UNIT IV

UNIT IV WEB SEARCH – LINK ANALYSIS AND SPECIALIZED SEARCH

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

Link Analysis

In this chapter we focus on the use of hyperlinks for ranking web search results. Such link analysis is one of many factors considered by web search engines in computing a composite score for a web page on any given query. There are two distinct methods for link analysis, Page rank and HITS

4.1 Web as a graph

Web consisting of static HTML pages together with the hyperlinks between them as a directed graph in which each web page is a node and each hyperlink a directed edge.

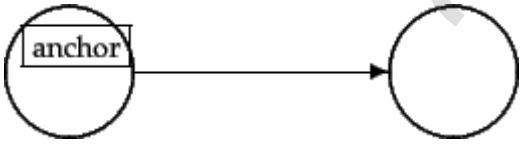


Figure4.1.1 Two nodes of the web graph joined by a link.

Figure 4.1.1 shows two nodes A and B from the web graph, each corresponding to a web page, with a hyperlink from A to B. We refer to the set of all such nodes and directed edges as the web graph. The Figure illustrates that there is some text surrounding the origin of the hyperlink on page A. This text is generally encapsulated in the href attribute of the <a> (for anchor) tag that encodes the hyperlink in the HTML code of page A, and is referred to as *anchor text* .

4.2 Types of Links

(1) Inbound links or Inlinks

- Inbound links are links into the site from the outside.
- Inlinks are one way to increase a site's total Page Rank.
- Sites are not penalized for inlinks.

(2) Outbound links or Outlinks

Outbound links are links from a page to other pages in a site or other sites.

(3) Dangling links
Dangling links are simply links that point to any page with no outgoing links.

4.3 Link Spam

It is well known that techniques such as Page Rank and anchor text extraction are used in commercial search engines, so unscrupulous web page designers may try to create useless links just to improve the search engine placement of one of their web pages. This is called ***link spam***.

4.4 Page Rank

It is a scoring measure based only on the link structure of web pages. A web page is important if it is pointed to by other important web pages. Our first technique for link analysis assigns to every node in the web graph a numerical score between 0 and 1 known as it page rank.

Given a query, a web search engine computes a composite score for each web page that combines hundreds of features such as cosine similarity and term proximity together with the Page rank score.

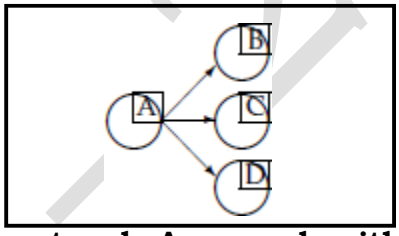


Fig:4.4.1 The random surfer at node A proceeds with probability 1/3 to each of B, C and D

Consider a random surfer who begins at a web page (a node of the web graph) and executes a random walk on the Web as follows. At each time step, the surfer proceeds from his current page A to a randomly chosen web page that A hyperlinks to. The above fig shows the surfer at a node A, out of which there are three hyperlinks to nodes B, C and D; the surfer proceeds at the next time step to one of these three nodes, with equal probabilities 1/3.

As the surfer proceeds in this random walk from node to node, he visits some nodes more often than others; intuitively, these are nodes with many links coming in from other frequently visited nodes. The idea behind Page Rank is that pages visited more often in this walk are more important.

Teleport

If the current location of the surfer, the node A, has no out-links? To address this we introduce an additional operation for our random surfer that is the teleport operation. In the teleport operation the surfer jumps from a node to any other node in the web graph. The destination of a teleport operation is modeled as being chosen uniformly at random from all web pages. In other

words, if N is the total number of nodes in the web graph 1, the teleport operation takes the surfer to each node with probability 1/N.

4.5 Markov chain

A Markov chain is a discrete-time stochastic process: a process that occurs in a series of time-steps in each of which a random choice is made. A Markov chain consists of N states.
A Markov chain is characterized by an N × N transition probability matrix P each of whose entries is in the interval [0, 1]; the entries in each row of P add up to 1.Each entry Pij is known as a transition probability and depends only on the current state i; this is known as the Markov property. Thus, by the Markov property,

$$\forall i, j, P_{ij} \in [0, 1]$$

and

$$\forall i, \sum_{j=1}^{N} P_{ij} = 1.$$

Equation 4.5.1

A matrix with non-negative entries that satisfies Equation 4.5.1 is known as a stochastic matrix. A key property of a stochastic matrix is that it has a principal left eigenvector corresponding to its largest eigenvalue, which is 1.

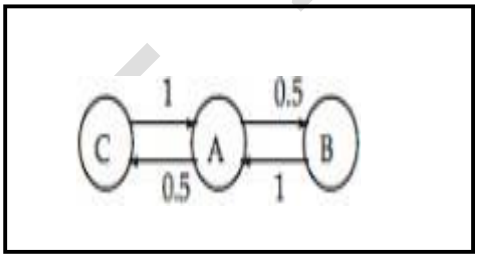


Fig:4.5.1 A simple Markov chain with three states , the numbers on the links indicate the transition probabilities

In a Markov chain, the probability distribution of next states for a Markov chain depends only on the current state, and not on how the Markov chain arrived at the current state. Figure 4.5.1 shows a simple Markov chain with three states. From the middle state A, we proceed with (equal) probabilities
of 0.5 to either B or C. From either B or C, we proceed with probability 1 to A.

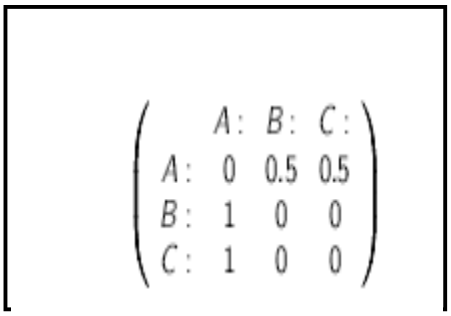


Fig:4.5.2 The transition probability matrix of this Markov chain

A Markov chain‟s probability distribution over its states may be viewed as a probability vector: a vector all of whose entries are in the interval [0, 1], and the entries add up to 1. For our simple Markov chain of Figure 4.5.1, the probability vector would have 3 components that sum to 1.We can view a random surfer on the web graph as a Markov chain, with one state for each web page, and each transition probability representing the probability of moving from one web page to another. The teleport operation contributes to these transition probabilities. The adjacency matrix A of the web graph is defined as follows: if there is a hyperlink from page i to page j, then Aij = 1, otherwise Aij = 0. We can readily derive the transition probability matrix P for our Markov chain from the N × N matrix A:

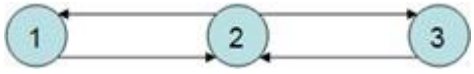
1. If a row of A has no 1‟s, then replace each element by 1/N. For all other rows proceed as follows.
2. Divide each 1 in A by the number of 1‟s in its row. Thus, if there is a row with three 1‟s, then each of them is replaced by 1/3.
3. Multiply the resulting matrix by 1 − α.
4. Add α/N to every entry of the resulting matrix, to obtain P.

For a Markov chain to be **ergodic**, two technical conditions are required of its states and the non-zero transition probabilities, these conditions are known as **irreducibility** and **aperiodicity**. Informally, the first ensures that there is a sequence of transitions of non-zero probability from any state to any other, while the latter ensures that the states are not partitioned into sets such that all state transitions occur cyclically from one set to another.

4.6 Page rank Computation using Power iteration

Steps to compute page rank

1.Given graph of links, build Matrix A(Adjacency Matrix)



$$A = \begin{matrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{matrix}$$

2.Apply teleportation, build Matrix P(Teleportation Matrix)

$$P = \begin{matrix} 1/6 & 2/3 & 1/6 \\ 5/12 & 1/6 & 5/12 \\ 1/6 & 2/3 & 1/6 \end{matrix}$$

$$\begin{matrix} 0 \\ 1 \\ 1 \end{matrix} X \begin{matrix} 1/6 & 2/3 & 1/6 \\ 2 & 3 & 6 \end{matrix} + \begin{matrix} 1/6 \\ 6 \end{matrix} = \begin{matrix} 1/6 \\ 6 \end{matrix}$$

$$1 X \begin{matrix} 1/6 & 2/3 & 1/6 \\ 2 & 3 & 6 \end{matrix} + \begin{matrix} 1/6 \\ 6 \end{matrix} = \begin{matrix} 2/3 \\ 3 \end{matrix}$$

3. From Modified Matrix, compute π using power Iteration Method

$$\vec{x}_0 = (1,0,0)$$

$$\vec{x}_0 P = (1/6 \quad 2/3 \quad 1/6) = \vec{x}_1$$

$$\vec{x}_1 P = (1/6 \quad 2/3 \quad 1/6) \begin{pmatrix} 1/6 & 2/3 & 1/6 \\ 5/12 & 1/6 & 5/12 \\ 1/6 & 2/3 & 1/6 \end{pmatrix}$$

$$= (1/3 \quad 1/3 \quad 1/3) = \vec{x}_2$$

4. $\vec{\pi}_i$ is the page rank of page i

Sequence of Probability Vector

$\vec{x}_0$	1	0	0
$\vec{x}_1$	1/6	2/3	1/6
$\vec{x}_2$	1/3	1/3	1/3
$\vec{x}_3$	1/4	1/2	1/4
$\vec{x}_4$	7/24	5/12	7/24
...	...	...	...
$\vec{x}$	5/18	4/9	5/18

Steady state

5. Page Rank => $\vec{\pi}$ =(5/18 4/9 5/18)

Page 1 5/18 =0.278
Page 2 4/ 9 =0.444
Page 3 5/18 =0.278

4.7 Hubs and Authorities

Given a query, every web page is assigned two scores. One is called its hub score and the other HUB SCORE its authority score. For any query, we compute two ranked lists of results rather than one. The ranking of one list is induced by the hub scores and that of the other by the authority scores.

