information behaviors* in order to learn about the situations when a user is, or feels, limited by only having the ability to perform a keyword search.

In the last few years, a series of workshops have been held at various related and key events. In 2005, the Exploratory Search Interfaces workshop focused on beginning to define some of the key challenges in the field. Since then a series of other workshops have been held at related conferences: Evaluating Exploratory Search at SIGIR06 and Exploratory Search and HCI at CHI07 (in order to meet with the experts in human–computer interaction).

In March 2008, an *Information Processing and Management* special issue has focused particularly on the challenges of evaluating exploratory search, given the reduced assumptions that can be made about scenarios of use.

In June 2008, the National Science Foundation sponsored an invitational workshop to identify a research agenda for exploratory search and similar fields for the coming years.

Research challenges

Important scenarios

With the majority of research in the information retrieval community focusing on typical keyword search scenarios, one challenge for exploratory search is to further understand the scenarios of use for when keyword search is not sufficient. An example scenario, often used to motivate the research by mSpace states: if a user does not know much about classical music, how should they even begin to find a piece that they might like.

Designing new interfaces

With one of the motivations being to support users when keyword search is not enough, some research has focused on identifying alternative user interfaces and interaction models that support the user in different ways. An example is faceted search which presents diverse category-style options to the user, so that they can choose from a list instead of guess a possible keyword query.

Many of the interactive forms of search, including faceted browsers, are being considered for their support of exploratory search conditions.

Computational cognitive models of exploratory search have been developed to capture the cognitive complexities involved in exploratory search. Model-based dynamic presentation of information cues are proposed to facilitate exploratory search performance.

Evaluating interfaces

As the tasks and goals involved with exploratory search are largely undefined or unpredictable, it is very hard to evaluate systems with the measures often used in information retrieval. Accuracy was typically used to show that a user had found a correct answer, but when the user is trying to summarize a domain of information, the *correct* answer is near impossible to identify, if not entirely subjective (for example: possible hotels to stay in Paris). In exploration, it is also arguable that spending more time (where time efficiency is typically desirable) researching a topic shows that a system provides increased support for investigation. Finally, and perhaps most importantly, giving study participants a well specified task could immediately prevent them from exhibiting exploratory behavior.

Models of exploratory search behavior

There has been recent attempts to develop process model of exploratory search behavior, especially in social information system (The process model assumes that user-generated information cues, such as social tags, can act as navigational cues that facilitate exploration of information that others have found and shared with other users on a social information system (such as social bookmarking system). These models provided extension to existing process model of information search that characterizes information-seeking behavior in traditional fact-retrievals using search engines.

Major figures

Key figures, including experts from both information seeking and human–computer interaction, are:

- Ryen White
- Gary Marchionini
- Nicholas Belkin
- m.c. schraefel

Faceted search

Faceted search, also called **faceted navigation** or **faceted browsing**, is a technique for accessing a collection of information represented using a faceted classification, allowing users to explore by filtering available information. A faceted classification system allows the assignment of multiple classifications to an object, enabling the classifications to be ordered in multiple ways, rather than in a single, pre-determined, taxonomic order. Each facet typically corresponds to the possible values of a property common to a set of digital objects.

Facets are often derived by analysis of the text of an item using entity extraction techniques or from pre-existing fields in the database such as author, descriptor, language, and format. This approach permits existing web-pages, product descriptions or articles to have this extra metadata extracted and presented as a navigation facet.

Development

The Association for Computing Machinery's Special Interest Group on Information Retrieval provided the following description of the role of Faceted Search for a 2006 workshop:

The web search world, since its very beginning, has offered two paradigms:

- Navigational search uses a hierarchy structure (taxonomy) to enable users to browse the information space by iteratively narrowing the scope of their quest in a predetermined order, as exemplified by Yahoo! Directory, DMOZ, etc.
- Direct search allows users to simply write their queries as a bag of words in a text box. This approach has been made enormously popular by Web search engines.

Over the last few years, the direct search paradigm has gained dominance and the navigational approach became less and less popular. Recently a new approach has emerged, combining both paradigms, namely the faceted search approach. Faceted search enables users to navigate a multi-dimensional information space by combining text search with a progressive narrowing of choices in each dimension. It has become the prevailing user interaction mechanism in e-commerce sites and is being extended to deal with semi-structured data, continuous dimensions, and folksonomies.

Academic work

Within the academic community, faceted search has attracted interest primarily among library and information science researchers, and to some extent among computer science researchers specializing in information retrieval.

Major academic efforts in faceted search include the following:

- Research on view-based systems, led by Steve Pollitt at the University of Huddersfield.

- The *Flamenco* project, led by Marti Hearst at the University of California, Berkeley.

- The *Relation Browser* project, led by Gary Marchionini at the University of North Carolina.

- The *Haystack* and *SIMILE* projects, led by David Karger at the Massachusetts Institute of Technology.

- The *mSpace* project, led by m.c. schraefel at the University of Southampton.

- The CiteSeerX project at the Pennsylvania State University allows faceted search for academic documents and continues to expand into other facets such as table search.

Mass market use

Faceted search has become a popular technique in commercial search applications, particularly for online retailers and libraries. An increasing number of enterprise search vendors provide software for implementing faceted search applications.

Online retail

Online retail catalogs were among the earliest applications of faceted search, reflecting both the faceted nature of product data (i.e., most products have a type, brand, price, etc.) and the ready availability of the data in retailers' existing information systems. In the early 2000s, retailers started using faceted search, leading to its ubiquity today on their online storefronts.

Libraries

Although the noted librarian Ranganathan was a strong proponent of a faceted classification system for library materials, he did not succeed in replacing the pre-coordinated Dewey Decimal Classification system with his faceted colon classification scheme. Nonetheless, online library catalogs, also known as OPACs, have increasingly adopted faceted search interfaces. Noted examples include the North Carolina State University library catalog (part of the Triangle Research Libraries Network) and the OCLC Open WorldCat system.

Federated search

Federated search is an information retrieval technology that allows the simultaneous search of multiple searchable resources. A user makes a single query request which is distributed to the search engines participating in the federation. The federated search then aggregates the results that are received from the search engines for presentation to the user.

Purpose

The federated search is the deepweb search. In one query we can search multiple database at one time and arrange the informations in useful form and it returns the results to the user. federated search is cheap to implement.

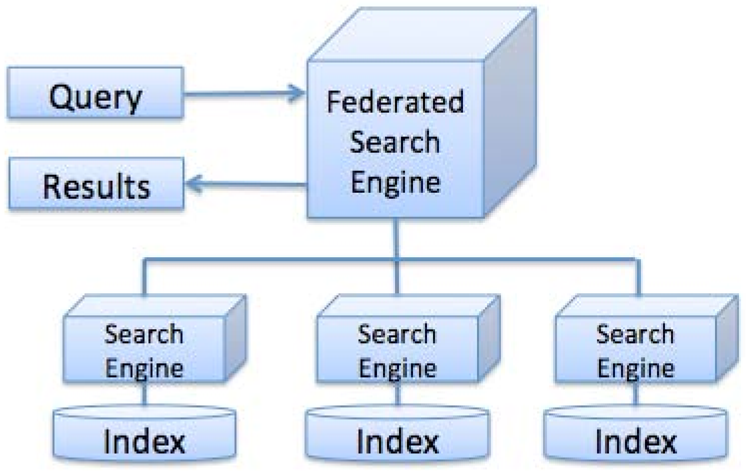
Process

As described by Peter Jacso (2004), federated searching consists of (1) transforming a query and broadcasting it to a group of disparate databases or other web resources, with the appropriate syntax, (2) merging the results collected from the databases, (3) presenting them in a succinct and unified format with minimal duplication, and (4) providing a means, performed either automatically or by the portal user, to sort the merged result set.

Federated search portals, either commercial or open access, generally search public access bibliographic databases, public access Web-based library catalogues (OPACs), Web-based search engines like Google and/or open-access, government-operated or corporate data collections. These individual information sources send back to the portal's interface a list of results from the search query. The user can review this hit list. Some portals will merely screen scrape the actual database results and not directly allow a user to enter the information source's application. More sophisticated ones will de-dupe the results list by merging and removing duplicates. There are additional features available in many portals, but the basic idea is the same: to improve the accuracy and relevance of individual searches as well as reduce the amount of time required to search for resources.

This process allows federated search some key advantages when compared with existing crawler-based search engines. Federated search need not place any requirements or burdens on owners of the individual information sources, other than handling increased traffic. Federated searches are inherently as current as the individual information sources, as they are searched in real time.

Implementation



Federating across three search engines

One application of federated searching is the metasearch engine; however, this is not a complete solution as many documents are not currently indexed. These documents are on what is known as the deep Web, or invisible Web. Many more information sources are not yet stored in electronic form. Google Scholar is one example of many projects trying to address this.

When the search vocabulary or data model of the search system is different from the data model of one or more of the foreign target systems the query must be translated into each of the foreign target systems. This can be done using simple data-element translation or may require semantic translation.

A challenge faced in the implementation of federated search engines is scalability, in other words, the performance of the site as the number of information sources comprising the federated search engine increase. One federated search engine that has begun to address this issue is WorldWideScience, hosted by the U.S. Department of Energy's Office of Scientific and Technical Information. WorldWideScience is composed of more than 40 information sources, several of which are federated search portals themselves. One such portal is Science.gov which itself federates more than 30 information sources representing most of the R&D output of the U.S. Federal government. Science.gov returns its highest ranked results to WorldWideScience, which then merges and ranks these results with the search returned by the other information sources that comprise WorldWideScience. This approach of cascaded federated search enables large number of information sources to be searched via a single query.

Another application Sesam running in both Norway and Sweden has been built on top of an open sourced platform specialised for federated search solutions. Sesat, an acronym for Sesam Search Application Toolkit, is a platform that provides much of the framework and functionality required for handling parallel and pipelined searches and displaying them elegantly in a user interface, allowing engineers to focus on the index/database configuration tuning.

Challenges

When federated search is performed against secure data sources, the users' credentials must be passed on to each underlying search engine, so that appropriate security is maintained. If the user has different login credentials for different systems, there must be a means to map their login ID to each search engine's security domain.

Another challenge is mapping results list navigators into a common form. Suppose 3 real-estate sites are searched, each provides a list of hyperlinked city names to click on, to see matches only in each city. Ideally these facets would be combined into one set, but that presents additional technical challenges. The system also needs to understand "next page" links if it's going to allow the user to page through the combined results.

Chapter-8

Human–computer Information Retrieval and Full Text Search

Human–computer information retrieval

Human–computer information retrieval (**HCIR**) is the study of information retrieval techniques that bring human intelligence into the search process. The fields of human–computer interaction (HCI) and information retrieval (IR) have both developed innovative techniques to address the challenge of navigating complex information spaces, but their insights have often failed to cross disciplinary borders. Human–computer information retrieval has emerged in academic research and industry practice to bring together research in the fields of IR and HCI, in order to create new kinds of search systems that depend on continuous human control of the search process.

History

This term *human–computer information retrieval* was coined by Gary Marchionini in a series of lectures delivered between 2004 and 2006. Marchionini's main thesis is that "HCIR aims to empower people to explore large-scale information bases but demands that people also take responsibility for this control by expending cognitive and physical energy."

In 1996 and 1998, a pair of workshops at the University of Glasgow on information retrieval and human–computer interaction sought to address the overlap between these two fields. Marchionini notes the impact of the World Wide Web and the sudden increase in information literacy – changes that were only embryonic in the late 1990s.

A few workshops have focused on the intersection of IR and HCI. The Workshop on Exploratory Search, initiated by the University of Maryland Human-Computer Interaction Lab in 2005, alternates between the Association for Computing Machinery Special Interest Group on Information Retrieval (SIGIR) and Special Interest Group on Computer-Human Interaction (CHI) conferences. Also in 2005, the European Science Foundation held an Exploratory Workshop on Information Retrieval in Context. Then, the first Workshop on Human Computer Information Retrieval was held in 2007 at the Massachusetts Institute of Technology.

What is HCIR?

