



Page Rank Analysis - Analysis for Indian Languages

¹ Dr.Parashuram Baraki,² Dr.Sunil Kumar G,

³ Prashanth K, ⁴ Anitha G, ⁵ Malatesh K

¹Professor & HOD Dept of CSE, HIT Nidasoshi,
hod.cse@hsit.ac.in, +919686042385

²Professor, Dept of CSE, VVIT Bangalore,
gsuneel.k@gmail.com, +919916382926

³Asst.Professor, Dept of CSE, PDIT, Hosapete,
prashanthkogali@gmail.com, +919449255277

⁴Asst.Professor, Dept of ISE, BIET, Davanagere,
basutha.anitha@gmail.com, +919036926295

⁵Asst.Professor, Dept of CSE, PDIT, Hosapete,
maltkpl@gmail.com, +919972005003

June 19, 2018

Abstract

Information retrieval is a branch of computer science which is gaining exponential popularity due to ever increasing amount of data being hosted on Internet. With increase in amount of data hosted on Internet, comes an increase in need of a search engine. Ranking models are techniques used in search engine to find the query relevant documents and to rank them in decreasing order of relevance and hence is an integral part of a search engine. This ranking depends on many factors one of which is the link structure of the web. In this paper, we will provide some of the methods which can be used to exploit this link structure of the web-graph.

We will then dig deeper into the PageRank algorithm- an algorithm which provides scores to documents based on links. We will conduct experiments to see the impact of PageRank algorithm on Indian language Hindi. We will combine the PageRank score with relevance score. Finally, we analyze the results and propose some solutions to overcome the limitations of the standard PageRank algorithm.

Key Words:PageRank,Crawling,Indexing,Retrieval and Ranking

1 INTRODUCTION

In a standard Information Retrieval System, the flow of the process is as follows: The documents from the internet are downloaded and converted into a representational form like metadata or index or inverted index. Then user can enter a search-query as input to the search-engine which is then matched with the index. This matching can be metadata matching or full-text matching. Matching documents are retrieved based on the similarity with query-model. Retrieved documents are then scored based on some ranking models, e.g. Vector Space Model, Probabilistic Model etc. from which top k documents are shown to the user. User can click on any document and see its content. Additionally, retrieved documents can also be summarized and converted into snippets which can direct the user in opening related document. The procedure explained above can be summarized into 3 main steps [10]:

1. Crawling
2. Indexing
3. Retrieval and Ranking

1.1 Introduction to NLP

Natural Language Processing is a field of computer science, artificial intelligence and linguistics that explores how computers can be used to understand and manipulate natural language text or speech to do useful things. It is the task of analyzing and generating by computers, languages that humans speak, read and write.

The tasks in NLP might be to translate to another language, to comprehend and represent the content of text, to build a database or generate summaries, or to maintain a dialogue with a user as part of an interface for database/information retrieval. As shown in the following figure, NLP is concerned with questions involving three dimensions: language, algorithm and problem [7].

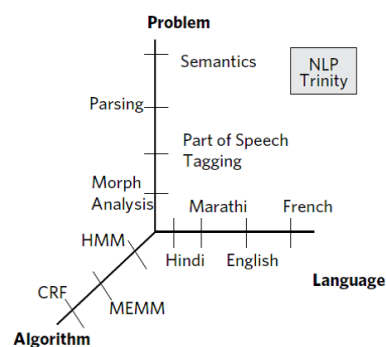


Fig. 1.1 Three dimensions of NLP

The Language axis includes different natural languages and linguistics. The Problem axis mentions different NLP tasks like morphology, part of speech tagging etc. The Algorithm axis represents different algorithms like Hidden Markov Model (HMM), Maximum Entropy Markov Model (MEMM) and Conditional Random Fields (CRF) to solve NLP problems.

1.1.1 Stages of NLP and associated ambiguities

Following are the traditionally accepted stages of NLP [7]:

Phonology and Phonetics

At this stage, utterances are processed. Two common problems are homophony and word boundary recognition. Homophony arises when two words sound same, but their meanings are widely different. For example, bank(river-bank) and bank(financial bank). Word boundary recognition arises in case of rapid speech. For example, I got up late and I got a plate - both the sentences sound very similar.

Morphology

Words are formed from root words or lexemes through processes of inflexion, derivation, back formation, clitics and portmanteauing. Languages differ in their morphological richness. For example, Dravidian languages have rich morphology whereas English has relatively simpler morphology. The ambiguity at this level arises from the choices available for breaking the word into stem and suffix.

Lexicon

Words are stored in the lexicon along with some useful information that helps processing in further stages of NLP. For example, word dog might be stored in the lexicon with the information like: POS (noun) Semantic Tag (Animate, 4-legged) Morphology (takes s in plural) Words may have multiple meanings even in the same part of speech, e.g. dog means an animal and a very detestable person. Ambiguities at this stage are homography and polysemy. Homography means two words spelt same and having different meanings. The example is bank (river-bank) and bank(financial bank) as explained earlier. Polysemy implies shades of meaning, e.g. falling of tree and falling of a kingdom.

