TERM WEIGHTING REVISITED

A Dissertation
Presented to the Faculty of the Graduate School
of Cornell University
in Partial Fulfillment of the Requirements for the Degree of
Doctor of Philosophy

by
Amitabh Kumar Singhal
January 1997

TERM WEIGHTING REVISITED

Amitabh Kumar Singhal, Ph.D.
Cornell University 1997

Term weighting is an essential part of the modern information retrieval systems. Out of the three main components of a term weighting strategy — term frequency, inverse document frequency, and document length normalization — the term frequency factor has been investigated recently by researchers. In this work, we study the inverse document frequency, and document length normalization components of term weights.

We observe that a document length normalization scheme that retrieves documents of all lengths with similar chances as their likelihood of relevance will outperform another scheme which retrieves documents with chances very different from their likelihood of relevance. We present *pivoted normalization*, a technique that can be used to modify normalization functions to reduce the gap between the relevance and the retrieval probabilities. We present two new normalization functions — *pivoted unique normalization* and *pivoted byte size normalization*, both of which yield significant improvements over the previous state of the art normalization functions.

When optical character recognition is used to create large information bases, term weighting schemes can be highly sensitive to the errors in the input text, introduced by the OCR process. This work examines the effects of the well known *cosine normalization* method in the presence of OCR errors, and proposes a new, more robust, normalization method. Experiments show that the new scheme is less sensitive to OCR errors and facilitates the use of more diverse basic weighting schemes. This study also explains why the use of cosine normalization in presence of the inverse document frequency factor is not advisable in large document collections.

When a user types a natural language query for an IR system, certain keywords in the query are more pertinent to the user's information need than others. Most modern IR systems incorporate these distinctions by using an inverse document frequency (*idf*) factor in term weighting. Preliminary experiments show that the usefulness of an *idf* type function is high at low ranks. We observe that the main reason for this effect is the widened gap between the weights of the rare terms and the non-rare query terms. The standard *idf* function works very well across query sets. Experiments show that there is room for improvement in the *idf* function. Further studies are needed to discover a better replacement for the standard *idf* function.

Biographical Sketch

Amit Singhal was born in Jhansi, a small town in the state of Uttar Pradesh (Hindi, Amit's native language, for Northern Province), India. After spending much of his boyhood in the foothills of the mighty Himalayas, Amit joined University of Roorkee to pursue a bachelor's degree in Computer Science. After Roorkee, Amit decided to quit school forever and took up a job as a database programmer in a software consultancy company, just to realize that school life was much better than real life. He quit work and joined the master's program in Computer Science in University of Minnesota, Duluth, Minnesota. After graduating from Duluth, once again deciding to quit school forever, Amit started working for West Publishing Company as a Systems Analyst. By now Amit had developed a habit of being in school and didn't last too long in a real job. He quit West and joined Cornell's Computer Science department to pursue a Ph.D. in Information Retrieval, a field he had developed a strong liking for during his M.S. degree. Amit has started a new experiment with a real job at AT&T Labs. Good thing, he can't get another degree in Computer Science ...

To Professor Gerard Salton ...

Acknowledgements

Nature was very cruel to take away Professor Salton, my mentor, before I could finish this work. Professor Salton was my guiding light. His dedication for the field of Information Retrieval was a true inspiration to me. Professor Salton had his own ways of mentoring. Realizing that I was a typical graduate student, out to solve all problems in all fields with my miniscule knowledge, in a casual conversation about a young researcher (who had switched fields lately), Professor Salton said, "When you are young, you should stick to a field and do your best. It doesn't help you much to jump from field to field." I got the idea. I miss him for numerous things that he did for me, but I miss him the most for the brief digressions into the history of Information Retrieval during our Tuesday IR meetings.

Young professors should learn from Chris Buckley what mentoring is all about. Chris went much beyond the call of duty to make sure that Professor Salton's students did not feel the affects of Professor Salton's loss. It would be very hard to find a researcher as committed to IR research as Chris. Chris taught me much of IR that I know, and he taught me how to do research in the field. I can never thank Chris enough for all I have learned, and am continually learning from him.

I thank Professor Claire Cardie for taking such good care of Professor Salton's students after his death. I thank her for serving as my committee chair, and for all her insightful comments about my work. Thanks to Professor Robert Constable and to Professor Adam Bojanczyk for serving on my committee, and for their valuable comments on earlier drafts of this thesis.

Mandar Mitra was an ideal colleague that I could have asked for. I would never forget his pleasant-natured style of working. In our discussions about work, when I would start making large leaps of faith, he would gently restrain me by saying, "Amit, I didn't get the last part, could you please go over that again." And, of course, I would discover that I had made a blunder in series of my thoughts. On a personal level, I thank Mandar for being such a good friend. People as helping as him are hard to find in this day and age.

