Extension of SMART with Divergence from Randomness (DFR) model

MTech Dissertation Report

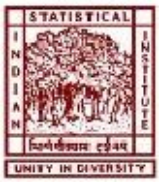
a dissertation submitted in partial fulfillment of the requirements for the M.Tech.(Computer Science) degree of the Indian Statistical Institute

by

Saurabh Bagchi
M.Tech-Computer Science
Roll No. - CS0812

under the supervision of

Dr. Mandar Mitra
of
CVPR Unit
ISI Kolkata



Indian Statistical Institute, Kolkata
July 2010

I certify that I have read this dissertation and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of Master of Technology in Computer Science.

Dr. Mandar Mitra (Principal Advisor)

I certify that I have read this dissertation and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of Master of Technology in Computer Science.

External Examiner

Abstract

The SMART information retrieval system is a sophisticated open-source text processing system based on the vector space model, developed over the last thirty five years. The SMART system automatically generates vectors for any given text collection and a set of queries and then uses the notion of vector similarity in computing the ranks of document vectors.[9]
The divergence from randomness (DFR) model of information retrieval was proposed in the year 2002 by Amati and Rijsbergen.[4] Amati reported his experimental findings in his PhD. thesis, which showed that his framework produces different nonparametric models forming baseline alternatives to the standard $tf - idf$ model. The DFR model is widely used as a benchmark model for testing the performance of other information retrieval models. Keeping in mind the need of the IR community for an open-source implementation of the DFR model, we decided to extend the SMART system by implementing the DFR modeled IR policy within its framework. This, we hope would enable IR researchers to do newer experiments on the DFR model, and possibly improve it.

Contents

List of Figures

List of Tables

Chapter 1

Introduction to IR

The meaning of the term information retrieval can be very broad. Just getting the credit card out of your wallet so that you can type in the card number is a form of information retrieval. Surveys show that about 85% of the users of the internet use popular interactive search engines to formulate queries, retrieve references of documents, inspect the documents(typically the top ranked ones) and possibly reformulate the queries, for buying goods, looking for vocation in companies, finding research papers and for so many other reasons. This is a good indication of the impact of search engines and information retrieval technology on ordinary people's life. If a technology is important enough, many people will adopt the discipline's technical vocabulary and new words eventually end up in language dictionaries. Infact people often say 'Google out the information' indicating the impact of Google on our everyday life.
Many modern information retrieval systems, like internet search engines, are specifically designed for users who are not familiar with the collection, the representation of the documents, and the use of Boolean operators. The main requirements for these systems are the following. Firstly, users should be able to enter any natural language word(s), phrase(s) or sentence(s) to the system, without the need to enter operators. This usually implies a full text information retrieval system, which is a system that potentially indexes every word in a document automatically. Secondly, the system should rank the retrieved documents by their estimated degree or probability of usefulness for the user. Thirdly, though not as important as the former two, the system should support automatic reformulation of the search statement from user feedback. However as an academic field of study information retrieval is defined as:

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

1.1 Basic Processes of Information Retrieval

There are three basic processes an information retrieval system has to support:

1. Representation of the content of the documents
2. Representation of the user's information need
3. comparison of the two representations.

The processes are visualised in figure (Croft 1993)[10]. In the figure 1.1, rectangles represent data and ovals represent processes.

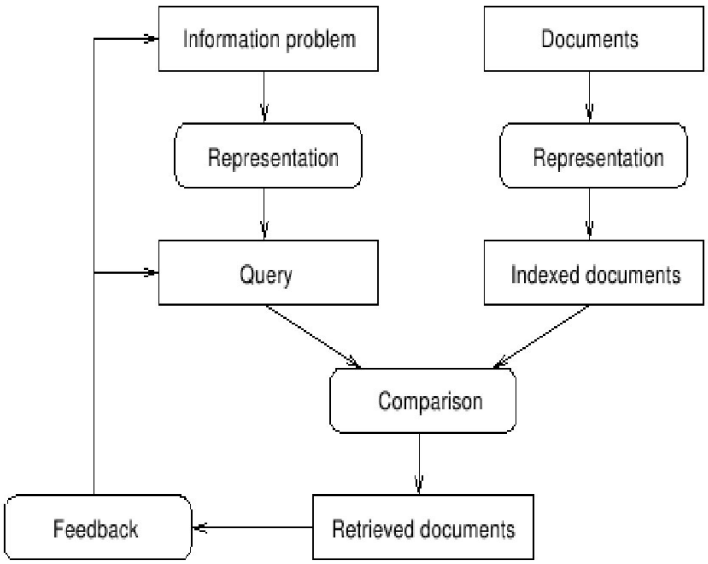


Figure 1.1: Information retrieval process

Representing the documents is usually called the indexing process. The process takes place off-line, that is, the end user of the information retrieval system is not directly involved. The indexing process results in a formal representation of the document: the index representation or document representation.The indexing process may include the actual storage of the document in the system, but often documents are only stored partly, for instance only title and abstract, plus information about the actual location of the document.
The process of representing the information problem or need is often referred to as the query formulation process. The resulting formal representation is the query. In a broad sense, query formulation might denote the complete interactive dialogue between system and user, leading not only to a suitable query but possibly also to a better understanding by the user of his information need. The retrieved set of documents can be further improved by a process known as relevance feedback, which iteratively tunes the parameters of the system, with reference to a set of documents, being judged as relevant by a group of experienced linguists. The aim of relevance feedback is to fetch more documents, which are relevant to a particular query, within the retrieved set.

1.2 Organization Of the Report

In the section to follow, we speak briefly about the tasks as performed by a standard IR engine followed by a section on evaluation methodologies and standard metrics for IR evaluation. These sections are very introductory and a reader aware of the IR terminologies can skip these sections and move onto chapter 2 (or directly to chapter 3 if he is aware of the vector space model). The rest of the report is organized as follows:

1. Chapter 2 introduces the vector space model of IR, the underlying model used in the SMART We also give a brief description of the SMART system. [9, 11, 12, 13].

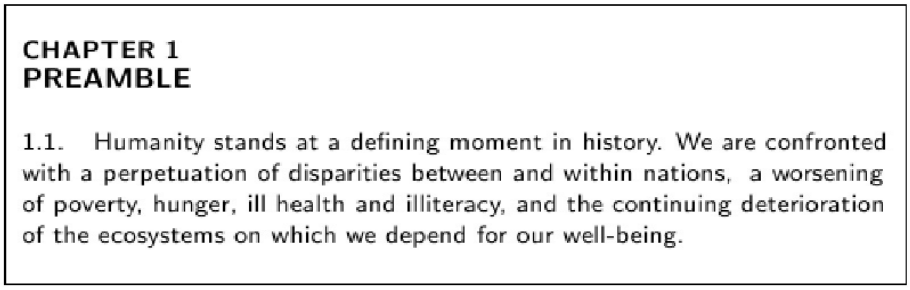


Figure 1.2: An example text - the opening lines of Agenda 21

2. Chapter 3 introduces the *DFR model* of IR with a mathematical description of the theory underneath this model and then follow it up with a discussion on the intuition for this model by which has been developed recently[4].