Parsing

Parsing (Syntactic Processing) includes representation of linear sequence of words into hierarchical structure. Ambiguity at this stage is **Structural Ambiguity**, which is further categorized into two subtypes - **scope ambiguity** and **attachment ambiguity**. Example of scope ambiguity: Old men and women were taken to safe locations.

The scope of the adjective is ambiguous. That is, is the structure (old men and women) or ((old men) and women)?

Example of attachment ambiguity:I saw the boy with a telescope. It is not clear who has the telescope, I or the boy?

Semantics

After the formation of hierarchical structure, this stage deals with the meaning extraction. The sentence is represented in one of the unambiguous forms like predicate calculus, semantic net, frame, conceptual dependency, conceptual structure etc. The ambiguity

at this stage arises out of ambiguities of semantic roles and representations, e.g. Visiting aunts can be trying. Here, are the aunts visitors (agent role) or are they being visited (object role)?

Pragmatics

This stage involves tasks like processing user intention, sentiment, belief world, modals etc. These are very complex tasks as they need to deal with larger context, history, intent, sentiment, tone etc. The following example illustrates how complex this task may be:

Tourist (checking out of the hotel): Waiter, go upstairs to my room and see if my sandals are there; do not be late; I have to catch the train in 15 minutes. Waiter (running upstairs and coming back panting): Yes sir, they are there.

It is clear that the waiter didn't understand the pragmatics of the situation and fell short of the expectation of the tourist.

Discourse

This is the task of processing connected sentences. The following sequence of sentences shows how, after each sentence, the thought-process of human changes.

John was coming dejected from the school.

(John is most likely a student.)

He could not control the class.

(John is Most likely the teacher.)

Teacher should not have made him responsible.

(John is Most likely the monitor.)

After all he is just a janitor.

(all previous assumptions are wrong.)

1.2 Introduction to Information Retrieval - A brief history of Information Retrieval

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 [11]. The first automated information retrieval systems were introduced in the 1950s and 1960s. Several works emerged in the mid-1950s that elaborated upon the basic idea of searching text with a computer. One of the most influential methods was described by H.P. Luhn in 1957, in which (put simply) he proposed using words as indexing units for documents and measuring word overlap as a criterion for retrieval. 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 with the aim to encourage research in IR from large text collections [9]. The introduction of web search engines has boosted the need for very large scale retrieval systems even further.

1.2.1 Steps of Information Retrieval system

In a standard Information Retrieval System, the flow of the process is as following: The documents from the internet are downloaded and converted into a representational form like metadata or index or inverted index. Then user can enter a search-query as input to the search-engine which is then matched with the index. This matching can be metadata matching or full-text matching. Matching documents are retrieved based on the similarity with query-model. Retrieved documents are then scored based on some ranking models, e.g. Vector Space Model, Probabilistic Model etc. from which top k documents are shown to the user. User can click on any document and see its content. Additionally, retrieved documents can also be summarized and converted into snippets which can direct the user in opening related document. The procedure explained above can be summarized into 3 main steps [10]:

1. Crawling

2. Indexing

3. Retrieval and Ranking

The following figure gives an idea about how these steps work as a whole:

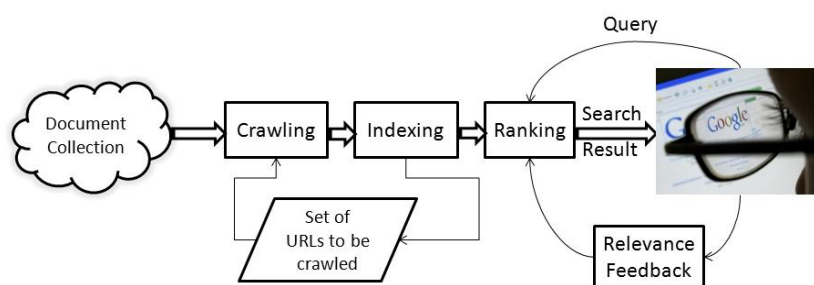


Fig.1.2.1. Components of an IR System

1.2.1.1. Crawling

Crawling is a process which periodically visits the web pages and copies them so that they can be indexed. It starts with a set of URLs and goes through the web pages represented by these URLs, adding each hyper link in web page to the list of URLs. Any new hyperlink can be blindly added to list of URLs or multiple checks can be performed before adding it to list of URLs. Thus, this process continues recursively.