Thanks to James Allan for his patient responses to my naive questions about the Smart system in my earlier years as a graduate student at Cornell, and for all the stimulating discussions about the field of IR. Professor Donald Crouch is largely responsible for motivating me to get a Ph.D. Without Don's help, I could never have done it.

I thank my parents for giving me the most wonderful childhood. I especially thank them for not stepping-in in my stupid mistakes, and letting me learn from my mistakes. They nurtured my independent thought, and gave me the foundation needed by a child to succeed in life. It took me a long time to realize how well my parent have raised me, and now, I am indebted for life.

And finally, I thank Shilpa, my wife, for always smiling during our hardships of a graduate student's life of destitute. And not only did she smile herself, she also made me laugh when the going got tough.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Determination of important keywords is crucial to the success of modern information retrieval (IR) systems. "How important is a keyword?", is an important question for statistical IR systems. Modern information retrieval systems use an importance predictor — the term weighting function[1] — to estimate the importance of keywords in a text. This work is an in-depth study of term weighting. Based on our observations, we propose new and better term weighting functions.

With an explosion in the availability of electronic information, the size of information collections is growing rapidly. Large text collections are commonplace now. These collections are a rich source for gathering accurate statistics about various aspects of languages. Most theories of term weights were developed on small text collections of short texts. It has been observed that the term weighting strategies developed for the small collections do a poor job of ranking articles from the new, large, more varied, full-text collections. Many of these strategies need reexamining for large collections. This work is aimed at studying term weighting for the large, more realistic, text collections.

Not many tools have been developed to study the direct effects of various components of a term weighting scheme on document ranking. Most work to study term weights is evaluated by the end result of the collective performance of the various components involved in a strategy. This work shows how we can develop tools to isolate the effects of individual components of a term weighting scheme. Based on such tools, we can improve individual components to achieve a higher collective performance from all the components.

1.1 Information Retrieval

The information age is here. Today mankind produces more information than any human being (actually, even many computers) can absorb. And with the advent of the World Wide Web, the saying that "all you need to know is out there, the key is to find it" is more true than ever. The field of information retrieval was born out of our need to know. When large amounts of information are available to be processed electronically, it should be possible to automatically find what we need, when we need. Most information in the modern world is text based; textual information retrieval is the most prevalent form of information retrieval (IR). Due to the usefulness of textual IR, the qualifier "textual" is oftentimes skipped, and the phrase "information retrieval" is used synonymously for textual information retrieval, as in the present study.

Large masses of text are now available for electronic processing. The aim of an IR system is to organize and store the information, and retrieve the useful information when a user poses a query to the system. Modern information retrieval systems usually accept a free-format natural language query from a user. Such a query represents the "information need" of the user. Given a large collection of articles (also called documents, information items, ...), keywords (usually words and phrases) are matched between a query and the articles to "predict" the potential usefulness of the articles for fulfilling the user's information need. Based on this prediction, all articles in the information base are ranked in decreasing order of their predicted usefulness. Using such a ranking, articles that are potentially most useful to the user are presented at high

[1] In the field of information retrieval, *term weight* is the common jargon used for term importance in a text. Important terms (typically words and phrases) are assigned higher weights in a text. Less important terms are assigned lower weights.

ranks, *i.e.*, earlier in the ranking, to the user. [SM83,Sal89]

Given a user query, a good information retrieval system should predict the usefulness of the articles well. An information retrieval system with a good predictor function will rank useful (relevant) articles ahead of the useless (non-relevant) articles, thereby allowing the user to peruse relevant information without having to read through much non-relevant information.

The two main aims of an information retrieval system are:

1. coverage: present most of the relevant articles, and
2. accuracy: *not* present most of the non-relevant articles.

The effectiveness with which an IR system achieves its first aim is measured in terms of *recall*. Recall is the proportion of the relevant articles presented to the user. Recall is maximized if most of the relevant articles in the database are presented to the user. It is quite easy for a system to achieve 100% recall by presenting all the articles in the information base to the user. If there is a relevant article in the collection, it is, of course, retrieved (presented). But this does not help the user in finding the good articles. For this reason, the accuracy of the search is also important. The effectiveness with which a system achieves accuracy is measured in terms of *precision*. Precision is the proportion of articles retrieved by a system that are relevant. If no non-relevant article is retrieved, the precision is 100%. Therefore, precision is maximized if most of the non-relevant articles are not retrieved. An IR system tries to maximize both recall and precision at the same time. Unfortunately these two goals have proven contradictory, and most IR systems try to strike a balance between the two. [Sal65,CMK66]

In recent years, the World Wide Web has emerged as the common platform for people to share information. This emergence has necessitated the availability of search capabilities for a user. One needs to find useful information in the colossal amount of information available on the web. Search engines based on information retrieval techniques have become an integral part of a web user's life. The field of information retrieval continually strives to enhance the search effectiveness of such systems. With the advent of the web, information retrieval techniques have had an enormous impact on how computer users use the Internet.

