MTech Seminar Report

Ranking in Information Retrieval

Author:
Joydip Datta
Roll No. - 09305014
joydip@cse.iitb.ac.in

Under the guidance of:
Dr. Pushpak Bhattacharyya
pb@cse.iitb.ac.in

Submitted in the partial completion of the course CS 694

April 16, 2010

Department of Computer Science and Engineering,
Indian Institute of Technology, Bombay
Powai, Mumbai - 400076

Acknowledgement

I express my sincere gratitude to my guide Prof. Pushpak Bhattacharyya for his constant guidance and motivation. Without his valuable suggestions and insights writing this document would not be possible. The long sessions we had at lab really cleared my conceptions a lot. I also thank my senior, Mr. Vishal Vachhani for the long, insightful discussions we had. I thank my parents, friends and roommates for being with me till date.

- Joydip Datta

Abstract

In this report we study several aspects of an information retrieval with focus on ranking. First we introduce basic concepts of information retrieval and several components of an information retrieval system. Then we discuss important theoretical models of IR. Web-specific topics like link analysis and anchor text are presented next. We discuss how IR systems are evaluated and different IR evaluation forums. We end the report with a case study of cross lingual information retrieval system at IIT Bombay. In the future scope, we introduce upcoming trends in Web IR like user modeling and Quantum IR.

Contents

1 Introduction **1**
1.1 What is Information Retrieval . . . 1
1.2 Difference between information retrieval and data retrieval . . . 1
1.3 Components of an Information Retrieval System . . . 2
1.3.1 Crawling . . . 2
1.3.1.1 Selection policy . . . 2
1.3.1.2 Re-visit policy . . . 3
1.3.1.3 Politeness Policy . . . 3
1.3.2 Indexing . . . 3
1.3.2.1 Tokenization . . . 3
1.3.2.2 Stop-word eliminator . . . 3
1.3.2.3 Stemmer . . . 4
1.3.2.4 Inverted index . . . 4
1.3.2.4.1 Example . . . 4
1.3.3 Ranking . . . 4
1.3.4 Relevance Feedback . . . 4
1.3.4.1 Types of relevance feedback . . . 5
1.3.4.2 Issues with relevance feedback . . . 5
1.4 End Notes . . . 5
1.4.1 Organization of this document . . . 5

2 Theoretical Models in Information Retrieval **7**
2.1 Boolean Model . . . 7
2.1.1 Discussion . . . 9
2.1.1.1 Advantages . . . 9
2.1.1.2 Disadvantages . . . 9
2.2 Vector Based Model . . . 10
2.2.1 Document and Query Representation . . . 10
2.2.2 Modelling as a clustering method . . . 10
2.2.3 Fixing the term-weights . . . 10
2.2.3.1 Term-frequency . . . 10
2.2.3.2 Inverse document frequency . . . 11
2.2.4 Similarity measure between two vectors . . . 11
2.2.4.1 Cosine Similarity . . . 11
2.2.4.2 Implementation of Cosine Similarity . . . 12
2.2.5 Discussion . . . 13
2.2.5.1 Advantage . . . 13

2.2.5.2 Disadvantage . . . 13
2.2.6 The Nutch Ranking . . . 13
2.2.6.1 The Nutch Ranking Expression . . . 13
2.3 Probabilistic Model . . . 14
2.3.1 The Probabilistic Ranking Principle . . . 14
2.3.2 The Binary Independence Model . . . 15
2.3.2.1 Derivation of ranking function . . . 15
2.3.2.2 Estimating the probabilities in theory . . . 18
2.3.2.3 Estimating the probability in practice . . . 18
2.3.2.4 Recursive approximation using Relevance Judgments . . . 19
2.3.3 Discussion . . . 19
2.3.4 Okapi BM25 Ranking Function . . . 20
2.4 Fuzzy Set Theory Model . . . 21
2.5 End notes . . . 22

3 Searching the Web: Link Analysis and Anchor Text 23
3.1 Difference between Web IR and Traditional IR . . . 23
3.2 Link Analysis . . . 24
3.2.1 Web as a graph . . . 24
3.2.2 PageRank algorithm . . . 24
3.2.2.1 Random surfer model . . . 25
3.2.2.2 Example . . . 25
3.2.2.3 Random Surfing as a Markov Chain . . . 25
3.2.2.4 Calculating PageRank as a eigenvalue computation . . . 26
3.2.2.5 Calculation of PageRank . . . 27
3.2.2.6 Convergence of PageRank algorithm . . . 27
3.2.2.7 Disadvantage of PageRank . . . 27
3.2.3 HITS algorithm . . . 27
3.2.3.1 As a Topic Specific Search . . . 28
3.2.3.2 Computation of hub and authority scores . . . 29
3.2.3.2.1 As a principal eigenvalue computation . . . 29
3.2.3.2.2 The HITS algorithm . . . 30
3.2.4 OPIC Algorithm . . . 30
3.3 Anchor Text . . . 30
3.3.1 Importance of Anchor texts in web IR . . . 31
3.3.2 Disadvantage . . . 31
3.4 End Notes . . . 31

4 Evaluation of IR Systems 32
4.1 Different IR evaluation measures . . . 32
4.1.1 Precision . . . 32
4.1.2 Precision at Rank k . . . 32
4.1.3 Mean average precision or MAP score . . . 33
4.1.4 Recall . . . 33
4.1.5 F-Score . . . 33
4.2 IR Evaluation Forums . . . 33
4.2.1 TREC . . . 34
4.2.1.1 Goals of TREC . . . 34
4.2.1.2 TREC Tracks . . . 34

4.2.1.3 TREC Topics . . . 34
4.2.1.4 TREC Document Collection . . . 35
4.2.1.5 TREC Evaluation . . . 35
4.2.1.6 TREC - CLIR Track . . . 35
4.2.1.7 Official Website . . . 35
4.2.2 CLEF . . . 35
4.2.2.1 Official Website . . . 36
4.2.3 FIRE . . . 36
4.2.3.1 FIRE Tracks . . . 36
4.2.3.2 Official Website . . . 37
4.2.4 NTCIR . . . 37
4.2.4.1 Official Website . . . 37
4.3 End Notes . . . 37

5 Cross Lingual Information Retrieval 38
5.1 The CLIR System Architecture . . . 39
5.1.1 Input processing module . . . 39
5.1.2 Search Module . . . 41
5.1.3 Output Generation Module . . . 41
5.2 Ranking Scheme in IITB-CLIR system . . . 42
5.2.1 Query Formulation . . . 42
5.2.1.1 Fall-back strategy . . . 42
5.2.1.2 Query Disambiguation . . . 42
5.2.1.3 Phrasal Query . . . 43
5.2.2 Effect of OPIC Score . . . 43
5.2.3 Experimental results . . . 43

6 Conclusion and Future Direction 44
6.1 Personalizing Web search . . . 45
6.1.1 Query Recommendation . . . 45
6.1.2 Query Expansion . . . 46
6.1.3 Ranking . . . 46
6.1.4 Relevance Feedback . . . 46
6.2 Quantum Probabilistic Retrieval Model . . . 46
6.2.1 Young's Double Slit Experiment . . . 46
6.2.2 Analogy with IR process and the Quantum Probabilistic Retrieval Principle 48

Chapter 1

Introduction

1.1 What is Information Retrieval

Information Retrieval is the art of presentation, storage, organization of and access to information items. The representation and organization of information should be in such a way that the user can access information to meet his information need. The definition of information retrieval according to (Manning et al., 2009) is:

> 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).

Another feature of information retrieval is that it does not actually fetch documents. It only informs the user on the existence and whereabouts of documents relating to his query.

1.2 Difference between information retrieval and data retrieval

The difference between information retrieval and data retrieval is summarized in the following table (Rijsbergen, 1979):

	Data Retrieval	Information Retrieval
Example	Database Query	WWW Search
Matching	Exact	Partial Match, Best Match
Inference	Deduction	Induction
Model	Deterministic	Probabilistic
Query Language	Artificial	Natural
Query Specification	Complete	Incomplete
Items Wanted	Matching	Relevant
Error Response	Sensitive	Insensitive

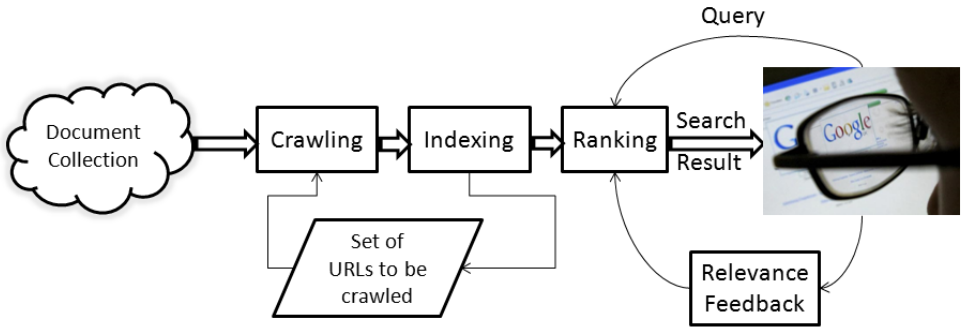

Figure 1.1: Important Processes in Web IR

1.3 Components of an Information Retrieval System

In this section we describe the components of a basic web information retrieval system. A general information retrieval functions in the following steps. It is shown in Figure 1.1.

1. The system browses the document collection and fetches documents. - Crawling
2. The system builds an index of the documents - Indexing
3. User gives the query
4. The system retrieves documents that are relevant to the query from the index and displays that to the user - Ranking
5. User may give relevance feedback to the search engine - Relevance Feedback.

The goal of any information retrieval system is to satisfy user's information need. Unfortunately, characterization of user information need is not simple. User's often do not know clearly about the information need. Query is only a vague and incomplete description of the information need. Query operations like query expansion, stop word removal etc. are usually done on the query.

1.3.1 Crawling

The web crawler automatically retrieves documents from the web as per some defined strategy. The crawler creates a copy of all the documents it crawls to be processed by the search engine. The crawler starts from a list of URLs (documents) called seed. The crawler visits the URLs, identifies the outgoing hyperlinks there and adds them to the list of URLs (documents) to be visited. This way the crawler traverses the web graph following hyperlinks. It saves a copy of each document it visits.

1.3.1.1 Selection policy

Selection policy determines which link to crawl first. Generally the web graph is traversed in a breadth first way to avoid being lost at infinite depth. As the number of documents is huge, the

selection strategy becomes critical so as to select which documents to crawl and which documents not to crawl. Generally page importance measures like PageRank are used as a selection policy.

1.3.1.2 Re-visit policy

The crawler needs to crawl frequently to keep the search results up-to-date. The revisit policy determines how frequently the crawling process should be restarted. There is a cost associated with an outdated copy of a document. The mostly used cost functions are freshness (is the stored copy outdated?) and age (how old is the stored copy). There may be two revisit policies:

Uniform policy Revisit all the documents in the collection with same frequency

Proportional policy Revisit documents that change frequently more often

It is interesting to note that proportional policy often incurs more freshness cost. The reason being, pages in the web either keep static or change so frequently that even the proportional policy cannot keep them up to date.

1.3.1.3 Politeness Policy

Being a bot, crawlers can retrieve documents much faster than human users. This way a crawler can easily overwhelm a website in terms of network resources, server overload etc. and degrade its performance. The politeness policy restricts a crawler so that these things do not happen. Different approaches in the politeness policy are listed below:

- Respect the robots exclusion principle: Do not crawl the portions of the web page indicated not to be crawled in the robots.txt file.
- Do not create multiple TCP connections with the same server
- Introduce a delay between two subsequent requests

1.3.2 Indexing

The documents crawled by the search engine are stored in an index for efficient retrieval. The documents are first parsed, and then tokenized, stop-word removed and stemmed. After that they are stored in an inverted index. The process is discussed below.

1.3.2.1 Tokenization

This stem extracts word tokens (index terms) from running text. For example, given a piece of text: “Places to be visited in Delhi” it outputs [places, to, be, visited, in, Delhi].

1.3.2.2 Stop-word eliminator

Stop-words are those words that do not have any disambiguation power. Common examples of stop words are articles, prepositions *etc.* In this step, stop words are removed from the list of tokens. For example, given the list of token generated by tokenizer, it strips it down to: [places, visited, Delhi].

1.3.2.3 Stemmer

The remaining tokens are then stemmed to the root form (*e.g.* visited → visit). For example, after stemming the list of tokens becomes this: [place, visit, Delhi].

1.3.2.4 Inverted index

The ordinary index would contain for each document, the index terms within it. But the inverted index stores for each term the list of documents where they appear. The benefit of using an inverted index comes from the fact that in IR we are interested in finding the documents that contain the index terms in the query. So, if we have an inverted index, we do not have to scan through all the documents in collection in search of the term. Often a hash-table is associated with the inverted index so that searching happens in O(1) time.

Inverted index may contain additional information like how many times the term appears in the document, the offset of the term within the document etc.

1.3.2.4.1 Example Say there are three documents.

Doc1 Milk is nutritious

Doc2 Bread and milk tastes good

Doc1 Brown bread is better

After stop-word elimination and stemming, the inverted index looks like:

milk	1,2
nutritious	1
bread	2, 3
taste	2
good	2
brown	3
better	3

1.3.3 Ranking

When the user gives a query, the index is consulted to get the documents most relevant to the query. The relevant documents are then ranked according to their degree of relevance, importance etc. Ranking is discussed elaborately in the subsequent chapters.

1.3.4 Relevance Feedback

Relevance feedback is one of the classical ways of refining search engine rankings. It works in the following way: Search engine firsts generate an initial set of rankings. Users select the relevant documents within this ranking. Based on the information in these documents a more appropriate ranking is presented (for example, the query may be expanded using the terms contained in the first set of relevant documents).

Sometimes users do not enough domain knowledge to form good queries. But they can select relevant documents from a list of documents once the documents are shown to him. For example, when the user fires a query 'matrix', initially documents on both the topics (movie and maths) are retrieved. Then say, the user selects the maths documents as relevant. This feedback can be used to refine the search and retrieve more documents from mathematics domain.

1.3.4.1 Types of relevance feedback

Explicit User gives feedback to help system to improve.

Implicit User doesn't know he is helping *e.g.* "similar pages" features in Google.