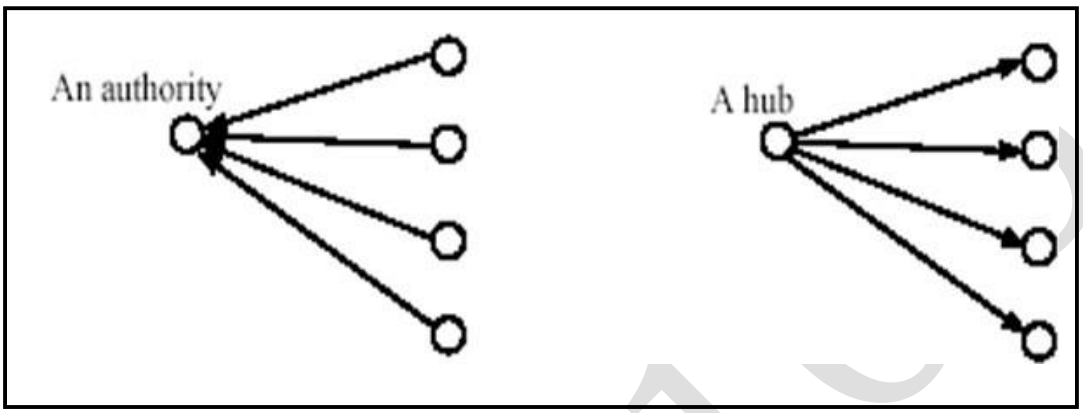


Fig 4.7.1 An authority page and a hub page

A good hub page is one that points to many good authorities, a good authority page is one that is pointed to by many good hub pages. We thus appear to have a circular definition of hubs and authorities, we will turn this into an iterative computation. For a web page v in our subset of the web, we use h(v) to denote its hub score and a(v) its authority score. Initially, we set h(v) = a(v) = 1 for all nodes v. We also denote by v → y the existence of a hyperlink from v to y. The core of the iterative algorithm is a pair of updates to the hub and authority scores of all pages given by Equation 4.7.2 which capture the intuitive notions that good hubs point to good authorities and that good authorities are pointed to by good hubs.

$$h(v) \leftarrow \sum_{v \mapsto y} a(y)$$
$$a(v) \leftarrow \sum_{y \mapsto v} h(y).$$

Equation 4.7.2

The first equation sets the hub score of page v to the sum of the authority scores of the pages it links to. In other words, if v links to pages with high authority scores, its hub score increases. The second line plays the reverse role; if page v is linked to by good hubs, its authority score increases. Recasting the equations 4.7.2 into matrix-vector form. Let h and a denote the vectors of all hub and all authority scores respectively, for the pages in our subset of the web graph. Let A denote the adjacency matrix of the subset of the web graph that we are dealing with: A is a square matrix with one row and one column for each page in the subset. The entry Aij is 1 if there is a hyperlink from page i to page j, and 0 otherwise. Then, we may write Equation 4.7.2 as

$$\vec{h} \leftarrow A\vec{a}$$
$$\vec{a} \leftarrow A^T\vec{h},$$

Equation 4.7.3

where AT denotes the transpose of the matrix A. Now the right hand side of each line of Equation 4.7.3 a vector that is the left hand side of the other line of Equation 4.7.3 . Substituting these into one another, we may rewrite Equation 4.7.3 as

$$\vec{h} \leftarrow AA^T\vec{h}$$
$$\vec{a} \leftarrow A^TA\vec{a}.$$

Equation 4.7.4

we replace the ← symbols by = symbols and introduce the (unknown) eigen value, the first line of Equation 4.7.4 becomes the equation for the eigenvectors of AA^T,while the second becomes the equation for the eigenvectors of A^TA:

$$\vec{h} = (1/\lambda_h)AA^T\vec{h}$$
$$\vec{a} = (1/\lambda_a)A^TA\vec{a}.$$

Equation 4.7.5

Here we have used λ_h to denote the eigenvalue of AA^T and λ_a to denote the eigenvalue of A^TA.

The resulting computation thus takes the following form:

1. Assemble the target subset of web pages, form the graph induced by their hyperlinks and compute AA^T and A^TA.
2. Compute the principal eigenvectors of AA^T and A^TA to form the vector of hub scores h and authority scores a.
3. Output the top-scoring hubs and the top-scoring authorities.

This method of link analysis is known as HITS, which is an acronym for Hyperlink-Induced Topic Search.

4.8 PageRank v.s. HITS

PageRank	HITS
Computed for all web pages stored prior to the query	Performed on the subset generated by each query
Computes authorities only	Computes authorities and hubs
Fast to compute	Easy to compute, real

	Time execution is hard.

5. Searching and Ranking

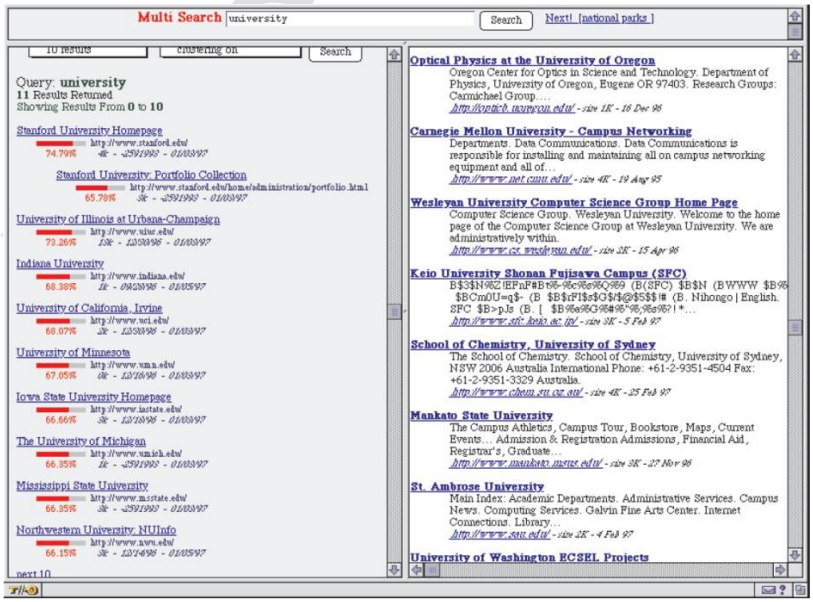
- **Two search engines:**
  - Title-based search engine
  - Full text search engine
- **Title-based search engine**
  - Searches only the "Titles"
  - Finds all the web pages whose titles contain all the query words
  - Sorts the results by Page Rank
  - Very simple and cheap to implement
  - Title match ensures high precision, and Page Rank ensures high quality

Full text search engine

- Examines all the words in every stored document and also performs Page Rank (Rank Merging)

-More precise but more complicated

4.4.1 Searching with PageRank



Web Page	PageRank (average is 1.0)
Download Netscape Software	11589.00
http://www.w3.org/	10717.70
Welcome to Netscape	8673.51
Point: It's What You're Searching For	7930.92
Web-Counter Home Page	7254.97
The Blue Ribbon Campaign for Online Free Speech	7010.39
CERN Welcome	6562.49
Yahoo!	6561.80
Welcome to Netscape	6203.47
Wusage 4.1: A Usage Statistics System For Web Servers	5963.27
The World Wide Web Consortium (W3C)	5672.21
Lycos, Inc. Home Page	4683.31
Starting Point	4501.98
Welcome to Magellan!	3866.82
Oracle Corporation	3587.63

4.4.2 Top 15 Page ranks in year 1996

Thus PageRank is a global ranking of all web pages based on their locations in the web graph structure . PageRank uses information which is external to the web pages – backlinks. Backlinks from important pages are more significant than backlinks from average pages.

4.4.3 Advantages of Page Rank

- **Fighting spam**. A page is important if the pages pointing to it are important.

  Since it is not easy for Web page owner to add in-links into his/her page from other important pages, it is thus not easy to influence Page Rank.

- **Page Rank is a global measure and is query independent**.

  Page Rank values of all the pages are computed and saved off-line rather than at the query time.

4.5 Relevance Scoring , ranking for Web, Similarity

Determining Relevance

When a user submits a query to a search engine, the first thing it must do is determine which pages in the index are related to the query and which are not. we can state it as follows:
Given a search query and a document, compute a relevance score that measures the similarity between the query and document.
The "document" in this context can also refer to things like the title tag, the meta description, incoming anchor text, or anything else that we think might help determine whether the query is related to the page. Practically, a search engine computes a number of relevance scores using different page elements and weights them all to arrive at one final score.

4.5.1 Relevance vs Ranking

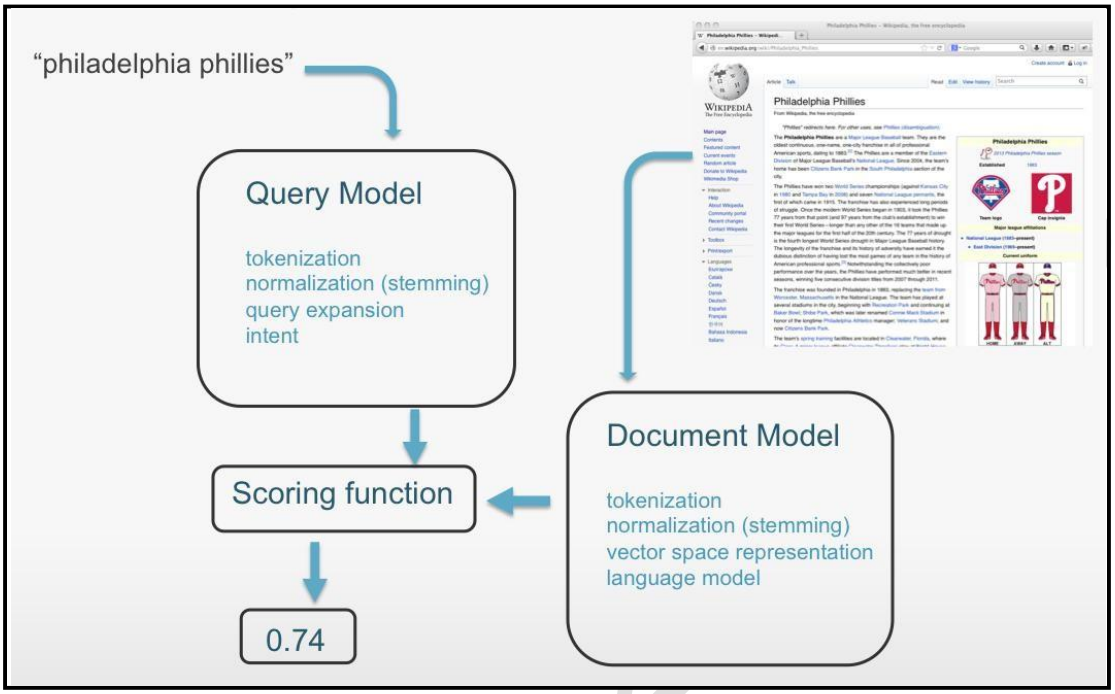
First the user separates relevance determination from ranking the relevant documents, even if they are implemented as a single step inside a search engine. The relevance step first makes a binary (True/False) decision for each page, then the ranking step orders the documents to return to the user.