1.2 Term Importance

When word– and phrase–matches between a query and an article are used to predict the usefulness of the article, it is essential to differentiate between the important and the less important matches. For example, in English, a function word like "the" is bound to be present in almost all documents, as well as most queries. The likelihood of a match between a query and an article on the word "the" is very high, but this does not indicate that the article is potentially useful to a user. On the other hand, if a relatively uncommon word, for example the word "telecommunications", matches between a query and an article, chances that the article is potentially useful to the user are much higher (than the chances of its usefulness if the matching word was "the"). This observation suggests that different words should have different importance in the matching process.

Term weighting is a sub-field of information retrieval that studies the question of how important a word or a phrase is in a given text. [SJ73,SYY75,SB88,SB90b] Contribution of a matching term between a query and an article towards the final predicted usefulness of the article can then be made proportional to the term's importance in the query, and its importance in the article. This way, the important matches have a greater impact on the predicted usefulness, and we get a better prediction of the potential usefulness of an article. Proper estimation of term weights can significantly enhance the ranking effectiveness of an information retrieval system. [Buc93]

Three main factors that affect the importance of a term for text retrieval are:

1. **Term Frequency**: Articles that repeatedly use a query term are potentially more useful to the user than articles that rarely use that query term. [Luh57] For example, given a query "telecommunications", an article that uses the word "telecommunications" ten times is potentially more useful to the user than another article that uses the word "telecommunications" just once or twice.
2. **Inverse Document Frequency**: Common words, *i.e.*, the words that are used in numerous different articles, are less important than uncommon words, *i.e.*, the words that are used in few articles. [SJ72,

SJ73,SYY75,RSJ76] A match on a function word like "the" should contribute less towards a document's predicted usefulness as opposed to a match on a content word like "telecommunications".

3. **Document Length**: Long documents often tend to repeat terms and thus, in general, have higher term frequencies (just because they are long). Also long documents use numerous different words. Thus the number of matches between a query and a long document tend to be high. For these reasons, long documents get a preference in retrieval over short documents just because of their length, and not because they are more informative for the user. For good ranking, modern IR systems use *document length normalization* to remove this bias in favor of long documents. [SSMB96,SBM96,RW94]

Various formulations to incorporate these three effects for term importance have been proposed over years. [MK60,SB88,RW94] Many of these formulations originate from completely different bases (vector based, probabilistic, Poisson, ...), but eventually all the formulations use these three factors in one way or another. Since good term importance estimation is essential for effective information retrieval, better importance estimators are always desired.

1.3 Present Research

This work is aimed at developing better term importance estimators. Much of this work is concentrated on the term weighting functions that have been developed in the studies with the Smart system. [Sal71] We examine every component that affects a term's importance and, based on our study of these functions, we propose new, more effective term importance estimators.

In the past, most information retrieval techniques were tested on small collections of abstracts of articles. With the availability of more processing power, large collections of full-text articles (not only abstracts) can be realistically processed these days. Also, many large full-text collections are now available for IR experimentation. These large full-text collections provide us with much richer set of statistics for word– and phrase–usage than those provided by the small, abstract-only collections. Many hypothesis about term weighting functions that were valid on small collections do not scale up to the new and large collections of full-text articles. Out of the three main factors that affect term importance, the term frequency factor has been studied independently by Buckley et al. and Robertson et al. in recent years for the new full-text collections. [BSA93,RW94] The two main aspects of term weighting that have not been revisited for the new collections are the inverse document frequency factor and the document length normalization schemes. We study these two aspects in more detail in this work.