Pseudo User doesn't do anything! Top 'k' judgments are taken as relevant. Being fully automated it has always this risk that results may drift completely away from the intended document set.

1.3.4.2 Issues with relevance feedback

- The user must have sufficient knowledge to form the initial query.
- This does not work too well in cases like: Misspellings, CLIR, and Mismatch in user's and document's vocabulary (Burma vs. Myanmar).
- Relevant documents has to be similar to each other (they need to cluster) while similarity between relevant and non-relevant document should be small. That is why this technique does not work too well for inherently disjunctive queries (Pop stars who once worked at Burger King) or generic topics (tigers) who often appear as disjunction of more specific concepts.
- Long queries generated may cause long response time.
- Users are often reluctant to participate in explicit feedback. [Spink et al. (2000): Only 4% users participate. 70% doesn't go beyond first page.]
- In web, clickstream data could be used as indirect relevance feedback (discussed in autorefchapter:conclusion).

1.4 End Notes

1.4.1 Organization of this document

In this document we try to study a Web IR system with principle focus on ranking strategies. In this chapter we introduced the concept of information retrieval (IR) and discussed the general structure of a Web IR system. In chapter 2 we introduce some theoretical models (Boolean, vector, probabilistic and fuzzy logic based) of IR ranking and show some practical implementation of them. In chapter 3 we show how the IR situation in web is different. We discuss how web specific techniques like link analysis and anchor texts can be used by search engines. Within link analysis we discuss three algorithms: the PageRank algorithm of Google, HITS algorithm and OPIC algorithm. In chapter 4 we discuss how IR systems are evaluated. We discuss various

evaluation measures and also some well-known evaluation forums like TREC, CLEF *etc.*. In chapter 5 we introduce Cross Lingual Information Retrieval (CLIR) and a case study of CLIR system at IIT Bombay. We conclude in chapter 6 with a brief discussion on how search engines can be made user specific using click study. There we also briefly discuss a relatively new IR model called Quantum Probabilistic Retrieval Model.

Chapter 2

Theoretical Models in Information Retrieval

Theoretical models of IR show many different ways in which the IR problem can be formulated and solved. Formally, the IR model can be defined as a 4-tuple $[\mathbf{D}, \mathbf{Q}, \mathcal{F}, \mathbf{R}(q_i, d_j)]$ where

1. **D** is document collection. In most of the modeling approaches (Boolean, Vector or probabilistic) each document is modeled as a bag of index terms where index terms are assumed to be independent of each other. This way the semantics of the document is lost.
2. **Q** is the query collection. The queries fired by the user belong to this set. It is also modeled as a bag of index terms in most of the cases.
3. $\mathcal{F}$ is the framework for modeling document representations, queries and their relationship.
4. $R(q_i, d_j)$ is a ranking function which associates a score (real number) with the pair (q_i, d_j) where $q_i \in Q$ and $d_j \in D$. Given the query (q_i) the documents are ranked according to this score.

In this chapter we will discuss three classical models of IR; *namely,* Boolean, Vector Space and Probabilistic Model.

2.1 Boolean Model

Boolean Model is one of the oldest and simplest models of Information Retrieval. It is based on set theory and Boolean algebra.(Baeza-Yates and Ribeiro-Neto, 1999) In this model, each document is taken as a bag of index terms. Index terms are simply words or phrases from the document that are important to establish the meaning of the document. The query is a Boolean algebra expression using connectives like $\wedge, \vee, \neg$ *etc.* The documents retrieved are the documents that completely match the given query. Partial matches are not retrieved. Also, the retrieved set of documents is not ordered.

For example, Say, there are four documents in the system.

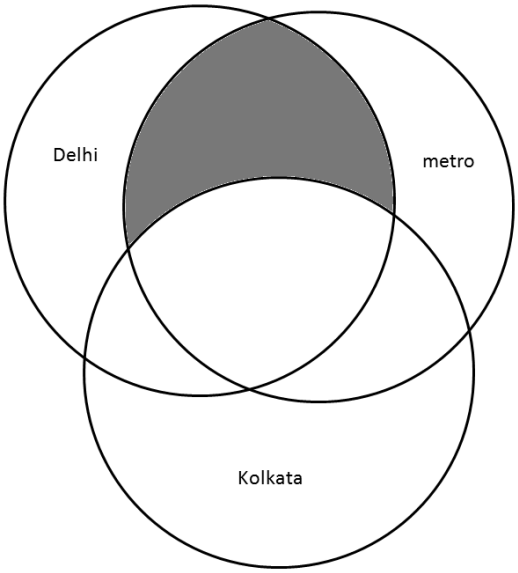

Figure 2.1: Venn Diagram to show the retrieved documents for query

Doc1 *The metro rail project in Delhi hit a major setback today.*

Doc2 *The Kolkata Metro Rail celebrated their anniversary this year.*

Doc3 *Unlike Kolkata, the Delhi metro often runs above the ground.*

Doc4 *Delhi is one of the biggest metro cities in India.*

Suppose the user wants to know specifically about Delhi Metro Rail project. A example query is given below:

$$Delhi \wedge Metro \wedge \neg Kolkata$$

For each term in the query, a list of documents that contain the term is created. Then the lists are merged according to the Boolean operators. The procedure will be clear from the Venn diagram in figure 2.1.

To know which of the documents contain a specified query term we cannot scan through the document to find out the word using tools like `grep`(Manning, 2009). BUt in practice, `grep` is often not suitable. The reasons are

1. `grep` will be very slow for large set of documents.
2. Here we are interested in documents rather than lines.

An Inverted Index comes very handy here (Manning, 2009). As discussed in the introduction section, the inverted index stores: for each terms the documents that contain the terms. So, from the inverted index we know that the term Delhi is in document {1, 3, 4}, the term metro is in documents $\{1, 2, 3, 4\}$ and the term Kolkata is in {2, 3}. We take the first two sets as it is

and perform a complement to the set of documents containing Kolkata which gives {1, 4}. We then perform a set intersection over the three sets to find out the result which is $\{1, 4\}$. The intersection operation can be done in $O(n)$ time. Where n is the length of the document lists (the posting lists should be kept sorted).

2.1.1 Discussion

2.1.1.1 Advantages

- It is simple, efficient and easy to implement.
- It was one of the earliest retrieval methods to be implemented. It remained the primary retrieval model for at least three decades.
- It is very precise in nature. The user exactly gets what is specified.
- Boolean model is still widely used in small scale searches like searching emails, files from local hard drives or in a mid-sized library.

2.1.1.2 Disadvantages

- In Boolean model, the retrieval strategy is based on binary criteria. So, **partial matches are not retrieved**. Only those documents that exactly match the query are retrieved. Hence, to effectively retrieve from a large set of documents users must have a good domain knowledge to form good queries.
- The retrieved documents are not ranked.
- Given a large set of documents, say, at web scale, the Boolean model either retrieves too many documents or very few documents.
- The reason of the above is: users usually do not form complex queries. Either they use very few (often a single) term fetching a tremendously large list of unordered documents. Else, they use a large set of terms joined by AND. This fetches very few documents. (Bhattacharyya, 2008) For example, the above query wrongly fetches Document 4 which is not relevant. To prevent this, user reformulates the query as

  $$Delhi \wedge Metro \wedge rail \wedge \neg Kolkata.$$

  But now document three is not retrieved although it was relevant. The query also does not capture synonymous terms like Kolkata and Calcutta. A still better query would be:

  $$Delhi \wedge (metro \vee tube) \wedge \neg(Kolkata \vee Calcutta)$$

  But users are often reluctant to form complex queries like the one above.
- The model does not use term weights. If a term occurs only once in a document or several times in a document, it is treated in same way.(Gao et al., 2004)

2.2 Vector Based Model

The main problem with Boolean model is its inability to fetch partial matches and the absence of any scoring procedure to rank the retrieved documents. This problem was addressed in the vector based model of Information retrieval.

2.2.1 Document and Query Representation

In this model documents are represented as a vector of index terms.

$$\vec{d_j} = \{w_{1j}, w_{2j}, \ldots, w_{tj}\}$$

where t is the total number of index terms in the collection of documents. Each $w_{ij} > 0$ if and only if the term i is present in document d_j. Unlike Boolean model, we do not consider only present or absence of terms. So, in vector model these `term weights` are not binary.

Like documents, queries are also represented as vectors in a similar way. The similarity between the query vector and document vector is a measure of relevance of the document and used as a ranking score. The similarity between document vector and query vector is usually calculated as the cosine similarity (discussed later in this chapter) between them. If the similarity is greater than a predefined threshold, the document is retrieved.

2.2.2 Modelling as a clustering method

What the vector based model of IR does is basically the following. Given a query as a possibly incomplete description of the information need of the user, the system tries to cluster the set of documents into two sets. One is the set of documents related to the query and the other is the set of documents unrelated to the query. For good clustering we need to ensure that the features that represent the documents are able to describe similar documents (intra-cluster similarity) as well as are able to distinguish dissimilar documents (inter-cluster dissimilarity). In the following section we will discuss how to fix term-weights in such a way that this balance is kept.

2.2.3 Fixing the term-weights

The term weights in the document and query vector represent the importance of the term for expressing the meaning of the document and query. There are two widely used factors in calculating term weights; *namely,* **term frequency** (tf) and **inverse document frequency** (idf). The term weights can be approximated by the product of the two factors. This is called the *tf-idf* measure.

2.2.3.1 Term-frequency

The term-frequency is simply the count of the term i in document j (how many times the term occur). The term-frequency of a term is normalized by the maximum term frequency of any term in the document to bring the range of the term-frequency 0 to 1. Otherwise, terms appearing in

larger documents would always have larger term frequency. Mathematically, the term-frequency of a term i in document j is given by:

$$tf_{i,j} = \frac{freq_{i,j}}{max_l(freq_{l,j})} \tag{2.1}$$

where $freq_{i,j}$ is the count of how many times the term i appear in document j.

2.2.3.2 Inverse document frequency

The term-frequency captures the inter-cluster dissimilarity. This factor is motivated from the fact that if a term appears in all documents in a set, then it loses its distinguishing power. For example the stop-words (articles, prepositions *etc.*) do not have any distinguishing power. The term cricket in a set of articles on cricket does not have any distinguishing power. The inverse document frequency (idf) of a term i is given by:

$$idf_i = \log \frac{N}{n_i} \tag{2.2}$$

where n_i is the number of documents in which the term i occurs and N is the total number of documents.

The expression of *idf* comes from information theory perspective. If the term i occurs in n_i number of documents among a total of N documents, then the term will occur in a randomly picked document with probability $\frac{n_i}{N}$. Therefore, the fraction of information carried by the statement 'A Document d contains the term i' is given by:

$$-\log \frac{n_i}{N} = \log \frac{N}{n_i}.$$

2.2.4 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. Here we will discuss a widely used measure of similarity called the **cosine similarity**.

2.2.4.1 Cosine Similarity

The cosine similarity between two vectors $\vec{d_j}$ (the document vector) and $\vec{q}$ (query vector) is given by:

$$similarity(\vec{d_j}, \vec{q}) = \cos\theta = \frac{\vec{d_j} \cdot \vec{q}}{|\vec{d_j}|\,|\vec{q}|} \tag{2.3}$$

$$= \frac{\sum_{i=1}^{t} w_{i,j} \times w_{i,q}}{\sqrt{\sum_{i=1}^{t} w_{i,j}^2} \times \sqrt{\sum_{i=1}^{t} w_{i,q}^2}} \tag{2.4}$$

Here θ is the angle between the two vectors, $w_{i,j}$ is the term-weight for i-th term of j-th document. $w_{1,q}$ is the term weight assigned to i-th term of the query. Since $w_{i,j} \geq 0$ and $w_{i,q} \geq 0$ for all i,j and $\vec{q}$, the similarity between $\vec{d_j}$ and $\vec{q}$ varies from 0 to 1.

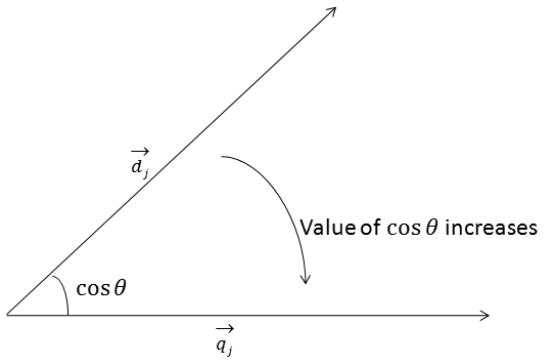

Figure 2.2: Cosine Similarity

The cosine similarity provides a nice metaphor. Cosine similarity gives maximum value when $\theta = 0$ or when the vectors coincide. It gives lowest value when the vectors are independent of each other. This is shown in figure 2.2

The retrieved set of documents are simply those documents $\vec{d_k}$ for which $similarity(\vec{d_j}, \vec{q})$ is greater than a threshold value. That threshold can be dependent on the query itself. For example, if for some query the highest similarity with any document is on the lower side, the cut-off threshold can be brought down. This way, vector based model does not enforce that all retrieved documents should be exact match of the query. It allows partial matches to be retrieved.

2.2.4.2 Implementation of Cosine Similarity

The actual computation of cosine similarity over full vectors can be quite computationally intensive. The reason being, for calculating dot product of query and document vector we need k multiplication operation where k is the total number of index terms (a very large number). In practice, the full document and query vectors are rarely implemented as they are long and parse (Vachhani, 2009). But we can take advantage from the inverted index while calculating cosine similarity. This is demonstrated in the following algorithm in (Vachhani, 2009).

Algorithm 1 Full vector space model implementation

```
for every query term q in Q do
 retrieve the postings list for q from the inverted index
 for each document d indexed in the postings list do
  score(d) = score(d) + tf * idf
 end
end
Normalize scores. // the denominator of cosine similarity
Sort documents according to n
```

2.2.5 Discussion

2.2.5.1 Advantage

- The cosine similarity measure returns value in the range 0 to 1. Hence, partial matching is possible.
- Ranking of the retrieved results according to the cosine similarity score is possible.

2.2.5.2 Disadvantage

- Index terms are considered to be mutually independent. Thus, this model does not capture the semantics of the query or the document.
- It cannot denote the "clear logic view" like Boolean model.(Gao et al., 2004)

Despite its simplicity, the vector based model works well with general collections. It is widely used in practical systems. In subsection 2.2.6 we discuss one such practical implementation.