HCIR includes various aspects of IR and HCI. These include exploratory search, in which users generally combine querying and browsing strategies to foster learning and investigation; information retrieval in context (i.e., taking into account aspects of the user or environment that are typically not reflected in a query); and interactive information retrieval, which Peter Ingwersen defines as "the interactive communication processes that occur during the retrieval of information by involving all the major participants in information retrieval (IR), i.e. the user, the intermediary, and the IR system."

A key concern of HCIR is that IR systems intended for human users be implemented and evaluated in a way that reflects the needs of those users.

Most modern IR systems employ a ranked retrieval model, in which the documents are scored based on the probability of the document's relevance to the query. In this model, the system only presents the top-ranked documents to the user. This systems are typically evaluated based on their mean average precision over a set of benchmark queries from organizations like the Text Retrieval Conference (TREC).

Because of its emphasis in using human intelligence in the information retrieval process, HCIR requires different evaluation models – one that combines evaluation of the IR and HCI components of the system. A key area of research in HCIR involves evaluation of these systems. Early work on interactive information retrieval, such as Juergen Koenemann and Nicholas J. Belkin's 1996 study of different levels of interaction for automatic query reformulation, leverage the standard IR measures of precision and recall but apply them to the results of multiple iterations of user interaction, rather than to a single query response. Other HCIR research, such as Pia Borlund's IIR evaluation model, applies a methodology more reminiscent of HCI, focusing on the characteristics of users, the details of experimental design, etc.

Goals

Marchionini put forth the following goals towards a system where the user has more control in determining relevant results:

- Systems should aim to get people closer to the information they need, especially to the meaning; that is, systems can no longer only deliver the relevant documents, but must also provide facilities for making meaning with those documents.

- Systems should increase user responsibility as well as control; that is, information systems require human intellectual effort, and good effort is rewarded.

- Systems should have flexible architectures so they may evolve and adapt to increasingly more demanding and knowledgeable installed bases of users over time.

- Systems should aim to be part of information ecology of personal and shared memories and tools rather than discrete standalone services.

- Systems should support the entire information life cycle (from creation to preservation) rather than only the dissemination or use phase.

- Systems should support tuning by end users and especially by information professionals who add value to information resources.

- Systems should be engaging and fun to use.

Techniques

The techniques associated with HCIR emphasize representations of information that use human intelligence to lead the user to relevant results. These techniques also strive to allow users to explore and digest the dataset without penalty, i.e., without expending unnecessary costs of time, mouse clicks, or context shift.

Many search engines have features that incorporate HCIR techniques. Spelling suggestions and automatic query reformulation provide mechanisms for suggesting potential search paths that can lead the user to relevant results. These suggestions are presented to the user, putting control of selection and interpretation in the user's hands.

Faceted search enables users to navigate information hierarchically, going from a category to its sub-categories, but choosing the order in which the categories are presented. This contrasts with traditional taxonomies in which the hierarchy of categories is fixed and unchanging. Faceted navigation, like taxonomic navigation, guides users by showing them available categories (or facets), but does not require them to browse through a hierarchy that may not precisely suit their needs or way of thinking.

Lookahead provides a general approach to penalty-free exploration. For example, various web applications employ AJAX to automatically complete query terms and suggest popular searches. Another common example of lookahead is the way in which search engines annotate results with summary information about those results, including both static information (e.g., metadata about the objects) and "snippets" of document text that are most pertinent to the words in the search query.

Relevance feedback allows users to guide an IR system by indicating whether particular results are more or less relevant.

Summarization and analytics help users digest the results that come back from the query. Summarization here is intended to encompass any means of aggregating or compressing the query results into a more human-consumable form. Faceted search, described above, is one such form of summarization. Another is clustering, which analyzes a set of documents by grouping similar or co-occurring documents or terms. Clustering allows the results to be partitioned into groups of related documents. For example, a search for "java" might return clusters for Java (programming language), Java (island), or Java (coffee).

Visual representation of data is also considered a key aspect of HCIR. The representation of summarization or analytics may be displayed as tables, charts, or summaries of aggregated data. Other kinds of information visualization that allow users access to summary views of search results include tag clouds and treemapping.

Full text search

In text retrieval, **full text search** refers to a technique for searching a computer-stored document or database. In a full text search, the search engine examines all of the words in every stored document as it tries to match search words supplied by the user. Full text searching techniques became common in online bibliographic databases in the 1970s. Many web sites and application programs (such as word processing software) provide full-text search capabilities. Some web search engines such as AltaVista employ full text search techniques while others index only a portion of the web pages examined by its indexing system.

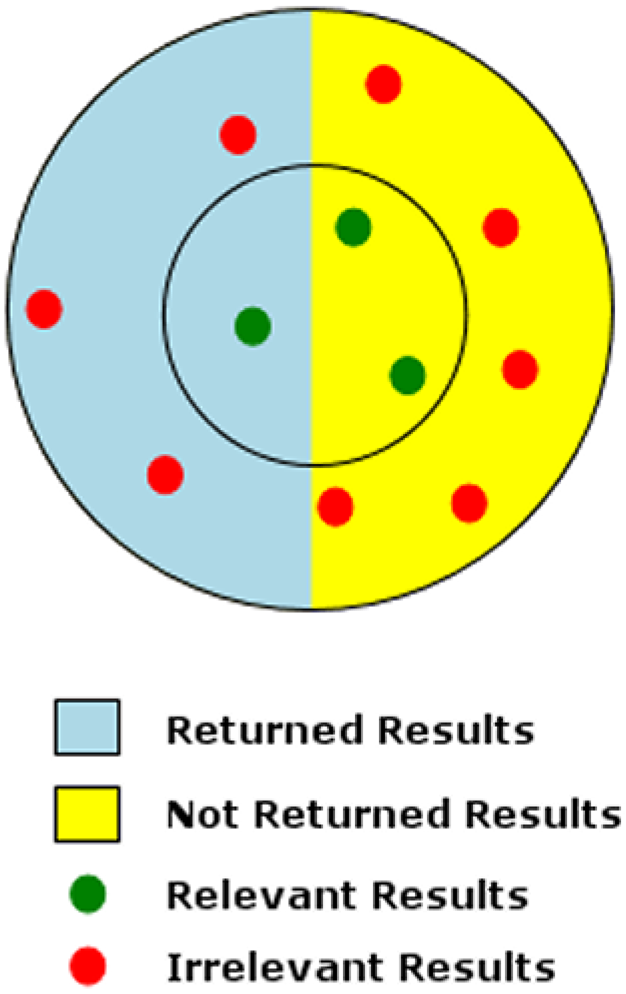
Indexing

When dealing with a small number of documents it is possible for the full-text search engine to directly scan the contents of the documents with each query, a strategy called serial scanning. This is what some rudimentary tools, such as grep, do when searching.

However, when the number of documents to search is potentially large or the quantity of search queries to perform is substantial, the problem of full text search is often divided into two tasks: indexing and searching. The indexing stage will scan the text of all the documents and build a list of search terms, often called an index, but more correctly named a concordance. In the search stage, when performing a specific query, only the index is referenced rather than the text of the original documents.

The indexer will make an entry in the index for each term or word found in a document and possibly its relative position within the document. Usually the indexer will ignore stop words, such as the English "the", which are both too common and carry too little meaning to be useful for searching. Some indexers also employ language-specific stemming on the words being indexed, so for example any of the words "drives", "drove", or "driven" will be recorded in the index under a single concept word "drive".

The precision vs. recall tradeoff



This diagram represents a low-precision, low-recall search as described in the text.

Recall measures the quantity of results returned by a search and precision is the measure of the quality of the results returned. Recall is the ratio of relevant results returned divided by all relevant results. Precision is the number of relevant results returned divided by the total number of results returned.

The diagram at right represents a low-precision, low-recall search. In the diagram the red and green dots represent the total population of potential search results for a given search. Red dots represent irrelevant results, and green dots represent relevant results. Relevancy is indicated by the proximity of search results to the center of the inner circle. Of all possible results shown, those that were actually returned by the search are shown on a light-blue background. In the example only one relevant result of three possible relevant results was returned, so the recall is a very low ratio of 1/3 or 33%. The precision for the example is a very low 1/4 or 25%, since only one of the four results returned was relevant.

Due to the ambiguities of natural language, full text search systems typically includes options like stop words to increase precision and stemming to increase recall. Controlled-vocabulary searching also helps alleviate low-precision issues by tagging documents in such a way that ambiguities are eliminated. The trade-off between precision and recall is simple: an increase in precision can lower overall recall while an increase in recall lowers precision.

False-positive problem

Free text searching is likely to retrieve many documents that are not relevant to the *intended* search question. Such documents are called *false positives*. The retrieval of irrelevant documents is often caused by the inherent ambiguity of natural language. In the sample diagram at right, false positives are represented by the irrelevant results (red dots) that were returned by the search (on a light-blue background).

Clustering techniques based on Bayesian algorithms can help reduce false positives. For a search term of "football", clustering can be used to categorize the document/data universe into "American football", "corporate football", etc. Depending on the occurrences of words relevant to the categories, search terms a search result can be placed in one or more of the categories. This technique is being extensively deployed in the e-discovery domain.

Performance improvements

The deficiencies of free text searching have been addressed in two ways: By providing users with tools that enable them to express their search questions more precisely, and by developing new search algorithms that improve retrieval precision.

Improved querying tools

- Keywords. Document creators (or trained indexers) are asked to supply a list of words that describe the subject of the text, including synonyms of words that describe this subject. Keywords improve recall, particularly if the keyword list includes a search word that is not in the document text.
- Field-restricted search. Some search engines enable users to limit free text searches to a particular field within a stored data record, such as "Title" or "Author."
- Boolean queries. Searches that use Boolean operators (for example, "encyclopedia" AND "online" NOT "Encarta") can dramatically increase the precision of a free text search. The AND operator says, in effect, "Do not retrieve any document unless it contains both of these terms." The NOT operator says, in effect, "Do not retrieve any document that contains this word." If the retrieval list retrieves too few documents, the OR operator can be used to increase recall; consider, for example, "encyclopedia" AND "online" OR "Internet" NOT "Encarta". This search will retrieve documents about online encyclopedias that use the term "Internet" instead of "online." This increase in precision is very

commonly counter-productive since it usually comes with a dramatic loss of recall.

- Phrase search. A phrase search matches only those documents that contain a specified phrase,
- Concept search. A search that is based on multi-word concepts, for example Compound term processing. This type of search is becoming popular in many e-Discovery solutions.
- Concordance search. A concordance search produces an alphabetical list of all principal words that occur in a text with their immediate context.
- Regular expression. A regular expression employs a complex but powerful querying syntax that can be used to specify retrieval conditions with precision.
- Fuzzy search will search for document that match the given terms and some variation around them (using for instance edit distance to threshold the multiple variation)
- Wildcard search. A search that substitutes one or more characters in a search query for a wildcard character such as an asterisk. For example using the asterisk in a search query "s*n" will find "sin", "son", "sun", etc. in a text.

Improved search algorithms

The PageRank algorithm developed by Google gives more prominence to documents to which other Web pages have linked.

Software

The following is a partial list of available software products whose predominant purpose is to perform full text indexing and searching. Some of these are accompanied with detailed descriptions of their theory of operation or internal algorithms, which can provide additional insight into how full text search may be accomplished.

Free and open source software

- DataparkSearch
- Ferret
- Ht-//Dig
- Hyper Estraier
- Lemur/Indri
- Lucene
- mnoGoSearch
- Sphinx
- Swish-e
- Xapian
- KinoSearch

Proprietary software

- MarkLogic
- Attivio
- Autonomy Corporation
- Brainware
- BRS/Search
- Concept Searching Limited
- Dieselpoint
- dtSearch
- Endeca
- Exalead
- Fast Search & Transfer
- Inktomi
- Vivísimo
- Lucid Imagination

Chapter-9

Index (search engine)

Search engine indexing collects, parses, and stores data to facilitate fast and accurate information retrieval. Index design incorporates interdisciplinary concepts from linguistics, cognitive psychology, mathematics, informatics, physics, and computer science. An alternate name for the process in the context of search engines designed to find web pages on the Internet is **Web indexing**.

Popular engines focus on the full-text indexing of online, natural language documents. Media types such as video and audio and graphics are also searchable.