We start our studies by examining the document length normalization component of term weighting functions. To study the document length normalization component, we first develop a new tool that helps us in isolating the document length effects in document ranking. We hypothesize that systems that retrieve articles of all lengths in accordance with their chances of relevance will have high retrieval effectiveness. Based on observations from our new tool and our hypothesis, we propose a novel method for modifying any existing length normalization function to improve its ranking effectiveness. Our observations also help us in developing new, more effective length normalization functions. Experiments show that our hypothesis is true. When systems are modified to retrieve articles of all lengths in accordance with their chances of relevance, retrieval effectiveness improves significantly. [SSMB96,SBM96]

We apply our knowledge of document length normalization functions to degraded text environments. Optical character recognition (OCR) is a commonly used technique to scan printed information, information not otherwise available in electronic form, and create a large repository of such information. When used in conjunction with an automatic IR system, this can be a very effective way to let people search information that would otherwise only be available on non-searchable paper. It is well known that the OCR process is erroneous. [NR94] The information base generated using OCR is degraded (has errors), and ranking of an IR system can be strongly affected by these OCR errors. We study the effects of OCR errors on term importance estimators and develop new, more robust (against OCR corruption) document length normalization techniques that do better importance estimation in degraded text collections. This part of our work also explains why combining inverse document frequency factor with a commonly used normalization technique (cosine normalization) yields poor term weights. [SSB96]

Next, we study the inverse document frequency component of term importance. In this part of our work, we observe that the effectiveness of inverse document frequency (idf) in document ranking increases as the

predicted usefulness of documents decreases. We show that this increased usefulness of an inverse function of term document frequency is mainly due to the increased gap between the weights of the rare and the non-rare terms. We show that the function $idf^{1.5}$, which is a stronger inverse function of term document frequency than the traditional *idf* function, yields better retrieval effectiveness that the traditional *idf* function for three different sets of user queries.

1.4 Research Contributions

The main contributions of this research are:

1. **Document Length Normalization**: This part of our work observes that a normalization scheme that retrieves documents of all lengths with similar chances as their likelihood of relevance will outperform other schemes which retrieve documents with chances very different from their likelihood of relevance. We show that the retrieval probabilities for a particular normalization method deviate systematically from the relevance probabilities across different collections. We present *pivoted normalization*, a technique that can be used to modify any normalization function thereby reducing the gap between the relevance and the retrieval probabilities. Training pivoted normalization on one collection, we can successfully use it on other (new) text collections, yielding a robust, *collection independent* normalization technique. We use the idea of pivoting with the well known cosine normalization function. We point out some shortcomings of the cosine function and present two new normalization functions — *pivoted unique normalization* and *pivoted byte size normalization*.

2. **Document Length Normalization in Degraded Text Collections**: Term weighting schemes can be highly sensitive to the errors in the input text introduced by an OCR process. This part of our work examines the effects of the well known *cosine normalization* method in the presence of OCR errors, and proposes a new, more robust, normalization method. Experiments show that the new scheme is less sensitive to OCR errors and facilitates the use of more diverse basic weighting schemes. This study also explains why the use of cosine normalization in presence of the inverse document frequency factor is not advisable in large document collections.

3. **Inverse Document Frequency**: It is well known that the words that are used in numerous different articles are less important for ranking documents than words that appear in few articles. A term importance factor — *inverse document frequency (idf)* — an inverse function of the number of articles a term appears in, is commonly used by modern information retrieval systems to incorporate this effect in the retrieval process. We show that the usefulness of inverse document frequency in ranking documents increases as we go down in document ranking. When a document has several query terms, the relative importance of the query terms is not very useful in ranking. For this reason, we can use a weaker idf factor in term weights for highly ranked articles (for example idf^1). On the other hand, when documents have a weak match to a query, *i.e.*, very few query terms are present in the documents, it is important that the matching terms are "good terms", and thus the relative importance of the query terms is quite useful in ranking. Therefore we should use a strong idf factor in term weights for poorly ranked articles (say idf^2). After examining various possible formulations in place of the standard *idf* function, we found the function $idf^{1.5}$ to be a better *across the board* replacement for *idf*. The function $idf^{1.5}$ yields important improvements over the standard *idf* function across query sets.

1.5 Organization

The rest of this thesis is organized as follows:

- Chapter 2 introduces the vector space model of information retrieval, the underlying model used in all our experimentation. We then elaborate upon the three main factors in term weighting: term frequency, inverse document frequency, and document length normalization. We show how ranking techniques are evaluated using recall–precision figures. Finally we introduce the TREC collections, the test-bed used for all out experiments.

- Chapter 3 introduces our new tool to study document length normalization. We compare the well-known cosine normalization scheme with normalization schemes from two other successful IR systems.
- Chapter 4 introduces the idea of pivoting, a new technique that we use to modify existing normalization functions to enhance their retrieval effectiveness. We present the results from pivoting the cosine normalization scheme. We observe some shortcomings of the cosine function and propose a new normalization function: the pivoted-unique normalization function.
- Chapter 5 studies the problem of document length normalization in degraded text collections. Based on our observations, we propose another new document length normalization function for degraded text environments: the pivoted-byte-size normalization function.
- Chapter 6 revisits the commonly used inverse document frequency function. We show that an inverse function of term document frequency is more effective at low ranks. Based on our observations, we propose that the traditional idf function be replaced by the function $idf^{1.5}$.
- Chapter 7 concludes this work and points towards some future directions.