2.2.6 The Nutch Ranking

Nutch is an open-source search engine platform based on Lucene java. Nutch is a fully fledged web search engine that supports crawling, indexing and ranking. It also has a link graph database and parsers for HTML and other document formats. The indexing and ranking component is based on apache Lucene. It is developed by Apache Software foundation. Current stable version is 1.0.0, released on March 23, 2009.

2.2.6.1 The Nutch Ranking Expression

The ranking in Nutch is done in two phases. In the first phase, an initial set of document is retrieved using Boolean model. Then it ranks the initial set using vector space model. The similarity of a query $\vec{q}$ and a document $\vec{d}$ is given by:

$$score(\vec{q},\vec{d}) = queryNorm(\vec{d}) \times coord(\vec{q},\vec{d}) \tag{2.5}$$

$$\times \sum_{t \in \vec{q}} tf(t \in \vec{d}) \times idf(t) \times t.boost(t.field \in \vec{d}) \times Norm(t.field \in \vec{d}) \tag{2.6}$$

Next, we will explain different terms in the expression given above. Let us first compare the above expression with the standard cosine similarity of vector space model [1]. The sum term gives the numerator of cosine similarity if we assume each term occurs once in the query. The normalization factor given in the denominator of cosine similarity is given by terms like $queryNorm(\vec{d})$ and $Norm(t.field \in \vec{d})$.

The different terms in the above expression is explained below:

[1] The tf-idf cosine similarity can be expressed as:

$$similarity(\vec{q},\vec{d}) = \sum_{t \in \vec{q}} idf_t \times \frac{tf_d}{|\vec{d}|}$$

we omit $|\vec{q}|$ as it is constant for all queries. The term $|\vec{d}|$ works as document length normalization.

tf(t in d) term frequency of term t in document $\vec{d}$.

idf(t) Inverse document frequency of term t

boost (t.field in d) The importance of a term to a document depends on where the term appears. The boost field captures that and gives higher weight if the term appears in an important area of a webpage. For example, a term appearing in title, URL or within headings of a page should be given higher weightage.

norm(t, d) This factor is calculated using the following expression:

$$norm(t, \vec{d}) = doc.getBoost() \times lengthNorm(field) \times \prod_{\text{field f in d as t}} f.getBoost() \quad (2.7)$$

d.getBoost() It captures the importance of the document in the collection of documents. It is calculated using a Link Analysis algorithm named Online Page Importance Calculation (OPIC). This algorithm is discussed in the next chapter.

lengthNorm(field) It captures the normalization factor that depends on the length (number of index terms) of the document.

f.getBoost() It captures the importance of a particular field. The product term captures whether the term appears more in the important part of the document than in non-important parts.

queryNorm(q) It is a normalizing factor used to make scores between queries comparable. This factor does not affect document ranking as it depends only on the query.

coord(q,d) It is a score factor based on how many of the query terms are found in the specified document.

2.3 Probabilistic Model

In probabilistic model we try to capture the information retrieval process from a probabilistic framework. The basic idea is to retrieve the documents according to the probability of the document being relevant.(Baeza-Yates and Ribeiro-Neto, 1999) There are many versions of Probabilistic Model like Rijsbergen(Rijsbergen, 1979), Robertson-Spark-Jones(Robertson, 1977) *etc.* Here, we will mainly go by the version of Robertson-Spark-Jones (1976)(Jones et al., 2000). In the following sections we describe the probabilistic model formally. Then we show how the probabilistic relevance is estimated. Finally we discuss Okapi-BM25 model as an extension to the probabilistic model. We will mainly follow (Manning et al., 2009) while deriving expressions.

2.3.1 The Probabilistic Ranking Principle

The basic question the system needs to ask before deciding whether to retrieve a document or not is: 'What is the probability of this document being relevant given this query' (Jones et al., 2000). Here it is assumed that the relevance of a document to a query is independent of other documents in collection(Robertson, 1977). Probabilistic ranking principle is thus(Robertson, 1977) the following:

> If a reference retrieval system's response to each request is a ranking of the documents in the collection in order of decreasing probability of relevance to the user who submitted the request, where the probabilities are estimated as accurately as possible on the basis of whatever data have been made available to the system for this purpose, the overall effectiveness of the system to its user will be the best that is obtainable on the basis of that data.

Formally, given the document vector $\vec{d}$ and query vector $\vec{q}$, we rank the documents according to the probability of the document being relevant. Mathematically, the scoring function is given by

$$P(R = 1|\vec{d}, \vec{q}) \tag{2.8}$$

where $\vec{d}$ is the document vector, $\vec{q}$ is the query vector and R is the indicator random variable that takes value 1 if $\vec{d}$ is relevant w.r.t. $\vec{q}$ and 0 if $\vec{d}$ is not-relevant w.r.t $\vec{q}$.

If we are interested in retrieving a set of documents rather than ordering the documents, we can use following rule(Rijsbergen, 1979) where we retrieve a document only if:

$$P(R = 1|\vec{d}, \vec{q}) > P(R = 0|\vec{d}, \vec{q}) \tag{2.9}$$

2.3.2 The Binary Independence Model

Probabilistic model of information retrieval is a generic model that allows many ways of interpretations. In this section we introduce the binary independence model of calculating the probability of relevance $P(R = 1|\vec{d}, \vec{q})$. In this model, the documents and queries are represented as binary (Boolean) term incidence vectors. Hence the model is called binary. It is also assumed that terms in a document are independent of each other, hence the model is called independence model.

Many assumptions are made in the Binary Independence Model. Let us summarize them here:

1. The documents are independent of each other.
2. The terms in a document are independent of each other.
3. The terms not present in query are equally likely to occur in any document *i.e.* do not affect the retrieval process.

2.3.2.1 Derivation of ranking function

We need to calculate the probability $P(R = 1|\vec{d}, \vec{q})$ with respect to which the documents will be ordered. We do not take the probability directly. But we calculate the odds of relevance. There is no harm in taking the odds ratio instead of $P(R = 1|\vec{d}, \vec{q})$. This is because we are not interested in actual value of the probability of relevance but only in the ordering of the documents. The 'odds of relevance' is monotonic with respect to probability of relevance. Hence it will return the same order of documents. Taking the odds ratio has some advantages like, in equation 2.11 we do not have to unnecessarily expand $P(\vec{d}|\vec{q})$. Taking the odds ratio also minimizes the probability of erroneous judgment(Baeza-Yates and Ribeiro-Neto, 1999).

Then our ranking function $rf()$ becomes:

$$rf() = O(R|\vec{d},\vec{q}) = \frac{P(R=1|\vec{d},\vec{q})}{P(R=0|\vec{d},\vec{q})} \tag{2.10}$$

We then use Bayes rule to convert it to *generative model*. The reason of using generative model will be clear when we will use naive-bayes assumption in equation 2.16

$$O(R|\vec{d},\vec{q}) = \frac{\frac{P(R=1|\vec{q})\cdot P(\vec{d}|R=1,\vec{q})}{P(\vec{d}|\vec{q})}}{\frac{P(R=0|\vec{q})\cdot P(\vec{d}|R=0,\vec{q})}{P(\vec{d}|\vec{q})}} \tag{2.11}$$

$$= \left[\frac{P(R=1|\vec{q}}{P(R=0|\vec{q})}\right] \cdot \left[\frac{P(\vec{d}|R=1,\vec{q})}{P(\vec{d}|R=0,\vec{q})}\right] \tag{2.12}$$

We can see that the left hand term in equation 2.12 does not depend on document vector $\vec{d}$. So, we can remove it safely from our ranking function as it is a constant term for a given query. We then break $\vec{d}$ in the right hand term into individual terms using chain rule.

$$rf() = \frac{P(\vec{d}|R=1,\vec{q})}{P(\vec{d}|R=0,\vec{q})} \tag{2.13}$$

$$= \frac{P(x_1|R=1,\vec{q})\cdot P(x_2|x_1,R=1,\vec{q})\cdots P(x_M|x_1\ldots x_{M-1},R=1,\vec{q})}{P(x_1|R=0,\vec{q})\cdot P(x_2|x_1,R=0,\vec{q})\cdots P(x_M|x_1\ldots x_{M-1},R=0,\vec{q})} \tag{2.14}$$

$$= \prod_{t=1}^{M} \frac{P(x_t|x_1\ldots x_{t-1},R=1,\vec{q})}{P(x_t|x_1\ldots x_{t-1},R=0,\vec{q})} \tag{2.15}$$

where M is total number of index terms.

We then apply our Naive-Bayes assumption which states that all index term x_t are independent *i.e.* does not depend on any other index term. This approximates 2.15 to

$$rf() \approx \prod_{t=1}^{M} \frac{P(x_t|R=1,\vec{q})}{P(x_t|R=0,\vec{q})} \tag{2.16}$$

Since each x_t is either 0 or 1 under Binary Independence Model, we can separate the terms to give

$$rf() = \prod_{t:x_t=1}^{M} \frac{P(x_t|R=1,\vec{q})}{P(x_t|R=0,\vec{q})} \cdot \prod_{t:x_t=0}^{M} \frac{P(x_t|R=1,\vec{q})}{P(x_t|R=0,\vec{q})} \tag{2.17}$$

Now let, $p_t = P(x_t|R=1,\vec{q})$ be the probability of a term x_t appearing in a relevant document and $u_t = P(x_t|R=0,\vec{q})$ be the probability that a term x_t appears in a non-relevant document. Rewriting equation 2.17 in terms of p_t and u_t gives:

$$rf() = \prod_{t:x_t=1}^{M} \frac{p_t}{u_t} \cdot \prod_{t:x_t=0}^{M} \frac{1-p_t}{1-u_t} \tag{2.18}$$

Now here is another assumption. Let's assume that terms not appearing in query are equally likely to appear in any document *i.e.* they do not have any classifying power. Thus our equation 2.18 becomes only over query terms.

$$rf() \approx \prod_{t:x_t=q_t=1}^{M} \frac{p_t}{u_t} \cdot \prod_{t:x_t=0,q_t=1}^{M} \frac{1-p_t}{1-u_t} \tag{2.19}$$

$$= \left[\prod_{t:x_t=q_t=1}^{M} \frac{p_t \cdot (1-u_t)}{u_t \cdot (1-p_t)} \right] \cdot \left[\prod_{t:[x_t=0 \vee 1],q_t=1}^{M} \frac{1-p_t}{1-u_t} \right] \tag{2.20}$$

What we have done in equation 2.20 is a simple algebraic manipulation. We multiply the right term in equation 2.19 with $\prod_{t:x_t=q_t=1}^{M} \frac{1-p_t}{1-u_t}$ and simultaneously divide it from the left product. Thus right term equation 2.20 becomes independent of x_t and we can ignore it. Thus our ranking function $rf()$ becomes:

$$rf() \approx \prod_{t:x_t=q_t=1}^{M} \frac{p_t \cdot (1-u_t)}{u_t \cdot (1-p_t)} \tag{2.21}$$

We now take logarithm of equation 2.21 to convert the product into summation. This is a standard practice to save the probability values from becoming too small. Thus the equation 2.21 become linear in nature and covers the whole real line (Cambridge) and Barcelona), 2007).

$$rf() = \log \prod_{t:x_t=q_t=1}^{M} \frac{p_t \cdot (1-u_t)}{u_t \cdot (1-p_t)} \tag{2.22}$$

$$= \sum_{t:x_t=q_t=1}^{M} \log \frac{p_t \cdot (1-u_t)}{u_t \cdot (1-p_t)} \tag{2.23}$$

$$= \sum_{t:x_t=q_t=1}^{M} \log \frac{p_t}{1-p_t} + \log \frac{1-u_t}{u_t} \tag{2.24}$$

$$= \sum_{t:x_t=q_t=1}^{M} c_t \tag{2.25}$$

where c_t is the log odds ratio[2] for the terms in query. This term c_t becomes zero when the term has equal odds of occurring in relevant and non-relevant documents and positive if the term is more likely to be present in a relevant document.

Thus the formal scoring function of probabilistic information retrieval model is given by equation 2.25. We now have to find out a way to measure the term c_t or the probability terms p_t and u_t

[2]We saw $c_t = \log \frac{\frac{p_t}{1-p_t}}{\frac{u_t}{1-u_t}}$ which is nothing but,

$\log \frac{\text{odds of term present in a relevant document}}{\text{odds of term being present in a non-relevant document}}$ or the log of odds ratio.

2.3.2.2 Estimating the probabilities in theory

We need some prior relevance judgment for approximating the parameter c_t. Information regarding the documents in which the term is present is already known to be relevant or non-relevant gives a way of measuring p_t and u_t (Jones et al., 2000). Suppose we have the following contingency table at our disposal:

	documents	relevant	non-relevant	total
Term present	$x_t = 1$	r_t	$n_t - r_t$	n_t
Term absent	$x_t = 0$	$R - r$	$(N - n) - (R - r)$	$N - n_t$
	Total	R	$N - R$	N

where N is the total number of documents. R is the total number of relevant documents, r_t is the number of relevant documents containing the term t and n_t is the total number of documents containing the term t.

This gives,

$$p_t = \frac{r}{R} \tag{2.26}$$

$$u_t = \frac{n_t - r_t}{N - R} \tag{2.27}$$

Putting these values in equation of c_t we get,

$$c_t = \log \frac{r_t(N - n_t - R + r_t)}{(R - r_t)(n_t - r_t)} \tag{2.28}$$

To avoid the possibility of zeros (say, a term is present/absent in all documents) we can add 0.5 to all the terms in equation 2.28 as a smoothing technique. This gives, Putting these values in equation of c_t we get,

$$c_t = \log \frac{(r_t + 0.5)(N - n_t - R + r_t + 0.5)}{(R - r_t + 0.5)(n_t - r_t + 0.5)} \tag{2.29}$$

2.3.2.3 Estimating the probability in practice

We again recall the equation of c_t from 2.25.

$$c_t = \log \frac{p_t}{1 - p_t} + \frac{1 - u_t}{u_t} \tag{2.30}$$

$$\tag{2.31}$$

When no relevance information is available, we can assume $p_t = 0.5$. This makes c_t depend only on u_t. We can further assume the distribution of index terms among the non-relevant documents can be approximated by the distribution of index terms among all the documents in collection. That is, $u_t \approx \frac{n_t}{N}$ (Baeza-Yates and Ribeiro-Neto, 1999). Putting these values in equation 2.31 we get,

$$c_t \approx \log \frac{0.5}{1-0.5} + \frac{1-\frac{n_t}{N}}{\frac{n_t}{N}} \tag{2.32}$$

$$= \log \frac{N-n_t}{n_t} \tag{2.33}$$

We can farther assume that n_t (number of documents in which the term appear) is very small compared to N (total number documents). Thus $(N - n_t) \approx N$. Putting this in equation 2.33 we get,

$$c_t \approx \log \frac{N}{n_t} \tag{2.34}$$

It is interesting to notice that the expression for c_t is same as the Inverse Document Frequency expression for term t (IDF_t) (with n_t being document frequency of term t). Thus the document ranking is determined by which query terms appear in the document scaled by their IDF weights. This idea was proposed by Croft and Harper in 1979. We will come back to this point in the next section while discussing Okapi-BM25 model.