Meta search engines reuse the indices of other services and do not store a local index, whereas cache-based search engines permanently store the index along with the corpus. Unlike full-text indices, partial-text services restrict the depth indexed to reduce index size. Larger services typically perform indexing at a predetermined time interval due to the required time and processing costs, while agent-based search engines index in real time.

Indexing

The purpose of storing an index is to optimize speed and performance in finding relevant documents for a search query. Without an index, the search engine would scan every document in the corpus, which would require considerable time and computing power. For example, while an index of 10,000 documents can be queried within milliseconds, a sequential scan of every word in 10,000 large documents could take hours. The additional computer storage required to store the index, as well as the considerable increase in the time required for an update to take place, are traded off for the time saved during information retrieval.

Index Design Factors

Major factors in designing a search engine's architecture include:

Merge factors

How data enters the index, or how words or subject features are added to the index during text corpus traversal, and whether multiple indexers can work asynchronously. The indexer must first check whether it is updating old content or adding new content. Traversal typically correlates to the data collection policy. Search engine index merging is similar in concept to the SQL Merge command and other merge algorithms.

Storage techniques

How to store the index data, that is, whether information should be data compressed or filtered.

Index size

How much computer storage is required to support the index.

Lookup speed

How quickly a word can be found in the inverted index. The speed of finding an entry in a data structure, compared with how quickly it can be updated or removed, is a central focus of computer science.

Maintenance

How the index is maintained over time.

Fault tolerance

How important it is for the service to be reliable. Issues include dealing with index corruption, determining whether bad data can be treated in isolation, dealing with bad hardware, partitioning, and schemes such as hash-based or composite partitioning, as well as replication.

Index Data Structures

Search engine architectures vary in the way indexing is performed and in methods of index storage to meet the various design factors. Types of indices include:

Suffix tree

Figuratively structured like a tree, supports linear time lookup. Built by storing the suffixes of words. The suffix tree is a type of trie. Tries support extendable hashing, which is important for search engine indexing. Used for searching for patterns in DNA sequences and clustering. A major drawback is that the storage of a word in the tree may require more storage than storing the word itself. An alternate representation is a suffix array, which is considered to require less virtual memory and supports data compression such as the BWT algorithm.

Inverted index

Stores a list of occurrences of each atomic search criterion, typically in the form of a hash table or binary tree.

Citation index

Stores citations or hyperlinks between documents to support citation analysis, a subject of Bibliometrics.

Ngram index

Stores sequences of length of data to support other types of retrieval or text mining.

Document-term matrix

Used in latent semantic analysis, stores the occurrences of words in documents in a two-dimensional sparse matrix.

Challenges in Parallelism

A major challenge in the design of search engines is the management of serial computing processes. There are many opportunities for race conditions and coherent faults. For example, a new document is added to the corpus and the index must be updated, but the index simultaneously needs to continue responding to search queries. This is a collision between two competing tasks. Consider that authors are producers of information, and a web crawler is the consumer of this information, grabbing the text and storing it in a cache (or corpus). The forward index is the consumer of the information produced by the corpus, and the inverted index is the consumer of information produced by the forward index. This is commonly referred to as a **producer-consumer model**. The indexer is the producer of searchable information and users are the consumers that need to search. The challenge is magnified when working with distributed storage and distributed processing. In an effort to scale with larger amounts of indexed information, the search engine's architecture may involve distributed computing, where the search engine consists of several machines operating in unison. This increases the possibilities for incoherency and makes it more difficult to maintain a fully synchronized, distributed, parallel architecture.

Inverted indices

Many search engines incorporate an inverted index when evaluating a search query to quickly locate documents containing the words in a query and then rank these documents by relevance. Because the inverted index stores a list of the documents containing each word, the search engine can use direct access to find the documents associated with each word in the query in order to retrieve the matching documents quickly. The following is a simplified illustration of an inverted index:

Inverted Index

Word	**Documents**
the	Document 1, Document 3, Document 4, Document 5
cow	Document 2, Document 3, Document 4
says	Document 5
moo	Document 7

This index can only determine whether a word exists within a particular document, since it stores no information regarding the frequency and position of the word; it is therefore considered to be a boolean index. Such an index determines which documents match a query but does not rank matched documents. In some designs the index includes additional information such as the frequency of each word in each document or the positions of a word in each document. Position information enables the search algorithm to identify word proximity to support searching for phrases; frequency can be used to help in ranking the relevance of documents to the query. Such topics are the central research focus of information retrieval.

The inverted index is a sparse matrix, since not all words are present in each document. To reduce computer storage memory requirements, it is stored differently from a two dimensional array. The index is similar to the term document matrices employed by latent semantic analysis. The inverted index can be considered a form of a hash table. In some cases the index is a form of a binary tree, which requires additional storage but may reduce the lookup time. In larger indices the architecture is typically a distributed hash table.

Index Merging

The inverted index is filled via a merge or rebuild. A rebuild is similar to a merge but first deletes the contents of the inverted index. The architecture may be designed to support incremental indexing, where a merge identifies the document or documents to be added or updated and then parses each document into words. For technical accuracy, a merge conflates newly indexed documents, typically residing in virtual memory, with the index cache residing on one or more computer hard drives.

After parsing, the indexer adds the referenced document to the document list for the appropriate words. In a larger search engine, the process of finding each word in the inverted index (in order to report that it occurred within a document) may be too time consuming, and so this process is commonly split up into two parts, the development of a forward index and a process which sorts the contents of the forward index into the inverted index. The inverted index is so named because it is an inversion of the forward index.

The Forward Index

The forward index stores a list of words for each document. The following is a simplified form of the forward index:

Forward Index

Document	**Words**
Document 1	the,cow,says,moo
Document 2	the,cat,and,the,hat
Document 3	the,dish,ran,away,with,the,fork

The rationale behind developing a forward index is that as documents are parsing, it is better to immediately store the words per document. The delineation enables Asynchronous system processing, which partially circumvents the inverted index update bottleneck. The forward index is sorted to transform it to an inverted index. The forward index is essentially a list of pairs consisting of a document and a word, collated by the document. Converting the forward index to an inverted index is only a matter of sorting the pairs by the words. In this regard, the inverted index is a word-sorted forward index.

Compression

Generating or maintaining a large-scale search engine index represents a significant storage and processing challenge. Many search engines utilize a form of compression to reduce the size of the indices on disk. Consider the following scenario for a full text, Internet search engine.

- An estimated 2,000,000,000 different web pages exist as of the year 2000
- Suppose there are 250 words on each webpage (based on the assumption they are similar to the pages of a novel.
- It takes 8 bits (or 1 byte) to store a single character. Some encodings use 2 bytes per character
- The average number of characters in any given word on a page may be estimated at 5**.**
- The average personal computer comes with 100 to 250 gigabytes of usable space

Given this scenario, an uncompressed index (assuming a non-conflated, simple, index) for 2 billion web pages would need to store 500 billion word entries. At 1 byte per character, or 5 bytes per word, this would require 2500 gigabytes of storage space alone, more than the average free disk space of 25 personal computers. This space requirement may be even larger for a fault-tolerant distributed storage architecture. Depending on the compression technique chosen, the index can be reduced to a fraction of this size. The tradeoff is the time and processing power required to perform compression and decompression.

Notably, large scale search engine designs incorporate the cost of storage as well as the costs of electricity to power the storage. Thus compression is a measure of cost.

Document parsing

Document parsing breaks apart the components (words) of a document or other form of media for insertion into the forward and inverted indices. The words found are called *tokens*, and so, in the context of search engine indexing and natural language processing, parsing is more commonly referred to as tokenization. It is also sometimes called word boundary disambiguation, tagging, text segmentation, content analysis, text analysis, text mining, concordance generation, speech segmentation, lexing, or lexical analysis. The terms 'indexing', 'parsing', and 'tokenization' are used interchangeably in corporate slang.

Natural language processing, as of 2006, is the subject of continuous research and technological improvement. Tokenization presents many challenges in extracting the necessary information from documents for indexing to support quality searching. Tokenization for indexing involves multiple technologies, the implementation of which are commonly kept as corporate secrets.

Challenges in Natural Language Processing

Word Boundary Ambiguity

Native English speakers may at first consider tokenization to be a straightforward task, but this is not the case with designing a multilingual indexer. In digital form, the texts of other languages such as Chinese, Japanese or Arabic represent a greater challenge, as words are not clearly delineated by whitespace. The goal during tokenization is to identify words for which users will search. Language-specific logic is employed to properly identify the boundaries of words, which is often the rationale for designing a parser for each language supported (or for groups of languages with similar boundary markers and syntax).

Language Ambiguity

To assist with properly ranking matching documents, many search engines collect additional information about each word, such as its language or lexical category (part of speech). These techniques are language-dependent, as the syntax varies among languages. Documents do not always clearly identify the language of the document or represent it accurately. In tokenizing the document, some search engines attempt to automatically identify the language of the document.

Diverse File Formats

In order to correctly identify which bytes of a document represent characters, the file format must be correctly handled. Search engines which support multiple file formats must be able to correctly open and access the document and be able to tokenize the characters of the document.

Faulty Storage

The quality of the natural language data may not always be perfect. An unspecified number of documents, particular on the Internet, do not closely obey proper file protocol. binary characters may be mistakenly encoded into various parts of a document. Without recognition of these characters and appropriate handling, the index quality or indexer performance could degrade.

Tokenization

Unlike literate humans, computers do not understand the structure of a natural language document and cannot automatically recognize words and sentences. To a computer, a document is only a sequence of bytes. Computers do not 'know' that a space character separates words in a document. Instead, humans must program the computer to identify what constitutes an individual or distinct word, referred to as a token. Such a program is commonly called a tokenizer or parser or lexer. Many search engines, as well as other natural language processing software, incorporate specialized programs for parsing, such as YACC or Lex.

During tokenization, the parser identifies sequences of characters which represent words and other elements, such as punctuation, which are represented by numeric codes, some of which are non-printing control characters. The parser can also identify entities such as email addresses, phone numbers, and URLs. When identifying each token, several characteristics may be stored, such as the token's case (upper, lower, mixed, proper), language or encoding, lexical category (part of speech, like 'noun' or 'verb'), position, sentence number, sentence position, length, and line number.

Language Recognition

If the search engine supports multiple languages, a common initial step during tokenization is to identify each document's language; many of the subsequent steps are language dependent (such as stemming and part of speech tagging). Language recognition is the process by which a computer program attempts to automatically identify, or categorize, the language of a document. Other names for language recognition include language classification, language analysis, language identification, and language tagging. Automated language recognition is the subject of ongoing research in natural language processing. Finding which language the words belongs to may involve the use of a language recognition chart.

Format Analysis

If the search engine supports multiple document formats, documents must be prepared for tokenization. The challenge is that many document formats contain formatting information in addition to textual content. For example, HTML documents contain HTML tags, which specify formatting information such as new line starts, **bold** emphasis, and font size or style. If the search engine were to ignore the difference between content and 'markup', extraneous information would be included in the index, leading to poor search results. Format analysis is the identification and handling of the formatting content embedded within documents which controls the way the document is rendered on a computer screen or interpreted by a software program. Format analysis is also referred to as structure analysis, format parsing, tag stripping, format stripping, text normalization, text cleaning, and text preparation. The challenge of format analysis is further complicated by the intricacies of various file formats. Certain file formats are proprietary with very little information disclosed, while others are well documented. Common, well-documented file formats that many search engines support include:

- HTML
- ASCII text files (a text document without specific computer readable formatting)
- Adobe's Portable Document Format (PDF)
- PostScript (PS)
- LaTeX
- UseNet netnews server formats
- XML and derivatives like RSS
- SGML
- Multimedia meta data formats like ID3
- Microsoft Word
- Microsoft Excel
- Microsoft Powerpoint
- IBM Lotus Notes

Options for dealing with various formats include using a publicly available commercial parsing tool that is offered by the organization which developed, maintains, or owns the format, and writing a custom parser.