Relevance determination from ranking

4.5.2 Query and Document Models

Translating the query and document from raw strings into something we can do computation with is the first hurdle in computing a similarity score. To do so, we make use of "query models" and "document models." The "models" here

are just a fancy way of saying that the strings are represented in some other way that makes computation possible.

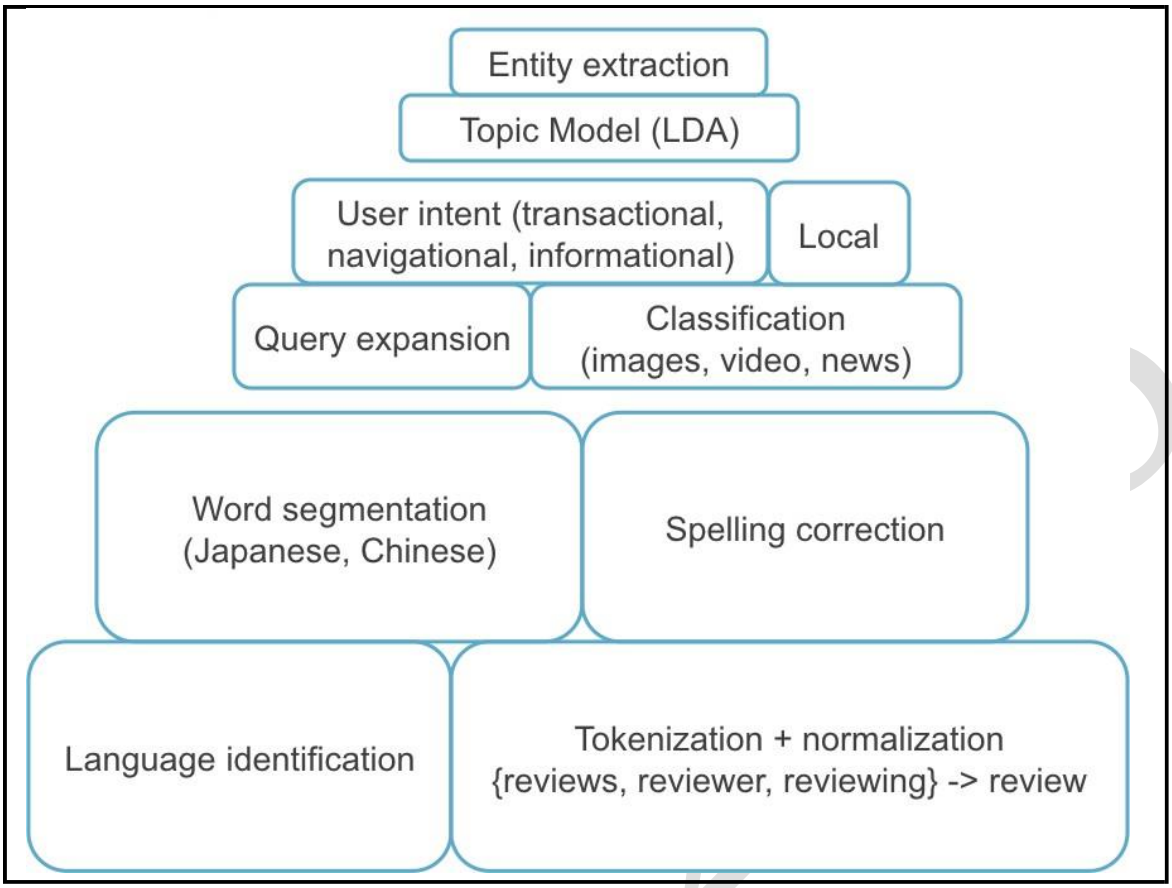


Query Model and Document Model

The above image illustrates this process for the query "philadelphia phillies" and the Wikipedia page about the Phillies. The final step in computing the similarity score runs the query and document representations through a scoring function.

4.5.3 Query Models

The following image illustrates some different types of query models:



Types of Query Models

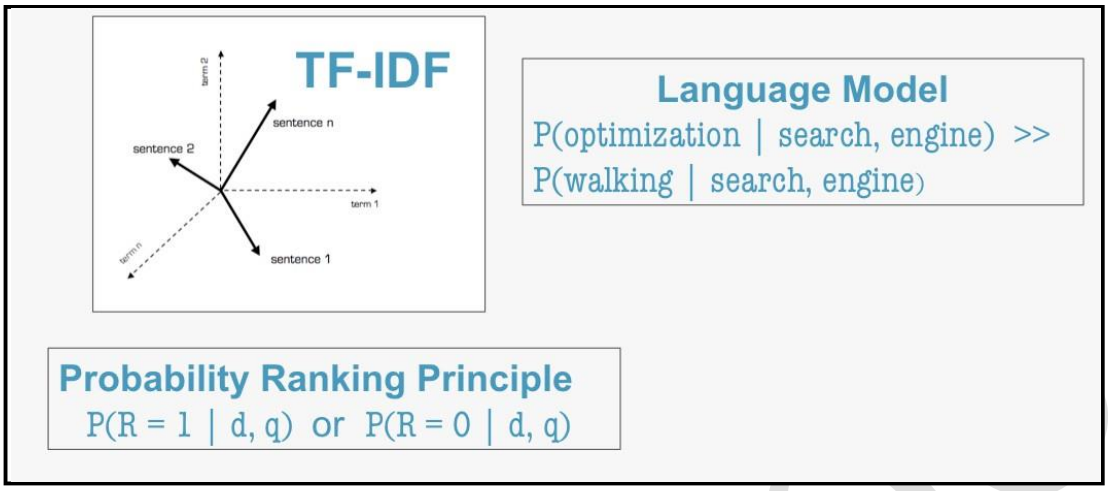
The building blocks at the bottom include things like tokenization (splitting the string into words), word normalization (such as stemming where common word endings are removed), and spelling correction (if a query contains a misspelled word, the search engine corrects it and returns results for the corrected word).

Built on top of these building blocks are things like query classification and intent. If the search engine determines that a particular query is time sensitive it will return news results, or if it thinks the query intent is transactional it will display shopping results. Finally, at the top of the pyramid are more abstract representations of the query such as entity extraction or latent topic representations (LDA).

4.5.4 Document Models

Like query models, there are several different types of document models

commonly used in search.



Types of Document Models

TF-IDF is one of the oldest and most well known approaches that represents each query and document as a vector and uses some variant of the cosine similarity as the scoring function. **Tf-IDF(term frequency–inverse document frequency)**,is a numerical statistic that is intended to reflect how important a word is to a document in a collection . It is often used as a weighting factor in information retrieval and text mining.

A language model encodes some information about the statistics of a language and includes knowledge such as the phrase "search engine optimization" is much more common then "search engine walking." Language models are used heavily in machine translation and speech recognition, among other applications. They are also extremely useful in information retrieval. Yet another class of models uses the probability ranking principle, which directly

models the probability of relevance given the query and document.

4.5.5 Scoring and Ranking Techniques

- Tf-idf term weighting

A term can also be assigned a weight that expresses its importance for a particular document. A commonly used term weighting method is tf-idf, which assigns a high weight to a term, if it occurs frequently in the document but rarely in the whole document collection. Contrarily, a term that occurs in

nearly all documents has hardly any discriminative power and is given a low weight. For calculating the tf-idf weight of a term in a particular document, it is necessary to know two things: how often does it occur in the document (term frequency = tf), and in how many documents of the collection does it appear (document frequency = df). Take the inverse of the document frequency (= idf) and you have both components to calculate the weight by multiplying tf by idf.

$$idf = \log\left(\frac{N}{df}\right)$$

- Vector space model

The tf-idf values can now be used to create vector representations of documents. Each component of a vector corresponds to the tf-idf value of a particular term in the corpus dictionary. Dictionary terms that do not occur in a document are weighted zero. Using this kind of representation in a common vector space is called vector space model (Salton et al., 1975), which is not only used in information retrieval but also in a variety of other research fields like machine learning (e.g. clustering, classification). Each dimension of the space - we are far beyond 3D when it comes to text - corresponds to a term in the dictionary. Remembering what we learned in trigonometry lessons at school, a vector space provides the possibility to perform calculations like computing differences or angles between vectors. Since documents are usually not of equal length, simply computing the difference between two vectors has the disadvantage that documents of similar content but different length are not regarded as similar in the vector space.

- Cosine similarity measure

  Cosine similarity is a measure of similarity between two vectors of an inner product space that measures the cosine of the angle between them. The cosine of 0° is 1, and it is less than 1 for any other angle. It is thus a judgment of orientation and not magnitude: two vectors with the same orientation have a cosine similarity of 1, two vectors at 90° have a similarity of 0, and two vectors diametrically opposed have a similarity of -1, independent of their magnitude.

Cosine similarity is particularly used in positive space, where the outcome is neatly bounded in [0,1].we can find out the similarity between any two documents.

Cosine Similarity (d1, d2) = Dot product(d1, d2) / ||d1|| * ||d2||

Dot product (d1,d2) = d1[0] * d2[0] + d1[1] * d2[1] * … * d1[n] * d2[n]
||d1|| = square root($d1[0]^2 + d1[1]^2 + … + d1[n]^2$)
||d2|| = square root($d2[0]^2 + d2[1]^2 + … + d2[n]^2$)

4.5.6 Similarity measure between two vectors

After fixing the term-weights, we have document and query vectors in k dimension (where k is number of index terms in vocabulary). Now we need to find the similarity between them. A widely used measure of similarity called the cosine similarity.