3. Chapter 4 reports the performance of our implemented DFR through experimental results followed by a suggestion on future experiments.

4. Chapter 5 summarization of our work and our future plans with extended SMART.

1.3 How an IR system works

With the emergence in the 1970's of models of ranked retrieval that process unstructured queries, automatic query systems became a fact. The main philosophy of automatic query systems is that indexing and query formulation should result in a representation that is closer to the actual meaning of the text, ignoring as many of the irregularities of natural language as possible. A typical approach to indexing and query formulation selects the query terms as follows. First a tokenisation process takes place, then stop words are removed, and finally the remaining words are stemmed. Additionally, natural language processing modules might provide the identification of phrases or splitting of compounds. Figure [1.2] shows an example text that will be used to illustrate the typical approach to query term selection.

1.3.1 Tokenization

As a first step in processing a document or a query, it has to be determined what the processing tokens are. One of the most simple approaches to tokenisation defines word symbols and inter-word symbols. In the example of figure [1.3] all characters that are non letters and non digits are considered to be inter-word symbols. The inter-word symbols are ignored during this phase, and the remaining sequences of word symbols are the processing tokens.

1.3.2 Stop word removal

Stop words are words with little meaning that are removed from the index and the query. Words might carry little meaning from a frequency (or information theoretic) point of

```
chapter 1 preamble 1 1 humanity stands at a defining moment in
history we are confronted with a perpetuation of disparities
between and within nations a worsening of poverty hunger ill
health and illiteracy and the continuing deterioration of the
ecosystems on which we depend for our well being
```

Figure 1.3: The Agenda 21 text after tokenization

```
chapter 1 preamble 1 1 humanity stands defining moment
history confronted perpetuation disparities nations
worsening poverty hunger ill health illiteracy continuing
deterioration ecosystems depend well being
```

Figure 1.4: The Agenda 21 text after stop word removal by Smart list

view, or alternatively from a linguistic point of view. Words that occur in many of the documents in the collection carry little meaning from a frequency point of view. If words carry little meaning from a linguistic point of view, they might be removed whether their frequency in the collection is high or low. In fact, they should especially be removed if their frequency is low, because these words affect document scores the most. Removing stop words for linguistic reasons can be done by using a stop list that enumerates all words with little meaning, like for instance "the", "it" and "a". Stop lists are used in many systems, but the lengths of the various stop lists may vary considerably. For instance, the Smart stop list contains 571 words[8], whereas the Okapi system uses a moderate stop list of about 220 words (Robertson and Walker)[14].

1.3.3 Stemming

A stemmer applies morphological 'rules of the thumb' to normalise words. The stemmers commonly used are those by Lovins[15] and Porter[12]. A stemmer can produce undesirable effects, for it may conflate two words with very different meanings to the same stem. For example 'operate', 'operating' and 'operations' are all stemmed to 'oper'. As a result, a query 'operating systems' can fetch documents related to 'operations research'. Figure 1.5 shows the result of Porter Stemmer on our sample text.

1.3.4 Phrase extraction

During indexing and automatic query formulation, multiple words may be treated as one processing token. The meaning of phrases might be quite different from what the two words independently suggest. A user who enters the query 'Stanford University' is less likely to be happy with a document which says 'Mr. Stanford never went to a university'. Maintaining the positional information of the terms is a generalized approach to deal with n-grams where a document is retrieved from the index if the positional information of the

```
chapter 1 preambl 1 1 human stand defin moment
histori confront perpetu dispar nation worsen
poverti hunger ill health illiteraci continu
deterior ecosystem depend well be
```

Figure 1.5: The Agenda 21 text after stemming by Porter's stemmer

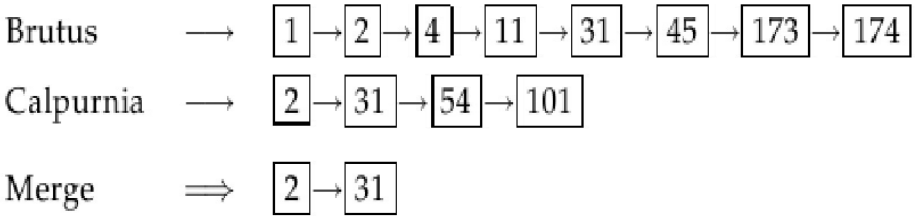


Figure 1.6: Merging two postings list

query terms conforms with itself. For example the query 'To be or not to be' is less likely to fetch Shakespeare's 'Hamlet' without positional information.

1.3.5 Index file structure

Within a document collection, we assume that each document has a unique number known as the document identifier(DocID). A weighted list of documents is constructed for every term in the collection, where the weight assigned to a document might be the number of occurences of that term in it. The terms, which act as keys to their corresponding lists are kept sorted and are typically kept in memory whereas the associated lists (commonly refered to as *postings* are kept sorted by the list members' weights and are typically stored on secondary storage. For each query term, their postings are merged to give the final set of documents. Figure [1.6] shows merging of two postings list.

1.4 Evaluation of IR systems

To measure ad hoc information retrieval effectiveness in the standard way, we need a test collection consisting of three things:

1. A test document collection.
2. A test suite of information needs, expressible as queries.
3. A set of relevance judgements, normally a binary assessment of either relevant or not relevant for each query-document pair.

The standard approach to information retrieval system evaluation revolves around the notion of relevant and not relevant documents. With respect to a user information need, a document is given a binary classification as either relevant or not relevant. To do a system evaluation, we require an overt expression of an information need, which can be used for judging returned documents as relevant or not relevant.

1.4.1 Evaluation of unranked retrieval sets

We define two standard metrics as follows:

1. Precision(P) is the fraction of retrieved documents that are relevant.

$$\frac{\#\ relevant\ documents\ retrieved}{\#\ retrieved\ documents}$$

2. Recall(R) is the fraction of relevant documents retrieved.

$$\frac{\#\ relevant\ documents\ retrieved}{\#\ relevant\ documents}$$

For binary valued relevance, retrieval performance is usually measured by the combination of *precision* and *recall*.

1.4.2 Evaluation of ranked retrieval sets

If the system ranks the documents in decreasing order of some document score, then the precision and recall measures should somehow be averaged over the number of documents retrieved. A number of fixed recall levels are chosen, for instance 10 levels {0.1, 0.2.....1.0}. The levels mimick users who are satisfied with 10%, 20%, ... 100% relevant documents. For each of these levels, the corresponding precision is determined by averaging the precision on that level over the topics.