1.2.1.2. Indexing

In this step, crawled pages in first step are converted into a representational form. Because of the index, we don't need to do a full-text search every time a query is fired. Thus, indexing makes search faster and efficient. There are many different techniques for indexing. Some of them are discussed here:

- **Suffix tree (Trie)** :Trie is sometimes referred to as Prefix tree, radix tree or digital tree. Trie is an ordered tree set data structure where keys are strings. Unlike binary tree no node stores the string

values but the path of the node from root depicts its value. All descendants of a node have common prefix. Values are normally associated with leaves.

• **Document matrix (Term frequency)** : This is a hash table base data structure where key represents the word. The value can be as simple as frequency of that word in document to a complex data structure which stores information such as frequency of word, word probability, root form, etc. This data structure is maintained for each document. The probability of a word depicts the importance of that word in the document.

• **Inverted index (Inverse document frequency)** : It is an index data structure storing a mapping from content to its locations in a document or a set of documents. The standard example of an inverted index is the index given at the end of a textbook which lets a person find the page- number from a keyword it is on. Thus, inverted index stores documents per word as opposed to the forward index which stores words per document. The purpose of an inverted index is to allow fast full-text search, at the cost of increased processing when a document is added to the database. The index is stored in the form of sparse matrix as all words belong to a very small subset of total documents. It will be more clear what an inverted index is from the following example:

Suppose there are 3 documents $T[0]$, $T[1]$ and $T[2]$ and their contents are as following:

$T[0]$: "it is what it is"

$T[1]$: "what is it"

$T[2]$: "it is a banana"

The inverted index for the above documents will be represented as following:

"a" : {2}

"banana" : {2}

"it" : {0, 1, 2}

"is" : {0, 1, 2}

"what" : {0, 1}

The above index indicates that the word "what" is contained in documents $T[0]$ and $T[1]$.

A search for the terms "what", "is" and "it" would give the set $\{0, 1\} \cap \{0, 1, 2\} \cap \{0, 1, 2\} = \{0, 1\}$.

Steps in Indexing

Consider a input query: "most powerful person of Indian history"

• Tokenization :

This process splits sentences of documents into morphological units. Each morphological unit must be a valid source language word. Incorrect words are corrected by spell correction.

The output for our query will be :

[most, power ful, person, of, Indian, history]

• Stop word elimination:

Stop words are the words of language which do not convey meaning of the context but improve fluency of sentence. Studies reveal that the words occurring with maximum frequencies are stop words. Stop words must be removed from query as well as document in order to truly match the concept of query with document. This module removes stop words like "to", "and", "as", "not" etc.

The output after stop-words elimination will be:

[power ful, person, Indian, history]

• Stemming :

Stemming is the process for reducing inflected (or sometimes derived) words to their stem, base or root forms (generally a written word form). It is usually sufficient that related words map to the same stem, even if this stem is not in itself a valid root. This module converts the words to stem forms, e.g. running, runner, runs are converted to run.

The output after stemming will be: [power, person, India, history]

1.2.1.3. Retrieval and Ranking

Boolean Ranking Model

The Boolean retrieval model is a model for information retrieval in which we can pose any query which is in the form of a Boolean expression of terms, that is, in which terms are combined with the operators AND, OR, and NOT. This is one of the simplest models which classifies the documents as relevant or irrelevant. The model views each document as just a set of words [1]. An example

query may be India OR England AND cricket AND NOT Pakistan.

Disadvantages of the Boolean Model:

- The weight assignment to documents are binary. So either the document is relevant or not relevant (1 or 0).
- Similarity function is boolean
- Exact-match only, no partial matches
- Retrieved documents not ranked, thus there is no way to express degree of relevance.
- All terms are equally important
- Query language is expressive but complicated

Vector Space Model

Vector space model or term vector model is an algebraic model for representing text documents as vectors of identifiers, such as, for example, index terms. Documents and queries are both vectors

$$d_j = (w_{1,j}, w_{2,j}, \dots, w_{t,j})$$

$$q = (w_{1,q}, w_{2,q}, \dots, w_{t,q})$$

where $w_{1,j}$ are weights associated with that word in document d and $w_{1,q}$ are weights associated with that word in query. It's a "bag-of-words representation".

If a term occurs in the document, its value in the vector is non-zero else zero. To find the weight of terms in the vectors i.e. term weights, there are many different ways. One of the best known schemes is term frequency-inverse document frequency (tf-idf) weighting.

• **Term frequency** of a word in a document is the frequency of that word in the documents.

• The **inverse document frequency** is a measure of how much information the word provides, that is, whether the term is common or rare across all documents. It is inversely proportional to frequency of that term in remaining documents.

• For calculating tf, Possibilities-

- Using **raw frequency** of a term in a document- $tf(t,d) = f(t,d)$ which can be obtained from the document term matrix.
- Using **boolean frequency** - $tf(t,d) = 1$ if occurs in d and 0

otherwise.

- Using **normalized frequency**: $tf(t,d) = f(t,d) / \sum f(t,d)$. Frequency is normalized to prevent bias towards longer documents. So after normalization, it denotes the portion of the document occupied by the index word.