The inner product of the two vectors (sum of the pairwise multiplied elements) is divided by the product of their vector lengths. This has the effect that the vectors are normalized to unit length and only the angle, more precisely the cosine of the angle, between the vectors accounts for their similarity.

$$sim(d_1, d_2) = \frac{\vec{v}(d_1) \cdot \vec{v}(d_2)}{\left|\vec{v}(d_1)\right|\left|\vec{v}(d_2)\right|}$$

Documents not sharing a single word get assigned a similarity value of zero because of the orthogonality of their vectors while documents sharing a similar vocabulary get higher values (up to one in the case of identical documents). Because a query can be considered a short document, it is of course possible to create a vector for the query, which can then be used to calculate the cosine similarities between the query vector and those of the matching documents. Finally, the similarity values between query and the retrieved documents are used to rank the results.

4.6 Hadoop & Map Reduce

What is Hadoop?

Hadoop is an Apache open-source framework that allows to store and process big data in a distributed environment across clusters of computers using simple programming models. It is designed to scale up from single servers to thousands of machines, each offering local computation and storage.

- **Open-source software** Open-source software is created and maintained by a network of developers from around the globe. It's free to download, use and

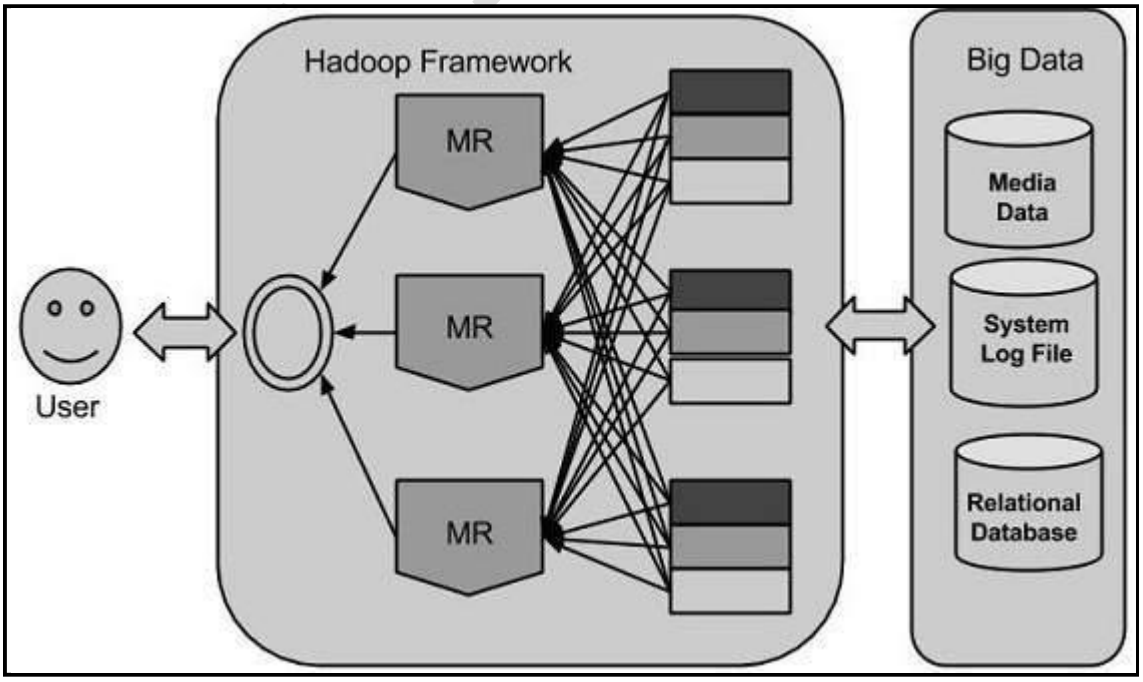
contribute to, though more and more commercial versions of Hadoop are becoming available.

- **Framework** In this case, it means that everything you need to develop and run software applications is provided – programs, connections, etc.
- **Massive storage** The Hadoop framework breaks big data into blocks, which are stored on clusters of commodity hardware.
- **Processing power** Hadoop concurrently processes large amounts of data using multiple low-cost computers for fast results.

4.6.1 Hadoop Architecture

Doug Cutting, Mike Cafarella and team took the solution provided by Google and started an Open Source Project called HADOOP in 2005 and Doug named it after his son's toy elephant. Now Apache Hadoop is a registered trademark of the Apache Software Foundation.

Hadoop runs applications using the MapReduce algorithm, where the data is processed in parallel on different CPU nodes. In short, Hadoop framework is capable enough to develop applications, capable of running on clusters of computers and they could perform complete statistical analysis for a huge amounts of data.

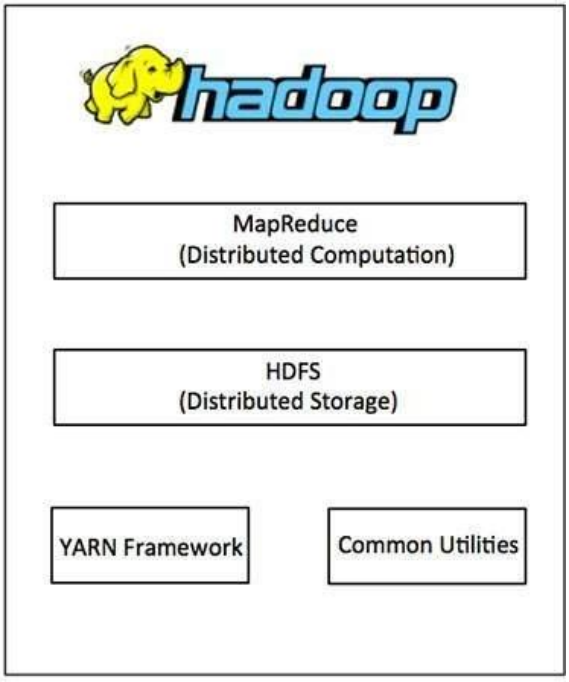


Hadoop Architecture

Hadoop framework includes following four modules:

- **Hadoop Common:** These are Java libraries and utilities required by other Hadoop modules. These libraries provides filesystem and OS level abstractions and contains the necessary Java files and scripts required to start Hadoop.
- **Hadoop YARN:** This is a framework for job scheduling and cluster resource management.
- **Hadoop Distributed File System (HDFS™):** A distributed file system that provides high-throughput access to application data.
- **Hadoop MapReduce:** This is YARN-based system for parallel processing of large data sets.

The following diagram depicts the four components available in Hadoop framework.



Since 2012, the term "Hadoop" often refers not just to the base modules mentioned above but also to the collection of additional software packages

that can be installed on top of or alongside Hadoop, such as Apache Pig, Apache Hive, Apache HBase, Apache Spark etc.

4.6.2 The benefits of Hadoop

One of the top reasons that organizations turn to Hadoop is its ability to store and process huge amounts of data any kind of data quickly. With data volumes and varieties constantly increasing, especially from social media and the Internet of Things, that‟s a key consideration. Benefits include:

- **Computing power.** Its distributed computing model quickly processes big data. The more computing nodes the user use, the more processing power the user have.
- **Flexibility.** Unlike traditional relational databases, user don‟t have to preprocess data before storing it. They can store as much data as they want and decide how to use it later. That includes unstructured data like text, images and videos.
- **Fault tolerance.** Data and application processing are protected against hardware failure. If a node goes down, jobs are automatically redirected to other nodes to make sure the distributed computing does not fail. And it automatically stores multiple copies of all data.
- **Low cost.** The open-source framework is free and uses commodity hardware to store large quantities of data.
- **Scalability**. The user can easily grow the system simply by adding more nodes. Little administration is required.

Big Data

Big data means really a big data, it is a collection of large datasets that cannot be processed using traditional computing techniques. Big data is not merely a data, rather it has become a complete subject, which involves various tools, techniques and frameworks.

4.6.3 Hadoop and Map Reduce

Map Reduce is designed for large computer clusters. The point of a cluster is to solve large computing problems on cheap commodity machines or nodes that are built from standard parts (processor, memory, disk) Although hundreds or thousands of machines are available in such clusters, individual machines can

fail at any time. One requirement for robust distributed indexing is, therefore, that we divide the work up into chunks that we can easily assign and in case of failure reassign. **A master node** directs the process of assigning and reassigning tasks to individual worker nodes.
The map and reduce phases of MapReduce split up the computing job into chunks that standard machines can process in a short time. The various steps of MapReduce are shown in Figure 4.6.3.2 and an example on a collection consisting of two documents is shown in Figure below

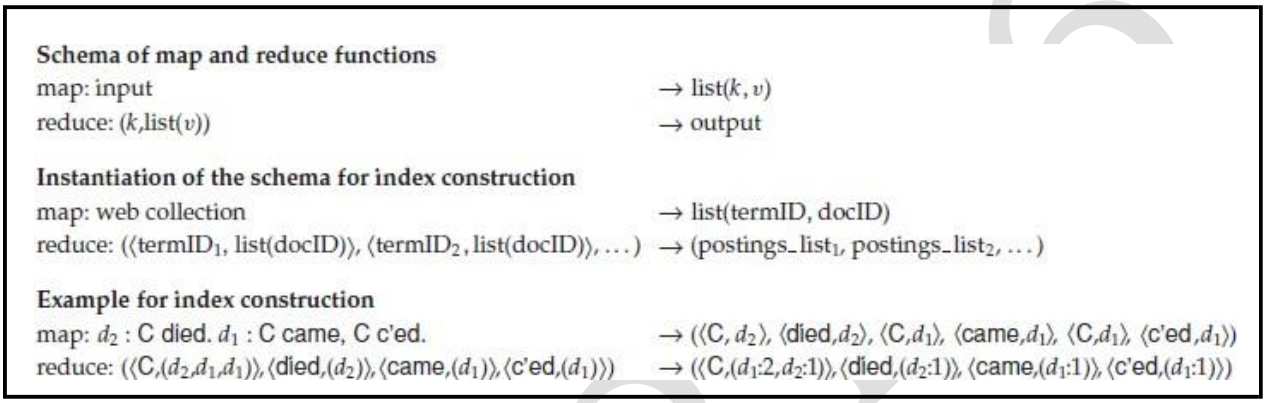


Figure 4.6.3.1 Map and reduce functions in MapReduce. In general, the map function produces a list of key-value pairs. All values for a key are collected into one list in the reduce phase. This list is then processed further. The instantiations of the two functions and an example are shown for index construction. Because the map phase processes documents in a distributed fashion, termID–docID pairs need not be ordered correctly initially as in this example. The example shows terms instead of termIDs for better readability.