Some search engines support inspection of files that are stored in a compressed or encrypted file format. When working with a compressed format, the indexer first decompresses the document; this step may result in one or more files, each of which must be indexed separately. Commonly supported compressed file formats include:

- ZIP - Zip archive file
- RAR - Roshal ARchive File
- CAB - Microsoft Windows Cabinet File
- Gzip - File compressed with gzip
- BZIP - File compressed using bzip2
- Tape ARchive (TAR), Unix archive file, not (itself) compressed
- TAR.Z, TAR.GZ or TAR.BZ2 - Unix archive files compressed with Compress, GZIP or BZIP2

Format analysis can involve quality improvement methods to avoid including 'bad information' in the index. Content can manipulate the formatting information to include additional content. Examples of abusing document formatting for spamdexing:

- Including hundreds or thousands of words in a section which is hidden from view on the computer screen, but visible to the indexer, by use of formatting (e.g. hidden "div" tag in HTML, which may incorporate the use of CSS or Javascript to do so).
- Setting the foreground font color of words to the same as the background color, making words hidden on the computer screen to a person viewing the document, but not hidden to the indexer.

Section Recognition

Some search engines incorporate section recognition, the identification of major parts of a document, prior to tokenization. Not all the documents in a corpus read like a well-written book, divided into organized chapters and pages. Many documents on the web, such as newsletters and corporate reports, contain erroneous content and side-sections which do not contain primary material (that which the document is about). For example, Here we, displays a side menu with links to other web pages. Some file formats, like HTML or PDF, allow for content to be displayed in columns. Even though the content is displayed, or rendered, in different areas of the view, the raw markup content may store this information sequentially. Words that appear sequentially in the raw source content are indexed sequentially, even though these sentences and paragraphs are rendered in different parts of the computer screen. If search engines index this content as if it were normal content, the quality of the index and search quality may be degraded due to the mixed content and improper word proximity. Two primary problems are noted:

- Content in different sections is treated as related in the index, when in reality it is not
- Organizational 'side bar' content is included in the index, but the side bar content does not contribute to the meaning of the document, and the index is filled with a poor representation of its documents.

Section analysis may require the search engine to implement the rendering logic of each document, essentially an abstract representation of the actual document, and then index the representation instead. For example, some content on the Internet is rendered via Javascript. If the search engine does not render the page and evaluate the Javascript within the page, it would not 'see' this content in the same way and would index the document incorrectly. Given that some search engines do not bother with rendering issues, many web page designers avoid displaying content via Javascript or use the Noscript tag to ensure that the web page is indexed properly. At the same time, this fact can also be exploited to cause the search engine indexer to 'see' different content than the viewer.

Meta Tag Indexing

Specific documents often contain embedded meta information such as author, keywords, description, and language. For HTML pages, the meta tag contains keywords which are also included in the index. Earlier Internet search engine technology would only index the keywords in the meta tags for the forward index; the full document would not be parsed. At that time full-text indexing was not as well established, nor was the hardware able to support such technology. The design of the HTML markup language initially included support for meta tags for the very purpose of being properly and easily indexed, without requiring tokenization.

As the Internet grew through the 1990s, many brick-and-mortar corporations went 'online' and established corporate websites. The keywords used to describe webpages (many of which were corporate-oriented webpages similar to product brochures) changed from descriptive to marketing-oriented keywords designed to drive sales by placing the webpage high in the search results for specific search queries. The fact that these keywords were subjectively specified was leading to spamdexing, which drove many search engines to adopt full-text indexing technologies in the 1990s. Search engine designers and companies could only place so many 'marketing keywords' into the content of a webpage before draining it of all interesting and useful information. Given that conflict of interest with the business goal of designing user-oriented websites which were 'sticky', the customer lifetime value equation was changed to incorporate more useful content into the website in hopes of retaining the visitor. In this sense, full-text indexing was more objective and increased the quality of search engine results, as it was one more step away from subjective control of search engine result placement, which in turn furthered research of full-text indexing technologies.

In Desktop search, many solutions incorporate meta tags to provide a way for authors to further customize how the search engine will index content from various files that is not evident from the file content. Desktop search is more under the control of the user, while Internet search engines must focus more on the full text index.

Chapter-10

Music Information Retrieval, IR Evaluation, Gain (information retrieval) and Harvester42

Music information retrieval

Music information retrieval (**MIR**) is the interdisciplinary science of retrieving information from music.

This includes:

- Computational methods for classification, clustering, and modelling — musical feature extraction for mono- and polyphonic music, similarity and pattern matching, retrieval
- Formal methods and databases — applications of automated music identification and recognition, such as score following, automatic accompaniment, routing and filtering for music and music queries, query languages, standards and other metadata or protocols for music information handling and retrieval, multi-agent systems, distributed search)
- Software for music information retrieval — Semantic Web and musical digital objects, intelligent agents, collaborative software, web-based search and semantic retrieval, query by humming, acoustic fingerprinting
- Human-computer interaction and interfaces — multi-modal interfaces, user interfaces and usability, mobile applications, user behavior
- Music perception, cognition, affect, and emotions — music similarity metrics, syntactical parameters, semantic parameters, musical forms, structures, styles and genres, music annotation methodologies
- Music analysis and knowledge representation — automatic summarization, citing, excerpting, downgrading, transformation, formal models of music, digital scores and representations, music indexing and metadata. Music Scalar Theory Analysis is a new addition to Music Analysis of MIR where the focus of finding the key and scale of a Blind Music Signal is a non trivial pursuit.

- Music archives, libraries, and digital collections — music digital libraries, public access to musical archives, benchmarks and research databases
- Intellectual property rights and music — national and international copyright issues, digital rights management, identification and traceability
- Sociology and Economy of music — music industry and use of MIR in the production, distribution, consumption chain, user profiling, validation, user needs and expectations, evaluation of music IR systems, building test collections, experimental design and metrics

IR evaluation

IR Evaluation

The evaluation of information retrieval system is the process of assessing how well a system meets the information needs of its users. Traditional evaluation metrics, designed for Boolean retrieval or top-k retrieval, include precision and recall.

- **Precision** is the fraction of retrieved documents that are relevant to the query:

$$\text{precision} = \frac{|\{\text{relevant documents}\} \cap \{\text{retrieved documents}\}|}{|\{\text{retrieved documents}\}|}$$

- **Recall** is the fraction of the documents relevant to the query that are successfully retrieved:

$$\text{recall} = \frac{|\{\text{relevant documents}\} \cap \{\text{retrieved documents}\}|}{|\{\text{relevant documents}\}|}$$

For modern (Web-scale) information retrieval, recall is no longer a meaningful metric, as many queries have thousands of relevant documents, and few users will be interested in reading all of them. Precision at k documents (P@k) is still a useful metric (e.g., P@10 corresponds to the number of relevant results on the first search results page), but fails to take into account the positions of the relevant documents among the top k.

Virtually all modern evaluation metrics (e.g., mean average precision, discounted cumulative gain) are designed for **ranked retrieval** without any explicit rank cutoff, taking into account the relative order of the documents retrieved by the search engines and giving more weight to documents returned at higher ranks.

Gain (information retrieval)

The **gain**, also called **improvement over random** can be specified for a classifier and is an important measure to describe the performance of it.

Definition

In the following a random classifier is defined such that it randomly predicts the same amount of either class.

The gain is defined as described in the following:

Gain in Precision

The random precision of a classifier is defined as

$$r = \frac{TP + FN}{TP + TN + FP + FN} = \frac{Positives}{N}$$

where TP, TN, FP and FN are the numbers of true positives, true negatives, false positives and false negatives respectively, positives is the number of positive instances in the target dataset and N is the size of the dataset.

The random precision defines the lowest baseline of a classifier.

And **Gain** is defined as

$$G = \frac{precision}{r}$$

which gives a factor by which a classifier is better when compared to its random counterpart. A Gain of 1 would indicate a classifier that is not better than random. The larger the gain, the better.

Gain in Overall Accuracy

The accuracy of a classifier in general is defined as

$$Acc = \frac{TP + TN}{TP + TN + FP + FN} = \frac{Corrects}{N}$$

Here, the random accuracy of a classifier can be defined as

$$r = \left(\frac{Positives}{N}\right)^2 + \left(\frac{Negatives}{N}\right)^2 = f(Positives)^2 + f(Negatives)^2$$

f(Positives) and f(Negatives) is the fraction of positive and negative classes in the dataset.

And again **gain** is

$$G = \frac{Acc}{r}$$

This time the gain is measured not only with respect to the prediction of a so called positive class, but with respect to the overall classifier ability to distinguish the two equally important classes.

Application

In Bioinformatics as an example, the gain is measured for methods that predict residue contacts in proteins.

Harvester42

The **Harvester42** is a meta search engine engine project hosted at KIT Karlsruhe Institute of Technology.

Harvester42 queries over 30 major search engines in parallel and presents a large result page with the individual search engine results. The name "Harvester42" originates from the Bioinformatic Harvester project, a meta search engine for genes and proteins from several species. Harvester42 is based on the same method integrating several distributed data sources.

Introduction

Currently no search engine covers the entire internet. Often a search across several search engines is more successful than a single search engine search. Meta search engines usually integrate several search engines with complementing indices. Harvester42 covers popularity based search engines (Google), proprietary search algorithms (cuil), open source search engines (YaCY), product search engines (Directindustry, Google products), Video search (youtube) and powerful meta search engines (Polymeta, Allplus, Mamma).

Harvester42 integrated search engines

- Clusty · Exalead · Entrez · EB-eye · YIF · nextbio · Bioinformatic Harvester · Google News · addictomatic · technology-review · 123people · eurekalert · ScienceDaily · physorg · google books · Google Scholar · GoPubmed · Scirus.com · Sciencenet · XClustering · iSEEK · YouTube · ovguide · Allplus · Stumbleupon · Delicious · YaCy · Cuil · Jove · Deepdyve · CiteULike · Twenga · Thefind · Ebay · Google patents · XCluster ...and several more.

Harvester42 technical details

All search results are presented in so called iframes without any modification of the original search engine result. Each iframe can be manipulated individually.

Chapter-11

Nearest Neighbor Search and Multi-Document Summarization

Nearest neighbor search

Nearest neighbor search (**NNS**), also known as **proximity search**, **similarity search** or **closest point search**, is an optimization problem for finding closest points in metric spaces. The problem is: given a set S of points in a metric space M and a query point $q \in M$, find the closest point in S to q. In many cases, M is taken to be d-dimensional Euclidean space and distance is measured by Euclidean distance or Manhattan distance.

Donald Knuth in vol. 3 of *The Art of Computer Programming* (1973) called it the **post-office problem**, referring to an application of assigning to a residence the nearest post office.

Applications

The nearest neighbor search problem arises in numerous fields of application, including:

- Pattern recognition - in particular for optical character recognition
- Statistical classification
- Computer vision
- Databases - e.g. content-based image retrieval
- Coding theory
- Data compression
- Recommendation systems
- Internet marketing
- DNA sequencing
- Spell checking - suggesting correct spelling
- Plagiarism detection
- Contact searching algorithms in FEA
- Similarity scores for predicting career paths of professional athletes.

- Cluster analysis - assignment of a set of observations into subsets (called clusters) so that observations in the same cluster are similar in some sense, usually based on Euclidean distance

Methods

Various solutions to the NNS problem have been proposed. The quality and usefulness of the algorithms are determined by the time complexity of queries as well as the space complexity of any search data structures that must be maintained. The informal observation usually referred to as the curse of dimensionality states that there is no general-purpose exact solution for NNS in high-dimensional Euclidean space using polynomial preprocessing and polylogarithmic search time.

Linear search

The simplest solution to the NNS problem is to compute the distance from the query point to every other point in the database, keeping track of the "best so far". This algorithm, sometimes referred to as the naive approach, has a running time of $O(Nd)$ where N is the cardinality of S and d is the dimensionality of M. There are no search data structures to maintain, so linear search has no space complexity beyond the storage of the database. Surprisingly, naive search outperforms space partitioning approaches on higher dimensional spaces.