For Calculating idf, possibilities-

- Logarithmically scaled fraction of the documents that contain the word. $idf(t,D) = \log (N / \text{no. of documents containing } t)$
- $idf(t,D) = 1 / \text{no. of documents containing } t$.

- The weight of an index term is directly proportional to frequency of that term in given document and inversely proportional to frequency of that term in remaining documents. Thus, the weight of index term is product of its Term frequency and Inverse Document Frequency popularly called as TF and IDF respectively.

$tf-idf(t, d, D) = \text{term frequency}(t, d) * \text{inverted document frequency}(t, D)$

- Then the relevance of document to query is calculated based on the similarity of the document vector to the query vector = cosine of the angle between the 2 vectors.

- Vector models give better results as the documents are retrieved based on the similarity between query and document.

- Further ranking on ranked set of documents is possible and thus this method can be used in combination with other ranking models. (The ranking model which we use to find the relevance score in the Analysis of PageRank on Indian Languages is based on this Vector Space Model)

Probabilistic Model

Given the query and document representations, an IR system has an uncertain guess of whether a document has content relevant to the information need. Probability theory provides a principled foundation for such reasoning under uncertainty. The Probabilistic Ranking Models are based on the Probability Theory and the Probability Ranking Principle [1]

Probability Theory:

The probability of joint events A and B is given by:

$$P(A, B) = P(A \cap B) = P(A | B)P(B) = P(B|A)P(A)$$

Also, as per the Partition Rule of Probability Theory:

$$P(B) = P(A, B) + P(A, \bar{B})$$

From this, we can derive Bayes Rule for inverting conditional probabilities:

$$P(A|B) = P(A, B) / P(B)$$

$$= P(A \cap B) / P(B)$$

$$= P(B|A) * P(A) / P(B)$$

$$= P(B|A) * P(A) / (P(A, B) + P(A, \bar{B}))$$

$$= P(B|A) * P(A) / (P(B-A) * P(A) + P(B-A) * P(\bar{A}))$$

$P(A)$: Prior Probability

$P(A|B)$: Posterior Probability after having seen the evidence/likelihood of occurrence of B.

Probability Ranking Principle:

- $P(R=1|d, q)$ Probability that the document d is relevant to the information need of query q

- $P(R=0|d, q)$ Probability that the document d is not relevant to the information need of query q

Probabilistic Ranking Principle (PRP) ranks the documents in the decreasing order of their probability of relevance to the information need i.e. in the decreasing order of $P(R=1|d, q)$.

Optimal Decision Rule, the decision which minimizes the risk of loss, is to simply return documents that are more likely relevant than non-relevant:

$$\text{dis relevant iff } P(R=1|d, q) \geq P(R=0|d, q)$$

2 LITERATURE REVIEW

[1] Information retrieval (IR) is finding material (usually documents) of an unstructured nature (usually text) that satisfies an information need from within large collections (usually stored on

computers). Obtaining the character sequence in a document, choosing a document unit, determining the vocabulary of terms like Tokenization, Dropping common terms: stop words, Normalization (equivalence classing of terms).

[2] In this paper we have studied Natural Language Processing from the perspective of ambiguity, multilinguality and resource constraint. First we described different kinds of ambiguity that obtain in NLP starting from the lowest level of processing, viz., morphology to the highest level, viz., pragmatics and discourse. Then we took up one specific ambiguity, viz., word sense and described ways of tackling it under constraints of resource, viz., annotated corpora. Multilinguality was leveraged in the sense of projecting sense distributions in the corpora from one language to another and wordnet parameters like distance between senses. Performance with and without projection were compared, and the idea of projection seemed well founded.

[3] Text Summarization is an active field of research in both the IR and NLP communities. Summarization is important for IR since it is a means to provide access to large repositories of data in an efficient way. It shares some basic techniques with indexing, since both indexing and summarization are concerned with identifying the essence of a document. High quality summarization requires sophisticated NLP techniques in addition, which are normally not studied in IR. In particular, for domains in which the aspects of interest can be pre-specified, summarization looks very much like Information Extraction. Summarization is therefore a good challenge problem to bring together techniques from different areas.

[4] Google is designed to be a scalable search engine. The primary goal is to provide high quality search results over a rapidly growing World Wide Web. Google employs a number of techniques to improve search quality including page rank, anchor text, and proximity information. Furthermore, Google is a complete architecture for gathering web pages, indexing them, and performing search queries over them.

[5] The CLIR System facilitates the Farmers of Tamil Nadu to

pose their query in transliterated format to translated target query words. Finally the documents are retrieved from a large corpus in English language. The system focuses the bilingual dictionary Translation technique to improve WSD rather than the word by word translation. The CLIR systems usually display the search result in English. This system can be additional comprehensive to Rank the pages and provide a summary (in English) of top pages. Document translation is to be considered for final targeted documents in future enhancement.