Chapter 2

Background

This chapter introduces the vector space model of information retrieval, the underlying model used in all our work. In the realm of the vector space model, we show how documents are ranked. We introduce how term weights are automatically assigned by modern information retrieval systems. We introduce the Smart text retrieval system, the system used for all our work. We then introduce how ranking effectiveness of an information retrieval system is measured.

2.1 The Vector Space Model

The vector space model of information retrieval was developed by Salton and his students in the late 1960's and the early 1970's. [SWY75] This model transforms any given text (article, query, portion of an article, ...) 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. Given several texts — the documents — the closeness of their vectors from the vector of a specific text — the query — can be measured. 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. [SM83,Sal89]

Salton and his students also implemented a system based on the vector space model, the Smart system. [Sal71] 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, soft boolean models, relevance feedback, document clustering, use of a thesaurus, ... 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 the field of IR.

2.1.1 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), 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*. Figure 2.1 shows 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

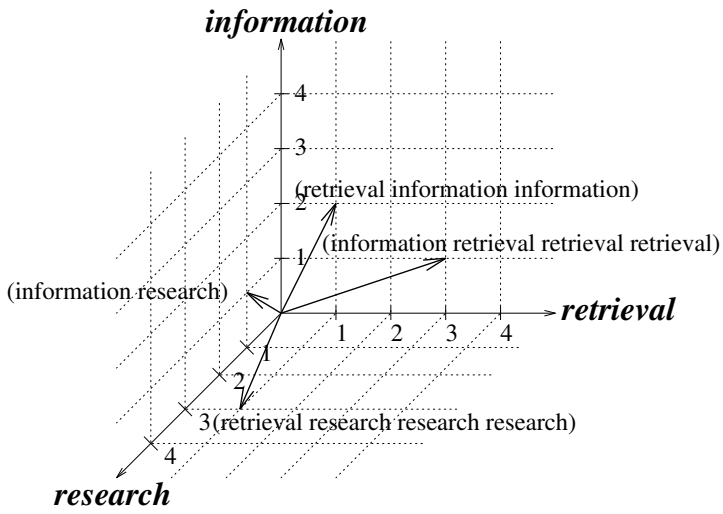


Figure 2.1: A three dimensional vector space

sub-vector in the corresponding dimension, and any text uses a very small subset of the entire vocabulary (except for when a dictionary is considered an article), most text vectors will be sparse, *i.e.*, they will have many zero components.

Several formulations propose themselves to be used as a measure of the closeness between two vectors (texts). Two texts are semantically related if they share some vocabulary; the more vocabulary they share, the stronger is this relationship. This suggests that our measure of closeness should increase with the number of word/phrase matches between two texts. Also, if the matching terms are important in either text, then the vectors should be considered closer than if the matching terms are not important in the texts. For example, a match on the word "telecommunications" is worth more than a match on the word "the".

In modern vector-based information retrieval systems, the length of the sub-vector in dimension-i is used to represent the importance, or the *weight*, of word-i in a text. Words that are absent in a text get a zero weight. A measure that increases with the number of word/phrase matches between two texts, and also with the importance of the matching terms is the *vector inner product* measure. Given two text vectors $\vec{P} = (p_1, p_2, \ldots, p_T)$ and $\vec{Q} = (q_1, q_2, \ldots, q_T)$ in a T dimensional vector space, where x_i is the sub-vector in dimension-i for vector $\vec{X}$, the inner product between the two vectors is

$$\vec{P} \cdot \vec{Q} = \sum_{i=1}^{T} \sum_{j=1}^{T} p_i \times \vec{u_i} \cdot q_j \times \vec{u_j}$$

or

$$\vec{P} \cdot \vec{Q} = \sum_{i=1}^{T} \sum_{j=1}^{T} p_i \times q_j \times (\vec{u_i} \cdot \vec{u_j})$$

where $\vec{u_i}$ and $\vec{u_j}$ are unit vectors in dimensions i and j. Making the (obviously false) binary term independence assumption — any two words are orthogonal, or unrelated[1] — we can assume that individual dimensions in our vector space are orthogonal, and thus $\vec{u_i} \cdot \vec{u_j} = 0$ whenever $i \neq j$. Calling the closeness between two vectors the *vector similarity*, and measuring it by the vector inner product formulation, we get:

$$similarity(\vec{P}, \vec{Q}) = \sum_{i=1}^{T} p_i \times q_i$$

This formulation has both the desirable properties of a closeness measure for texts.