Space partitioning

Starting from 1970s branch and bound methodology was applied to the problem. In the case of Euclidean space this approach is known as spatial index or spatial access methods. Several space-partitioning methods have been developed for solving the NNS problem. Perhaps the simplest is the kd-tree, which iteratively bisects the search space into two regions containing half of the points of the parent region. Queries are performed via traversal of the tree from the root to a leaf by evaluating the query point at each split. Depending on the distance specified in the query, also neighboring branches that might contain hits need to be evaluated. For constant dimension query time, average complexity is $O(\log N)$ in the case of randomly distributed points, worst case complexity analyses have been performed. Alternatively the R-tree data structure was designed to support nearest neighbor search in dynamic context, as it has efficient algorithms for insertions and deletions.

In case of general metric space branch and bound approach is known under the name of metric trees. Particular examples include VP-tree and Bk-tree.

Using a set of points taken from a 3-dimensional space and put into a BSP tree, and given a query point taken from the same space, a possible solution to the problem of finding the nearest point-cloud point to the query point is given in the following description of an algorithm. (Strictly speaking, no such point may exist, because it may not be unique. But in practice, usually we only care about finding any one of the subset of all point-cloud points that exist at the shortest distance to a given query point.) The idea is, for each branching of the tree, guess that the closest point in the cloud resides in the half-space containing the query point. This may not be the case, but it is a good heuristic. After having recursively gone through all the trouble of solving the problem for the guessed half-space, now compare the distance returned by this result with the shortest distance from the query point to the partitioning plane. This latter distance is that between the query point and the closest possible point that could exist in the half-space not searched. If this distance is greater than that returned in the earlier result, then clearly there is no need to search the other half-space. If there is such a need, then you must go through the trouble of solving the problem for the other half space, and then compare its result to the former result, and then return the proper result. The performance of this algorithm is nearer to logarithmic time than linear time when the query point is near the cloud, because as the distance between the query point and the closest point-cloud point nears zero, the algorithm needs only perform a look-up using the query point as a key to get the correct result.

Locality sensitive hashing

Locality sensitive hashing (LSH) is a technique for grouping points in space into 'buckets' based on some distance metric operating on the points. Points that are close to each other under the chosen metric are mapped to the same bucket with high probability.

Nearest neighbor search in spaces with small intrinsic dimension

The cover tree has a theoretical bound that is based on the dataset's doubling constant. The bound on search time is $O(c^{12} \log n)$ where c is the expansion constant of the dataset.

Vector Approximation Files

In high dimensional spaces tree indexing structures become useless because an increasing percentage of the nodes need to be examined anyway. To speed up linear search, a compressed version of the feature vectors stored in RAM is used to prefilter the datasets in a first run. The final candidates are determined in a second stage using the uncompressed data from the disk for distance calculation.

Variants

There are numerous variants of the NNS problem and the two most well-known are the *k*-nearest neighbor search and the ε-approximate nearest neighbor search.

K-nearest neighbor

k-nearest neighbor search identifies the top *k* nearest neighbors to the query. This technique is commonly used in predictive analytics to estimate or classify a point based on the consensus of its neighbors. *k*-nearest neighbor graphs are graphs in which every point is connected to its *k* nearest neighbors.

Approximate nearest neighbor

In some applications it may be acceptable to retrieve a "good guess" of the nearest neighbor. In those cases, we can use an algorithm which doesn't guarantee to return the actual nearest neighbor in every case, in return for improved speed or memory savings. Often such an algorithm will find the nearest neighbor in a majority of cases, but this depends strongly on the dataset being queried.

Algorithms which support the approximate nearest neighbor search include Best Bin First and Balanced Box-Decomposition Tree based search.

ε-approximate nearest neighbor search is becoming an increasingly popular tool for fighting the curse of dimensionality.

Nearest neighbor distance ratio

Nearest neighbor distance ratio do not apply the threshold on the direct distance from the original point to the challenger neighbor but on a ratio of it depending on the distance to the previous neighbor. It is used in CBIR to retrieve pictures through a "query by example" using the similarity between local features. More generally it is involved in several matching problems.

All nearest neighbors

For some applications (e.g. entropy estimation), we may have *N* data-points and wish to know which is the nearest neighbor *for every one of those N points*. This could of course be achieved by running a nearest-neighbor search once for every point, but an improved strategy would be an algorithm that exploits the information redundancy between these *N* queries to produce a more efficient search. As a simple example: when we find the distance from point *X* to point *Y*, that also tells us the distance from point *Y* to point *X*, so the same calculation can be reused in two different queries.

Given a fixed dimension, a semi-definite positive norm (thereby including every L^p norm), and *n* points in this space, the *m* nearest neighbours of every point can be found in $O(mn \log n)$ time.

Multi-document summarization

Multi-document summarization is an automatic procedure aimed at extraction of information from multiple texts written about the same topic. Resulting summary report allows individual users, so as professional information consumers, to quickly familiarize themselves with information contained in a large cluster of documents. In such a way, multi-document summarization systems are complementing the news aggregators performing the next step down the road of coping with information overload.

Key benefits

Multi-document summarization creates information reports that are both concise and comprehensive. With different opinions being put together & outlined, every topic is described from multiple perspectives within a single document. While the goal of a brief summary is to simplify information search and cut the time by pointing to the most relevant source documents, comprehensive multi-document summary should itself contain the required information, hence limiting the need for accessing original files to cases when refinement is required. Automatic summaries present information extracted from multiple sources algorithmically, without any editorial touch or subjective human intervention, thus making it completely unbiased.

Technological challenges

The multi-document summarization task has turned out to be much more complex than summarizing a single document, even a very large one. This difficulty arises from inevitable thematic diversity within a large set of documents. A good summarization technology aims to combine the main themes with completeness, readability, and conciseness. Document Understanding Conferences, conducted annually by NIST, have developed sophisticated evaluation criteria for techniques accepting the multi-document summarization challenge.

An ideal multi-document summarization system does not simply shorten the source texts but presents information organized around the key aspects to represent a wider diversity of views on the topic. When such quality is achieved, an automatic multi-document summary is perceived more like an overview of a given topic. The latter implies that such text compilations should also meet other basic requirements for an overview text compiled by a human. The multi-document summary quality criteria are as follows:

- clear structure, including an outline of the main content, from which it is easy to navigate to the full text sections
- text within sections is divided into meaningful paragraphs
- gradual transition from more general to more specific thematic aspects
- good readability

The latter point deserves additional note - special care is taken in order to ensure that the automatic overview shows:

- no paper-unrelated "information noise" from the respective documents (e.g., web pages)
- no dangling references to what is not mentioned or explained in the overview
- no text breaks across a sentence
- no semantic redundancy.

Real-life systems

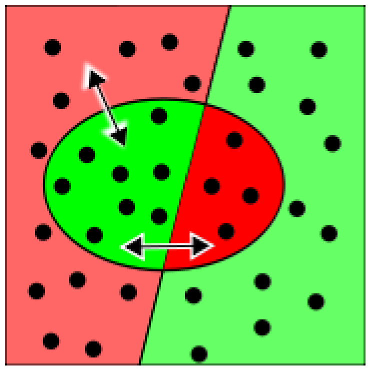
The multi-document summarization technology is now coming of age - a view supported by a choice of advanced web-based systems that are currently available.

- Ultimate Research Assistant - The Ultimate Research Assistant performs text mining on Internet search results to help summarize and organize them and make it easier for the user to perform online research. Specific text mining techniques used by the tool include concept extraction, text summarization, hierarchical concept clustering (e.g., automated taxonomy generation), and various visualization techniques, including tag clouds and mind maps. To use this tool, the user types in the name of a topic, and the tool will search the web for highly relevant resources, and organize the search results into a rich, easy-to-understand research report.
- iResearch Reporter - Commercial Text Extraction and Text Summarization system, free demo site accepts user-entered query, passes it on to Google search engine, retrieves multiple relevant documents, produces categorized, easily-readable natural language summary reports covering multiple documents in retrieved set, all extracts linked to original documents on the Web, post-processing, entity extraction, event and relationship extraction, text extraction, extract clustering, linguistic analysis, multi-document, full text, natural language processing, categorization rules, clustering, linguistic analysis, text summary construction tool set.
- Newsblaster is a system that helps users find the news that is of the most interest to them. The system automatically collects, clusters, categorizes, and summarizes news from several sites on the web (CNN, Reuters, Fox News, etc.) on a daily basis, and it provides users a user-friendly interface to browse the results.
- NewsInEssence may be used to retrieve and summarize a cluster of articles from the web. It can start from a URL and retrieve documents that are similar, or it can retrieve documents that match a given set of keywords. NewsInEssence also downloads hundreds of news articles daily and produces news clusters from them.
- NewsFeed Researcher is a news portal performing continuous automatic summarization of documents initially clustered by the news aggregators (e.g., Google News). NewsFeed Researcher is backed by the free online engine covering major events related to business, technology, U.S. and international news. This tool is also available in the on-demand mode allowing a user to build a summary on any selected topic.
- Shablast is a universal search engine that produces multi-document summaries from the top 50 results returned by Microsoft's Bing search engine for a set of keywords.

As the quality multi-document summaries are becoming to resemble the overviews written by a human, one cannot exclude that their use of extracted text snippets can one day face some copyright issues. This potential case should be regarded from the point of the fair use copyright concept.

Chapter-12

Precision and Recall



Recall and precision depend on the outcome (oval) of a query and its relation to all relevant documents (left) and the non-relevant documents (right). The more correct results (green), the better. **Precision**: horizontal arrow. **Recall**: diagonal arrow.

Precision and **recall** are two widely used metrics for evaluating the correctness of a pattern recognition algorithm. They can be seen as extended versions of *accuracy*, a simple metric that computes the fraction of instances for which the correct result is returned.

When using precision and recall, the set of possible labels for a given instance is divided into two subsets, one of which is considered "relevant" for the purposes of the metric. Recall is then computed as the fraction of correct instances among all instances that *actually* belong to the relevant subset, while precision is the fraction of correct instances among those that the algorithm *believes* to belong to the relevant subset.

Precision can be seen as a measure of exactness or fidelity, whereas recall is a measure of completeness.

Introduction

As an example, in an information retrieval scenario, the instances are documents and the task is to return a set of relevant documents given a search term; or equivalently, to assign each document to one of two categories, "relevant" and "not relevant". In this case, the "relevant" documents are simply those that belong to the "relevant" category. Recall is defined as the *number of relevant documents* retrieved by a search *divided by the total number of existing relevant documents*, while precision is defined as the *number of relevant documents* retrieved by a search *divided by the total number of documents retrieved* by that search.

In a classification task, the precision for a class is the *number of **true positives*** (i.e. the *number of items correctly labeled as belonging to the positive class*) *divided by the total number of elements labeled as belonging to the positive class* (i.e. the sum of true positives and **false positives**, which are items incorrectly labeled as belonging to the class). Recall in this context is defined as the *number of true positives divided by the total number of elements that actually belong to the positive class* (i.e. the sum of true positives and **false negatives**, which are items which were not labeled as belonging to the positive class but should have been).

In information retrieval, a perfect precision score of 1.0 means that every result retrieved by a search was relevant (but says nothing about whether all relevant documents were retrieved) whereas a perfect recall score of 1.0 means that all relevant documents were retrieved by the search (but says nothing about how many irrelevant documents were also retrieved).

In a classification task, a precision score of 1.0 for a class C means that every item labeled as belonging to class C does indeed belong to class C (but says nothing about the number of items from class C that were not labeled correctly) whereas a recall of 1.0 means that every item from class C was labeled as belonging to class C (but says nothing about how many other items were incorrectly also labeled as belonging to class C).

Often, there is an inverse relationship between precision and recall, where it is possible to increase one at the cost of reducing the other. For example, an information retrieval system (such as a search engine) can often increase its recall by retrieving more documents, at the cost of increasing number of irrelevant documents retrieved (decreasing precision). Similarly, a classification system for deciding whether or not, say, a fruit is an orange, can achieve high precision by only classifying fruits with the exact right shape and color as oranges, but at the cost of low recall due to the number of *false negatives* from oranges that did not quite match the specification.

Usually, precision and recall scores are not discussed in isolation. Instead, either values for one measure are compared for a fixed level at the other measure (e.g. *precision at a recall level of 0.75*) or both are combined into a single measure, such as the **F-measure**, which is the *weighted harmonic mean of precision and recall*, or the Matthews correlation coefficient.