PROBLEM STATEMENT To analyze and compare the search-results retrieved with and without PageRank for Hindi languages. PageRank Analysis for Hindi includes following major task:

- Indexing Wikipedia corpus with Solr.
- Obtaining relevance score given by Solr.
- Computing PageRank scores for each document in the corpus.
- Combining PageRank scores with relevance scores

3 SYSTEM ARCHITECTURE

3.1 Dataflow Diagram

The system architecture diagram basically gives overview about the complete system and the module which compose the complete system. The important modules of our system are specified in the below given architecture diagram. Each module in our system has its own functionality and predefined roles to handle while search operation execution.

1. User Input/output module
2. Indexing module (Solr)
3. Ranking module

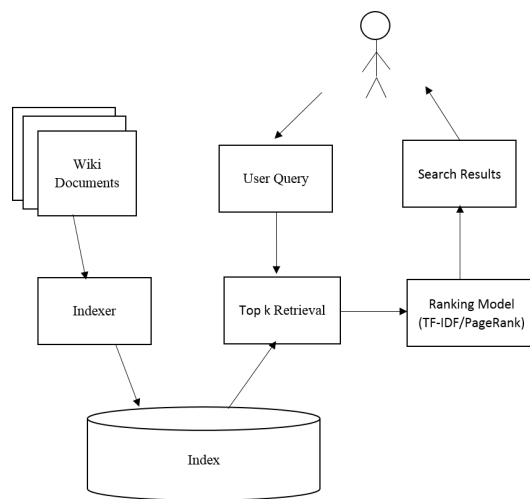


Fig. 3.1. Dataflow Diagram

User Input/Output Module

This module is used for the communication between an information processing system, such as a computer, and the outside world, possibly a human. Here inputs are queries received by the system and outputs are the links of the documents retrieved related to the given query.

Indexing Module(solr)

Apache Solr is a fast open-source Java search server. Solr enables you to easily create search engines which searches websites, databases and files. Here we have used it for indexing the Hindi Wikipedia documents. Solr is powered by Lucene under the hood. The relationship between Solr and Lucene, is like that of the relationship between a car and its engine - car is driven by its engine.

Ranking Module

This module provides ranks to the documents. After User presses the search button, the URLs of the top 10 documents (or less if total retrieved documents are less than 10) are displayed. After viewing a result, the user can rate the document as one of the following: Relevant, Partially Relevant, Not Relevant and Junk

3.2 Sequence Diagram

A sequence diagram is an interaction diagram that shows how objects operate with one another and in what order. A sequence diagram is a good way to visualize and validate various runtime scenarios. These can help to predict how a system will behave and to discover responsibilities a class may need to have in the process of modeling a new system.

In our system we have below mentioned two important interaction scenarios where user interact with the system for information search.

1. Query search with PageRank Algorithm
2. Query search with TF-IDF

In both the scenarios user has to provide input query and one of the ranking model (TF-IDF/PageRank) to search.

4 IMPLEMENTATION

4.1 Indexing Wikipedia Corpus WithSolr

In this section, the procedure for indexing Wikipedia corpus with Solr-4.10.3 is explained in detail. To setup Solr to index Wikipedia, 3 files - schema.xml, solrconfig.xml and data-config.xml(user-given name) should be configured.

4.1.1 schema.xml

The schema.xml file contains all of the details about which fields your documents can contain, what their field-types are and how those fields should be dealt with when adding documents to the index, or when querying those fields. Following are the fields of a Wikipedia documents indexed along with their types.

1. id - string (document-id)
2. title - string (document-title)

3. revision - integer (document revision number)
4. user - string (user of the document)
5. userId - integer (user-id of the document-user)
6. text - text (content of the document)
7. titleText - text (document-title)

In the above code, `jsolrQueryParserdefaultOperator="OR" / >` indicates that the query-terms will be combined by OR operator. That means that all the documents containing at least one of the query terms will be retrieved. `jsolrQueryParserdefaultOperator="AND" / >` indicates that all the documents containing all the query-terms should be retrieved.

4.1.2 solrconfig.xml

`solrconfig.xml` is the file that contains most of the parameters for configuring Solr itself. It is used to configure request-handlers. Request handlers are responsible for accepting HTTP requests, performing searches, then returning the results. For indexing Wikipedia, Data-import Handler is used.

After configuring these 3 files, start the Solr using command: `path-to-Solr/example java -jar start.jar`. To complete the indexing process, type-in the following in the browser:

`http://localhost:8983/solr/update/dih?command=full-import`

The Wikipedia corpus for Hindi are indexed using the procedure explained as above. By default, Solr removes Redirect pages from the Wikipedia corpus. After the removal of redirect pages, the number of documents in Hindi corpus are 78378. The standard relevance scores for the documents retrieved for any query can be retrieved by Solr admin user-interface as well as programmatically.