Another common measure used is the *Mean Average Precision* (MAP), which provides a single-figure measure of quality across recall levels. It is the average of the precision value obtained for the top set of k documents existing after each relevant document is retrieved. That is, if the set of relevant documents for a query $q_j \in Q$ is $\{d_1, d_2, ...d_m\}$ and R_k is the set of ranked retrieval results from the top result upto d_k,then

$$MAP(Q) = \frac{1}{Q}\sum_{j=1}^{|Q|}\frac{1}{m}\sum_{k=1}^{m}P(R_k) \tag{1.1}$$

Our experimental findings reported in chapter 4 use both the above mentioned measures.

Chapter 2

Vector Space Model

The vector space model of information retrieval was developed by Gerard Salton and his students in the late 1960's and the early 1970's[13]. This model transforms any given text such as an article, a query, a portion of an article etc. into a vector in a very high-dimensional vector space. The main power of this model comes from its ability to measure the proximity between any two vectors, i.e., the 'closeness' between any two texts. In terms of information retrieval, when two vectors are close, then the corresponding texts are semantically related. The documents can then be ranked in decreasing order of their closeness to the query, yielding a semantic relatedness ranking, as desired in modern information retrieval systems.
Salton and his students also implemented a information retrieval system based on the vector space model, the SMART system[8]. SMART has had an enormous impact on IR research over the last thirty years. Many theories and techniques in the field of information retrieval, for example, automatic indexing and term weighting, evaluation of ranked systems, boolean models, relevance feedback, document clustering, use of a thesaurus etc. were either developed directly on the SMART system, or were first tested for their applicability to IR tasks on the SMART system. Research out of the SMART group at Cornell has contributed numerous important results, and has stimulated lots of new work in the field of information retrieval. Through SMART, the vector space model has had a tremendous influence on IR.

2.1 The model

Let us introduce the model with an example. Imagine a hypothetical three word world with only three words in its vocabulary *information*, *retrieval*, and *research* (probably an IR graduate student's world). If we assign an independent dimension to every word in the vocabulary (a total of three), assuming that the terms are mutually independent of each other, any utterance in this world can now be represented by a vector in this three dimensional space. Assuming that the number of occurrences of a word indicate the length of the sub-vector in the dimension corresponding to the word, Figure [2.1] shows the vectors for some texts in this three word world. We see that the utterance 'information research' is a vector with a zero '*retrieval* component and a unit component in the *information* and the *research* dimensions. Similarly, the utterance 'retrieval information information' is a vector with zero *research* component, a unit *retrieval* component, and a component of length two in the *information* dimension. In a real world, however, the vector space will have a very high dimensionality - equal to the size of the vocabulary of the text collection at hand. Since words that are not used in a text have a zero length sub-vector in the

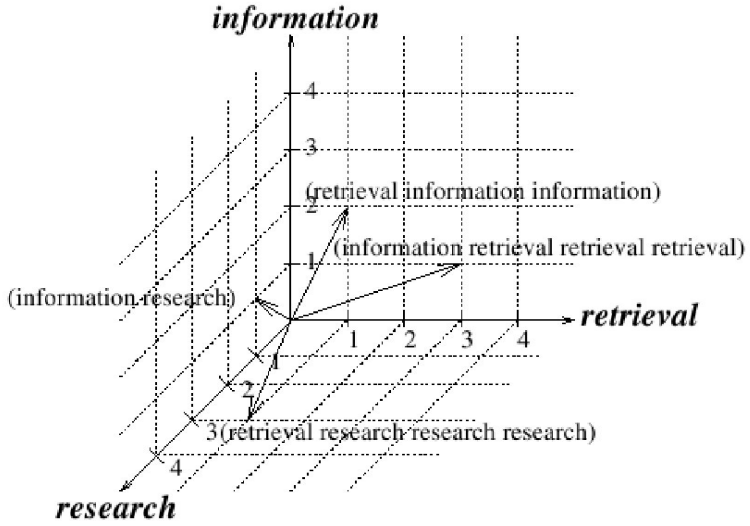


Figure 2.1: A three dimensional vector space

corresponding dimension, and any text uses a very small subset of the entire vocabulary most text vectors will be sparse, i.e., they will have many zero components.

The dot product of two vectors which is directly proportional to the cosine of the angle between the vectors, and hence inversely proportional to the angle value between the two and hence provides a suitable measure of similarity between two vectors. Given two text vectors $\vec{P} = (p_1, p_2, ..., p_n)$ and $\vec{Q} = (q_1, q_2, ..., q_n)$ in an 'n' dimensional vector space, their inner product is defined as

$$sim(\vec{P}, \vec{Q}) = \vec{P} \cdot \vec{Q} = \sum_{i=1}^{n} p_i \cdot q_i \tag{2.1}$$

Thus, the documents can be ranked by the above similarity measure which ensures that the documents in the top rank have a higher degree of vocabulary overlap with the user query and hence potentially more relevant to the users information need.

2.1.1 Term weights

The importance of a term increases with the number of occurrences of the term in a text. Therefore one can use some monotonically increasing function of the number of occurrences of a term in a text to estimate the term weight. The number of occurrences of a term in a text is called the *term frequency* of the term. The function of the term frequency used to compute a term's importance is called the *term frequency factor* or the *tf factor* in term weights. Some commonly used *tf* factors are:

1. **The raw *tf* factor** - This factor is simply the number of occurrences of a term in an article.

2. **The logarithmic *tf* factor** - This factor is computed as

$$1 + log(tf) \tag{2.2}$$

The justification of using this term weight is that the very high occurence of a single matching term in a document shoudn't outperform another document which has two matching query terms with somewhat lower term frequencies. Consider as an example the query 'recycling of tires', and two documents - D1 with ten occurrences of the word 'recycling', and D2 which uses both the words 'recycling' and 'tires' three times each. Everything else being equal, if raw *tf* s are used, D1 gets a higher similarity (proportional to 10) than D2 (which gets a similarity proportional to 3 + 3 = 6). But it is intuitive that D2 addresses the needs of the query much better than D1 because it addresses both recycling and tires, and not just recycling (like D1).

3. **The augmented term factor** - This factor reduces the range of the contributions from the term frequency of a term. This is done by compressing the range of the possible *tf* factor values (say between 0.5 and 1.0). The augmented *tf* factor is used with a belief that mere presence of a term in a text should have some default weight (say 0.5); additional occurrences of a term could increase the weight of the term to some maximum value (usually 1.0). Typically this factor is:

$$0.5 + 0.5 \cdot \frac{tf}{max\_tf} \tag{2.3}$$

2.1.2 Inverse Document Frequency

With reference to the discussion on stop words in section 1.4.2, we can say that such words occur with very high term frequencies across numerous articles and hence are not very informative. A match between a query and an article on words like 'put' does not mean much in terms of the semantic relationship between the query and the article. For this reason we also need to differentiate between the goodness of a matching term.
An inverse function of the number of articles a word appears in, has an intuitive appeal to be used to judge the importance of a word. The number of articles a word appears in is often called the *document frequency* of the word. An inverse function of the document frequency is used in term weights, and is called the *inverse document frequency* factor or the *idf* factor in term weights. The most commonly used inverse document frequency formulation is

$$log\left(\frac{N}{df}\right) \tag{2.4}$$

where N is the total number of articles in a collection and *df* is the document frequency of the word.