2.3.2.4 Recursive approximation using Relevance Judgments

With the initial approximation of $p_t = 0.5$ and $u_t = \frac{n_t}{N}$, we get expression of c_t as given in equation 2.34 (Baeza-Yates and Ribeiro-Neto, 1999). We can rank the documents using this model. The set of relevant documents V is then obtained either from the user (relevance feedback) or V can be approximated by the set of top r ranked documents where r is a previously determined threshold (Pseudo-relevance feedback). Let V_t be the subset of V that contain the term t. What we will do now is to approximate the set of relevant documents R with V. This way, set of relevant documents in which term t occurs become V_t. The non-relevant set of documents become $N - V$ and non-relevant set of documents in which t appears become $n_t - V_t$. Then,

$$p_t = \frac{V_t}{V} \tag{2.35}$$

$$u_t = \frac{n_t - V_t}{N - V} \tag{2.36}$$

This can be used for a better approximation of c_t. This process is repeated recursively (Baeza-Yates and Ribeiro-Neto, 1999). The equations in 2.35 and 2.36 can be smoothed to avoid problems caused by small values of V_i and V. For example,

$$p_t = \frac{V_t + 0.5}{V + 0.5} \tag{2.37}$$

$$u_t = \frac{n_t - V_t + 0.5}{N - V + 0.5} \tag{2.38}$$

2.3.3 Discussion

The probabilistic model of IR in its pure form have been implemented only in small scale IR tasks like library catalog search (Manning et al., 2009). Generally models like vector based

model out-performs probabilistic model in large scale information retrieval tasks like web search (Baeza-Yates and Ribeiro-Neto, 1999). The main reasons are, probabilistic IR in its pure form incorporates too many assumptions many of which could have side effect.

For example, Binary Independence Model considers only term presence or absence. It does not use term frequency. It also assumes each document as a bag of index terms. It assumes (i) Sequence in which the terms appear in the document is not important, (ii) all index terms are independent of each other and (iii) ranking of some documents does not affect the relevance judgment of other documents.

Another drawback of probabilistic IR model is for calculating p_t and u_t in equation 2.25, approximate relevance judgment is required. While this is possible for small scale IR process, it is not possible for large scale IR scenario.

For these reasons probabilistic model of information retrieval in its original form is hardly implemented in large scale IR process. But as probabilistic IR model is very generic in nature, many versions of probabilistic IR exist which are used practically. One such probabilistic IR based algorithm is Okapi-BM25 which we will discuss in the next section.

2.3.4 Okapi BM25 Ranking Function

In this section we discuss the Okapi-BM25 ranking function. It was first implemented in the "Okapi Information retrieval system" in London's City University in the 1980s and 1990s. It is based on the probabilistic retrieval model discussed above but pays attention to the term frequencies and document length.

We saw in equation 2.34 that when we have no relevance estimate and the number of relevent documents are very small compared to total number of documents then the term weights are approximately equal to the idf score of the term.

In that case,

$$rf() \approx \sum_{t \in q} \log \frac{N}{n_t} \tag{2.39}$$

Now we factor in the term frequencies in each term (Manning et al., 2009). In that case,

$$rf() = \sum_{t \in q} \log \frac{N}{n_t} \cdot \frac{(k_1 + 1) tf_{td}}{k_1 + tf_{td}} \tag{2.40}$$

k_1 is a tuning parameter whose value determines the importance given to the term frequency factor. When k_1 is 0, no term frequency is considered. But we also need to think of document length which might affect the term frequency. So, we consider normalizing the term frequency with the document length. A document may be large for two reasons (Cambridge) and Barcelona), 2007):

- **Verbosity** Some authors are more verbose
- **Scope** The author has more to say.

We want to normalize the term-frequencies in the first case. Not in the second case. So what we need is a kind of soft normalizing factor.(Cambridge) and Barcelona), 2007)

The soft-normalization factor used in Okapi-BM25 is:

$$B = \left((1-b) + b\frac{L_d}{L_{ave}}\right) \tag{2.41}$$

where L_d is the document length given by $L_d = \sum_{t \in \vec{d}} tf_t$ and L_{ave} is the average document length over all document collection. We now normalize tf using B before applying in equation 2.40. Putting $\frac{tf_t}{B}$ in equation 2.40 we get,

$$rf() = \sum_{t \in q} \log \frac{N}{n_t} \cdot \frac{(k_1+1)\frac{tf_{td}}{B}}{k_1 + \frac{tf_{td}}{B}} \tag{2.42}$$

$$= \sum_{t \in q} \log \frac{N}{n_t} \cdot \frac{\frac{(k_1+1)tf_{td}}{B}}{\frac{k_1 \times B + tf_{td}}{B}} \tag{2.43}$$

$$= \sum_{t \in q} \log \frac{N}{n_t} \cdot \frac{(k_1+1)tf_{td}}{k_1((1-b) + b \times \frac{L_d}{L_{ave}}) + tf_{td}} \tag{2.44}$$

It is interesting to note that this equation 2.44 is very similar to the tf-idf similarity equation in vector based model.[3]

If the query is long, one could use similar weighting for query terms also. The Okapi-BM25 algorithm can also be adapted if there is relevance judgment available.

2.4 Fuzzy Set Theory Model

The IR models discussed so far assumed that index terms are independent of each other. They all represented document as a collection of index terms and this way lost the semantics of the document. As a result the matching of the query and document is often a vague one.

In fuzzy set theory each query term q_i defines a fuzzy set of documents. Each document d_j in the collection has a degree of membership ($\mu_{i,j} < 1$) in this set. The term $\mu_{i,j}$ is defined as:

$$\mu_{i,j} = 1 - \prod_{k_l \in d_j} (1 - c_{i,l}) \tag{2.45}$$

Where $c_{i,l}$ is the correlation of the index term i and index term l (a query term is also an index term). The correlation is calculated as the odds of the term appearing together and not appearing together; given by:

$$c_{i,l} = \frac{\text{\# times terms i and l appear together}}{\text{\# times terms i and l does not appear together}} \tag{2.46}$$

$$= \frac{n_{i,l}}{n_i + n_l - n_{i,l}} \tag{2.47}$$

[3] The tf-idf cosine similarity can be expressed as:

$$similarity(\vec{q}, \vec{d}) = \sum_{t \in \vec{q}} idf_t \times \frac{tf_d}{|\vec{d}|}$$

We omit $|\vec{q}|$ as it is constant for all queries. The term $|\vec{d}|$ works as document length normalization.

Equation 2.45 actually calculates the algebraic sum of correlations of query term q_i with all the terms in the document. The sum is implemented as complemented of a negated algebraic product. This has many benefits as will be clear with our discussions. Firstly, this formulation ensures that whenever there is one index term in the document which is strongly related to q_i (*i.e.* $c_{i,l} \approx 1$) then $\mu_{i,j}$ will also be ≈ 1. The degree of membership is calculated using an algebraic sum overall index terms instead of a usual max function to allow smooth transition for the values of the $\mu_{i,j}$ factor.

The user specifies his information need using a Boolean logic expression. This expression is converted to disjunctions of conjunctions (disjunctive normal form). A fuzzy set of documents is created for each of the conjunctive components. These sets are combined to get the final fuzzy set of documents. The fuzzy set is nothing but a degree of membership value for all the documents in collection. The documents can be ranked according to this value.

2.5 End notes

In this chapter we discussed various theoretical models of IR. Practically they all are implemented using an inverted index of documents. The key area of focus of any retrieval process is how to represent documents and features. They are either represented as a bag of index terms (Boolean model) or a vector of index terms. The index terms can be included as a binary basis (1 if present, 0 if absent) or various complex term-weights can be used. The major factors that influence term-weighting are (Singhal, 2001): 1) term frequency 2) inverse document frequency and 3) document length. One popular method of term-weighting is ($\frac{tf \times idf}{\text{doc length}}$). But using raw term frequencies is non-optimal (Singhal, 2001). Using a dampened frequency (log of tf) may be better (Singhal, 2001). State of the art scoring methods like Okapi-BM25 include many different factors and are far superior to normal tf-idf scoring.

Chapter 3

Searching the Web: Link Analysis and Anchor Text

In the last chapter we discussed theoretical models of IR and their practical applications. They all share a commonality. All the models rank the documents based on the similarity of them with the query and for doing this they use features from the document only. This approach was suitable for information retrieval from a well-controlled collection of documents like research papers or library catalog. But the scenario in the case of Web Search is substantially different.

3.1 Difference between Web IR and Traditional IR

But in the case of ranking of web documents, using features only within the document is not sufficient. One reason is, as the number of documents is very huge in case of web, a large number documents are often very relevant to the query which cannot be further ranked based only on the internal features of the document.

As the web is growing really fast, the search process must be scalable, algorithms must be efficient and storage should be used properly, queries should be handled quickly (hundreds of thousands per second). In web, thousands of documents would match the query but only the top few are those who count. A Web IR system must avoid junk results at top (Brin and Page, 1998). So, "very high precision is important even at the expense of recall".

The web is a huge set of totally uncontrolled heterogeneous set of documents. There are many languages and many different document formats involved. Anyone can publish anything in the web. Not all documents are of same importance. For example, a random blog and the BBC website cannot be treated in the same way.

Search engines take a very big role in routing traffic towards a web page. As there is virtually no control over what people can put on the web, a malicious user may put random interesting words in his page (probably do things to make these terms not easily visible to the visitor) and get a high ranking in any term-frequency based ranking method. Meta fields like "keywords" are often used for this purpose as they are not directly visible. This makes clear that we cannot rely only on the document to get its rank.

In this chapter we discuss two features specific to web that can be exploited to carry on Web IR. One is Link Analysis and another is Anchor Text.

3.2 Link Analysis

Fortunately, web gives us a unique way to measure the importance of a document. In web, documents are often connected using hyperlinks. If a page B has a link to page A, then we say page A has a backlink from page B. We can view back-links as a type of endorsement. The more backlinks a page have, the more important the page is. While ranking if two pages have similar relevance to the query, the more important page should be ranked higher. In the next section we formalize the notions.

3.2.1 Web as a graph

Web is a collection of hyperlinked documents. We can view the web as a directed graph where each page[1] is a node and a hyperlink from one page to another is captured by a directed edge . If page A has a link to page B, then there should be a directed edge from node A to node B. Every page has a number of forward edges (out edges) and backlinks (in edges). Page A has a backlink from page B if there is a hyperlink to page A from page B. Backlink can be viewed as a type of endorsement and the count of backlinks is regarded as a measure of importance of a page. This idea was used earlier in the field of citation analysis.

But deciding the importance of a page based on backlink count pose another problem in terms of link farms. Link farms are a group of web pages that link to each other. In this way any malicious creator of web page can have high backlink count by setting up another n number of web pages each having a hyperlink to that page. The algorithm PageRank solves this problem by not only counting the backlinks but also noting the page from which the link is coming from.

3.2.2 PageRank algorithm

PageRank is a link analysis algorithm that estimates the importance of a document by analyzing the link structure of a hyperlinked set of documents. It is named after Larry Page (co-founder of Google).

The simple backlink count was sufficient for well controlled document collection of citation analysis. But in web, there is hardly any control. Millions of pages can be automatically created and linked with each other to manipulate the backlink count (Page et al., 1998). As web consists of conflicting profit making ventures, any evaluation strategy which counts replicable features of web pages is bound to be manipulated.

PageRank extends the idea of backlink by "not counting links from all pages equally, and by normalizing by the number of links on a page."(Brin and Page, 1998). Here, we assume page A has pages $T_1 \dots T_n$ 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 (a normalizing factor). The PageRank of a page A is a value in the range 0 to 1 and is given by:

$$PR(A) = \frac{1-d}{N} + d\left(\frac{PR(T_1)}{C(T_1)} + \cdots + \frac{PR(T_n)}{C(T_n)}\right) \tag{3.1}$$

[1] in context of web, a document is often called a page. Throughout this chapter, the term document and page have been used interchangeably

3.2.2.1 Random surfer model

The PageRank computation is based on a random surfer model. It models a random surfer who starts browsing from a random page and follows the web-graph choosing a forward link randomly at each node (Brin and Page, 1998). The random surfer does not hit the back button but may get bored and restarts the process from any other random node. The PageRank of page A is simply the probability of a random surfer to visit page A. The term d is a damping factor which signifies that the probability of the random surfer becoming bored at a page and restarting from any other random page is $(1 - d)/N$ (N is the total number of pages). When the random surfer visits a page with no out-link, it restarts from any other random page. In other terms, pages with no out-links are assumed to have out-links to all other pages in the web.

The PageRank of a page (probability of reaching that page by random surfing) can be high in two cases. First, when a lot of pages connect to that page or a few pages with high PageRank connects to that page.

3.2.2.2 Example

This example is taken from Wikipedia. See Figure 3.1. Here, page C has a higher PageRank than Page E, even though it has fewer links to it; the link it has is of a much higher value. A web surfer who chooses a random link on every page (but with 15% likelihood jumps to a random page on the whole web) is going to be on Page E for 8.1% of the time. (The 15% likelihood of jumping to an arbitrary page corresponds to a damping factor of 85%.) Without damping, all web surfers would eventually end up on Pages A, B, or C, and all other pages would have PageRank zero. Page A is assumed to link to all pages in the web, because it has no outgoing links.