4.2 Computing PageRank scores for each document in the corpus

4.2.1 PageRank Algorithm

This chapter covers more details and features of PageRank. PageRank was one of the 1st algorithm created by Google to rank documents. It is used by Google and many search engines even today however its importance has relatively decreased as there are several other factors now contributing to the final score of a document. PageRank is the measure of a pages quality or importance in the web and is based on the link structure of the graph. Its intuition is that the more number of in-links a page has, the more will be its authority and usefulness to the user. The behavior of PageRank is like that of a Random surfer. We assume that a user who is a "random surfer" is given a web page at random and keeps clicking on links, never hitting "back" but eventually gets bored, stops and starts on another random page. The damping factor d is the probability that at each page the "random surfer" will get bored and request another random page. Thus $(1-d)$ is the probability that user follows a hyperlink given on the current page. 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. When the surfer lands up on a page with no out-links, a random jump is taken to any page in the web-graph which is called **Teleport** operation. In other words, if N is the total number of nodes in the web graph, the teleport operation takes the surfer to each node with probability $1/N$. At each node, the random surfer jumps to a random page with probability d and follows a hyperlink on the current page with probability $(1-d)$. In this random walk, the surfer visits each node v of the web graph a fixed fraction of the time $\pi(v)$ that depends on (1) the structure of the web graph and (2) the value of α . We call this value $\pi(v)$ the PageRank of v [1]. In the coming sections, we have shown how PageRank is computed and an algorithm for it. We have then focused on some key features of PageRank, its limitations, usefulness, and finally we end with showing some variants of PageRank.

4.2.2 Calculation of PageRank

As per the Google Paper on PageRank [9], it is defined as: We assume page A has pages T1...Tn which point to it (i.e., are citations). The parameter d is a damping factor which can be set between 0 and 1. We usually set d to 0.85. Also C(A) is defined as the number of links going out of page A. The PageRank of a page A is given as follows:

$$\text{PR(A)} = (1-d) + d (\text{PR(T1)} / \text{C(T1)} + \dots + \text{PR(Tn)} / \text{C(Tn)})$$

- Thus this is an iterative formula.
- Initialize the PageRank of all pages to some value to prevent infinite looping. Often it is initialized to 1.
- A key Principle is that it doesn't matter where you start your guess, once the PageRank calculations have settled down, the normalized probability distribution (the average PageRank for all pages) will be 1.0
- PageRank corresponds to the principal eigenvector of the normalized link matrix of the web.
- PageRank scores are query independent i.e. they are computed once after indexing and before firing any query and can be used then for ranking all query results.

4.3 Combining PageRank Score and Relevance Score

In the previous section, we went through different algorithms for computing the score of a web page based on the link structure. Now once we have computed the scores for all documents, the next task is to use these to rank the results of a query q. We wish to combine the Link score of a page with the relevance score of that page obtained by some text-based ranking model, say, Boolean model, Vector-Space model, Probabilistic model, Okapi BM25 model etc. Some of the possible ways to combine these scores are-

If page P satisfies query Q, Score of P, S(P,Q) can be computed as:

- Use a **linear combination** of different ranking signals [2]

$$R(P,Q) = (\alpha * \text{Text-Based Rank Score}(P,Q)) + ((1 - \alpha) * \text{Link-}$$

Score(P)) $\alpha = 1$: text-based ranking, early search engines

$\alpha = 0$: link-based ranking, independent of the query

The value of α can be found out experimentally or through machine learning techniques.

- Use **multiplicative combination** of the scores $R(P,Q) = \text{Text-Based Rank Score}(P, Q) * \text{Link-Score}(P)$
- Use the link score to reorder the pages obtained by relevance score.

In practical Search engines like Google, Yahoo etc. there exits many other parameters other than these two scores which are used together to form a final score.

4.4 Analysis of PageRank for Indian Languages

In this chapter, we analyze the effect of PageRank algorithm on Indian Languages, namely Hindi. The web has large amount of data in English. However, the percentage of the web containing documents in Indian Languages is very less. We have already seen that the web can be viewed as a directed graph with nodes as pages and edges as links from one page to another. It has been an observation that interlinking between the documents in Indian languages is less as compared to English and so the web-graph of Indian Languages is sparser than that for English language. Now PageRank is directly related to the interlinking between the documents in the web-graph. We can say that more the interlinking in the web-graph, more the endorsement among pages and so more effective the PageRank scores are. Our aim is to check the effect of PageRank algorithm for sparsely connected web graph having less interlinking among the pages. We have conducted experiments for Hindi documents retrieval from Wikipedia corpus. We then analyze the results obtained for both Hindi and compare the search-results for a query With PageRank and Without PageRank. Section 4.2 provides the details of the experiment conducted for the analysis and section 4.3 gives the detail of a web-application developed through which one can fire queries and submit one's response for the search-results.