2.1.3 Document length normalization

$tf \cdot idf$ weights don't always suffice as good estimates to term importance in a text, because it ignores the document lengths often leading to prioritizing longer documents due to the possiblity of presence of higher term frequencies on a per term basis and also due to

Table 2.1: Term weights in SMART

Term frequency		Inverse document frequency		Document length normalization	
First letter	$f(tf)$	Second letter	$f\left(\frac{1}{df}\right)$	Third letter	$f(length)$
n(natural)	tf	n(no)	1	n(no)	1
l(logarithmic)	$1 + ln(tf)$	t(full)	$log(\frac{N}{df})$	c(cosine)	$\parallel D \parallel$
a(augmented)	$0.5 + 0.5 \cdot \frac{tf}{maxm.tf}$				

the mere presence of more terms. Hence as a compensation, the weights of the terms in longer documents are depressed giving the shorter documents a chance to compete with the longer ones. This technique is called 'document length normalization'. Following is a review of some commonly used length normalization schemes:

1. **Cosine normalization** - It is the most common and perhaps most intuitive normalization technique of the vector space model. The inner-product equation as defined in (2.1) can be modified to compute the actual cosine of the angle between the two vectors, which is independent of the length of the vectors.

$$cos(Q, D) = \frac{\vec{Q} \cdot \vec{D}}{||Q||||D||} \tag{2.5}$$

2. **Maximum** *tf* **Normalization** - Another popular normalization technique is normalization of individual *tf* weights for a document by the maximum *tf* in the document. The SMART system's augmented *tf* factor see eqn. (2.3) is an example of such normalization.

2.2 The SMART system

The SMART system is an open source IR engine[8]. The processing steps are the same as described in section 1.4. Most of the functionalities in general and term weighting in particular are configurable in SMART. The subsections to follow contain some digression materials on the SMART system from an engineering point of view. If the reader is already aware of the technical details, he can skip to the beginning of the next chapter.

2.2.1 Term weighting

Term weights are configured with three letters - the first is a short hand for the *tf*, the second denotes the *idf*, and the third corresponds to the document length normalization.

A retrieval experiment can now be characterized by a pair of triples *ddd.qqq* where the first triple corresponds to the term weighting used for the documents, and the second triple corresponds to the query term weights. For example, when *ltc.nnn* is used to symbolize a retrieval run, the document term weights used in the run are

$$\frac{(1 + \log(tf)) \cdot \log(N/df)}{\sqrt{\sum_{i=1}^{n} d_i^2}} \tag{2.6}$$

and the query weights are tf.

2.2.2 Code overview

Since the SMART is an open source system of about 1,50,000 lines of code, the person contributing to its developement is expected to follow the original design so as to ensure future maintainability. This demands that any new functionalities has to be plugged in proper places without violating the original design goals.

The directory structure of the *smart/src* is as follows:

- **libconvert:** Contains the conversion routines from one term weighting scheme to another.
- **libevaluate:** Evaluation measures are computed by these procedures.
- **libfeedback:** Various feedback strategies are implemented here.
- **libfile:** Contains low level routines for manipulating file structures.
- **libgeneral:** Contains some utility routines like reading from configuration files, logging, reading, writing and copying vector etc. These procedures are used almost everywhere throughout the rest of the code.
- **libindexing:** Contains indexing procedures for documents and queries. Strategies during the indexing steps to be taken are all implemented as separate procedures and are invoked through the configuration file. For example the default configuration file *spec.default* suggests that document vectors are to be stored as inverted lists and queries are to be stored in the vector format. The user can override this default behavior by configuring the parameters *doc.store* and *query.store*. Please refer to the 'libproc' section where we discuss the function calling mechanism in SMART.
- **liblocal:** This directory, structure-wise is an exact replica of the *src* folder, adopting the convention that any small changes necessary to be done in the any of the files (say a file in the libconvert folder), is done by copying the file in the corresponding folder inside liblocal (i.e. liblocal/libconvert in our example) and then making the necessary changes instead of changing the original file.
- **libprint:** Contains procedures to print the file structures as an aid for debugging.
- **libproc:** Each file in this folder defines a procedure hiearchy providing a top level action to take, and a choice of configurable sub-actions to be chosen from the implemented available options. If for example the user wants to index documents, he simply invokes the program as follows:

  ```
  smart  index.doc  spec.default
  ```

  The file *spec.default* is provided by the system and below we provide a small snippet of that file for quick reference relevant to our discussion.

  ```
  store          index.store.store_vec
  doc.store      index.store.store_aux
  index.doc      index.top.doc_coll
  ```

Since the top level action is specified as *index.doc*, the function invoked is *doc_coll* defined as a member of the array of function pointers named *top* which in turn is a member of an array of function pointers named *index*, in accordance with the configuration of the third line of *spec.default*.

- **libretrieve:** The functions in this procedure compute the similarity measures of a given query with the documents in the collection and forms a list of retrieved documents ordered by decreasing similarity measures.

Chapter 3

Divergence from Randomness Model

This is a probabilistic model of Information Retrieval based on measuring the divergence from randomness. The models are nonparametric models of IR. Term-weighting models are derived by measuring the divergence of the actual term distribution from that obtained under a random process. Among the random processes, the binomial distribution and Bose-Einstein statistics are studied. Two types of term frequency normalization are used for tuning term weights in the document query matching process. The first normalization assumes that documents have the same length and measures the information gain with the observed term once it has been accepted as a good descriptor of the observed document. The second normalization is related to document length and other statistics. We have incorporated five other normalizations based on the DFR model currently implemented in Terrier-3.0.[16]

3.1 Analytical Charaterization

The DFR model proposed by Amati and Rijsbergen is currently one of the most successful IR models [4]. It is based on the informative content provided by the occurrence of terms in a document, a quantity that is then corrected by the risk of accepting a term as a descriptor for the document (first normalization principle) and by normalizing the raw occurrences by the length of the document (second normalization principle). The weight of a term in a document is expressed as the function of two probabilities $Prob_1$ and $Prob_2$. The following terms appear frequently in subsequent discussion so for convenience they are described below.

Speciality words: Words belonging to technical vocabulary, which being informative, tend to appear more densely in some *elite* documents.

Non Speciality words: Words that usually are included in a stop list and are randomly distributed over the collection.

Elite Set: The documents containing the speciality words.

Non Elite Set: All the documents in the collection excepting the *elite* documents.

$Prob_1$: It is derived mainly from the work of Harter[17,18] on automatic keyword indexing. It is supposed that words that bring little information are randomly distributed *on the whole set of documents*. In other words it defines the notion of randomness in the

context of information retrieval.