3.2.2.3 Random Surfing as a Markov Chain

The random surfing on web graph can be modeled as a Markov Chain where the next state depends on the current state and all transitions are equally probable (Manning et al., 2009). A Markov chain is characterized by its transition probability matrix. Each entry $P(i, j)$ in the transition probability matrix stores the probability of going to state j given current state i. For a transition probability matrix the following properties hold:

$$\forall i, j, P(i, j) \in [0, 1] \tag{3.2}$$

$$\forall i, \sum_{j=1}^{N} P(i, j) = 1 \tag{3.3}$$

Any matrix with non-negative entries that follow the above rules is known as a stochastic matrix. A key property of stochastic matrix is that, it has a principal left eigenvector corresponding to largest eigenvector which is 1 (Manning et al., 2009).

From the web-graph as captured by the crawler and HTML parser, an adjacency matrix is created. We convert the adjacency matrix to a stochastic matrix P as per Equation 3.1 (Manning et al., 2009).

- If no entry in a row is 1, each entry is replaced by $\frac{1}{N}$. It captures the rule that, when a node has no out-links it is assumed to have link to all other pages.

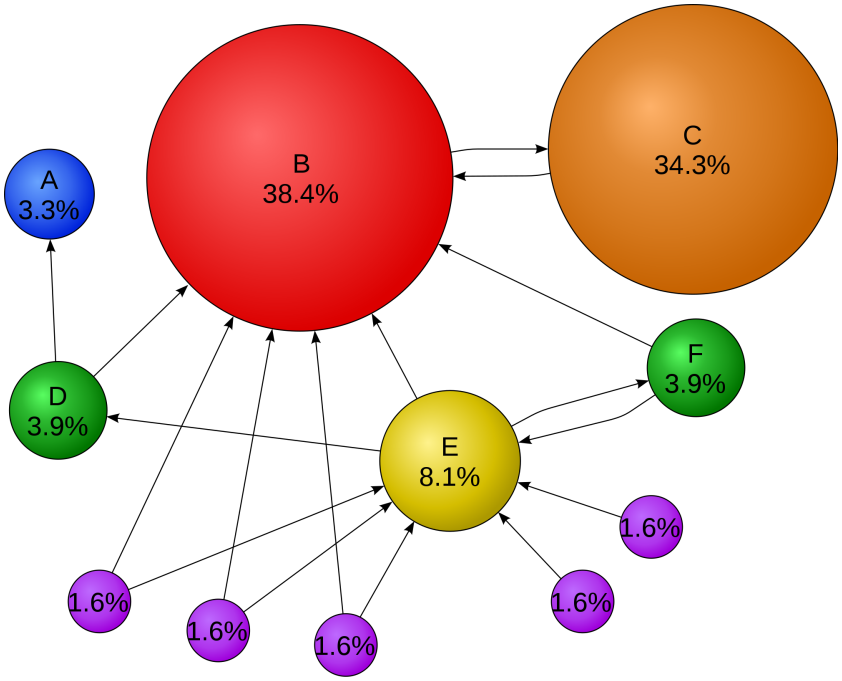

Figure 3.1: PageRank example from Wikipedia

- Divide each 1 in a row by the number of 1s in that row. This way, we equally distribute the transition probabilities over all possible transitions.
- Multiply the matrix by d. Here we introduce the damping factor.
- Add $\frac{1-d}{N}$ to each entry of the matrix. This incorporates the assumption that at any page the user may be bored and restart from any random page with probability $\frac{1-d}{N}$.

Each entry $P(i,j)$ in this transition holds the probability of visiting page j given the current page as i. This incorporates the Markov assumption: the next page to be visited depends only on the current page and the probability is equally distributed over all forward links. This way, the random walk (surfing) on web-graph is nothing but a Markov Chain.

3.2.2.4 Calculating PageRank as a eigenvalue computation

The probability distribution of the surfer's position over all possible pages can be viewed as a vector $\vec{x}$ of cardinality N where N is the number of pages in the web. At start ($t = 0$), the random surfer could begin serfing from any position and the corresponding entry in $\vec{x}$ will be 1 and the rest will be 0. At $t = 1$, the probability distribution is given by $\vec{x}P$. At $t = 2$ the probability distribution of the surfer's position is given by $(\vec{xP})P = \vec{x}P^2$ and so on. If we carry on this process multiple times, the probability distribution of the surfer's position ($\vec{x}$) eventually converges to a limiting distribution regardless of the initial value of $\vec{x}$ (this can be proved if the underlying graph is strongly connected and aperiodic - which we have ensured while transforming the adjacency matrix in subsubsection 3.2.2.3).

It is interesting to note that the probability distribution of the random surfer's position gives the PageRank of the pages. So, the limiting distribution of $\vec{x}$ we just found, is nothing but the PageRank values. When $\vec{x}$ reaches a limiting distribution, farther application of matrix P on it does not change $\vec{x}$. Let the limiting distribution of $\vec{x}$ is $\vec{\pi}$. In that case,

$$\vec{\pi}P = 1 \cdot \vec{\pi} \tag{3.4}$$

From the above equation it is clear that 1 is an eigenvalue of the matrix P. And $\vec{\pi}$ is the principal left eigenvector of the matrix P which is nothing but modified adjacency matrix of the web graph. We can now use any eigenvector finding algorithm to calculate the PageRank of the pages.

3.2.2.5 Calculation of PageRank

Although the expression for PageRank is a recursive one, it is calculated in an iterative way. At first all the pages are given uniform PageRank ($= 1/N$ where N is the number of pages in collection). With every iteration, the PageRank values are approximated by Equation 3.1. After a point of time the process converges.

3.2.2.6 Convergence of PageRank algorithm

We discussed in subsubsection 3.2.2.4 that the distribution of PageRank eventually converges to a limiting distribution. Experiments done in (Page et al., 1998) show that, PageRank algorithm over a large 322 million link database converge to a reasonable tolerance within roughly 52 iterations.

A Markov chain or a random walk on a graph is expander if every subset of nodes S has a neighborhood at least α times greater than S. A graph has a good expansion factor if it has a good eigenvalue separation. It can be proved that if the graph is expander, then it quickly (in logarithmic time of the size of the graph) converges to a limiting distribution. Therefore, it can be proved that the PageRank computation terminates in logarithmic time. Figure 3.2 shows the convergence of PageRank algorithm.

3.2.2.7 Disadvantage of PageRank

PageRank favors older pages than newer ones. The older pages are expected to have more number of citations from important page than a page just introduced. Therefore, page rank should not be used as a standalone metric. It should be used as a parameter only.

3.2.3 HITS algorithm

Hyperlink Induced Topic Search or HITS algorithm is a Link Analysis algorithm developed by Jon Kleinberg (Kleinberg, 1999). This algorithm computes two values for any page:

Hub Score The value of its links to other pages

Authority Score The value of the page's content.

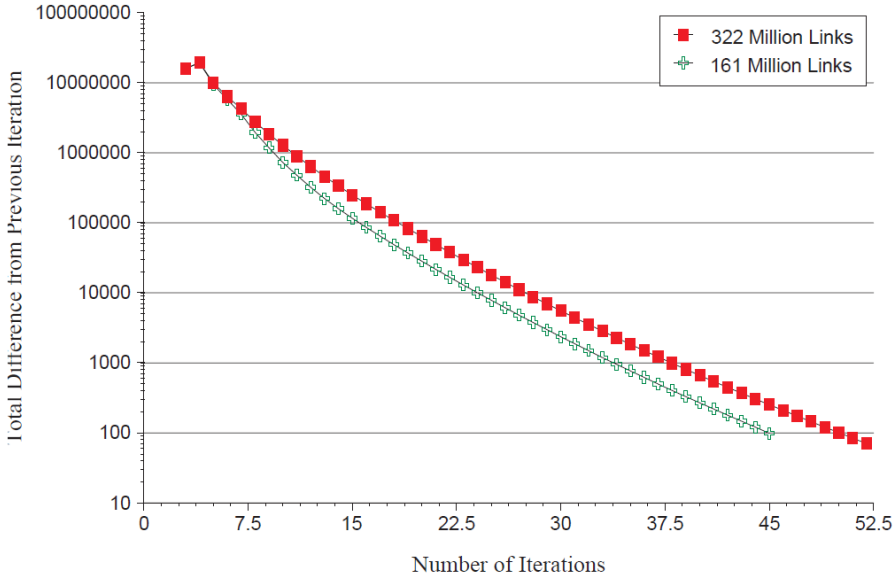

Figure 3.2: Convergence of PageRank algorithm (Courtesy: (Page et al., 1998))

Both the scores are calculated using each other.

A page with good authority score are authoritative sources of information in that topic. Pages with good hub score do not carry much information in themselves but contain links to good authoritative pages (Manning et al., 2009). Hub pages are generally manually compiled collection of links on some specific topic. A good hub page is one that has links to many good authoritative pages. A good authoritative page should have backlinks from many good hub pages.

3.2.3.1 As a Topic Specific Search

HITS is a topic specific search. First of all a subset of web pages containing good hub and authority pages with respect to a query is created. This is done by first firing the query and getting an initial set of documents relevant to the query (say 200 documents) (Manning et al., 2009). This is called the root set for the query. The subset of pages is created by including this root set and all the pages that either links to any of the pages in the root set or is linked by a page within this root set.

The intuition behind this is the following. A good authoritative page in the query may not contain the text of the query (for example, home page of IBM does not contain the word "computer"). So, it may not be retrieved as a result of the query. But the good hub pages are expected to contain the query words. If we manage to capture a good hub pages, we include the good authoritative pages by traversing the 'out-links' of that pages. Similarly, if the root set contains a good authoritative page, we will get good hub pages by traversing the backlinks from that page.

3.2.3.2 Computation of hub and authority scores

Formally, the hub score is computed as the sum of the authority scores of all pages it links to. The authority score is the sum of hub scores of all the pages connects to it. So, the authority score increases if many hub pages links to that page. Mathematically,

$$h(v) \leftarrow \sum_{v \longmapsto y} a(y) \tag{3.5}$$

$$a(v) \leftarrow \sum_{y \longmapsto v} h(y) \tag{3.6}$$

$$\tag{3.7}$$

where $h(v)$ is the hub score of page v, $a(v)$ is the authority score of page v, $v \longmapsto y$ denote the existence of a hyper link from v to y.

Given a subset of web pages containing good hub and authoritative pages with respect to a query, the hub score and authority score is computed using iterative approximation. Initially we set the hub and authority scores of all the pages in subset to 1. Then we compute new hub scores using authority scores and vice versa. This iterative process continues until the hub and authority values converge.

3.2.3.2.1 As a principal eigenvalue computation Let $\vec{h}$ and $\vec{a}$ denote the hub and authority scores of all the pages respectively. Let, A be the adjacency matrix of the subset of the web graph we are dealing with. Then, one step of approximation in Equation 3.5 and Equation 3.6 can be written as a linear transformation of the vectors $\vec{h}$ and $\vec{a}$.

$$\vec{h} \leftarrow A\vec{a} \tag{3.8}$$

$$\vec{a} \leftarrow A^T\vec{h} \tag{3.9}$$

Substituting the values into one another in Equation 3.9 and Equation 3.8 we get,

$$\vec{h} \leftarrow AA^T\vec{h} \tag{3.10}$$

$$\vec{a} \leftarrow A^TA\vec{a} \tag{3.11}$$

When there is only scalar changes in the values of $\vec{h}$ and $\vec{a}$, we say the algorithm has converged. (We are only interested in relative hubs and authority values). In that case, we can write,

$$\vec{h} = (1/\lambda_h)AA^T\vec{h} \tag{3.12}$$

$$\vec{a} = (1/\lambda_a) \leftarrow A^TA\vec{a} \tag{3.13}$$

Where λ_h is the eigenvalue and $\vec{h}$ is the corresponding principle eigenvector of the matrix AA^T. Similarly, λ_a is the eigenvalue and $\vec{a}$ is the corresponding principle eigenvector of the matrix A^TA.

3.2.3.2.2 The HITS algorithm The HITS algorithm works in following way (Manning et al., 2009):

1. It assembles the root set of relevant documents by firing the query first.
2. It creates the base set of hubs and authorities from the root set through the procedure given in subsubsection 3.2.3.1.
3. It computes AA^T and A^TA from the adjacency matrix of the base set of documents (a subset of web-graph).
4. It computes the value of $\vec{h}$ and $\vec{a}$ by computing the eigenvectors of AA^T and A^TA respectively. Any algorithm for computing eigenvector like the power iteration method (Manning et al., 2009) can be used.
5. Output the top scoring hubs and authorities.

3.2.4 OPIC Algorithm

Online Page Importance Calculation (OPIC) is a link analysis algorithm that can compute the page importance at the time of crawling. Contrast this with PageRank, where the web is crawled and the web-graph is computed and frozen. A separate offline process computes page importance, which may take hours or days for a very large graph. OPIC, on the contrary, computes page importance online (directly at the time of crawling) and with limited resource (Abiteboul et al., 2003).

This algorithm maintains two values for each page: cash and history. All pages are given some initial cash. When the page is crawled, it distributes its current cash equally to all pages it points to. The cash of a node records the sum of the cash obtained by the page since the last time it was crawled. The history variable stores the sum of the cash obtained by the page since the start of the algorithm until the last time it was crawled. The importance of a page is given by the value in the history of the page. The intuition is that the flow of cash through a page is proportional to its importance (Abiteboul et al., 2003). When a page is retrieved by the web agent (while crawling) we record its cash value to the history and distribute its cash equally to all the pages it links to.

To make the graph strongly connected (it is required for convergence) a virtual root node is added. The root node has links to every other node and every other node has link to it. The algorithm starts by giving a fixed amount of cash to the root node which it equally distributes over all other pages. Whenever a new page is crawled, the root gets back some cash. The nodes with no out edges give all their cash to the root node. As an online process, OPIC does not require the complete graph to be present beforehand. It can be adapted to the dynamic nature of the web. OPIC algorithm can also be used for selecting pages for crawling.

3.3 Anchor Text

Anchor texts are visible, clickable text in a hyperlink. It is the text between the two anchor tags in a HTML hyperlink definition. The anchor text generally describes the page where the link points to.