4.4.1 PageRank Analysis for Hindi

The major tasks are:

Indexing Wikipedia corpus with Solr

Obtaining relevance score given by Solr

Computing PageRank scores for each document in the corpus

Combining PageRank scores with relevance scores

The following sub-sections will illustrate this procedure in detail.

4.4.2 Indexing and searching using Solr

The Wikipedia corpus for Hindi are indexed using the procedure explained in chapter-6. By default, Solr removes Redirect pages from the Wikipedia corpus. After the removal of redirect pages, the number of documents in Hindi corpus are 78378. The standard relevance scores for the documents retrieved for any query can be retrieved by Solr admin user-interface as well as programmatically.

4.4.3 Computation of PageRank scores

As explained in the previous sub-section, as Solr removes redirect pages from the corpus, removal of such pages from the web-graph while computing the PageRank scores is also necessary to maintain the consistency between the set of documents considered by Solr for indexing and the set of documents in the web-graph considered for PageRank calculation. After removal of redirect pages from web-graph too, the PageRank Algorithm is applied to calculate the PageRank scores of each document in this new web-graph.

Key-points:

- Initial PageRanks of all pages set to 1.
- Iterations go on till PageRanks of all the documents stabilize up to 2 decimal points.

• Document IDs and corresponding PageRanks stored in file. **Observations:**

- The stabilization of PageRank values took about 55 iterations for Hindi corpus.

4.4.4 Combining PageRank scores with relevance scores

As explained in the section 7.5, there are different approaches possible to combine PageRank score with relevance score. Out of these approaches, we will take the multiplication approach into account for the analysis purpose. In this approach, the formula for the final score S of a document D for query Q is given by,

$$S(D,Q) = \text{Relevance score}(D,Q) * \text{PageRank score}(D)$$

4.5 Web Application for PageRank Analysis

Usually, to find out the efficiency of a search-system, a relevance document is available, but for Wikipedia corpus, there is no relevance document available. So, we cannot measure Precision and Recall values. So, to analyze and compare the results retrieved with and without PageRank, Subjective analysis has been done. We have developed a **Web-application** for our search system so that we can usually, to find out the efficiency of a search-system, a relevance document is available, but for Wikipedia corpus, there is no relevance document available. So, we cannot measure Precision and Recall values. So, to analyze and compare the results retrieved with and without PageRank, Subjective analysis has been done. We have developed a **Web-application** for our search system so that we can get the evaluation done by multiple users which will help in better subjective analysis of the results.

4.5.1 The features of the web application are:

- The web application supports searching in Hindi languages.
- Transliteration is provided for the input query. So a user can input a query either in directly the target language selected or can enter the target language words as they are in English. In this case, pressing space or tab after each word will convert the English word in the target language word. For example, if we have selected Hindi search and we want to search for "House" which is "ghar" in Hindi, we can enter either directly "ghar" or Ghar and press space or tab to convert it to "ghar".
- After entering a query, user can search with both With PageRank

and Without PageRank.

- After User presses the search button, the URLs of the top 10 documents (or less if total retrieved documents are less than 10) are displayed. After viewing a result, the user can rate the document as one of the following:

Relevant

Partially Relevant

Not Relevant

Junk

For evaluation, we have assigned following scores to each category-

Relevant : 1

Partially Relevant : 0.5

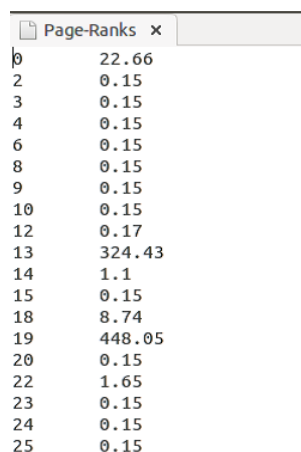
Not Relevant : 0

Junk : 0

For Analysis and comparison of results With and Without PageRank, on a user-level basis or on a query level basis, we can find a score by- $\text{Score } S = (\text{No. of documents judged as relevant} * 1) + (\text{No. of documents judged as partially-relevant} * 0.5) + (\text{No. of documents judged as not relevant} * 0) + (\text{No. of documents judged as junk} * 0)$

1. The class Dump_Preprocessor pre-processes the corpus and removes redirect pages from it. 2. The class ParseImpl parses the Wikipedia dump and extracts the outlinks on each document and stores it in file links which contains 2 fields - Web-graph id and

3. The Matrix-generator class calculates the PageRank scores of all the documents using the links file and stores them in PageRanksfile which contains 2 fields - Web-graph id and its PageRank score. 4. The classes mentioned above are controlled by the class PageRank.Now; the query-independent PageRank scores are computed.