Definition (information retrieval context)

In information retrieval contexts, precision and recall are defined in terms of a set of **retrieved documents** (e.g. the list of documents produced by a web search engine for a query) and a set of **relevant documents** (e.g. the list of all documents on the internet that are relevant for a certain topic).

Precision

In the field of information retrieval, **precision** is the fraction of retrieved documents that are relevant to the search:

$$\text{precision} = \frac{|\{\text{relevant documents}\} \cap \{\text{retrieved documents}\}|}{|\{\text{retrieved documents}\}|}$$

Precision takes all retrieved documents into account, but it can also be evaluated at a given cut-off rank, considering only the topmost results returned by the system. This measure is called **precision at n** or **P@n**.

For example for a text search on a set of documents precision is the number of correct results divided by the number of all returned results.

Precision is also used with recall, the percent of *all* relevant documents that is returned by the search. The two measures are sometimes used together in the F1 Score (or f-measure) to provide a single measurement for a system.

Note that the meaning and usage of "precision" in the field of Information Retrieval differs from the definition of accuracy and precision within other branches of science and technology.

Recall

Recall in Information Retrieval is the fraction of the documents that are relevant to the query that are successfully retrieved.

$$\text{recall} = \frac{|\{\text{relevant documents}\} \cap \{\text{retrieved documents}\}|}{|\{\text{relevant documents}\}|}$$

For example for text search on a set of documents recall is the number of correct results divided by the number of results that should have been returned

In binary classification, recall is called sensitivity. So it can be looked at as the probability that a relevant document is retrieved by the query.

It is trivial to achieve recall of 100% by returning all documents in response to any query. Therefore recall alone is not enough but one needs to measure the number of non-relevant documents also, for example by computing the precision.

Definition (classification context)

In the context of classification tasks, the terms **true positives**, **true negatives**, **false positives** and **false negatives** are used to compare the given classification of an item (the class label assigned to the item by a classifier) with the desired correct classification (the class the item actually belongs to). This is illustrated by the table below:

		correct result / classification	
		E1	**E2**
obtained result / classification	**E1**	**tp** (true positive)	**fp** (false positive)
	E2	**fn** (false negative)	**tn** (true negative)

Precision and recall are then defined as:

$$\text{Precision} = \frac{tp}{tp + fp}$$
$$\text{Recall} = \frac{tp}{tp + fn}$$

Recall in this context is also referred to as the True Positive Rate, other related measures used in classification include True Negative Rate and Accuracy:. True Negative Rate is also called Specificity.

$$\text{True Negative Rate} = \frac{tn}{tn + fp}$$
$$\text{Accuracy} = \frac{tp + tn}{tp + tn + fp + fn}$$

Probabilistic interpretation

It is possible to interpret precision and recall not as ratios but as probabilities:

- **Precision** is the probability that a (randomly selected) retrieved document is relevant.
- **Recall** is the probability that a (randomly selected) relevant document is retrieved in a search.

Note that the random selection refers to a uniform distribution over the appropriate pool of documents; i.e. by **randomly selected retrieved document**, we mean selecting a document from the set of retrieved documents in a random fashion. The random selection should be such that all documents in the set are equally likely to be selected.

Note that, in a typical classification system, the probability that a retrieved document is relevant depends on the document. The above interpretation extends to that scenario also (needs explanation).

Another interpretation for precision and retrieval is as follows. Precision is the average probability of relevant retrieval. Recall is the average probability of complete retrieval. Here we average over multiple retrieval queries.

F-measure

A measure that combines precision and recall is the harmonic mean of precision and recall, the traditional F-measure or balanced F-score:

$$F = 2 \cdot \frac{\mathrm{precision} \cdot \mathrm{recall}}{\mathrm{precision} + \mathrm{recall}}$$

This is also known as the F_1 measure, because recall and precision are evenly weighted.

It is a special case of the general F_β measure (for non-negative real values of β):

$$F_\beta = (1 + \beta^2) \cdot \frac{\mathrm{precision} \cdot \mathrm{recall}}{\beta^2 \cdot \mathrm{precision} + \mathrm{recall}}$$

Two other commonly used F measures are the F_2 measure, which weights recall higher than precision, and the $F_{0.5}$ measure, which puts more emphasis on precision than recall.

The F-measure was derived by van Rijsbergen (1979) so that F_β "measures the effectiveness of retrieval with respect to a user who attaches β times as much importance to recall as precision". It is based on van Rijsbergen's effectiveness measure $E = 1 - (1/(\alpha/P + (1 - \alpha)/R))$. Their relationship is $F_\beta = 1 - E$ where $\alpha = 1/(\beta^2 + 1)$.

Chapter-13

Question Answering

In information retrieval and natural language processing (NLP), **question answering** (**QA**) is the task of automatically answering a question posed in natural language. To find the answer to a question, a QA computer program may use either a pre-structured database or a collection of natural language documents (a text corpus such as the World Wide Web or some local collection).

QA research attempts to deal with a wide range of question types including: fact, list, definition, *How*, *Why*, hypothetical, semantically constrained, and cross-lingual questions. Search collections vary from small local document collections, to internal organization documents, to compiled newswire reports, to the World Wide Web.

- *Closed-domain* question answering deals with questions under a specific domain (for example, medicine or automotive maintenance), and can be seen as an easier task because NLP systems can exploit domain-specific knowledge frequently formalized in ontologies. Alternatively, *closed-domain* might refer to a situation where only a limited type of questions are accepted, such as questions asking for descriptive rather than procedural information.
- *Open-domain* question answering deals with questions about nearly anything, and can only rely on general ontologies and world knowledge. On the other hand, these systems usually have much more data available from which to extract the answer.

QA is regarded as requiring more complex natural language processing (NLP) techniques than other types of information retrieval such as document retrieval, thus natural language search engines are sometimes regarded as the next step beyond current search engines.

Architecture

The first QA systems were developed in the 1960s and they were basically natural-language interfaces to expert systems that were tailored to specific domains. In contrast, current QA systems use text documents as their underlying knowledge source and combine various natural language processing techniques to search for the answers.

Current QA systems typically include a **question classifier** module that determines the type of question and the type of answer. After the question is analysed, the system typically uses several modules that apply increasingly complex NLP techniques on a gradually reduced amount of text. Thus, a **document retrieval module** uses search engines to identify the documents or paragraphs in the document set that are likely to contain the answer. Subsequently a **filter** preselects small text fragments that contain strings of the same type as the expected answer. For example, if the question is "Who invented Penicillin" the filter returns text that contain names of people. Finally, an **answer extraction** module looks for further clues in the text to determine if the answer candidate can indeed answer the question.

Question answering methods

QA is very dependent on a good search corpus - for without documents containing the answer, there is little any QA system can do. It thus makes sense that larger collection sizes generally lend well to better QA performance, unless the question domain is orthogonal to the collection. The notion of data redundancy in massive collections, such as the web, means that nuggets of information are likely to be phrased in many different ways in differing contexts and documents , leading to two benefits:

> **(1)** By having the right information appear in many forms, the burden on the QA system to perform complex NLP techniques to understand the text is lessened.
> **(2)** Correct answers can be filtered from false positives by relying on the correct answer to appear more times in the documents than instances of incorrect ones.

Shallow

Some methods of QA use keyword-based techniques to locate interesting passages and sentences from the retrieved documents and then filter based on the presence of the desired answer type within that candidate text. Ranking is then done based on syntactic features such as word order or location and similarity to query.

When using massive collections with good data redundancy, some systems use templates to find the final answer in the hope that the answer is just a reformulation of the question. If you posed the question "What is a dog?", the system would detect the substring "What is a X" and look for documents which start with "X is a Y". This often works well on simple "factoid" questions seeking factual tidbits of information such as names, dates, locations, and quantities.

Deep

In cases where simple question reformulation or keyword techniques do not suffice, syntactic, semantic and contextual processing must be performed to extract or construct the answer. Such techniques include named-entity recognition, relation detection, coreference resolution, syntactic alternations, word sense disambiguation, logic form transformation, logical inferences (abduction) and commonsense reasoning, temporal or spatial reasoning and so on. Systems also very often utilize world knowledge that can be found in ontologies, such as WordNet or the Suggested Upper Merged Ontology (SUMO), to augment the available reasoning resources through semantic connections and definitions. An example of semantic connection is the use of VerbNet's thematic roles to model verb arguments. The Conceptual Graph Formalism (CGF), a semantic knowledge representation methodology, is used to model knowledge in documents and questions, and then compare them using the projection algorithm defined in CGF.

More difficult queries such as *Why* or *How* questions, hypothetical postulations, spatially or temporally constrained questions, dialog queries, badly worded or ambiguous questions will all need these types of deeper understanding of the question. Complex or ambiguous document passages likewise need more NLP techniques applied to understand the text.

Statistical QA, which introduces statistical question processing and answer extraction modules, is also growing in popularity in the research community. Many of the lower-level NLP tools used, such as part-of-speech tagging, parsing, named-entity detection, sentence boundary detection, and document retrieval, are already available as probabilistic applications.

AQ (Answer Questioning) Methodology; introduces a working cycle to the QA methods. This method may be used in conjunction with any of the known or newly founded methods. AQ Method may be used upon perception of a posed question or answer. The means by which it is utilized can be manipulated beyond its primary usage; however, the primary usage is taking an answer and questioning it turning that very answer into a question. Example; A"I like sushi." Q"(Why do) I like sushi(?)" A"The flavor." Q"(What about) the flavor of sushi (do) I like?" Inadvertently, this may unveil different methods of thinking and perception as well. While most would agree that this seems to be the endall stratagem, it is only a starting point with endless possibilities. Any number of question methods may be used to derive the number of WHY as in, $A = \infty(Q)$, the answer may yield any number of questions to be asked; thereby unveiling an ongoing process constantly being reborn into the research being performed. The QA methodology utilizes just the opposite where, $1(Q) = (\infty(A)-\infty) = 1(A)$, supposedly there is only one true answer in reality everything else is perception or plausibility. Utilized alongside other forms of communication; debate may be greatly improved. Even this methodology should be questioned.

Issues

In 2002 a group of researchers wrote a roadmap of research in question answering. The following issues were identified.

Question classes

Different types of questions (e.g., "What is the capital of Lichtenstein?" vs. "Why does a rainbow form?" vs. "Did Marilyn Monroe and Cary Grant ever appear in a movie together?") require the use of different strategies to find the answer. Question classes are arranged hierarchically in taxonomies.

Question processing

The same information request can be expressed in various ways, some interrogative ("Who is the president of the United States?") and some assertive ("Tell me the name of the president of the United States."). A semantic model of question understanding and processing would recognize equivalent questions, regardless of how they are presented. This model would enable the translation of a complex question into a series of simpler questions, would identify ambiguities and treat them in context or by interactive clarification.

Context and QA

Questions are usually asked within a context and answers are provided within that specific context. The context can be used to clarify a question, resolve ambiguities or keep track of an investigation performed through a series of questions. (For example, the question, "Why did Joe Biden visit Iraq in January 2010?" might be asking why Vice President Biden visited and not President Obama, why he went to Iraq and not Afghanistan or some other country, why he went in January 2010 and not before or after, or what Biden was hoping to accomplish with his visit. If the question is one of a series of related questions, the previous questions and their answers might shed light on the questioner's intent.)

Data sources for QA

Before a question can be answered, it must be known what knowledge sources are available and relevant. If the answer to a question is not present in the data sources, no matter how well the question processing, information retrieval and answer extraction is performed, a correct result will not be obtained.

Answer extraction

Answer extraction depends on the complexity of the question, on the answer type provided by question processing, on the actual data where the answer is searched, on the search method and on the question focus and context.

Answer formulation

The result of a QA system should be presented in a way as natural as possible. In some cases, simple extraction is sufficient. For example, when the question classification indicates that the answer type is a name (of a person, organization, shop or disease, etc.), a quantity (monetary value, length, size, distance, etc.) or a date (e.g. the answer to the question, "On what day did Christmas fall in 1989?") the extraction of a single datum is sufficient. For other cases, the presentation of the answer may require the use of fusion techniques that combine the partial answers from multiple documents.