First, the input data, in our case a collection of web pages, are split into n splits where the size of the split is chosen to ensure that the work can be distributed evenly and efficiently , 16 or 64MB are good sizes in distributed indexing. Splits are not pre-assigned to machines, but are instead assigned by the master node on an ongoing basis: As a machine finishes processing one split, it is assigned the next one. If a machine dies or becomes a laggard due to hardware problems, the split it is working on is simply reassigned to
another machine.
In general, MapReduce breaks a large computing problem into smaller KEY-VALUE PAIRS parts by recasting it in terms of manipulation of key-value pairs. For indexing, a key-value pair has the form (termID,docID). In distributed indexing, the mapping from terms to termIDs is also distributed and therefore more complex than in single-machine indexing. A simple solution is to maintain a (perhaps precomputed) mapping for frequent terms that is copied to all nodes and to use terms directly (instead of termIDs) for infrequent terms.

We do not address this problem here and assume that all nodes share a consistent
term →termID mapping.
The map phase of MapReduce consists of mapping splits of the input data to key-value pairs. Each parser writes its output to local intermediate files, the segment files (shown as a-f g-p q-z in Figure 4.6.3.2).
For the reduce phase, we want all values for a given key to be stored close together, so that they can be read and processed quickly. This is achieved by

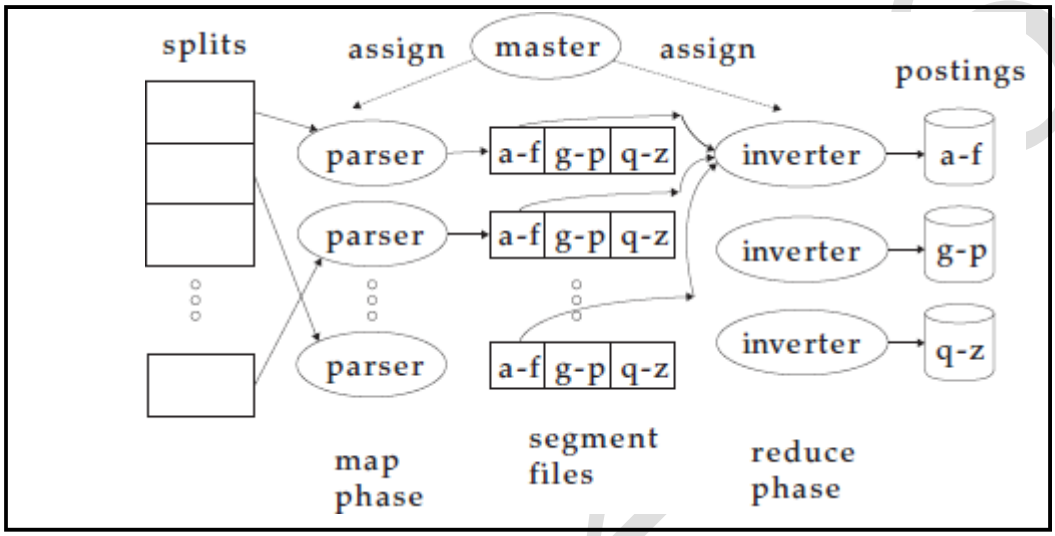


Fig 4.6.3.2 An example of distributed indexing with MapReduce

partitioning the keys into j term partitions and having the parsers write keyvalue pairs for each term partition into a separate segment file. In the above Figure the term partitions are according to first letter: a–f, g–p, q–z, and j = 3. The term partitions are defined by the person who operates the indexing system (Exercise 4.10). The parsers then write corresponding segment files, one for each term partition. Each term partition thus corresponds to r segments files, where r is the number of parsers. For instance, Figure 4.5 shows three a–f segment files of the a–f partition, corresponding to the three parsers shown in the figure 4.6.3.2
Collecting all values (here: docIDs) for a given key (here: termID) into one INVERTER list is the task of the inverters in the reduce phase. The master assigns each term partition to a different inverter and, as in the case of parsers, reassigns term partitions in case of failing or slow inverters. Each term partition (corresponding to r segment files, one on each parser) is processed by one inverter.
We assume here that segment files are of a size that a single machine can handle .Finally, the list of values is sorted for each key and

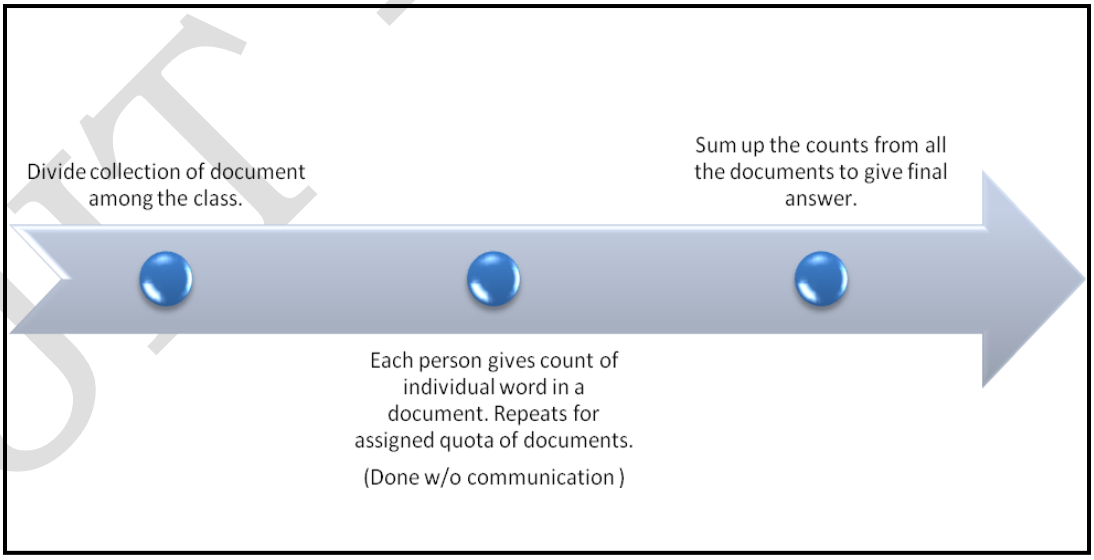
written to the final sorted postings list. The data flow is shown for a–f in Figure 4.6.3.2. This completes the construction of the inverted index. Parsers and inverters are not separate sets of machines. The master identifies idle machines and assigns tasks to them. The same machine can be a parser in the map phase and an inverter in the reduce phase. And there are often other jobs that run in parallel with index construction, so in between being a parser and an inverter a machine might do some crawling or another unrelated task.

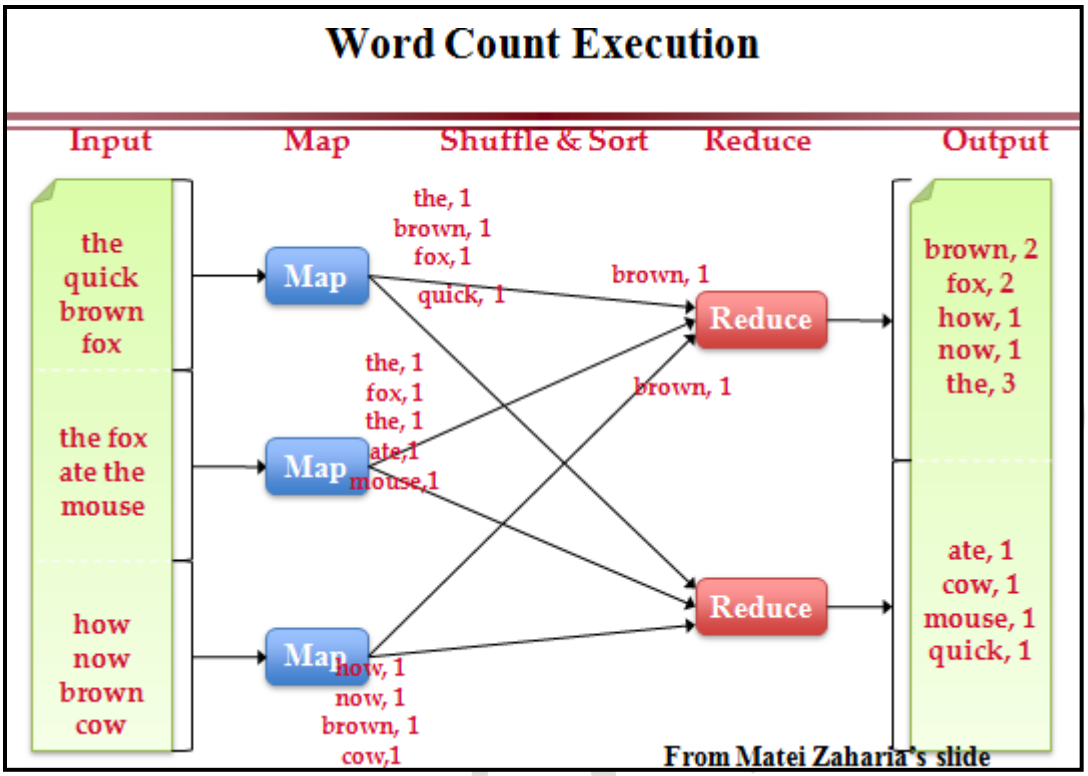
To minimize write times before inverters reduce the data, each parser writes its segment files to its local disk. In the reduce phase, the master communicates to an inverter the locations of the relevant segment files (e.g., of the r segment files of the a–f partition). Each segment file only requires one sequential read because all data relevant to a particular inverter were written to a single segment file by the parser. This setup minimizes the amount of network traffic needed during indexing.