An example of HTML anchor is:
`<a href=http://www.cse.iitb.ac.in>Computer Science Department, IIT Bombay</a>.`

3.3.1 Importance of Anchor texts in web IR

In the web it is often found that pages do not contain a good description of themselves. For example, the home page of IBM does not contain the terms "Computer". Web search users often use query terms that are not present in the relevant web pages. In that case, it is hard for a search engine to come up with some of the most relevant documents.

But we can exploit the fact that although pages may not contain terms to describe themselves, the anchor texts pointing to the page will contain terms describing of the page. For example, links to IBM homepage includes the term Computer. We can add the terms appearing in those anchor texts to the description of the page as a special indicator. Using anchor texts has two advantages (Brin and Page, 1998): 1) they often describe the target page more accurately than the target page itself and 2) anchors may exists for documents that cannot be indexed (*e.g.* programs, images *etc.*) The weight given to the anchor texts are usually more than usual terms in the web page.

Like anchor text, the text surrounding the anchor can also be used (Manning et al., 2009). Take, for example, the fragment of text "Click here for a good discussion on Information Retrieval". It is a matter of research of how much of the surrounding text should be considered. Anchor texts are primary technology for image search engines.

3.3.2 Disadvantage

Using anchor texts in this way has a room for manipulation. Since a website developer can create a misleading anchor text, it is often used for orchestrated campaign against specific sites (Manning et al., 2009). In this case lots of links are created with often derogatory anchor texts, all pointing to a particular website. This way, it may happen that when a user searches with those terms, that particular website comes as a top result. This kind of spamming is often called "Google Bomb". Visit `http://en.wikipedia.org/wiki/Google_bomb` for examples of different funny Google Bombs.

3.4 End Notes

In this chapter we discussed the differences and difficulties involved in web search. We studied features specific to web like link analysis anchor text and how they can be exploited in web search engines. A web document generally has many parts. Different weightage are given to the terms appearing in different parts of the pages. For example, the terms appearing in web page title, url, anchor texts and within headings are given more importance than terms appearing in body. The terms that appear in larger font size are also given more importance. The anchor texts generally describe the page they are pointing to rather than the page they are in. So, they should be part of the target page and not the source page.

Chapter 4

Evaluation of IR Systems

After proposing a new IR technique, it is usually evaluated against standard benchmarks.(Baeza-Yates and Ribeiro-Neto, 1999) The most common measures of evaluation are precision, recall, f-score *etc.* For a given IR system, the measure to evaluate depends on the goal of that IR system(Baeza-Yates and Ribeiro-Neto, 1999). For evaluation of IR systems we also generally need large universal document collections, a test suite which can be expressed as queries and a set of relevance judgment on the documents (usually each document is manually tagged positive or negative) (Manning et al., 2009). The different IR evaluation forums provide these. For a newly proposed IR system, it is often necessary to get a good performance against standard benchmarks in some reputed IR evaluation forums like TREC.

4.1 Different IR evaluation measures

Here we discuss different evaluation measures of Information Retrieval techniques. We first describe standard measures like precision and recall. Then we discuss different combined measures.

4.1.1 Precision

Precision measures the exactness of the retrieval process. If the actual set of relevant document is denoted by I and the retrieved set of document is denoted by O, then the precision is given by:

$$Precision = \frac{|I \cap O|}{|O|} \tag{4.1}$$

Precision measures among the retrieved documents, how many are relevant. It does not care if we do not retrieve all the relevant documents but penalizes if we retrieve non-relevant documents.

4.1.2 Precision at Rank k

In web search systems the set of retrieved document is usually huge. But it makes a lot of difference if the relevant document is retrieved early in the rank list that late. To take this into

account, precision at a cut-off rank k is introduced. Here the list of relevant documents I is cut-off at rank k. Only documents up to rank k are considered to be retrieved set of documents.

4.1.3 Mean average precision or MAP score

To represent the order in which the result was given, the mean average precision or MAP measure gives the average of precision at various cut-off ranks. It is given by:

$$MAP = \frac{\sum_{i=0}^{k} precision@i}{k} \tag{4.2}$$

4.1.4 Recall

Recall is a measure of completeness of the IR process. If the actual set of relevant document is denoted by I and the retrieved set of document is denoted by O, then the recall is given by:

$$Recall = \frac{|I \cap O|}{|I|} \tag{4.3}$$

Recall measures how much of the relevant set of documents we can retrieve. It does not care if we retrieve non-relevant documents also in the process.

Precision and Recall are not independent measures. It is seen that if we try to increase precision, the recall is reduced and vice versa.

4.1.5 F-Score

F-Score tries to combine the precision and Recall measure. It is the harmonic mean of the two. If P is the precision and R is the recall then the F-Score is given by:

$$F - Score = \frac{2 \cdot P \cdot R}{P + R} \tag{4.4}$$

4.2 IR Evaluation Forums

IR Evaluation forums provide framework to test various IR techniques against standard benchmarks. They provide the following things: (Manning et al., 2009)

- A Document collection
- A test suite of information needs expressible as queries
- A set of relevance judgments. Usually, a pool of documents is manually judged as relevant or non-relevant.

One key thing to note is that relevance is judged against information needs and not queries (Manning et al., 2009). The task of converting an Information Need to a `system query` must be done by the system itself and becomes a part to be evaluated (Baeza-Yates and Ribeiro-Neto, 1999). For example, the information need could be

whether drinking red wine is helpful for heart diseases or not

and the query could be:

$$Red \wedge wine \wedge heart \wedge attack$$

. A document containing all those terms is not necessarily relevant.

Here we discuss three such evaluation forums namely TREC, CLEF and FIRE.

4.2.1 TREC

TREC stands for Text REtrieval Conference. It is co-sponsored by NIST (US Gov.) and the Information Technology office of Defense Advanced Research Project Agency (DARPA). It started in 1992.

The purpose of TREC as stated in TREC website is:

> to support research within the information retrieval community by providing the infrastructure necessary for large-scale evaluation of text retrieval methodologies

Seventeenth TREC was organized in 2008 at Gaithersburg, Maryland.

4.2.1.1 Goals of TREC

The goals of TREC as given in TREC website are:

- to encourage research in information retrieval based on large test collections;
- to increase communication among industry, academia, and government by creating an open forum for the exchange of research ideas;
- to speed the transfer of technology from research labs into commercial products by demonstrating substantial improvements in retrieval methodologies on real-world problems;
- to increase the availability of appropriate evaluation techniques for use by industry and academia, including development of new evaluation techniques more applicable to current systems.

4.2.1.2 TREC Tracks

TREC is divided in different research areas called TREC Tracks. These tracks act as incubators of new areas and creates necessary infrastructure for the area. Each track has a task to be performed. Multiple groups across the globe participate to do the task. The participating groups are provided with data sets and test problems. Each Track has an open-to-all mailing list to discuss the task. Tracks may be added or removed from TREC depending on changing research needs. Currently there are 7 TREC Tracks.

4.2.1.3 TREC Topics

TREC retrieval tasks are to be performed with reference to certain test information requests (needs) specified in TREC topics. Each topic is a description of information need in natural language. The task of convert a topic into a system query must be done by the system itself (Baeza-Yates and Ribeiro-Neto, 1999).

4.2.1.4 TREC Document Collection

TREC provide an ever growing document collection to their participants. Participants build their retrieval systems to carry out the search. They send their results to TREC. TREC compares results coming from different participants and this way evaluates different approaches. Mainly precision and recall of the system are judged against human evaluated relevance information.

4.2.1.5 TREC Evaluation

Evaluation is done by comparing the results obtained by an IR system with human evaluated relevance results (Baeza-Yates and Ribeiro-Neto, 1999). For this pool of relevant of documents are marked for each topic by human assessors. This is done by taking the top K (say K = 100) documents from the rankings of each participants. The set obtained by taking union of these sets are then relevance judged by human. This procedure is called pooling. It is assumed that the documents beyond rank K can be considered non-relevant by both the TREC organization and the participants.

4.2.1.6 TREC - CLIR Track

The Cross-Lingual Information Retrieval track in TREC was introduced in TREC-6 (1997) and continued up to TREC-11 (2002) (Source: TREC Website).

4.2.1.7 Official Website

`http://trec.nist.gov/`

4.2.2 CLEF

CLEF or Cross Language Evaluation Forum is an IR evaluation forum dealing in multilingual information retrieval in European languages. It was founded by European Union in 2000 (Manning et al., 2009). It is co-ordinated by the Istituto di Scienza e Tecnologie dell'Informazione, Consiglio Nazionale delle Ricerche, Pisa. The forum promotes R&D in multilingual and multimedia information access [1].

The aims of CLEF according to the official website are as follows:

- Developing an infrastructure for the testing, tuning and evaluation of information retrieval systems operating on European languages in both monolingual and cross-language contexts
- Creating test-suites of reusable data which can be employed by system developers for benchmarking purposes.

The evaluations are concentrated on cross-language information retrieval in European languages. The 10th CLEF workshop was held between 30 September and 2 October at Corfu, Greece.

[1] Source:`http://www.daedalus.es/en/r-d-i/clef-cross-language-evaluation-forum/`

Like TREC, CLEF is also divided into topics (tracks) that researches on different areas. The key areas are [2].: searching on a text (ad-hoc task), geographical information search (GeoCLEF), search of information on the web (WebCLEF), image retrieval (ImageCLEF), question answering systems (QA@CLEF), *etc.*. Each track offers a task (problem) which is relevant to that research area. A collection of objects (documents, news, images *etc.*) is given to the participants. They are also given certain search objectives (topics) and queries. Based on these, the participants build indexes and the ranking system. They execute their searches and send the result in a standard format. The organizers select the best groups based on the precision they achieve.

4.2.2.1 Official Website

`http://www.clef-campaign.org/`

4.2.3 FIRE

Forum for Information Retrieval Evaluation (FIRE) is an evaluation forum co-ordinated by Indian Statistical Institute, Kolkata. The goals of FIRE as given in their website are:

- To encourage research in South Asian language Information Access technologies by providing reusable large-scale test collections for ILIR experiments
- To explore new Information Retrieval / Access tasks that arise as our information needs evolve, and new needs emerge
- To provide a common evaluation infrastructure for comparing the performance of different IR systems
- To investigate evaluation methods for Information Access techniques and methods for constructing a reusable large-scale data set for ILIR experiments.

FIRE-2010 was organized in 19-21 February 2010 at DAIICT, Gandhinagar, Gujarat, India. The organization of FIRE is quite similar to that of TREC or CLEF.

4.2.3.1 FIRE Tracks

The principal tracks of FIRE'2010 were (as given in official website of FIRE):

1. **Ad-hoc monolingual document retrieval in Bengali, Hindi Marathi and English.**
2. **Ad-hoc cross-lingual document retrieval**
   - documents in Bengali, Hindi, Marathi, and English,
   - queries in Bengali, Hindi, Marathi, Tamil, Telugu, Gujarati and English.
3. **Retrieval and classification from mailing lists and forums.** This is a pilot task being offered by IBM India Research Lab.
4. **Ad-hoc Wikipedia-entity retrieval from news documents**
   - Entities mined from English Wikipedia
   - Query documents from English news website

[2]Source:`http://www.daedalus.es/en/r-d-i/clef-cross-language-evaluation-forum/`

4.2.3.2 Official Website

`http://www.isical.ac.in/~fire/`

4.2.4 NTCIR

NII Test Collection for IR Systems (NTCIR) is co-ordinated by National Institute of Informatics (NII), Japan. It is an information retrieval evaluation forum like TREC or CLEF, specifically focused towards East Asian languages like Japanese, Chinese and Korean. The goals as given in the official website are:

- To encourage research in Information Access technologies by providing large-scale test collections reusable for experiments and a common evaluation infrastructure allowing cross-system comparisons
- To provide a forum for research groups interested in cross-system comparison and exchanging research ideas in an informal atmosphere
- To investigate evaluation methods of Information Access techniques and methods for constructing a large-scale data set reusable for experiments.

The structure and procedures are quite similar to other forums like TREC and CLEF. The 8th NTCIR workshop was on June15-18, 2010 at NII, Tokyo, Japan.

4.2.4.1 Official Website

`http://research.nii.ac.jp/ntcir/index-en.html`

4.3 End Notes

In this chapter we discussed various parameters on which an IR system is evaluated. We also discussed the need of IR evaluation forums for IR evaluation. We then discussed some of the prominent IR evaluation forums like TREC, CLEF and FIRE.

Chapter 5

Cross Lingual Information Retrieval

Internet is a great repository of information. With state of the art search engines like Google, Yahoo *etc.*, the information is easily available. But there is still a problem. Most of the information in web is in English (56.4%[1]). So, for a person who does not know English, this vast information is simply not available.

This problem is more appropriate for multi-cultural and multi-lingual country like India. Only 21.09% of the total population of India can speak English. Compared to urban areas, very few people in rural areas can speak English. Imagine how nice it would be if a farmer in a village could access internet to know the right ways of farming. With this idea in mind, the subfield Cross Lingual Information Retrieval (CLIR) was started. In CLIR, the language of the documents is different from the language of the query.

There are three basic approaches to CLIR. These are:

Query translation approach The query is translated into the language of the document. Many translation schemes could be possible like word-to-word dictionary based translation or more sophisticated machine translations. The basic problem in this approach is, the user is often knows only the query language. So, even if the foreign language documents are retrieved, they are of no use as the user cannot read them.

Document translation approach This approach translates the documents instead of the query. Although this approach alleviates the problem stated above, this approach has scalability issues. There are too many documents to be translated and each document is very big compared to a little query. This makes this approach practically unsuitable.

Interlingua based approach In this case, the document and the query are both translated into some common Interlingua (like UNL [2] or SRS [3]). This approach is not much investigated yet. It generally requires huge resources as the translation needs to be done online.

[1] Source: `http://www.netz-tipp.de/languages.html`
[2] `http://www.undl.org/`
[3] `http://mt-archive.info/MTS-2005-Mohanty.pdf`

To overcome the problems in the query translation and document translation based approaches, a quick hack could be used. Here the query is translated and documents are retrieved. Snippets are extracted from the retrieved documents and only the snippets are translated in the language of the query. The user then selects one of the retrieved documents based on the document URL or translated snippet. The selected document is then fully translated in the document of the user.