Real time question answering

There is need for developing Q&A systems that are capable of extracting answers from large data sets in several seconds, regardless of the complexity of the question, the size and multitude of the data sources or the ambiguity of the question.

Multilingual (or cross-lingual) question answering

The ability to answer a question posed in one language using an answer corpus in another language (or even several). This allows users to consult information that they cannot use directly.

Interactive QA

It is often the case that the information need is not well captured by a QA system, as the question processing part may fail to classify properly the question or the information needed for extracting and generating the answer is not easily retrieved. In such cases, the questioner might want not only to reformulate the question, but to have a dialogue with the system. (For example, the system might ask for a clarification of what sense a word is being used, or what type of information is being asked for.)

Advanced reasoning for QA

More sophisticated questioners expect answers that are outside the scope of written texts or structured databases. To upgrade a QA system with such capabilities, it would be necessary to integrate reasoning components operating on a variety of knowledge bases, encoding world knowledge and common-sense reasoning mechanisms, as well as knowledge specific to a variety of domains.

User profiling for QA

The user profile captures data about the questioner, comprising context data, domain of interest, reasoning schemes frequently used by the questioner, common ground established within different dialogues between the system and the user, and so forth. The profile may be represented as a predefined template, where each template slot represents a different profile feature. Profile templates may be nested one within another.

History

Some of the early AI systems were question answering systems. Two of the most famous QA systems of that time are BASEBALL and LUNAR, both of which were developed in the 1960s. BASEBALL answered questions about the US baseball league over a period of one year. LUNAR, in turn, answered questions about the geological analysis of rocks returned by the Apollo moon missions. Both QA systems were very effective in their chosen domains. In fact, LUNAR was demonstrated at a lunar science convention in 1971 and it was able to answer 90% of the questions in its domain posed by people untrained on the system. Further restricted-domain QA systems were developed in the following years. The common feature of all these systems is that they had a core database or knowledge system that was hand-written by experts of the chosen domain.

Some of the early AI systems included question-answering abilities. Two of the most famous early systems are SHRDLU and ELIZA. SHRDLU simulated the operation of a robot in a toy world (the "blocks world"), and it offered the possibility to ask the robot questions about the state of the world. Again, the strength of this system was the choice of a very specific domain and a very simple world with rules of physics that were easy to encode in a computer program. ELIZA, in contrast, simulated a conversation with a psychologist. ELIZA was able to converse on any topic by resorting to very simple rules that detected important words in the person's input. It had a very rudimentary way to answer questions, and on its own it led to a series of chatterbots such as the ones that participate in the annual Loebner prize.

The 1970s and 1980s saw the development of comprehensive theories in computational linguistics, which led to the development of ambitious projects in text comprehension and question answering. One example of such a system was the Unix Consultant (UC), a system that answered questions pertaining to the Unix operating system. The system had a comprehensive hand-crafted knowledge base of its domain, and it aimed at phrasing the answer to accommodate various types of users. Another project was LILOG, a text-understanding system that operated on the domain of tourism information in a German city. The systems developed in the UC and LILOG projects never went past the stage of simple demonstrations, but they helped the development of theories on computational linguistics and reasoning.

In the late 1990s the annual Text Retrieval Conference (TREC) included a question-answering track which has been running until the present. Systems participating in this competition were expected to answer questions on any topic by searching a corpus of text that varied from year to year. This competition fostered research and development in open-domain text-based question answering. The best system of the 2004 competition achieved 77% correct fact-based questions.

In 2007 the annual TREC included a blog data corpus for question answering. The blog data corpus contained both "clean" English as well as noisy text that includes badly formed English and spam. The introduction of noisy text moved the question answering to a more realistic setting. Real-life data is inherently noisy as people are less careful when writing in spontaneous media like blogs. In earlier years the TREC data corpus consisted of only newswire data that was very clean.

An increasing number of systems include the World Wide Web as one more corpus of text. Currently there is an increasing interest in the integration of question answering with web search. Ask.com is an early example of a such a system, which was followed in subsequent years by other natural language search engines. Google and Microsoft have also started to integrate question-answering facilities in their search engines. However, these tools mostly work by using shallow methods, as described above — thus returning a list of documents, usually with an excerpt containing the probable answer highlighted, plus some context. More recently, tools to actually reason and compute the answer were developed, such as True Knowledge and Wolfram Alpha. Furthermore, highly-specialized natural language question-answering engines, such as EAGLi for health and life scientists, have been made available.

The Future of Question Answering

QA systems have been extended in recent years to explore critical new scientific and practical dimensions For example, systems have been developed to automatically answer temporal and geospatial questions, definitional questions, biographical questions, multilingual questions, and questions from multimedia (e.g., audio, imagery, video).

Additional aspects such as interactivity (often required for clarification of questions or answers), answer reuse, and knowledge representation and reasoning to support question answering have been explored. Future research may explore what kinds of questions can be asked and answered about social media, including sentiment analysis.

Chapter-14

Query Expansion, Relevance (information retrieval) and Relevance Feedback

Query expansion

Query expansion (**QE**) is the process of reformulating a seed query to improve retrieval performance in information retrieval operations. In the context of web search engines, query expansion involves evaluating a user's input (what words were typed into the search query area, and sometimes other types of data) and expanding the search query to match additional documents. Query expansion involves techniques such as:

- Finding synonyms of words, and searching for the synonyms as well
- Finding all the various morphological forms of words by stemming each word in the search query
- Fixing spelling errors and automatically searching for the corrected form or suggesting it in the results
- Re-weighting the terms in the original query

Query expansion is a methodology studied in the field of computer science, particularly within the realm of natural language processing and information retrieval.

Precision and recall tradeoffs

Search engines invoke query expansion to increase the quality of user search results. It is assumed that users do not always formulate search queries using the best terms. Best in this case may be because the database does not contain the user entered terms.

By stemming a user-entered term, more documents are matched, as the alternate word forms for a user entered term are matched as well, increasing the total recall. This comes at the expense of reducing the precision. By expanding a search query to search for the synonyms of a user entered term, the recall is also increased at the expense of precision. This is due to the nature of the equation of how precision is calculated, in that a larger recall implicitly causes a decrease in precision, given that factors of recall are part of the denominator. It is also inferred that a larger recall negatively impacts overall search result quality, given that many users do not want more results to comb through, regardless of the precision.

The goal of query expansion in this regard is by increasing recall, precision can potentially increase (rather than decrease as mathematically equated), by including in the result set pages which are more relevant (of higher quality), or at least equally relevant. Pages which would not be included in the result set, which have the potential to be more relevant to the user's desired query, are included, and without query expansion would not have, regardless of relevance. At the same time, many of the current commercial search engines use word frequency (Tf-idf) to assist in ranking. By ranking the occurrences of both the user entered words and synonyms and alternate morphological forms, documents with a higher density (high frequency and close proximity) tend to migrate higher up in the search results, leading to a higher quality of the search results near the top of the results, despite the larger recall.

This tradeoff is one of the defining problems in query expansion, regarding whether it is worthwhile to perform given the questionable effects on precision and recall. Critics state one of the problems is that the dictionaries and thesauri, and the stemming algorithm, are driven by human bias and while this is implicitly handled by the query expansion algorithm, this explicitly affects the results in a non-automated manner (similar to how statisticians can 'lie' with statistics). Other critics point out potential for corporate influence on the dictionaries, promoting advertising of online web pages in the case of web search engines.

Relevance

In information science and information retrieval, **relevance** denotes how well a retrieved document or set of documents meets the information need of the user.

Types

Relevance most commonly refers to *topical* relevance or *aboutness*, i.e. to what extent the topic of a result matches the topic of the query or information need. Relevance can also be interpreted more broadly, referring to generally how "good" a retrieved result is with regard to the information need. The latter definition of relevance, sometimes referred to as *user* relevance, encompasses *topical* relevance and possibly other concerns of the user such as timeliness, authority or novelty of the result.

History

The concern with the problem of finding relevant information dates back at least to the first publication of scientific journals in 17th Century.

The formal study of relevance began in the 20th Century with the study of what would later be called bibliometrics. In the 1930s and 1940s, S. C. Bradford used the term "relevant" to characterize articles relevant to a subject (cf., Bradford's law). In the 1950s, the first information retrieval systems emerged, and researchers noted the retrieval of irrelevant articles as a significant concern. In 1958, B. C. Vickery made the concept of relevance explicit in an address at the International Conference on Scientific Information.

Since 1958, information scientists have explored and debated definitions of relevance. A particular focus of the debate was the distinction between "relevance to a subject" or "topical relevance" and "user relevance".

Evaluation

The information retrieval community has emphasized the use of test collections and benchmark tasks to measure topical relevance, starting with the Cranfield Experiments of the early 1960s and culminating in the TREC evaluations that continue to this day as the main evaluation framework for information retrieval research.

In order to evaluate how well an information retrieval system retrieved topically relevant results, the relevance of retrieved results must be quantified. In Cranfield-style evaluations, this typically involves assigning a *relevance level* to each retrieved result, a process known as *relevance assessment*. Relevance levels can be binary (indicating a result is relevant or that it is not relevant), or graded (indicating results have a varying degree of match between the topic of the result and the information need). Once relevance levels have been assigned to the retrieved results, information retrieval performance measures can be used to assess the quality of a retrieval system's output.

In contrast to this focus solely on topical relevance, the information science community has emphasized user studies that consider user relevance. These studies often focus on aspects of human-computer interaction.

Clustering and relevance

The cluster hypothesis, proposed by C. J. van Rijsbergen in 1979, asserts that two documents that are similar to each other have a high likelihood of being relevant to the same information need. With respect to the embedding similarity space, the cluster hypothesis can be interpreted globally or locally. The global interpretation assumes that there exist some fixed set of underlying topics derived from inter-document similarity. These global clusters or their representatives can then be used to relate relevance of two documents (e.g. two documents in the same cluster should both be relevant to the same request). Methods in this spirit include,

- cluster-based information retrieval
- cluster-based document expansion such as latent semantic analysis or its language modeling equivalents. It is important to ensure that clusters – either in isolation or combination – successfully model the set of possible relevant documents.

A second interpretation, most notably advanced by Ellen Voorhees, focuses on the local relationships between documents. The local interpretation avoids having to model the number or size of clusters in the collection and allow relevance at multiple scales. Methods in this spirit include,

- multiple cluster retrieval
- spreading activation and relevance propagation methods
- local document expansion
- score regularization

Local methods require an accurate and appropriate document similarity measure.

Epistemological issues

Are users best at evaluating the relevance of a given document, or is it better to use experts? Most research about relevance in information retrieval in recent years have implicitly assumed that the users' evaluation of the output a given system should be used to increase "relevance" output. An alternative strategy would be to use journal impact factor to rank output and thus base relevance on expert evaluations. Other strategies, such as including diversity of the search results, may be used as well. The important thing to recognize is, however, that relevance is fundamentally a question of epistemology, not psychology. (Peoples' psychology reflects certain epistemological influences).

Relevance feedback

Relevance feedback is a feature of some information retrieval systems. The idea behind relevance feedback is to take the results that are initially returned from a given query and to use information about whether or not those results are relevant to perform a new query. We can usefully distinguish between three types of feedback: explicit feedback, implicit feedback, and blind or "pseudo" feedback.

Explicit feedback

Explicit feedback is obtained from assessors of relevance indicating the relevance of a document retrieved for a query. This type of feedback is defined as explicit only when the assessors (or other users of a system) know that the feedback provided is interpreted as relevance judgments.

Users may indicate relevance explicitly using a *binary* or *graded* relevance system. Binary relevance feedback indicates that a document is either relevant or irrelevant for a given query. Graded relevance feedback indicates the relevance of a document to a query on a scale using numbers, letters, or descriptions (such as "not relevant", somewhat relevant", "relevant", or "very relevant"). Graded relevance may also take the form of a cardinal ordering of documents created by an assessor; that is, the assessor places documents of a result set in order of (usually descending) relevance.

A performance metric which became popular around 2005 to measure the usefulness of a ranking algorithm based on the explicit relevance feedback is NDCG. Other measures include precision at *k* and mean average precision.