Figure 4.6 shows the general schema of the MapReduce functions. Input and output are often lists of key-value pairs themselves, so that several MapReduce jobs can run in sequence. Another MapReduce operation transforms the term-partitioned index we just created into a document-partitioned one. MapReduce offers a robust and conceptually simple framework for implementing index construction in a distributed environment. By providing a semiautomatic method for splitting index construction into smaller tasks, it can scale to almost arbitrarily large collections, given computer clusters of sufficient size.

Example: Word counting

"Consider the problem of counting the number of occurrences of each word in a large collection of documents"





4.7 Personalized search

There is more and more information and it gets harder to find what one is looking for. A solution to this problem is still to be found , yet the personalization of web search could be a good idea. For example when a user is searching for an ”apple” she might be looking for recipes and cooking tips. Categorization could improve the search results easily, or may be the user has been looking for recipes recently. In this case the search history or even the browser history could be taken into consideration. There are different approaches to personalize search results. A basic and common idea is to create a user profile.

4.7.1

User profiling can be the first step towards personalization of information search and re-trieval. User profiling is basically a composition of three parts: Firstly, the information about the user is collected. Secondly, the user profile is created from that information. Thirdly, the profile has to be represented and stored.

Collecting information

There are a lot of sources for collecting input data about a user. The previous search queries or the search results which were clicked by the user can be collected. The browser history is a data source as well. A lot of data can be gathered for every previously visited page: The visit time, the last visit and number of total visits, the page title or the text contents of the page, the number of out-links followed by the user and more. The bookmarks of the user could also be analyzed. Another source could be the data about the client: For example the IP address (the location can be determined through the IP address), the access method (desktop or mobile device) or the user‟s browser and operating system can be recognized.

4.7.2.Creating profiles

The actual creation of user profiles consists of two parts. In the first part, some Pre-Processing needs to be done before building the profiles: First of all, the data has to be filtered and cleaned up (elimination of noise). Furthermore, user identification and session identification have to be ensured. In the second part, refining of the data is done to get representative user profiles through Pattern Discovery: Knowledge about the user is discovered by applying machine learning and statistical techniques, association discovery and sequential pattern discovery to the data. One approach is to cluster text to create browsing patterns from the collected data of visited websites. Classification of the user is another way of creating a profile. There are several methods to do this. For example the sites of the user‟s history can be looked up in the Open Directory Project.

Open Directory Project (ODP) is a human-edited index of Web sites, also known as DMOZ, an acronym for "Directory Mozilla."The purpose of the ODP is to list and categorize sites, not to rank or promote them.

Pattern Discovery

The pattern used as a word or phrase that is extracted from the text document. There are numbers of patterns which may be discovered from a text document, but not all of them are interesting. Only those evaluated to be interesting in some manner are viewed as useful knowledge. Decision trees may also be used for this. Finally, some Post-Processing (like evaluations) can be applied to further refine the profiling.

4.7.3.Profile representation

After the actual profile creation, the extracted information has to be stored somehow.For Personalized Web Search the extracted knowledge is incorporated in a Personalization module in order to facilitate the personalization functions. Many different forms of data representation can be applied here. The two most common representations seem to be the rather simple ”list of URLs” and the ”bag of words”. Different tables or tensors can be used as well (for example: all

queries by a user and/or clicked results for every query). the use of classification is also quite common. The categories for the classification can be taken from the Open Directory Project (or from the Yahoo! Directory), selected from another given hierarchy or self-defined.

4.8 Collaborative filtering and content-based recommendation of documents and products

What is CF?

In a nutshell it is recommendation by "word of mouth" . Oxford dictionary defines CF as

"The process in which the purchaser of a product or service tells friends, family,neighbors, and associates about its virtues, especially when this happens in advance of media advertising."

Example,(in social networking) suppose that I read a book, I like it, and recommend it to my friends. Those of my friends who have a similar taste in books to mine may decide to read the book and then recommend it to their friends.

In an e-commerce site, this process may be automated as follows When I buy a book, this in itself is an , but the site could ask me for an explicit rating of the book, say on a scale of 1 to 10.

When my friend logs onto the site, the CF system will be able to deduce that his taste in books is similar to mine, since we have purchased similar items in the past. The system will also notice that he has not yet bought the book that I have rated highly, and then recommend this book to my friend. This is the essence of collaborative filtering.

The system will collect as many recommendations as it can and score them according to their overall popularity before presenting the top recommendations to the user. CF has applications also in e-learning in order to recommend content to teachers and students, and, in general, it can be used in web navigation to recommend links to like-minded surfers.

a) User-Based Collaborative Filtering

	Item				
User	Data Mining	Search Engines	Databases	XML	...
Alex	1		5	4	...
George	2	3	4		...
Mark	4	5		2	...
Peter			4	5	...

User-Item Rating Matrix

Consider the user–item matrix shown in Table 9.3. Each row represents a user,and each column represents an item. A number in the *i* th row and *j* th column is the rating that user *i* assigned to item *j* ; an empty cell indicates that the userdid *not* rate that item. The ellipsis (· · ·) at the end of each row indicates that we have shown only a small fraction of the items.
when an item has not been rated yet, questioning how can it be recommended. An e-commerce site may still want to promote items having no rating, and in this case a *content-based* approach is necessary.

b)Item-Based Collaborative Filtering

Item-to-item recommendation systems try to match similar items that have been co-rated by different users, rather than similar users or customers that have overlapping interests in terms of their rated items. Item-to-item CF looks at column similarity rather than row similarity, in user-based methods.

c)Model based collaborative filtering

There have been several other methods, which use machine learning techniques to build a statistical model of the user–item matrix that is then used to make predictions. One such technique trains a neural network for each user, which learns to predict the user rating for a new item. Another technique builds association rules such as"90% of users who like items i and j also like item k, 30% of all users like all these items."

The rules are generally of the form X ⇒ Y, where X is a set of items and Y is another item, as in user-based algorithms. In this case, the rule is {i, j} ⇒ {k}.The 30% in the rule refers to its support,(i.e) out of all the users in the user–item matrix, 30% like all three items (this includes the items in both X and Y). The 90% refers to the confidence of the rule; that is, it is the proportion of users who like all three items (this includes the items in either X or Y) out of the proportion of users who like only i and j (this includes only the items in X).

Yet another technique uses the naive Bayes classifier with the user–item matrix being the input states an item is liked by the active user, given other user ratings, as being proportional to the product of the probabilities of each user liking an item given that the active user likes an item.

4.8.1 Content-based recommendation of documents

There are two problems in User-Based Collaborative Filtering .A user would normally rate (or purchase) only a few products, say 30, out of the millions that may be available, so that the user–item matrix is very sparse. Another related problem is the first-rater problem, when an item has not been rated. An e-commerce site may still want to promote items having no rating, and in this case a content-based approach is necessary. For content-based systems to work, the system must be able to build a profile of the users interests.

The user interests include the categories he/she prefers in relation to the application. Example, does the user prefer fiction to nonfiction books, and pop music to classical music. Once the system has a user profile, it can check similarity of the item (or content) a user is viewing to the profile, and according to the degree of similarity create a rating for the item (or content). This is much like the search process, where, in this case, the profile acts as a query and the items presented to the user acts as the query results. The higher the item is rated, the higher is its ranking when presented to the user.

4.9 Handling "invisible" Web

What we access every day through popular search engines like Google, Yahoo or Bing is referred to as the Surface Web. These familiar search engines crawl through tens of trillions of pages of available content and bring that content to us on demand. However, this represents only the tip of the iceberg. Beneath the Surface Web is what is referred to as the Deep or Invisible Web. It is comprised of:

- Private websites, such as VPN (Virtual Private networks) and sites that require passwords and logins
- Limited access content sites (which limit access in a technical way, such as using Captcha, Robots Exclusion Standard or no-cache HTTP headers that prevent search engines from browsing or caching them)
- Unlinked content, without hyperlinks to other pages, which prevents web crawlers from accessing information
- Textual content, often encoded in image or video files or in specific file formats not handled by search engines
- Dynamic content created for a single purpose and not part of a larger collection of items
- Scripted content, pages only accessible using Java Script, as well as content downloaded using Flash and Ajax solutions

There are many high-value collections to be found within the invisible web. Some of the material found there that most people would recognize and, potentially, find useful include:

- Academic studies and papers
- Blog platforms
- Pages created but not yet published
- Scientific research
- Academic and corporate databases
- Government publications
- Electronic books
- Bulletin boards
- Mailing lists

- Online card catalogs
- Directories
- Many subscription journals
- Archived videos
- Images

4.9.1 Open Access Journal Databases

Open access journal databases (OAJD) are compilations of free scholarly journals maintained in a manner that facilitates access by researchers and others who are seeking specific information or knowledge. Because these databases are comprised of unlinked content, they are located in the invisible web. A sample list of well-regarded and reputable databases are "AGRIS" (International Information System for Agricultural Science and Technology), "BioMed Central", "Copernicus Publications", "Directory of Open Access Journals etc.

4.9.2 Invisible Web Search Engines

The search engines that deliver results from the invisible web are distinctly different. Narrower in scope, these deep web engines tend to access only a single type of data. This is due to the fact that each type of data has the potential to offer up an outrageous number of results. An inexact deep web search would quickly turn into a needle in a haystack. That"s why deep web searches tend to be more thoughtful in their initial query requirements. Below is a list of popular invisible web search engines:

- "Clusty" is a meta search engine that not only combines data from a variety of different source documents, but also creates "clustered" responses, automatically sorting by category.
- "CompletePlanet" searches more than 70,000 databases and specialty search engines found only in the invisible web. A search engine as well-suited to casual searchers as it is to researchers.
- "DigitalLibrarian": A Librarian"s Choice of the Best of the Web is maintained by a real librarian. With an eclectic mix of some 45 broad categories, Digital Librarian offers data from categories as diverse as Activism/Non Profits and Railroads and Waterways.
- "InfoMine" is another librarian-developed internet resource collection, this time from The Regents of the University of California.
- "InternetArchive" has an eclectic array of categories, starting with the „Wayback Machine," which allows the searcher to locate archived documents, and including an archive of Grateful Dead audience and soundboard recordings. They offer 6 million texts, 1.5 million videos, 1.9 million audio recordings and 126K live music concerts.
- "The Internet Public Library" (ipl and ipl2) is a non-profit, student-run website at Drexel University. Students volunteer to act as librarians and

respond to questions from visitors. Categories of data include those directed to Children and Teens.