[1] This assumption is commonly made in modern IR systems. The use of inter-word correlations makes information retrieval models more complex. [YBLS83] It is obvious that the word "school" is related to the word "children" but measuring this relatedness is hard. There are other techniques to extract the benefits that we would get from a detailed knowledge of inter-word relationships.

- **Increases with the number of matches**: When a term is present in both the texts, it gets a non-zero weight in both the texts. As two texts have more and more common vocabulary, the number of non-zero products in the similarity (which is a sum of such products: $p_1 \times q_1 + p_2 \times q_2 + \ldots + p_T \times q_T$) increases, increasing the total similarity between the texts.
- **Increases with the importance of matching terms**: If the matching words are important, *i.e.*, they have a high weight in either of the texts (high p_i or high q_i or both), then the contribution of that single match towards the total similarity is high, once again yielding a high similarity between texts.

2.1.2 Document Ranking

A natural language query entered by a user is converted into a weighted term vector and, using the inner-product function, a numeric similarity is computed between the query vector and the vector for every document in the collection. [Sal89]

If Q is the query vector and D_i is the vector representation for document-i, a numerical similarity between the query and the document is computed as,

$$Sim(Q, D_i) = \sum_{common\ terms\ t_j} q_j \times d_{ij}$$

Where t_j is a term present in both the query and the document, q_j is the weight of term t_j in the query, and d_{ij} is its weight in document-i. The summation is performed over all such terms t_j present in both the query and the document. Such a similarity supposedly measures the potential usefulness of a document for the user-query. The documents in the information base are ranked by their decreasing similarity to the query and are presented to the user in this order. Documents presented earlier in a search have a higher degree of vocabulary overlap to the user-query, and are potentially more relevant to the user's information need.

2.2 Term Weights

With a simple inner product similarity computation formula, the goodness of the document ranking is completely dependent upon the assignment of proper weights to terms in the texts (documents and queries). Judicious assignment of term weights can substantially improve the ranking effectiveness of a system. [SB88, Buc93] Most modern information retrieval systems automatically assign weights to the terms in a text. The two basic factors responsible for determining the importance of a term in a text are the *term frequency factor* and the *inverse document frequency* factor.

2.2.1 Term Frequency Factor

The importance of a term increases with the number of occurrences of the term in a text. [Luh57,SB88] Therefore we should 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* factor 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

   $$1 + ln(tf)$$

   where *tf* is the term frequency of a term. This function, proposed by Buckley, [BSA93, BAS94] is motivated by the following fact: if a document has one query term with a very high term frequency, then that document is (often) not better than another document that has two query terms with somewhat lower term frequencies. Therefore, in general, more occurrences of a match should not out-contribute fewer occurrences of several matches.

Consider as an example the query "recycling of tires", and two documents — D_1 with ten occurrences of the word "recycling", and D_2 which uses both the words "recycling" and "tires" three times each. Everything else being equal, if raw *tf*s are used, D_1 gets a higher similarity (proportional to 10) than D_2 (which gets a similarity proportional to $3 + 3 = 6$). But it is intuitive that D_2 addresses the needs of the query much better than D_1 because it addresses both recycling and tires, and not just recycling (like D_1). When a logarithmic term frequency factor is used, the usefulness of D_2 is reflected in the similarities. D_1 gets a similarity related to $1 + ln(10) = 3.3$ whereas D_2 gets a similarity related to $[1 + ln(3)] + [1 + ln(3)] = 4.1$, and D_2 is ranked above D_1, as is desirable.

3. **The binary tf factor**: This *tf* factor completely disregards the number of occurrences of a term; it is either one or zero depending upon the presence (one) or the absence (zero) of the term in a text. This factor gives a nice baseline to measure the usefulness of the term frequency factors in document ranking.

4. **The augmented tf factor**: This *tf* 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: [SB88]

$$0.5 + 0.5 \times \frac{tf}{\text{maximum tf in text}}$$

The augmented *tf* factor, like the logarithmic tf factor, also emphasizes that more matches are more important than fewer matches. A single match contributes at least a 0.5 and high *tf*s can only contribute at most another 0.5. This factor was motivated by document length considerations (see Section 2.2.3), and does not work as well as the logarithmic *tf* factor in practice.

5. **Okapi's tf factor**: Recently, Robertson et al. have developed another *tf* factor (in the Okapi information retrieval system) based on approximations to the 2-Poisson model. This *tf*-factor [RW94]

$$\frac{tf}{2 + tf}$$

is quite close to the logarithmic *tf* factor in its behavior.

Tests with large full-text collections show that the logarithmic *tf* factor is most effective for good document ranking.

2.2.2 Inverse Document Frequency

Just using the *tf* factors to estimate term importance does not suffice. Consider the common words (for example the word "put") that occur with very high term frequencies across numerous articles. Such words 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.