$Prob_2$: It is the probability of occurrence of the term whithin a document with respect to its *elite* set and is related to the risk $1 - Prob_2$ of accepting a term as a good descriptor of the document when the document is compared with the *elite* set of the term.

$$w = (1 - Prob_2) \cdot (-\log_2 Prob_1) \tag{3.1}$$

or alternatively

$$w = Inf_1 \cdot Inf_2 \tag{3.2}$$

where

$$Inf_1 = -\log_2 Prob_1 \tag{3.3}$$

and

$$Inf_1 = (1 - \log_2 Prob_2) \tag{3.4}$$

3.2 Models for $Prob_1$

The intution behind Inf_1 is simple. If $Prob_1$ is low for a term t in a document d then the distribution of t in d deviates from its distribution in the collection and thus t is useful to describe the contents of d. In this case, Inf_1 will be high. On the contrary if $Prob_1$ is high then the distribution of t in d is same as that in the collection and thus does not provide much information about d. Inf_1 thus captures the importance of a term in a document through its deviation from an average behaviour estimated on the whole collection. Amati and Rijsbergen[4] consider five basic probability models for estimating $Prob_1$. A brief description of the model and the corresponding formulae for $Prob_1$ are given, in the following sections.

3.2.1 The Bernoulli Model of Randomness

Here the assumption is that the tokens of a nonspecialty word should distribute over the N documents according to the binomial law. The problem in estimating probabilities using this model is that the factorials are too large to calculate and hence need to be approximated. The first approximation of the Bernoulli process is the Poisson process; the second is obtained by means of the information theoretic divergence D. The first method gives the basic model P where the factorials are approximated using Stirling's

Table 3.1: Description of various terms involved in the formulae

N	The total number of documents in the collection
F	The total number of occurrences of observed term t in collection
n	The total number of documents containing term t in collection
tf	Number of occurrences of term t in given document
tfn	Normalized version of term frequency
avg_l	Average length of documents in collection
l	Length of a specific document
c	An experimental constant
n_e	Expected size of *elite* set if total F tokens are assumed

formula

$$Inf_1(tf) = tf \cdot \log_2 \frac{tf}{\lambda} + \left(\lambda + \frac{1}{12 \cdot tf} - tf\right) \cdot \log_2 e + 0.5 \cdot \log_2(2\pi \cdot tf) \tag{3.5}$$

The second method uses Renyi's approximation[19] to get the model D where

$$\Phi = \frac{tf}{F} \tag{3.6}$$

$$p = \frac{1}{N} \tag{3.7}$$

$$D(\Phi, p) = \Phi \cdot \log_2 \left(\frac{\Phi}{p}\right) + (1 - \Phi) \cdot \log_2 \left(\frac{1 - \Phi}{1 - p}\right) \tag{3.8}$$

$$Inf_1(tf) = F \cdot (D(\Phi, p)) + 0.5 \log_2(2\pi \cdot tf(1 - \Phi)) \tag{3.9}$$

3.2.2 The Bose-Einstein Model of Randomness

In this model it is supposed that the F tokens of a word are randomly placed in N documents. Once the random allocation of tokens to documents is completed, this event is completely described by its occupancy numbers: $tf_1, \cdots, tf_N$ where tf_k stands for the term frequency of the word in the kth document. So a solution of the occupancy problem is found under the assumption that $N \gg tf$ and taking $\lambda = \frac{F}{N}$ to be the mean frequency of the term t in the collection D. Then the probability that a term occurs tf times in a document is

$$Prob_1(tf) = \left(\frac{1}{1 + \lambda}\right) \cdot \left(\frac{\lambda}{1 + \lambda}\right)^{tf} \tag{3.10}$$

The right side of the above equation is known as the geometric distribution with probability $p = \frac{1}{1+\lambda}$. This model of randomness is called G and the corresponding equation is

$$Inf_1(tf) = -\log_2\left(\frac{1}{1+\lambda}\right) - tf \cdot \log_2\left(\frac{\lambda}{1+\lambda}\right) \tag{3.11}$$

The second operational model associated with the Bose-Einstein statistics is constructed by approximating the factorials by Stirling's formula. Thus we have

$$Inf_1(tf) = -\log_2(N-1) - \log_2(e) + f(N+F-1, N+F-tf-2) - f(F, F-tf) \tag{3.12}$$

where

$$f(n,m) = (m+0.5) \cdot \log_2(\frac{n}{m}) + (n-m) \cdot \log_2 n \tag{3.13}$$

3.2.3 The tf-idf and tf-itf Model of Randomness

The probability $Prob_1(tf)$ is obtained by first computing the unknown probability p of choosing a document at random and then computing the probability of having tf occurrences of that term in that document.

We suppose that any token of the term is independent of all other tokens both of the same and different type, namely, the probability that a given document contains tf tokens of the given term is

$$Prob_1(tf) = \left(\frac{n+0.5}{N+1}\right)^{tf} \tag{3.14}$$

Hence we obtain the basic model $I(n)$:

$$Inf_1(tf) = tf \cdot \log_2 \frac{N+1}{n+0.5} \tag{3.15}$$

A different computation can be obtained from Bernoulli's law. Let n_e be the expected number of documents containing the term under the assumption that there are F tokens in the collection. Then

$$n_e = N \cdot (Prob(tf \neq 0)) = N \cdot \left(1 - \left(\frac{N-1}{N}\right)^F\right) \tag{3.16}$$

The third basic model is the tf-Expected_idf model $I(n_e)$:

$$Inf_1(tf) = tf \cdot \log_2 \frac{N+1}{n_e+0.5} \tag{3.17}$$

Again from the term independence assumption, we obtain with a smoothing of the probability, the $tf - itf$ basic model $I(F)$

$$Inf_1(tf) = tf \cdot \log_2 \frac{N+1}{F+0.5} \tag{3.18}$$

3.3 Models for $Prob_2$

Inf_1 thus captures the importance of a term in a document through its deviation from average behaviour estimated on the whole collection. The question is that why do we need to normalize it. The answer lies in the fact that there are certain principles or heuristics that seem to cause good retrieval performance. Such heuristics are discussed extensively by Fang [5]. However for the purpose of DFR we need to know four heuristics that are given below.

We consider here matching functions (denoted RSV), of the form:

$$RSV(q,d) = \sum_{w \in q \cap d} h(tf, l(d), z_w, \theta) \tag{3.19}$$

where θ is a set of parameters and h depends on the form of the IR model considered; z_w can take the value n or F normalized by N . In particular the following four principles should be necessarily satisfied:

Principle 1: Documents with more occurrences of query terms should get higher scores than documents with less occurrences.

Principle 2: However the increase in the retrieval score should be smaller for larger term frequencies, in as much as the difference between say 110 and 111 is not as important as the increase from 1 to 2.

Principle 3: Longer documents when compared to shorter ones with exactly the same number of occurrences of the query term should be penalized as they are likely to cover additional topics beyond the ones that are present in the query.

Principle 4: It is important when evaluating the retrieval score of a document, to down-weight terms occurring in many documents i.e which have a high document/collection frequency as these terms have a lower discriminating power.