0	22.66
2	0.15
3	0.15
4	0.15
6	0.15
8	0.15
9	0.15
10	0.15
12	0.17
13	324.43
14	1.1
15	0.15
18	8.74
19	448.05
20	0.15
22	1.65
23	0.15
24	0.15
25	0.15

Fig.4.3 : 'PageRanks' file

The process under the hood for searching:

1. When user enters a query, connection with Solr is established.
2. The search-results and their relevance scores given by Solr are retrieved programmatically.
3. The PageRank score obtained from PageRanksfile is multiplied with relevance score for each corresponding result-document to generate the final score.
4. The results are sorted according to the final score and shown to the user.

5 SYSTEM TESTING

System Testing is a black box testing technique performed to evaluate the complete system the system's compliance against specified requirements. In System testing, the functionalities of the system are tested from an end-to-end perspective

In our system testing we have two major test scenarios mentioned below. The functionality verified here are the searching the information which available in system and searching the information which is not available in the system.

Search with data available

In this scenario user has to enter the input query for searching, the expected information that is relevant to the user input query should be present in the system. Once the query executed over the indexed data the relevant documents will be retrieved as result set. The selected ranking model will be applied on this retrieved result set and the top k results will be computed and displayed for the user.

Input: Valid search query with ranking model.

Expected Output: Top k retrieved relevant documents

Search with data not available

In this scenario user has to enter the input query for searching, the expected information that is relevant to the user input query should not be present in the system. Once the query executed over the indexed data, there will not be any relevant documents retrieved and result set will be empty. The user search result page will not be displayed any information and result is empty.

Input: Valid search query with ranking model. (Unavailable data)

Expected Output: No relevant information (empty results)

6 CONCLUSION

In this paper, following topics have been covered:

- Introduction to NLP
- Steps of Information Retrieval and some tools for it
- Some of the Text-based Relevance Ranking Models and Link Based Ranking Models and issues regarding combining these two
- PageRank Algorithm, in some detail and limitations of its query-independent behavior
- Indexing and Searching with Solr

- Analysis of PageRank for Indian Languages

The experiments and subjective analysis conducted after applying PageRank on Hindi showed that the search-results without PageRank are better as compared to search-results with PageRank. The reason for this could be either one of the following or combination of both:

- Sparsely connected web-graph for Hindi
- The query-independent behavior of PageRank
- The Ranking function: Total Score= Relevance score * PageRank score

FUTURE SCOPE Making PageRank query dependent or domain dependent seems to overcome the problems with standard query-dependent model of PageRank. Future work can be done on:

- Computing PageRank on the sub-graph of the graph obtained as in HITS algorithm. However, as PageRank will have to be computed for each query, it may become time consuming and response time will increase. As the response time is a very important factor when we fire queries in any search engine, this issue has to be looked upon and appropriate methods will be needed to lessen the response time.
- Topic specific PageRank in which PageRank scores for each topic are pre-computed as in the standard PageRank, but here, a page will have multiple PageRank scores each for a different topic. Ways to infer the topic from the query have to be investigated for this.
- Use of Linear Ranking function:
Total score= α * Relevance Score + $(1-\alpha)$ * PageRank score
The value of α here is to be found out. It is possible that α is machine learnt.

References

- [1] An Introduction To Information Retrieval; Christopher D. Manning, PrabhakarRaghavan, HinrichSchtze; Cambridge University Press; Cambridge, England.
- [2] Pushpak Bhattacharyya (2012), Natural Language Processing: A Perspective from Computation in Presence of Ambiguity,

Resource Constraint and Multilinguality, Computer Science and Engineering at IIT Bombay.

- [3] University of Massachusetts Amherst (2002), Challenges in Information Retrieval and Language Modeling, Report of a Workshop held at the Center for Intelligent Information Retrieval.
- [4] Sergey Brin and Lawrence Page, The Anatomy of a Large-Scale Hypertextual Web Search Engine, Computer Science Department, Stanford University, Stanford.
- [5] AshishKankaria (2014), Cross Lingual Information Retrieval, Department of Computer Science and Engineering, Indian Institute of Technology - Bombay.
- [6] A. Jain et al. Page ranking algorithm in web mining, limitations of existing methods and a new method for indexing web pages. International Conference on Communication Systems and Network Technologies, IEEE, 2013.
- [7] M. Shamiul Amin et al. A score based web page ranking algorithm International Journal of Computer Applications, Volume 110, No.12, January 2015.
- [8] S. Gupta et al. New combined page ranking scheme in information retrieval system. International Journal of Scientific and Research Publications, Volume 4, Issue 4, ISSN: 2250-3153, April 2014.
- [9] A. Jain et al. Page ranking algorithm in web mining, limitations of existing methods and a new method for indexing web pages. International Conference on Communication Systems and Network Technologies, IEEE, 2013.
- [10] T. S. Govada and N. L. Prasanna Comparative study of various page ranking algorithms in web content mining International Journal of Advanced Research (IJAR), Volume 2, Issue 7, pp. 457-464, 2014.
- [11] http://en.wikipedia.org/wiki/Information_retrieval