In this chapter we will discuss the CLIR system being developed at IIT-Bombay as a part of the nation (India) wide project funded by TDIL, Department of IT, Government of India. The interface is available at: `http://www.clia.iitb.ac.in/clia-beta-ext/` [4]. This chapter also demonstrates various aspects of an IR system through this case study.

5.1 The CLIR System Architecture

The present CLIR system accepts the query in one of the six languages (English, Hindi, Bengali, Marathi, Tamil, Telegu and Punjabi) and fetches documents in English, Hindi and that of the query language. The overall architecture is defined in Figure 5.1. The whole system is being developed across many institutes all over India (as indicated in the figure).

There are three basic modules in this system:

Input processing Module This module process the query. The output of this module is the query properly translated in foreign language.

Search Module With translated query as input, this module searches the document index. The system uses the search engine Nutch as the underlying searching technology with some tweaks in it.

Output generation module From the ranked results given by the search module, this module generates snippets of the result documents and translates the snippets back to query language. This module presents the output of the system to the user in a web interface.

5.1.1 Input processing module

Let us now discuss the steps involved in input processing module one by one:

Language Analyzer It works on both the query and the document. It involves the following steps:

- **Tokenizer** Extracts word tokens (index terms) from running text. For example, given a piece of text: "Places to be visited in Delhi" it outputs [places, to, be, visited, in, Delhi].
- **Stop-word eliminator** Stop-words are those words that do not have any disambiguation power. Common examples of stop words are articles, prepositions *etc.* In this step, stop words are removed from the list of tokens. For example, given the list of token generated by tokenizer, it strips it down to: [places, visited, Delhi].
- **Stemmer** The remaining tokens are then stemmed to the root form (*e.g.* visited → visit). For example, after stemming the list of tokens becomes this: [place, visit, Delhi].

[4] Use `http://www.clia.iitb.ac.in:8080/clia-beta-ext/` inside IIT Bombay

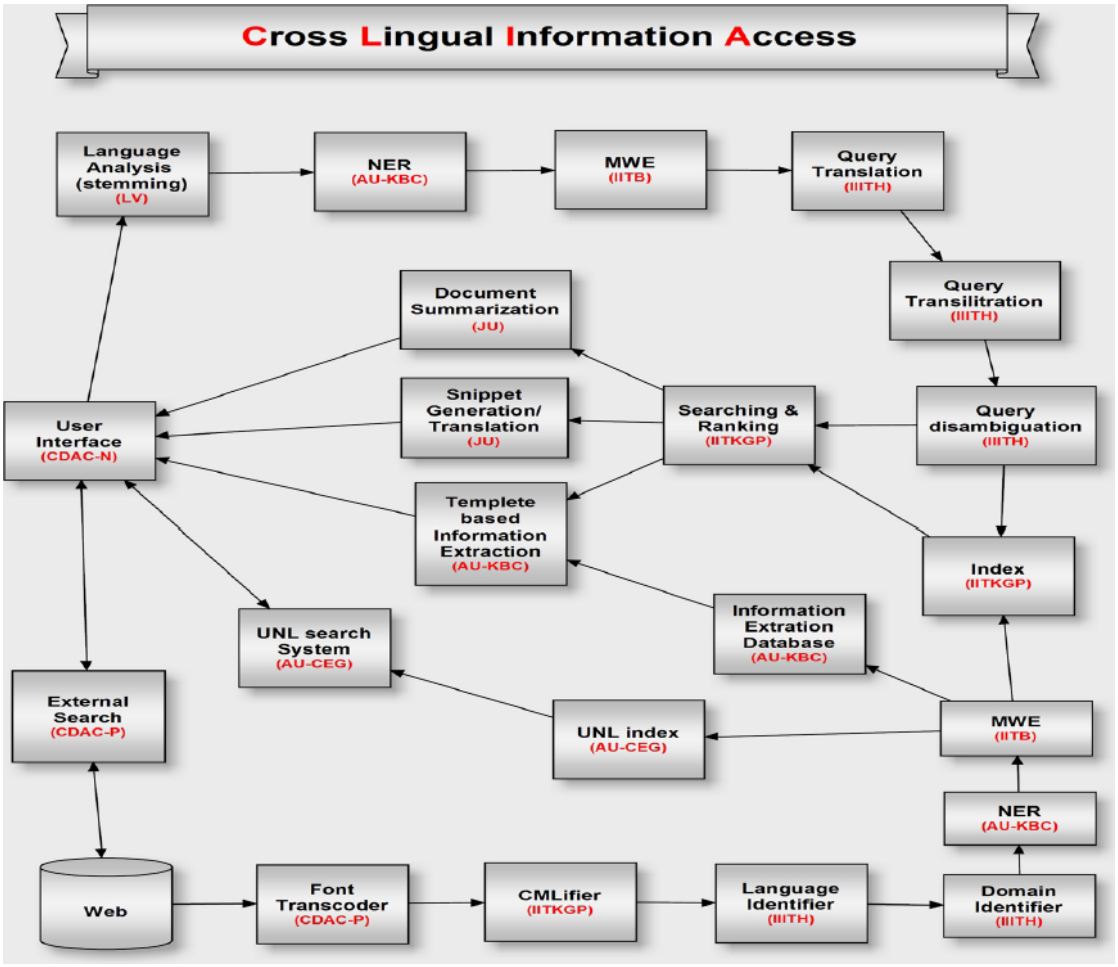

Figure 5.1: CLIR System Architecture at IIT Bombay

Named Entity Recognizer This step detects proper nouns from the list of tokens. The named entities are marked not to be translated. For example, given a list of tokens [places, visited, Delhi] this step recognizes Delhi as a named entity.

Multi-Word Expression Recognizer It detects phrases to make "dictionary based phrase translation" possible.

Query Translation In this section query tokens are translated into foreign language. Simple dictionary based word-to-word translation is used as we have already lost the semantics through tokenization process of the query and document and we have a list of tokens to be translated (not sentences). Translation is done using two sub modules:

Query Translation Module This module translates the using multi-lingual and bi-lingual dictionaries.

Query Transliteration Module This module transliterates the tokens from one script to other script. Named entities and the entries that are not found in dictionary are transliterated.

5.1.2 Search Module

The search module is responsible for

- Crawling and indexing the documents
- Given the query it returns a ranked set of documents as a result of the query

The system uses Apache Nutch as an underlying platform for carrying out these tasks. The system presently crawls and indexes documents only from the tourism domain. The sub-modules of Search are:

Font Transcoder Many Indian language documents are written using proprietary fonts. An open source font transcoder Padma is used to convert them to Unicode.

Focused Crawler This is a special type of crawler that is trained to crawl documents only from tourism domain.

Language Identifier Identifies the language of the document using Natural Language Processing.

Indexer The pages crawled are tokenized, stop-word removed and stemmed using the language analyzer module. Then it is put into an inverted index.

Ranking Given a query, relevant documents are ranked by the system in order of relevance. Default scoring function of Nutch with some modification is used as the ranking function.

5.1.3 Output Generation Module

This module presents the search results along with snippets in a web interface. The sub-modules involved here are:

Snippet Generator Snippets are generated for all the web documents that are marked as relevant with the query. Snippet consists of the title of the document and a very brief summary of the document keeping the query in mind.

Summary Generator Summaries of web pages may be provided on request.

Snippet translator Snippets generated are translated from the document language to the query language.

5.2 Ranking Scheme in IITB-CLIR system

The CLIR system at IITB uses the default Nutch ranking with some modifications. The default ranking scheme in Nutch is described in subsection 2.2.6. Here we will discuss the modifications done.

5.2.1 Query Formulation

Most of the modifications are done in the query formulation part. Query formulation is very important to get required precision and recall. For web search engines usually precision is more important than recall as the total number of relevant documents is too huge to be considered by the user. While designing different strategies for query formulation, we keep this in mind. The different strategies for query formulation in IITB cross-lingual system are:

5.2.1.1 Fall-back strategy

The set of index terms in query as returned by the Language Analyzer (section) are joined using AND operator. This means only those documents that contain ALL of the query terms will be considered for ranking [5]. This formulation of query improves precision but affects recall negatively. To circumvent this we have a fall-back strategy. If no results are returned by the ANDed query, the system relaxes the query and puts OR operator instead of the AND. This increases recall.

5.2.1.2 Query Disambiguation

Sometimes a single query has more than one translation/transliteration candidates in the document language. This is due to the polysemous words in the language of the query. The system disambiguates various alternative translation candidates through the following strategy. It generates all the translation/transliteration candidates in the document language and keeps only that candidate which gives highest number of matches (hits) in the document collection. This naïve way of disambiguation performs well in practice.

[5] As discussed in subsection 2.2.6, Nutch first uses a Boolean model to retrieve an initial set of documents before ranking them.

5.2.1.3 Phrasal Query

Documents which contain all the query term in same sequence should be ranked higher in the result. A phrasal query only matches those documents that meet this requirement. A parameter called "slop" is used to control the number of other words allowed to occur between words in a phrase. Slop distance between two phrases is defined as the number of moves required to construct one phrase from other. If slop is zero, it is an exact phrase search. The slop value by which the phrase matches with the document determines how weight the document should be given. Hence, the weightage given to a document for phrasal matching is given by:

$$weight = \frac{1}{\text{total slop} + 1} \tag{5.1}$$

The total slop value in Equation 5.1 is calculated as a sum of slops of all sloppy phrase matches in a document. In the denominator, 1 is added to circumvent the problem caused by slop = 0.

5.2.2 Effect of OPIC Score

As discussed in subsection 2.2.6, in the default ranking scheme in Nutch, the field doc.getBoost() capture the OPIC score of that document. OPIC algorithm was discussed in subsection 3.2.4. In fact, it was found in experiment that the OPIC factor takes a decisive role in ranking (Vachhani, 2009).

But the OPIC score behaved badly in some kinds of situations. Firstly the OPIC score vary greatly from document to document. For some documents it was zero and for some documents the OPIC score was more than 500. It also had other side effects (Vachhani, 2009). This problem was solved by limiting the OPIC score at 100 (any document having OPIC score more than 100 is given and OPIC score of 100).

5.2.3 Experimental results

In this section we compared the results obtained by two systems (Vachhani, 2009):

CLIA-IITB It incorporates all the above mentioned modifications to the default similarity based ranking scheme in Nutch *namely,* Fall-back strategy, query disambiguation, phrasal query and limiting the OPIC score.

CLIA-BETA It uses the default ranking of Nutch with none of these modifications.

The following table summarizes the results obtained by the two systems. Clearly, the modifications that were done greatly improved performance.

Precision k	CLIA-IITB	CLIA-BETA	Improvement
P1	0.76	0.23	330
P2	0.76	0.19	400
P3	0.76	0.15	584
P5	0.65	0.13	500
P10	0.52	0.13	400

Chapter 6

Conclusion and Future Direction

The goal of any information retrieval system is to retrieve documents that match the information need of the user. But it is very difficult to specify the information need to the IR system. Sometimes, even the users don't know their information need precisely. The information need may not be represented by the query given to the search engine. Given this possibly vague description of the information need (the query), The IR system searches the document index to retrieve and rank documents according to their relevance to the query given.

To calculate the relevance, classical IR systems consider features only from the document itself. The features are often represented as term-weights given to individual index terms appearing in the document (after stop-word removal and stemming). Two well-known parameters used for fixing the term-weights are term-frequency and inverse document frequency of index terms. This approach was suitable for retrieval from well-controlled repositories like citation analysis. But for IR situations like Web Search, where there is practically no control over the document collection, depending only on internal features of the document is prone to manipulation. That is why, document independent features like link analysis and anchor text is used for ranking in such cases.

Many IR classical IR systems represent both queries and documents as a collection of independent index terms. While reducing the complexity of the system, this suffers when there is dependency between the index terms e.g. phrases, famous quotes etc. Example: If the query given is "To be or not to be", the search engine, if does not recognize phrases, may remove all the terms as stop-words. Modern search engines use different techniques to catch the dependency among terms.

Many different techniques are used by modern IR systems to merge the gap between the information need and its query representation. These include query expansion, relevance feedback etc. Past interactions of the user with the search engine can also be studied to predict the information need. This way, the search results may be customized and directed to the specific information needs of individual users.

Another assumption taken by most of the IR systems is, the documents in the result page are independent to each other. This may not be true as users are more interested in finding novel information in different documents in the search results than the same information in all the search results.

In this chapter we discuss how the search engines can be made even better. First we discuss how we can personalize web-search in section 6.1. Then we discuss a new concept called Quantum

IR that takes inter-document dependency into account.

6.1 Personalizing Web search

Different users have different types of information need. Queries never fully represent the complete information need. For example, the query 'Matrix' can refer to the matrix concept of mathematics or the 1999 movie by Wachowski brothers. The idea behind is, as the information need varies from person to person, the search results should also be different for different people.

A search engine can predict user's information need by using user's past interactions with the search engine like which queries they fired earlier, which links they visited and so on. This is similar to modeling the user for e-Marketing websites.

User interactions are mainly collected in two ways:

1. The queries fired by users earlier are logged in a **Query Log**
2. The links visited by the user in the Search Results page is recorded (**Click-through data**)
3. Various other types of parameters could be used, like the rank of the result document the user clicked, the time user took to return to the search page, did the user visit second result page, did he reformulated the query immediately *etc.* All these information can be used to refine the above two parameters.

Goal of Web Search is to return the document the user wants to see. But only the user knows what he wants to see and the query does not completely represent the information need. But the intended information need gets clear when the user chooses the document he wants to see from the list of documents returned by the IR system. Clickstream data logs that information. We can use that information to make next search results better and more user-oriented. For example, when the user fires a query 'Matrix' we can use his earlier interactions with the search engine to cater him customized search results. If he searched topics mostly from mathematics domain earlier, it is more likely that he is interested in the concept of matrix in mathematics so the mathematical documents will be ranked higher than movie reviews.

Also, User often cannot frame good queries. This can be solved by recommending better queries to the user. We can make use of Query Logs of other users in this purpose. But we should only suggest queries yielding good quality result (using clickstream data).

Clickstreams and query logs *etc.* can be used to enhance web search in at least the following ways:

6.1.1 Query Recommendation

When a user gives a query, we can suggest user more queries he might be interested in. For example, if a user gives a query "Places to visit in New Delhi", he might be interested in visiting Delhi and Agra together. So, the query "Places to visit in Agra" can be suggested to him. Similarly "Hotels in New Delhi", "Hotels in Pahadganj", "How to reach Agra from Delhi" are examples of other few queries that can be suggested.