The above four principles can be mathematically stated as

$$\forall (l(d), z_w, \theta), \frac{\partial h(tf, l(d), z_w, \theta)}{\partial tf} > 0 \tag{3.20}$$

$$\forall (l(d), z_w, \theta), \frac{\partial^2 h(tf, l(d), z_w, \theta)}{\partial^2 tf} < 0 \tag{3.21}$$

$$\forall(tf, z_w, \theta), \frac{\partial h(tf, l(d), z_w, \theta)}{\partial l(d)} < 0 \tag{3.22}$$

$$\forall(tf, l(d), \theta), \frac{\partial h(tf, l(d), z_w, \theta)}{\partial z_w} < 0 \tag{3.23}$$

Here equation (3.20), (3.21) and (3.23) directly state that h should be increasing with term frequency and decreasing with the document length and document collection frequency. Equations (3.20) and (3.21) state that h should be an increasing concave function of term frequency, the concavity ensuring that the increase in retrieval score will be smaller for large term frequency. If we base a retrieval function solely on the above formulation of Inf_1 only then we can see that model $I(n)$, $I(F)$ and $I(n_e)$ verify conditions (3.20), (3.22) and (3.23) and the model for geometric distribution verifies condition (3.20) and (3.22) but partially (3.23) as the derivative may be positive for some values. All the models fail to satisfy (3.21) however as for each of them

$$\forall(l(d), z_w, \theta), \frac{\partial^2 h(tf, l(d), z_w, \theta)}{\partial^2 tf} = 0 \tag{3.24}$$

Hence Inf_1 alone for the geometric distribution and the models $I(n)$, $I(F)$ and $I(n_e)$ is not able to give a valid IR model which satisfies the above four principles. The role of Inf_2 is thus to make the IR model a valid one by satisfying equation (3.21) Two models have been proposed for Inf_2. These are discussed in the following sections.

3.3.1 The Normalization L

The first model is given by Laplace's law of succession. The law of succession in this context is used when we have no advance knowledge of how many tokens of a term should occur in a relevant document of arbitrarily large size. For understanding this model we need to understand the phenomenon of *aftereffect in the elite set*. A short description is given below.

Aftereffect in elite set: If we are searching for tokens of a term and after a long unsuccessful search we find a few of them in a portion of a document. It is quite likely that we have finally reached a document in which we expect increased success in our search. The more we find, the higher is the expectation. This expectation is given by $Prob_2(tf)$ in formula (3.1), and has been called by statisticians an apparent aftereffect of future sampling. [20]

In other words aftereffect is satisfying formula (3.21). The probability $Prob_2$ modeling the aftereffect in the elite set in formula (3.1) is given by the conditional probability of having one more token of the term in the document (i.e., passing from tf observed occurrences to $tf + 1$) assuming that the length of a relevant document is very large.

$$Prob_2(tf) = \frac{tf + 1}{tf + 2} \tag{3.25}$$

Similarly, if $tf \geq 1$ then $Prob_2(tf)$ can be given by the conditional probability of having tf occurrences assuming that $tf - 1$ have been observed. Equation (3.23) with $tf1$ instead of tf leads to the following equation,

$$Prob_2(tf) = \frac{tf}{tf+2} \tag{3.26}$$

So the corresponding value of Inf_2 is

$$Inf_2(tf) = 1 - Prob_2(tf) = \frac{1}{tf+1} \tag{3.27}$$

3.3.2 The Normalization B

The disadvantage of the Bernoulli model is that it lacks memory i.e. previous successes and failures do not influence successive outcomes. So for estimating $Prob_2$ with Bernoulli trial we use the following urn model. We add a new token of the term to the collection, thus having $F+1$ tokens instead of F. We then compute the probability $B(n, F+1, tf+1)$ that this new token falls into the observed document, thus having a within document term frequency $tf+1$ instead tf. Similarly we calculate $B(n, F, tf)$. Therefore we can calculate the incremental rate α of term occurrence in the elite set which is given by

$$\alpha = \frac{B(n, F, tf) - B(n, F+1, tf+1)}{B(n, F, tf)} = 1 - \frac{B(n, F+1, tf+1)}{B(n, F, tf)} \tag{3.28}$$

On simplifying we get

$$\alpha = 1 - \frac{F+1}{n \cdot (tf+1)} \tag{3.29}$$

So the corresponding value for Inf_2 is

$$Inf_2(tf) = 1 - \alpha = \frac{F+1}{n \cdot (tf+1)} \tag{3.30}$$

3.4 Length Normalization

The next concern is to introduce an additional methodology that can normalize the random variables tf by the length of the document. In other words, we would like to obtain the expected number of tokens of a term in a document and in the collection if the lengths of the documents in the collection were equal to a fixed value, for example, to their average length.

Here a density function $\rho(l)$ of the term frequency is defined and then for each document d of length $l(d)$ the term frequency is computed on the same interval $[l(d), l(d) + \Delta l]$ of given length l as a normalized term frequency. l can be chosen as either the median or the mean avg_l of the distribution. Experiments show that normalization with $l = avg\_l$

is the most appropriate choice.

Hypothesis 1: The distribution of a term is uniform in the document. The term frequency density $\rho(l)$ is a constant ρ. So the normalized term frequency *tfn* is given by

$$tfn = tf \cdot \frac{avg\_l}{l(d)} \tag{3.31}$$

Hypothesis 2: The term frequency density $\rho(l)$ is a decreasing function of the length l. So the normalized term frequency *tfn* is given by

$$tfn = tf \cdot \log_2 \left(1 + \frac{avg\_l}{l(d)}\right) \tag{3.32}$$

Chapter 4

Experimental Results

4.1 New First Normalization Models

These models are implemented in the DFR model of another IR system Terrier 3.0. So for comparison purpose they have also been implemented in SMART. A brief description of these normalization techniques follows:

4.1.1 Log of Laplace law of succession

This is the same as the Laplace model except that negative logarithm of the corresponding Inf_1 value is taken. The formula is as follows:

$$tfn = \log_2 \frac{1 + tf}{tf} \tag{4.1}$$

4.1.2 Laplace law of succession with prior

This is an extension of the Laplace model where the prior information is used as a function of term frequency and average length. The formula is as follows:

$$prior = \frac{tf}{c \cdot avg\_l} \tag{4.2}$$

$$tfn = \frac{(1 + prior)^2}{1 + tf} \tag{4.3}$$

4.2 New Length Normalization Models

Amati and Rijsbergen have suggested that along with the two assumptions in Hypothesis 1 and 2 other choices are equally possible and that they think this is a crucial research issue which should be extensively studied and explored. So we have implemented some of the length normalization techniques that are implemented in the DFR model of another IR system TERRIER 3.0. A brief description of these normalization techniques follows:

4.2.1 No Normalization

This implements an empty normalisation. It does no frequency normalisation but returns the original raw term frequency.