4.9.3 Ways to Make Content More Visible

The idea of making content more visible spawned the Search Engine Optimization (SEO) industry. Some ways to improve your search optimization include:

- **Categorize your database**. If you have a database of products, you could publish select information to static category and overview pages, thereby making content available without form-based or query-generated access. This works best for information that does not become outdated, like job postings.
- **Build links within your website, interlinking between your own pages**. Each hyperlink will be indexed by spiders, making your site more visible.
- **Publish a sitemap**. It is crucial to publish a serially linked, current sitemap to your site. It‟s no longer considered a best practice to publicize it to your viewers, but publish it and keep it up to date so that spiders can make the best assessment of your site‟s content.
- **Write about it elsewhere**. One of the easiest forms of Search Enging Optimization (SEO) is to find ways to publish links to your site on other webpages. This will help make it more visible.
- Use **social media to promote your site**. Link to your site on Twitter, Instagram, Facebook or any other social media platform that suits you. You‟ll drive traffic to your site and increase the number of links on the Internet.
- **Remove access restrictions**. Avoid login or time-limit requirements unless you are soliciting subscriptions.
- **Write clean code**. Even if you use a pre-packaged website template without customizing the code, validate your site‟s code so that spiders can navigate it easily.
- Match your site‟s page titles and link names to other text within the site, and **pay attention to keywords that are relevant to your content**.

4.9.4 How to Access and Search for Invisible Content

If a site is inaccessible by conventional means, there are still ways to access the content, if not the actual pages. For invisible content that cannot or should not be visible, there are still a number of ways to get access:

- Join a professional or research association that provides access to records, research and peer-reviewed journals.
- Access a virtual private network via an employer.
- Request access; this could be as simple as a free registration.

- Pay for a subscription.
- Use a suitable resource. Use an invisible Web directory, portal or specialized search engine such as Google Book Search, Librarian‟s Internet Index, or BrightPlanet‟s Complete Planet.

4.9.5 Invisible Web Search Tools

Here is a small sampling of invisible web search tools (directories, portals, engines) to help you find invisible content.

- Art – Musie du Louvre
- Books Online – The Online Books Page
- Economic and Job Data – FreeLunch.com
- Finance and Investing – Bankrate.com
- General Research – GPO‟s Catalog of US Government Publications
- Government Data – Copyright Records (LOCIS)
- International – International Data Base (IDB)
- Law and Politics – THOMAS (Library of Congress)
- Library of Congress – Library of Congress
- Medical and Health – PubMed
- Transportation – FAA Flight Delay Information

4.10 Snippet generation

Snippet , a short summary of the document, which is designed so as to allow the user to decide its relevance. Typically, the snippet consists of the document title and a short summary, which is automatically extracted.

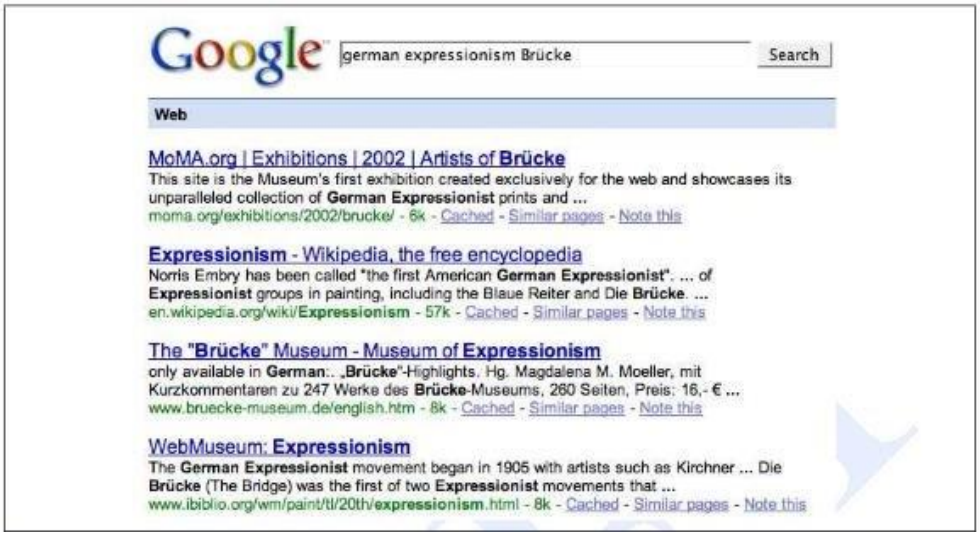


Fig 4.10.1 The first 4 snippets from Google for German Expressionism Brucke

For example, the above Fig. shows some sample snippets from Google summarizing the first
four documents returned from the query German Expressionism Brucke.

Fig 4.10.2 Example of product reviews rich snippets:

4.11 Summarization, Question Answering

Question answering (QA) and summarization, tasks produces specific phrases, sentences, or short passages, often in response to a user‟s need for information expressed in a natural language query. Rather than make the user read through an entire document, we‟d often prefer to give a single concise short answer. The simplest form of question answering is dealing with factoid questions .The answers to factoid questions are simple facts that can be found in short text strings. The following are examples of this kind.

- Who founded Virgin Airlines?
- What is the average age of the onset of autism?
- Where is Apple Computer based?

Each of these questions can be answered directly with a text string that contain the name of person, a temporal expression, or a location, respectively. Factoid questions, therefore, are questions whose answers can be found in short spans of text and correspond to a specific, easily characterized, category, often a named entity of the kind. These answers may be found on the Web, or alternatively within some smaller text collection. Sometimes we are seeking information whose scope is greater than a single factoid, but less than an entire document. In such cases we might need a summary of a document or set of documents.

The goal of text summarization is to produce an abridged version of a text which contains the important or relevant information. The two basic kinds of summaries are static , which are always the same regardless of the query,

and dynamic (or query-dependent), which are customized according to the user's information need as deduced from a query.

A static summary is generally comprised of either or both a subset of the document and *metadata* associated with the document. The simplest form of summary takes the first two sentences or 50 words of a document, or extracts particular zones of a document, such as the title and author.

Dynamic summaries display one or more ``windows'' on the document, aiming to present the pieces that have the most utility to the user in evaluating the document with respect to their information need. Usually these windows contain one or several of the query terms, and so are often referred to as *keyword-in-context* ( ) snippets.

Important kinds of summaries that are the focus of current research include:

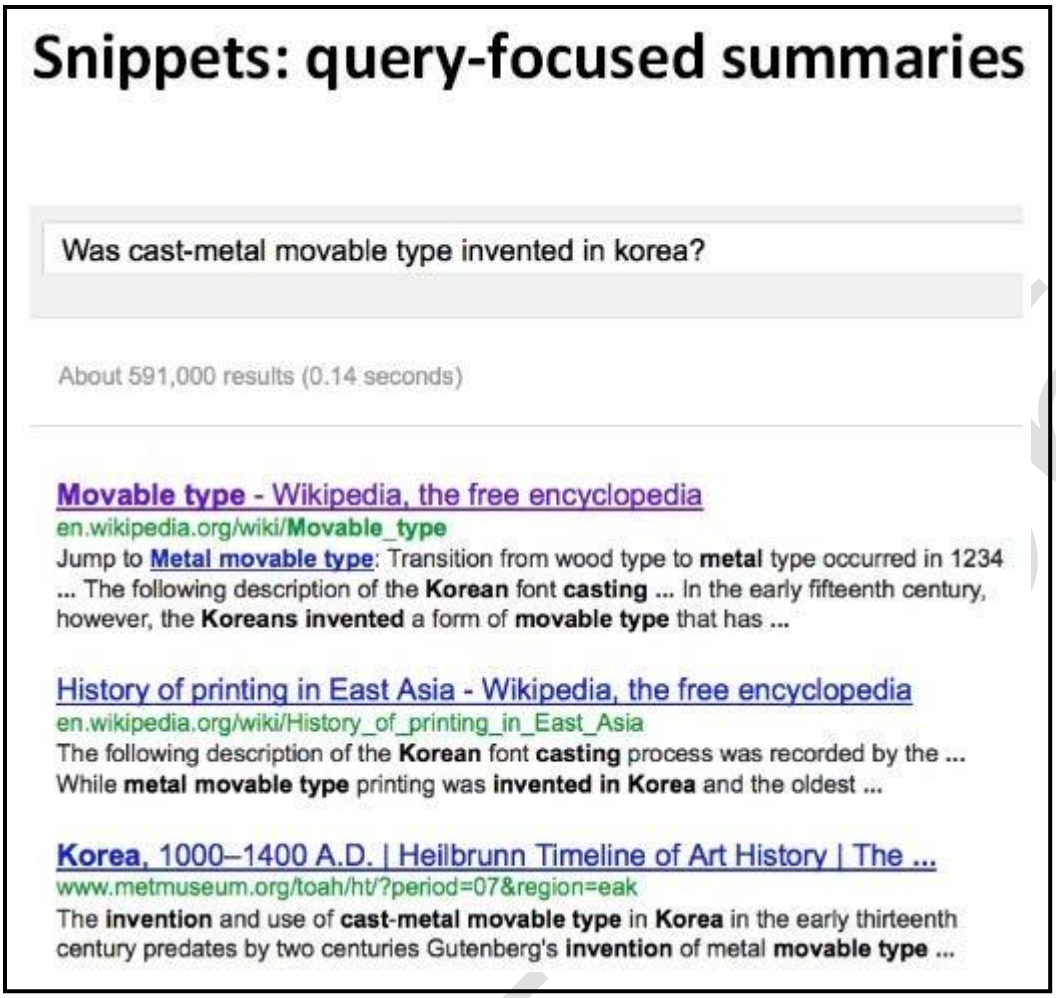
•outlines of any document
•abstracts of a scientific article
•headlines of a news article
•snippets summarizing a web page on a search engine results page
•action items or other summaries of a (spoken) business meeting
•summaries of email threads
•compressed sentences for producing simplified or compressed text
•answers to complex questions, constructed by summarizing multiple documents

These kinds of summarization goals are often characterized by their position on two dimension

- single document versus multiple document summarization
- generic summarization versus query-focused summarization

A generic summary is one in which we don‟t consider a particular user or a particular information need; the summary simply gives the important information in the document(s).