It has been observed that commonality of a word can be related to its usage. Words that are used in numerous articles are more common and are less useful for document ranking than words that appear in very few articles. This suggests that an inverse function of the number of articles a word appears in 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

$$log(\frac{N}{df})$$

where N is the total number of articles in a collection and df is the document frequency of the word. A word that appears in all the articles in the collection will get a zero weight. [SJ72,SJ73]

Other importance estimators based on term discrimination value theory have been proposed in place of the inverse document frequency factor. [SY73,Sal75,SYY75] The term discrimination theory is based upon the hypothesis that terms that bring the document vectors close to each–other in the vector space are poor discriminators and should be given low importance; on the other hand, if using a term spreads the document vectors apart, then it is a good index term and should be assigned a high weight.

Other researchers have derived importance functions based upon probabilistic *term relevance* weight estimation. [BS75] The term relevance weight is defined as the proportion of the relevant documents in which a term occurs divided by the proportion of the non-relevant documents in which the term is present. This factor is not immediately available without a detailed knowledge of relevance for queries, and can not be used in this form for ad-hoc querying. Croft and Harper have shown that under reasonable assumptions, the term relevance weight reduces to $log(\frac{N-df}{df})$, a function quite similar to the inverse document frequency factor. [CH79]

The final weight of a term in a text is then: *tf–factor*×*idf–factor*, and is called the $tf \times idf$ weight.

2.2.3 Document Length Normalization

Even though $tf \times idf$ weights are a good first approximation to term importance in a text, they are not adequate. They ignore a very important aspect on which the importance of a term depends — the document length. In real life, documents are of different lengths, and this length variation favors the similarities of the longer documents due to the following two reasons: [SBM96]

1. **Higher term frequencies:** Long documents usually use the same terms repeatedly. As a result, the term frequency factors may be large for long documents. This increases the average weight of terms in the long documents, which, in turn, increases the contribution of a long document's individual matches (to a query) towards the query–document similarity, resulting in a high overall similarity.

2. **More terms:** Long documents also have numerous distinct terms. This increases the number of matches between a query and a long document, increasing the query–document similarity and the chances of retrieval of long documents in preference over shorter documents.

Therefore we need to compensate for the length differences between articles in a text collection.

Document length normalization of term weights is used to remove the advantage that the long documents have in retrieval over the short documents. [SSMB96,SBM96] If the weights of the terms in a long documents are (somehow) depressed, then the contributions of individual matches between a long document and a query will go down, resulting in a lower overall query–document similarity for long documents. This would give the shorter documents a chance to compete with longer documents in terms of document similarity. Document length normalization is a way of penalizing the term weights for a document in accordance with its length. Various normalization techniques are used in information retrieval systems. Following is a review of some commonly used normalization techniques:

- **Cosine Normalization:** Cosine normalization is the most commonly used normalization technique in the vector space model. [SB88] In section 2.1.1, we introduced the vector inner product measure to compute the closeness (also called correlation) between two vectors. The inner product similarity measure does not compensate for document length differences, instead, one has to introduce length normalization in the term weights when using the inner product measure. As people had realized the need to normalize for document length differences since the beginning of experimental IR, vector correlation measures (other than the inner product measure) that compensated for vector length differences have been proposed earlier. These measures, for example, the Dice correlation, the Cosine correlation, the Jaccard correlation, the overlap correlation ..., were bounded measures as opposed to the vector inner product measure which can be unbounded in principle. [Sal89,Sal71]

  The bounding factor for such measures restricts the correlation values between strict bounds (for example, 0 in case there were no matching terms between two vector and 1 in case of identical vectors). Such bounding factor, in turn, acts as the document length normalization factor. For example, consider the Cosine correlation between a query vector Q and a document vector D:

$$cos(Q, D) = \frac{\sum_{i=1}^{T} w_{q_i} \times w_{d_i}}{\sqrt{{w_{q_1}}^2 + {w_{q_2}}^2 + \ldots + {w_{q_T}}^2} \times \sqrt{{w_{d_1}}^2 + {w_{d_2}}^2 + \ldots + {w_{d_T}}^2}}$$

where w_{q_i} is the $tf \times idf$ weight of term-i in the query vector, and w_{d_i} is its $tf \times idf$ weight in the document vector. The cosine correlation is bound between 0 and 1 by the use of the Euclidean lengths of the individual vectors in the denominator. It implicitly normalizes for length variation of documents. The cosine correlation can also be written as:

$$cos(Q, D) = \sum_{i=1}^{T} \left(\frac{w_{q_i}}{\sqrt{{w_{q_1}}^2 + {w_{q_2}}^2 + \ldots + {w_{q_T}}^2}} \times \frac{w_{d_i}}{\sqrt{{w_{d_1}}^2 + {w_{d_2}}^2 + \ldots + {w_{d_T}}^2}} \right)$$

which is same as the vector inner product similarity measure if individual $tf \times idf$ weights of a vector were divided by the Euclidean length of the vector.