$$tfn = tf \tag{4.4}$$

4.2.2 BM25 Normalization

It is also known as Okapi BM25 normalization and is based on the probabilistic retrieval framework developed in the 1970s and 1980s by Stephen E. Robertson, Karen Sparck Jones, and others [21]. The formula is as follows:

$$tfn = \frac{tf}{1 - c + c \cdot avg\_l/l(d)} \tag{4.5}$$

4.2.3 Increasing density function

As in Hypothesis 2, the DFR model assumed $\rho(l)$ to be a decreasing function of the length. Here we assume it as an increasing function of l. The formula is as follows:

$$tfn = c \cdot tf \cdot \frac{avg\_l}{l(d)} \tag{4.6}$$

4.2.4 Pareto Distribution Normalization

The Pareto distribution, named after the Italian economist Vilfredo Pareto, is a power law probability distribution that coincides with social, scientific, geophysical, actuarial, and many other types of observable phenomena. Outside the field of economics, it is at times referred to as the Bradford distribution. The formula is as follows:

$$tfn = tf \cdot \left(\frac{avg\_l}{l(d)}\right)^c \tag{4.7}$$

4.2.5 Jelinek - Mercer smoothing

It is a smoothing technique which is primarily used in language models and is considered as an interpolation between the unigram and the bigram model. Since DFR is also considered as an extension of the language model approach so it is used here. The formula is as follows:

$$tfn = \left((1 - c) \cdot \frac{tf}{l(d)} + c \cdot \frac{n}{N}\right) \cdot l(d) \tag{4.8}$$

4.2.6 Normalization with natural logarithm

This is same as the normalization of Hypothesis 2 in DFR model except that here the logarithm is to the base e and there it was to the base 2. The formula is as follows:

$$tfn = tf \cdot \log\left(1 + c \cdot \frac{avg\_l}{l(d)}\right) \tag{4.9}$$

4.3 Naming convention followed in SMART

Here we give a table 4.1 of the naming convention that is used in SMART and the corresponding model in Amati's paper alongwith that of TERRIER. The nil fields indicate that the model is absent in the current implementation.

Table 4.1: Naming Convention Table

Type	DFR	SMART code	TERRIER	Model Description
Inf_1	$I(F)$	f	IF	Approximation of $I(n_e)$
	$I(n)$	n	In	Inverse document frequency
	$I(n_e)$	e	In_exp	Mixture of Poisson and inverse document frequency
	G	g	-	Geometric as limiting form of Bose Einstein
	B_E	b	-	Limiting form of Bose-Einstein
	P	p	P	Poisson approximation of the binomial model
	D	d	D	Approximation of the binomial model with divergence
Inf_2	L	L	L	Laplace law of succession
	-	M	LL	Log of Laplace law of succession
	-	P	L5	Laplace law of succession with prior
	B	B	B	Ratio of two Bernoulli processes
tfn	-	0	0	Raw term frequency
	$H1$	1	1	Uniform distribution of the term frequency
	$H2$	2	2	The term frequency density is decreasing function
	-	-	3	Drichlet's Prior
	-	3	B	BM25 normalization
	-	4	F	Increasing density function
	-	-	J	Jelinek Mercer smoothing
	-	5	P	Pareto Distribution
	-	6	JN	Jelinek Mercer smoothing 2
	-	7	C	Same as 2 but with natural log

4.4 Results

The experiments models such as $I(F)B2$, $I(n)L2$, $I(n_e)B2$, $I(n_e)L2$ showed good results as reported in Amati's implementation of the DFR model. So these models were checked against the TREC-6 collection. It consisted of 556,000 documents, from the Congressional Record, Federal Register, Financial Times, Foreign Broadcast Information Service, and

Table 4.2: Disks 4 and 5 of TREC 6,Topics $301 - 350$ Rel. Doc: 4290

SMART						
Model	AvegPr	Pr5	Pr10	Pr30	Pr100	Rel Ret
$I(F)B2$	0.2458	0.5080	0.4220	0.3213	0.1990	2476
$I(n)L2$	0.2440	0.5080	0.4400	0.3120	0.1974	2394
$I(n_e)B2$	**0.2488**	**0.5200**	**0.4420**	**0.3273**	**0.2010**	**2497**
$I(n_e)L2$	0.2446	0.4840	0.4300	0.2980	0.1896	2325
Amati						
Model	AvegPr	Pr5	Pr10	Pr30	Pr100	Rel Ret
$I(F)B2$	-	-	-	-	-	-
$I(n)L2$	0.2705	0.5560	**0.4840**	0.3267	0.2088	2510
$I(n_e)B2$	0.2622	**0.5680**	0.4680	**0.3373**	**0.2100**	**2566**
$I(n_e)L2$	**0.2751**	0.5440	0.4620	0.3213	0.2044	2493

Table 4.3: Disks 4 and 5 of TREC 7, Topics $351 - 400$ Rel. Doc: 4674

SMART						
Model	AvegPr	Pr5	Pr10	Pr30	Pr100	Rel Ret
$I(F)B2$	0.2458	0.5720	0.5020	**0.3733**	0.2314	**2818**
$I(n)L2$	0.2312	0.5440	0.4880	0.3513	0.2216	2700
$I(n_e)B2$	**0.2486**	**0.5800**	**0.5100**	0.3680	**0.2316**	2799
$I(n_e)L2$	0.2287	0.5240	0.4740	0.3507	0.2134	2656
Amati						
Model	AvegPr	Pr5	Pr10	Pr30	Pr100	Rel Ret
$I(F)B2$	**0.2484**	**0.5800**	**0.5200**	**0.3813**	0.2374	2833
$I(n)L2$	0.2360	0.5400	0.4960	0.3687	0.2278	2845
$I(n_e)B2$	0.2482	**0.5800**	0.5100	**0.3813**	**0.2386**	**2881**
$I(n_e)L2$	0.2320	0.5400	0.4980	0.3613	0.2174	2810

LA Times collections. Differently from TREC-6, in TREC-7 and TREC-8, the collection CR (about 28,000 transcripts from the Congressional Record) was not indexed.

Each of the 50 topics consists of three fields: a title (from one to three words), a description (one or two sentences), and a narrative (a paragraph listing specific criteria for accepting or rejecting a document). In our experiments we used all these three fields. The value of the constant c is taken as 1.