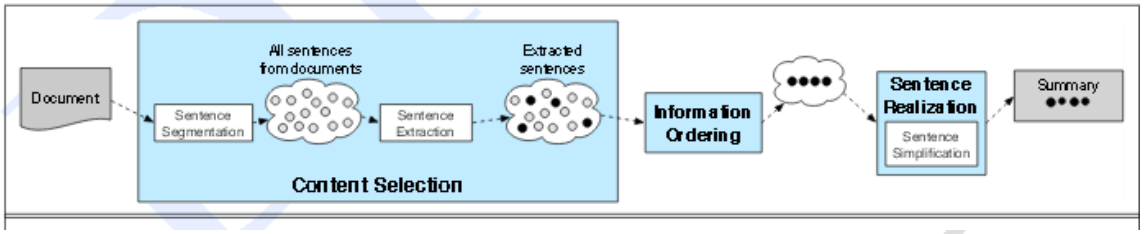
By contrast, in query-focused summarization , also called focused summarization, topic-based summarization and user-focused summarization, the summary is produced in response to a user query. We can think of query-focused summarization as a kind of longer, non-factoid answer to a user question.



4.11.1 Summarizing Single Documents

In single document summarization we are given a single document and produce a summary. Single document summarization is thus used in situations like producing a headline or an outline, where the final goal is to characterize the content of a single document.

Consider the task of building an extractive summary for a single document.
Assuming that the units being extracted are at the level of the sentence.
The three summarization stages for this task are:
1.Content Selection:
Choose sentences to extract from the document
2.Information Ordering:
Choose an order to place these sentences in the summary
3.Sentence Realization:
Clean up the sentences, for example by removing non-essential phrases from each sentence, or fusing multiple sentences into a single sentence, or by fixing problems in coherence.

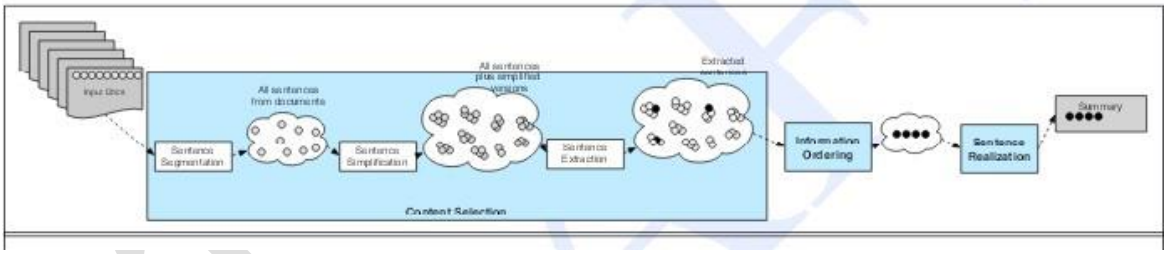


The basic architecture of a generic single document summarizer

4.11.2 Summarizing Multiple Documents

In multiple document summarization , the input is a group of documents, and our goal is to produce a condensation of the content of the entire group. We might use multiple document summarization when we are summarizing a series of news stories on the same event, or whenever we have web content on the same topic that we"d like to synthesize and condense.

Multi-document summarization algorithms are based on the same three steps we"ve seen before. In many cases we assume that we start with a cluster of documents that we"d like to summarize, and we must then perform content selection, information or-dering, and sentence realization.



The basic architecture of a multi-document summarize

4.12 Cross- Lingual Retrieval.

Cross-language information retrieval (CLIR) is a subfield of information retrieval dealing with retrieving information written in a language different from the language of the user's query. For example, a user may pose their query in English but retrieve relevant documents written in French. To do so, most of CLIR systems use translation techniques.

CLIR techniques can be classified into different categories based on different translation resources:

- Dictionary-based CLIR techniques
- Parallel corpora based CLIR techniques
- Comparable corpora based CLIR techniques
- Machine translator based CLIR techniques

4.12.1 Challenges in CLIR

We face the following challenges in creating a CLIR system:

1. Translation ambiguity: • While translating from source language to target language, more than one translation may be possible. Selecting appropriate translation is a challenge. • For example, the word मान (maan, respect/neck) has two meanings neck and respect.

2. Phrase identification and translation • Indentifying phrases in limited context and translating them as a whole entity rather than individual word translation is difficult.

3. Translate/transliterate a term: • There are ambiguous names which need to be transliterated instead of translation. • For example, भास्कर (Bhaskar, Sun) in Marathi refers to a person"s name as well as sun. Detecting these cases based on available context is a challenge.

4. Transliteration errors: • Errors while transliteration might end up fetching the wrong word in target language.

5. Dictionary coverage • For translations using bi-lingual dictionary, the exhaustiveness of the dictionary is important criteria for performance on system.

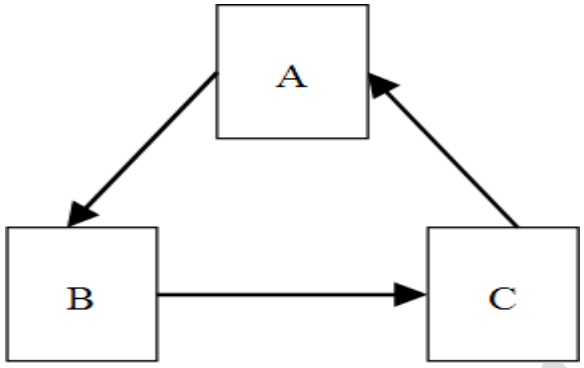
6. Font: • Many documents on web are not in Unicode format. These documents need to be converted in Unicode format for further processing and storage.

7. Morphological analysis (different for different languages)

8. Out-of-Vocabulary (OOV) problems • New words get added to language which may not be recognized by the system.

PageRank Examples

Example 1



The number of web pages $N = 3$
The damping parameter $d = 0.7$

$$PR(A) = (1 - d) \times (1 / N) + d \times (PR(C) / 1)$$
$$PR(B) = (1 - d) \times (1 / N) + d \times (PR(A) / 1)$$
$$PR(C) = (1 - d) \times (1 / N) + d \times (PR(B) / 1)$$

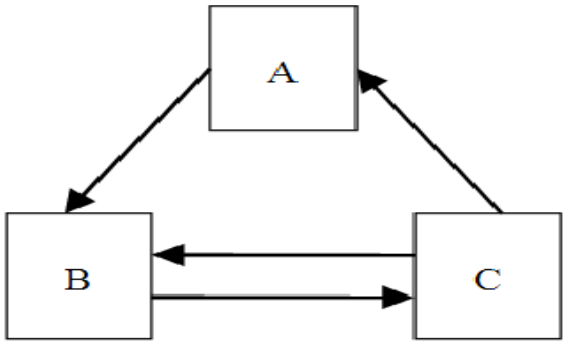
So

$$PR(A) = 0.1 + 0.7 \times PR(C)$$
$$PR(B) = 0.1 + 0.7 \times PR(A)$$
$$PR(C) = 0.1 + 0.7 \times PR(B)$$

By solving the above system of linear equations, we get

$$PR(A) = 1/3 = 0.33$$
$$PR(B) = 1/3 = 0.33$$
$$PR(C) = 1/3 = 0.33$$

Example 2



The number of web pages $N = 3$
The damping parameter $d = 0.7$

$$PR(A) = (1 - d) \times (1 / N) + d \times (PR(C) / 2)$$
$$PR(B) = (1 - d) \times (1 / N) + d \times (PR(A) / 1 + PR(C) / 2)$$
$$PR(C) = (1 - d) \times (1 / N) + d \times (PR(B) / 1)$$

$$PR(A) = (1 - d) \times (1 / N) + d \times (PR(C) / 2)$$
$$PR(B) = (1 - d) \times (1 / N) + d \times (PR(A) / 1 + PR(C) / 2)$$
$$PR(C) = (1 - d) \times (1 / N) + d \times (PR(B) / 1)$$

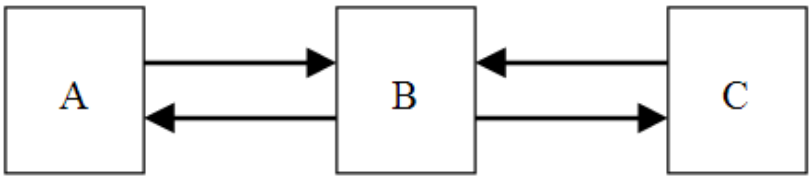
So

$$PR(A) = 0.1 + 0.35 \times PR(C)$$
$$PR(B) = 0.1 + 0.70 \times PR(A) + 0.35 \times PR(C)$$
$$PR(C) = 0.1 + 0.70 \times PR(B)$$

By solving the above system of linear equations, we get

$$PR(A) = 0.2314$$
$$PR(B) = 0.3933$$
$$PR(C) = 0.3753$$

Example 3



The number of web pages $N = 3$
The damping parameter $d = 0.7$

$$PR(A) = (1 - d) \times (1 / N) + d \times (PR(B) / 2)$$
$$PR(B) = (1 - d) \times (1 / N) + d \times (PR(A) / 1 + PR(C) / 1)$$
$$PR(C) = (1 - d) \times (1 / N) + d \times (PR(B) / 2)$$

So

$$PR(A) = 0.1 + 0.35 \times PR(B)$$
$$PR(B) = 0.1 + 0.70 \times PR(A) + 0.70 \times PR(C)$$
$$PR(C) = 0.1 + 0.35 \times PR(B)$$

By solving the above system of linear equations, we get

$$PR(A) = 0.2647$$
$$PR(B) = 0.4706$$
$$PR(C) = 0.2647$$