Implicit feedback

Implicit feedback is inferred from user behavior, such as noting which documents they do and do not select for viewing, the duration of time spent viewing a document, or page browsing or scrolling actions .

The key differences of implicit relevance feedback from that of explicit include :

1. the user is not assessing relevance for the benefit of the IR system, but only satisfying their own needs and
2. the user is not necessarily informed that their behavior (selected documents) will be used as relevance feedback

An example of this is the Surf Canyon browser extension, which advances search results from later pages of the result set based on both user interaction (clicking an icon) and time spent viewing the page linked to in a search result.

Blind feedback

Pseudo relevance feedback, also known as blind relevance feedback, provides a method for automatic local analysis. It automates the manual part of relevance feedback, so that the user gets improved retrieval performance without an extended interaction. The method is to do normal retrieval to find an initial set of most relevant documents, to then assume that the top "k" ranked documents are relevant, and finally to do relevance feedback as before under this assumption. The procedure is:

1. Take the results returned by initial query as relevant results (only top k with k being between 10 to 50 in most experiments).
2. Select top 20-30 (indicative number) terms from these documents using for instance tf-idf weights.
3. Do Query Expansion, add these terms to query, and then match the returned documents for this query and finally return the most relevant documents.

Some experiments such as results from the Cornell SMART system published in (Buckley et al.1995), show improvement of retrieval systems performances using pseudo-relevance feedback in the context of TREC 4 experiments.

This automatic technique mostly works. Evidence suggests that it tends to work better than global analysis. Through a query expansion, some relevant documents missed in the initial round can then be retrieved to improve the overall performance. Clearly, the effect of this method strongly relies on the quality of selected expansion terms. It has been found to improve performance in the TREC ad hoc task. But it is not without the dangers of an automatic process. For example, if the query is about copper mines and the top several documents are all about mines in Chile, then there may be query drift in the direction of documents on Chile. In addition, if the words added to the original query are unrelated to the topic, the quality of the retrieval is likely to be degraded. Since the Web is a highly volatile and heterogeneous information source and contains a lot of low-quality documents, such a naive pseudo-relevance feedback is not capable of producing a satisfactory result.

Blind feedback automates the manual part of relevance feedback and has the advantage that assessors are not required.

Using relevance information

Relevance information is utilized by using the contents of the relevant documents to either adjust the weights of terms in the original query, or by using those contents to add words to the query. Relevance feedback is often implemented using the Rocchio Algorithm.

Chapter-15

Search Engine (computing) and Search Engine Technology

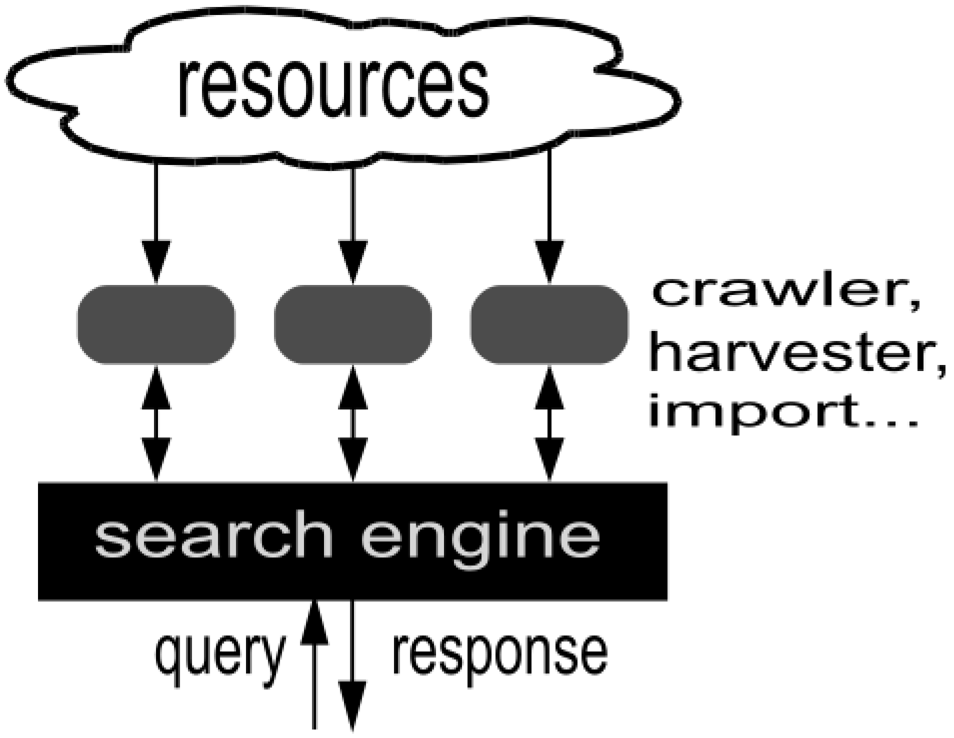
Search engine

A **search engine** is an information retrieval system designed to help find information stored on a computer system. The search results are usually presented in a list and are commonly called *hits*. Search engines help to minimize the time required to find information and the amount of information which must be consulted, akin to other techniques for managing information overload.

The most public, visible form of a search engine is a Web search engine which searches for information on the World Wide Web.

How search engines work

Search engines provide an interface to a group of items that enables users to specify criteria about an item of interest and have the engine find the matching items. The criteria are referred to as a search query. In the case of text search engines, the search query is typically expressed as a set of words that identify the desired concept that one or more documents may contain. There are several styles of search query syntax that vary in strictness. It can also switch names within the search engines from previous sites. Whereas some text search engines require users to enter two or three words separated by white space, other search engines may enable users to specify entire documents, pictures, sounds, and various forms of natural language. Some search engines apply improvements to search queries to increase the likelihood of providing a quality set of items through a process known as query expansion.



index-based search engine

The list of items that meet the criteria specified by the query is typically sorted, or ranked. Ranking items by relevance (from highest to lowest) reduces the time required to find the desired information. Probabilistic search engines rank items based on measures of similarity (between each item and the query, typically on a scale of 1 to 0, 1 being most similar) and sometimes popularity or authority or use relevance feedback. Boolean search engines typically only return items which match exactly without regard to order, although the term *boolean search engine* may simply refer to the use of boolean-style syntax (the use of operators AND, OR, NOT, and XOR) in a probabilistic context.

To provide a set of matching items that are sorted according to some criteria quickly, a search engine will typically collect metadata about the group of items under consideration beforehand through a process referred to as indexing. The index typically requires a smaller amount of computer storage, which is why some search engines only store the indexed information and not the full content of each item, and instead provide a method of navigating to the items in the search engine result page. Alternatively, the search engine may store a copy of each item in a cache so that users can see the state of the item at the time it was indexed or for archive purposes or to make repetitive processes work more efficiently and quickly.

Other types of search engines do not store an index. Crawler, or spider type search engines (a.k.a. real-time search engines) may collect and assess items at the time of the search query, dynamically considering additional items based on the contents of a starting item (known as a seed, or seed URL in the case of an Internet crawler). Meta search engines store neither an index nor a cache and instead simply reuse the index or results of one or more other search engines to provide an aggregated, final set of results.

Search engine technology

Modern web search engines are complex software systems using the technology that has evolved over the years. There are several categories of search engine software: Web search engines (example: Lucene), database or structured data search engines (example: Dieselpoint), and mixed search engines or enterprise search (example: Google Search Appliance). The largest web search engines such as Google and Yahoo! utilize tens or hundreds of thousands of computers to process billions of web pages and return results for thousands of searches per second. High volume of queries and text processing requires the software to run in highly distributed environment with high degree of redundancy. Modern search engines have the following main components:

Search Engine Categories

Web Search Engines

Search engines designed for searching web pages and documents are designed to allow searching through these largely unstructured units of content. They are built to follow a multi-stage process: crawling the pages or documents to discover their contents, indexing their content in a structured form (database or other), and finally resolving user queries to return results and links to the documents or pages from the index.

Crawl

In the case of full-text search for the web search, the first step in preparing web pages for search is to find and index them. In the past, search engines started with a small list of URLs as seed list, fetched the content, parsed for the links on those pages, fetched the web pages pointed to by those links which provided new links and the cycle continued until enough pages were found. Most modern search engines now utilize a continuous crawl method rather than **discovery** based on a seed list. The continuous crawl method is just an extension of discovery method but there is no seed list because the crawl never stops. The current list of pages is visited on regular intervals and new pages are found when links are added or deleted from those pages. Many search engines use sophisticated scheduling algorithms to decide when to revisit a particular page. These algorithms range from constant visit-interval with higher priority for more frequently changing pages to adaptive visit-interval based on several criteria such as frequency of change, popularity and overall quality of site, speed of web server serving the page and resource constraints like amount of hardware and bandwidth of Internet connection. Search engines crawl many more pages than they make available for searching because crawler find lots duplicate content pages on the web and many pages don't have useful content. Duplicate and useless content often represents more than half the pages available for indexing.

Link Map

Pages discovered by crawlers are fed into (often distributed) service that creates a **link map** of the pages. Link map is a graph structure in which pages are represented as nodes connected by the links among those pages. This data is stored in data structures that allow fast access to the data by certain algorithms which compute the popularity score of pages on the web, essentially based on how many links point to a web page and the quality of those links. One such algorithm, PageRank, proposed by Google founders Larry Page and Sergey Brin, is well known and has attracted a lot of attention. The idea of doing link analysis to compute a popularity rank is older than PageRank and many variants of the same idea are currently in use. These ideas can be categorized in three main categories: rank of individual pages, rank of web sites, and nature of web site content (Jon Kleinberg's HITS algorithm). Search engines often differentiate between internal links and external links, with the assumption that links on a page pointing other pages on the same site are less valuable because they are often created by web site owners to artificially increase the rank of their web sites and pages. Link map data structures typically also store the anchor text embedded in the links because anchor text often provides a very good quality short-summary of a web page's content.

Index

Indexing is the process of extracting text from web pages, tokenizing it and then creating an index structure (inverted index) that can be used to quickly find which pages contain a particular word. Search engines differ quite a lot in tokenization process. The issues involved in tokenization are: detecting the encoding used for the page, determining the language of the content (some pages use multiple languages), finding word, sentence and paragraph boundaries, combining multiple adjacent-words into one phrase and changing the case of text and stemming the words into their roots (lower-casing and stemming is applicable only to some languages). This phase also decides which sections of page to index and how much text from very large pages (such as technical manuals) to index. Search engines also differ in the document formats they interpret and extract the text from.

Some search engines go through the indexing process every few weeks and refresh the complete index used for web search requests while others keep updating small fragments of the index continuously. Before web pages can be indexed, an algorithm decides which node (a server in a distributed service) will index any given page and makes the information available as metadata for other components in the search engine. The index structure is complex and typically employs some compression algorithm. The choice of compression algorithm involves a trade-off between on-disk storage space and speed of decompression when needed to satisfy search requests. The largest search engines use thousands of computers to index pages in parallel.

Database Search Engines

Searching for text-based content in databases presents some special challenges and opportunities which a number of specialized search engines resolve. Databases are slow when solving complex queries (with multiple logical or string matching arguments. Databases allow logical queries which full-text search doesn't (use of multi-field boolean logic for instance). There is no crawling necessary for a database since the data is already structured but it is often necessary to index the data in a more compact form designed to allow for faster search.

Database search systems relational databases are indexed by compounding multiple tables into a single table containing only the fields that need to be queried (or displayed in search results). The actual data matching engines can include any functions from basic string matching, normalization, transformation, Database search technology is heavily used by government database services, e-commerce companies, web advertising platforms, telecommunications service providers, etc.

Mixed Search Engines

In cases where the data searched contains both database content and webpages or documents, search engine technology has been developed to respond to both sets of requirements. Most mixed search engines are large Web search engines (example: Google) or enterprise search software products (example: Autonomy). They search both through structured and unstructured data sources. Pages and documents are crawled and indexed in a separate index. Databases are indexed also from various sources. Search results are then generated for users by querying these multiple indices in parallel and compounding the results according to rules.

Much of the incremental value of these search systems comes from their ability to connect to multiple sources of content and data and their ability to interpret their multiple formats.