Table 4.4: Disks 4 and 5 of TREC 8, Topics 401 − 450 Rel. Doc: 4728

SMART						
Model	AvegPr	Pr5	Pr10	Pr30	Pr100	Rel Ret
$I(F)B2$	0.2747	0.5560	**0.4980**	0.3720	0.2300	2959
$I(n)L2$	0.2640	0.5440	**0.4980**	0.3653	0.2262	2861
$I(n_e)B2$	**0.2764**	**0.5600**	0.4940	**0.3753**	**0.2310**	**2967**
$I(n_e)L2$	0.2644	0.5320	0.4880	0.3507	0.2246	2880
Amati						
Model	AvegPr	Pr5	Pr10	Pr30	Pr100	Rel Ret
$I(F)B2$	0.2833	**0.5520**	0.5060	**0.3967**	0.2528	**3189**
$I(n)L2$	0.2792	0.5360	0.5040	0.3927	0.2492	3073
$I(n_e)B2$	**0.2841**	**0.5520**	**0.5080**	**0.3967**	**0.2532**	3178
$I(n_e)L2$	0.2769	0.5200	0.4940	0.3887	0.2452	3067

Table 4.5: Comparison of Average Precision Values

TREC 6			
Model	SMART	TERRIER	Amati
$I(F)B2$	0.2458	0.2338	-
$I(n_e)C2$	0.2451	0.2293	-
$I(n)L2$	0.2312	**0.2357**	0.2705
$I(n_e)B2$	**0.2486**	0.2342	0.2662
$I(n_e)L2$	0.2287	-	**0.2751**
TREC 7			
Model	SMART	TERRIER	Amati
$I(F)B2$	0.2458	**0.2465**	**0.2484**
$I(n_e)C2$	0.2447	0.2421	-
$I(n)L2$	0.2312	0.2306	0.2360
$I(n_e)B2$	**0.2486**	0.2457	0.2482
$I(n_e)L2$	0.2287	-	0.2320
TREC 8			
Model	SMART	TERRIER	Amati
$I(F)B2$	0.2747	0.2753	0.2833
$I(n_e)C2$	0.2672	0.2654	-
$I(n)L2$	0.2640	0.2645	0.2792
$I(n_e)B2$	**0.2764**	**0.2760**	**0.2841**
$I(n_e)L2$	0.2644	-	0.2769

Some observations from the results obtained in SMART are as follows:

- We could not match performance reported by Amati and Rijsbergen[4] in many cases. This is possibly owing to the other components (besides term-weighing) involved in a full IR system.
- We therefore compared against TERRIER which may be taken as an authoritative implementation. Our implementation performs at par with TERRIER as can be seen in the Table 4.5 with the default parameter settings.
- We have inferred that model $I(n_e)B2$ performs consistently better than other models in our implementation.

Chapter 5

Future Work

We discussed various aspects for the divergence from randomness model and its present implementation in Smart system. The future work will involve the following:

1. Different Query Expansion Models need to be tried out and the performance evaluated.

2. The two length normalization procedures namely Drichlet's Prior and Jelinek Mercer Smoothing for language modelling needs to be implemented.

3. The DFR system needs to be validated under the TREC 10 data to compare retrieval with and without query expansion.

Acknowledgment

Words are not enough to express my gratitude to my supervisor Dr. Mandar Mitra. He has truly been an inspiration to me. My motivation to work under him can be traced back a year, when I used to remain spellbound during his lecture sessions on 'Operating Systems' and 'Compilers' wondering how he was able to illustrate quite involved areas of computer science in apparently simple terms. His enthusiasm and passion about 'Information Retrieval' spurred me to do research in this field of study. Without his help in particular, it woudn't have been possible to come into terms with the huge codebase of the SMART IR engine, let aside extending it. Also his patience and support at all times made this dissertation implementable.
I would also like to thank Dr. Debashis Ganguly whose LaTeX files and dissertation report helped me prepare the initial part of my report.
I would also take the opportunity to thank Dr. Prasenjit Majumdar for answering a lot of my amateurish questions on IR and research fellow Mr. Sukomal Pal for helping me out with Smart installation and newbie questions regarding smart at all times. I would also like to thank research fellow Ms. Dipasree Pal, Mr. Jiaul Paik and Mr. Kripa Bondhu Ghosh for helping me understand terrier, its running and other details. It would be no less than a sin, of not thanking the project linked personnels , Ayan Bandopadhyay, Samaresh Maiti, Sukanya Mitra and Aparajita Sen for providing such a nice and friendly atmosphere in the lab and lending their helping hands whenever needed.

Saurabh Bagchi
July 2010
Indian Statistical Institute, Kolkata

References

[1] Christopher D. Manning, Prabhakar Raghavan, Hinrich Schutze "Introduction to Information Retrieval", Cambridge University, New York, USA, 2008

[2] Hans Paijmans's Smart Tutorial For beginners Site, "http://www.tcnj.edu/ mm-martin/CSC485IMME321/Papers/SMART/SmartCourse.html"

[3] Brian W. Kernighan, Dennis M. Ritchie, "The C - Programming Language, (ANSI C Version)", Prentice-Hall of India Pvt. Ltd., New Delhi, 1998

[4] G. Amati and C.J.V. Rijsbergen. Probabilistic Models of Information Retrieval Based on Measuring the Divergence from Randomness. ACM Transactions on Information Systems, 20(4):357-389, 2002

[5] ChengXiang Zhai, Tao Tao and Hui Fang.A Formal Study of Information Retrieval Heuristics In:SIGIR 2004

[6] Leif Azzopardi, Gabriella Kazai and Stephen Robertson. Advances in Information Retrieval Theory In:ICTIR September 2009

[7] Stephane Clinchant and Eric Gaussier. Bridging Language Modeling and Divergence from Randomness Models: A Log-Logistic Model for IR In: ICTIR September 2009

[8] Smart Code "ftp://ftp.cs.cornell.edu/pub/smart"

[9] Gerard Salton editor, The SMART retrieval system - Experiments in Automatic Document Retrieval,Prentice-Hall, Englewood Cliffs, NJ, 1971.

[10] Croft, Knowledge based and statistical approaches to text retrieval, IEEE Expert 8(2), 8-12, 1993.

[11] Djoerd Hiemstra, Using Language Models for Information Retrieval, Ph.D. thesis, Center of Telematics and Information Technology, AE Enschede The Netherlands, 2000.

[12] M.F. Porter, An algorithm for suffix stripping, Program 14, pp. 130-137, 1980.

[13] C.S. Yang Gerard Salton, A. Wong, A vector space model for Information Retrieval, .

[14] S.E. Robertson and S.Walker, Okapi, Keenbow at TREC-8, 2000, Eigth Text Retrieval Conference (TREC-8).

[15] Lovins, Development of a stemming algorithm, Mechanical translations and Computational Linguis- tics, vol. 11, pp. 22-31, 1993.

[16] Terrier 3.0 and 2.0 code "http://terrier.org/download/"

[17] S.P. Harter. A probabilistic approach to automatic keyword indexing. Part I: On the distribution of specialty words in a technical literature. In: J. ASIS 26, 197-216.

[18] S.P. Harter. A probabilistic approach to automatic keyword indexing. Part II: An algorithm for probabilistic indexing. In: J. ASIS 26, 280-289.

[19] A.Renyi Foundations of Probability. Holden-Day, San Francisco. 1969

[20] W. Feller "An Introduction to Probability Theory and Its Applications", Vol. I, third ed. Wiley, New York. 1968

[21] Karen Sparck Jones, Steve Walker, and Stephen E. Robertson. A Probabilistic Model of Information Retrieval: Development and Comparative Experiments (parts 1 and 2). Information Processing and Management, 36(6):779-840. 2000.