6.1.2 Query Expansion

User's often cannot give good quality queries. Additional terms can be added based on related queries found in the query log.

6.1.3 Ranking

While deciding the ranks of the two documents, user modeling can be helpful in the following way. Say, there are two documents A and B. A is from sports domain, another is from banking. If the previous behavior of the user suggests that the user generally fires queries in the banking domain, visits finance related websites most then the document B will be ranked higher than document A (vice versa).

6.1.4 Relevance Feedback

Relevance feedback is one of the classical ways of refining search engine rankings. It works in the following way: Search engine firsts generate an initial set of rankings. Users select the relevant documents within this ranking. Based on the information in these documents a more appropriate ranking is presented (for example, the query may be expanded using the terms contained in the first set of relevant documents). Like in the example given before, when the user fires a query 'matrix', initially documents on both the topics (movie and mathematics) are retrieved. Then say, the user selects the mathematical documents as relevant. This feedback can be used to refine the search and retrieve more documents from mathematics domain. But users often do not want to participate in giving feedback. We can use clickstream data to approximate relevance feedback. If a user clicks on a document at say rank 3 and skips rank 2, we can say that document at rank 3 is more relevant than document at rank 2.

6.2 Quantum Probabilistic Retrieval Model

All the ranking procedures we discussed so far, assume that the ranking of one document does not influence the ranking of other documents. But it may not be always true for real world situations. Often multiple relevant documents carry exactly same information. So, after seeing a document A, the user will possibly like to see another document B that caters information need in some different aspect. After fixing the documents up to rank K, the document at rank $K+1$ should be chosen in such a way that it gives some new information. This is denoted as the interference between the documents. The ranking should be determined by probability of relevance as well as the interference with previously ranked documents.

This has a strong analogy with the interference of particles in quantum physics. We first discuss the Young's double slit experiment to set the context of discussion:

6.2.1 Young's Double Slit Experiment

Young's double slit experiment show the difference between classical probabilities and quantum probability. This experiment consists of shooting a physical particle (say electron) towards a screen with two slits (A and B). Let the random experiment be the particle passing through one of the slits and hitting a detector positioned at X. This is shown in Figure 6.1.

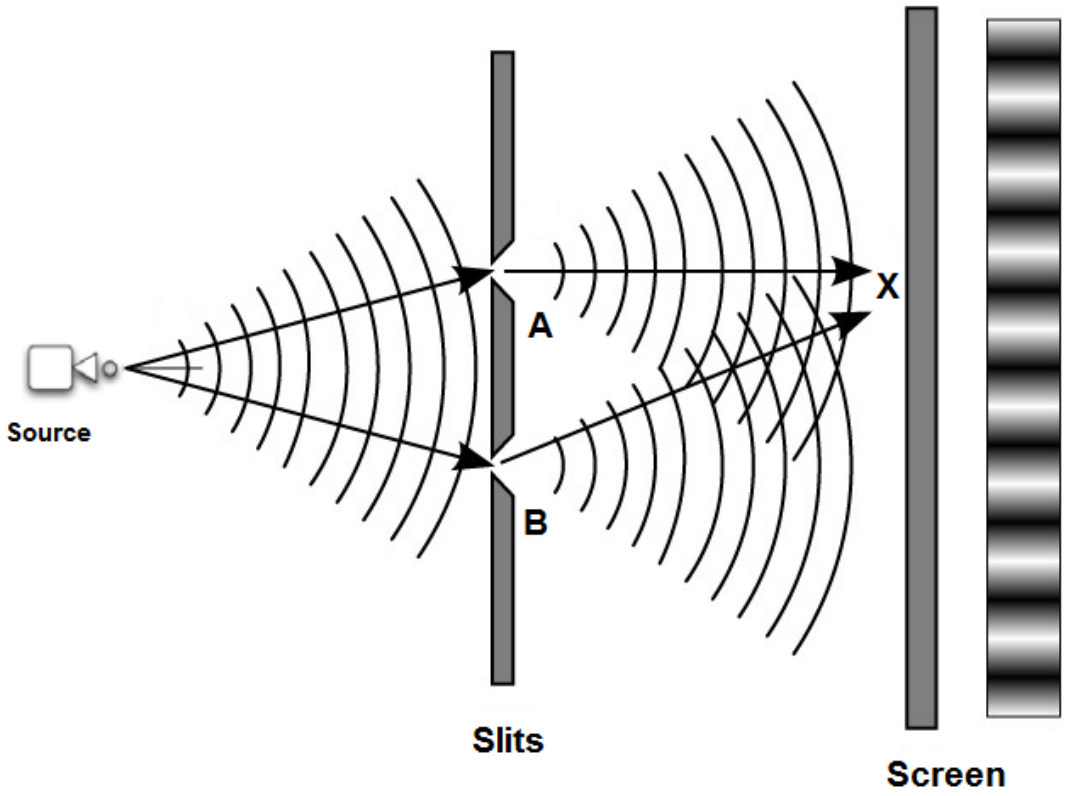

Figure 6.1: Young's Double Slit Experiment. Courtesy: `http://www.physicsoftheuniverse.com`

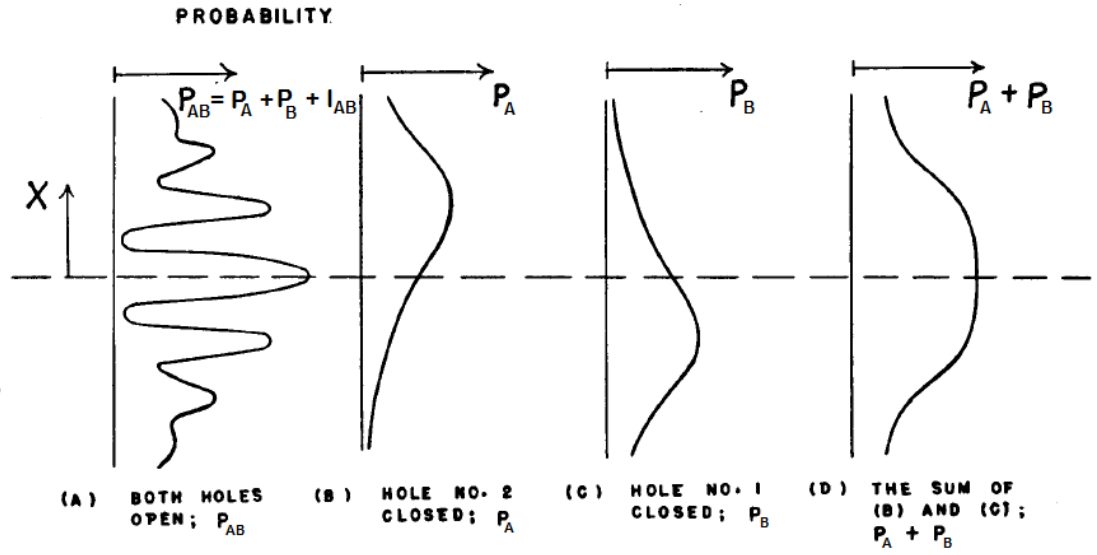

Figure 6.2: Influence of interference: Probability of arrival of electrons at X plotted against the position X of the detector (Courtesy: (Feynman, 1951)).

Now say slit B is closed and only slit A is open. Then say the probability of detecting the particle at point X (where the distance from the center of the plate is x) is $P_A(x)$. Similarly, the probability of detecting the particle at X with slit B open and slit A closed is $P_B(x)$. Let $P_{AB}(x)$ be the probability of detecting the particle at point X with both of the slits open. Then, using classical theory of probability, $P_{AB}(x) = P_A(x) + P_B(x)$. But this is experimentally shown that $P_{AB}(x) \neq P_A(x) + P_B(x)$. The reason is, quantum interference which needs to be accounted for. It can be showed that $P_{AB}(x) = PA(x) + PB(x) + I_{AB}(x)$ where $I_{AB}(x)$ represent quantum interference between the events associated with $P_A(x)$ and $P_B(x)$. This thing is explained in the Figure 6.2 taken from a paper by R. P. Feynman.

6.2.2 Analogy with IR process and the Quantum Probabilistic Retrieval Principle

In our analogy, the particle is associated with the user and his information need. The event of passing from the left of the screen to the right (through a slit) is the act of examining the ranking of the documents. Detecting the particle at position x is the decision of user to stop his search (user is satisfied). The event of not detecting the particle is the fact that user is not satisfied.

Now let us follow this analogy to classical theory of probability. Here, to increase the probability of the user getting satisfied is similar to increase the probability of detecting the particle at X ($P_{AB}(x)$). Now say, with slit A kept constant, we are varying the position, size *etc.* of slit B. Then $P_{AB}(x)$ can be maximized by maximizing $P_B(x)$ only. The problem is to find out the combination of A and $B_i \in \mathfrak{B}$ (set of all possible slit B) that maximizes $P_{AB}(x)$. So, after fixing the topmost ranked document A, the next document B will be ranked according to their probability of relevance [1]. This is nothing but classical Probabilistic Ranking Principle.

[1] probability of relevance is nothing but the probability of user stopping the search when only document B is present

This does not consider any interference term. If the ranking of the first document A is fixed and if there are two more documents B and C then it will rank B before C if and only if $P_B(x) \geq PC(x)$.

But as we said earlier, the presence of document A can influence the ranking of the other documents. This can be modeled using analogy with quantum probability principle. Here for we maximize $P_{AB}(x) = P_A(x) + P_B(x) + I_{AB}(x)$. If we fix A as first document as before, the problem is to find out the B_i for which $P_{AB}(x)$ is maximum. In other terms we need to maximize $P_B(x)+I_{AB}(x)$. If the ranking of the first document A is fixed and if there are two more documents B and C then it will rank B before C if and only if $P_B(x)+I_{AB}(x) \geq PC(x)+I_{AC}(x)$. This will not only consider the relevance of the document while ranking but also the interference of the document with already ranked documents.

Formally the Quantum Probability Ranking Principle (QPRP) is stated as (Zuccon et al., 2009):

> in order to maximize the effectiveness of an IR system, document B should be ranked after the set $\mathfrak{A}$ of documents already ranked and before any other document C in the list returned to the user who submitted the query if and only if $P_B(x) + I_{\mathfrak{A}B} \geq p_C(x) + I_{\mathfrak{A}C}$, where $I_{\mathfrak{A}Y}$ is the sum of all the interference terms associated to each pair of documents Y and $X \in A$.

In practical terms interference depends on the novelty and diversity of information contained in the document with respect to already ranked documents.

In physics, the interference term depends on the phase shift θ between two slits. In information retrieval context, phase shift θ between two documents can be calculated as:

$$\theta_{AB} \approx \arccos(sim(A, B)) + \pi \tag{6.1}$$

where $sim(A, B)$ is the cosine similarity between the two documents. But more research has to be carried out in this field and estimation of θ is still an open question. Besides, empirical studies are yet to be carried out to verify the influence of the interference term in ranking.

References

C. D Manning, P. Raghavan, and H. Schütze. *An introduction to information retrieval*. Cambridge University Press, 2009.

C. J. Van Rijsbergen. *Information retrieval*. Butterworths, 1979.

Ricardo Baeza-Yates and Berthier Ribeiro-Neto. *Modern Information Retrieval*. ACM Press, 1999.

Cristopher D. Manning. Introduction to information retrieval - cs 276 lecture slides. In *Introduction to Information Retrieval*. Stanford University, 2009. `http://nlp.stanford.edu/IR-book/newslides.html`.

Pushpak Bhattacharyya. Artificial intelligence - cs 621 lecture slides. IIT Bombay, 2008. `http://www.cse.iitb.ac.in/~pb/cs621`.

Kai Gao, Yongcheng Wang, and Zhiqi Wang. An efficient relevant evaluation model in information retrieval and its application. In *International Conference on Computer and Information Technology*, volume 0, pages 845–850, Los Alamitos, CA, USA, 2004. doi: http://doi.ieeecomputersociety.org/10.1109/CIT.2004.1357300.

Vishal Vachhani. Cross-language information access. *MTP Stage 1 Report,Indian Institute of Technology,Bombay*, 2009.

S.E. Robertson. The Probability Ranking Principle in IR. *Journal of Documentation*, 33(4):294 – 304, 1977. doi: 10.1108/eb026647. `http://www.emeraldinsight.com/10.1108/eb026647`.

K. Sparck Jones, S. Walker, and S. E. Robertson. A probabilistic model of information retrieval: development and comparative experiments: Part 2. *Information Processing & Management*, 36(6):809–840, November 2000. doi: 10.1016/S0306-4573(00)00016-9. `http://www.sciencedirect.com/science/article/B6VC8-40V4CH5-2/2/99b7ca71c12a8845a56ae46c78497f80`.

Stephen Robertson (Microsoft Research Cambridge) and Hugo Zaragoza (Yahoo! Research Barcelona). The probabilistic relevance method: Bm25 and beyond - slides from sigir 2007. 2007. `http://www.zaragozas.info/hugo/academic/pdf/tutorial_sigir07_2d.pdf`.

A. Singhal. Modern information retrieval: A brief overview. *IEEE Data Engineering Bulletin*, 24(4):3543, 2001.

S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. *Computer networks and ISDN systems*, 30(1-7):107–117, 1998.

L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. *Technical report,Stanford Digital Library Technologies Project*, 1998.

J.M. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM (JACM)*, 46(5):604–632, 1999.

Serge Abiteboul, Mihai Preda, and Gregory Cobena. Adaptive on-line page importance computation. In *Proceedings of the 12th international conference on World Wide Web*, pages 280–290, Budapest, Hungary, 2003. ACM. doi: 10.1145/775152.775192. `http://portal.acm.org/citation.cfm?id=775192`.

R. P Feynman. The concept of probability in quantum mechanics. In *Proc. II Berkeley Symp. Math. Stat. and Prob*, pages 533–541, 1951.

Guido Zuccon, Leif A. Azzopardi, and Keith Rijsbergen. The quantum probability ranking principle for information retrieval. pages 232–240, Cambridge, UK, 2009. Springer-Verlag. `http://portal.acm.org/citation.cfm?id=1612333.1612361`.