Since it is easier to implement inner product similarity in IR systems with inverted lists, as an implementational convenience, the denominator for individual vectors is precomputed, and the $tf \times idf$ weights for terms are divided by it before being stored in the inverted lists. Doing this, one can now use vector inner product similarity to obtain cosine correlation between two vectors. The denominator, over years, has been named as the *cosine normalization factor*.

Thus, the cosine normalization factor is computed as

$$\sqrt{{w_1}^2 + {w_2}^2 + \ldots + {w_t}^2}$$

where w_i is the raw $tf \times idf$ weight for a term. Cosine normalization attacks both the reasons for normalization (*higher tfs* and *more terms*) in one step. Higher individual term frequencies increase individual w_i values, increasing the penalty on the term weights. Also, if a document has more terms, the number of individual weights in the cosine factor (t in the above formula) increases, yielding a higher normalization factor.

- **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 ($0.5 + 0.5 \times \frac{tf}{max\_tf}$), [SB88] and the tf weights used in the INQUERY system ($0.4 + 0.6 \times \frac{tf}{max\_tf}$) [Tur90,CCH92] are examples of such normalization.

  By restricting the tf factors to a maximum value of 1.0, this technique only compensates for the first reason (*higher tfs*) for normalization. When used without any correction for the second reason (*more terms*) this turns out to be a "weak" form of normalization and favors the retrieval of long documents. [BCCN95] This normalization technique was proposed when the second reason for normalization (more matches for long articles) was not considered harmful for ranking. It is true that as the number of matches between a query and an article increase, so do the chances of relevance of the article. But still, for long articles, the advantage due to more matches should not be left unchecked; the advantage must be supervised by some (possibly weak) normalization. Normalization by maximum tf leaves this advantage completely unchecked and favors long documents.

- **Byte Length Normalization:** More recently, a length normalization scheme based on the byte size of documents has been used in the Okapi system. This normalization technique is based on approximations to the 2-Poisson model. This normalization formulation is used in conjunction with Okapi's tf formulation (see Section 2.2.1) and the final normalized term weight is:

  $$\frac{tf}{2 \times (1 - b + b \times \frac{document\ length}{average\ document\ length}) + tf}$$

  where b is some constant, typically 0.75. [RWJ+95] The byte size of an article grows if it uses the same word repeatedly, as well as if it uses new words. Therefore, this normalization factor attacks both the reasons for normalization in one shot.

2.3 The Smart System

The Smart system is a sophisticated text processing system based on the vector space model, developed over the last thirty five years. [Sal71] The Smart system generates vectors for any given text automatically by indexing the text. Automatic indexing of a text usually involves the following steps: [Sal81]

Table 2.1: Term Weights in the Smart System

Term Frequency		Inverse Document Frequency		Normalization	
First Letter	$f(tf)$	Second Letter	$f(\frac{1}{df})$	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)	$\sqrt{w_1^2 + w_2^2 + \ldots + w_n^2}$
a (augmented)	$0.5+0.5 \times \frac{tf}{max\ tf}$				

- **Tokenization:** The text is first tokenized into individual words and other tokens.
- **Stop word removal:** Common function words (like *the*, *of*, *an*, ...), also called stop words, are removed from this list of tokens. The Smart system uses a predefined list of 571 stop words.
- **Stemming:** Various morhological variants of a word are normalized to the same *stem*. [Lov68] Usually simple rules for suffix stripping are used in this process.
- **Phrase formulation:** Optionally, phrases in a text are recognized and are used in addition to the list of single words to index the text. [Fag87]
- **Weighting:** The term (words and phrases) vector, thus created for a text, is weighted using tf, idf, and length normalization considerations.

In the Smart system, term weighting schemes are denoted by triples of letters. The first letter in a triple is a shorthand for the term frequency factor being used in the term weights, the second letter corresponds to the inverse document frequency function, and the third letter corresponds to the normalization factor applied to the term weights. [SB88] Table 2.1 summarizes the most commonly used functions in Smart's term weighting. The *max tf* in the augmented *tf* weight formula is the maximum term frequency of any term in the document under consideration. The N in the full inverse document frequency formula corresponds to the total number of documents in the collection.

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 *anc.ltn* is used to symbolize a retrieval run, the document term weights used in the run are,

$$\frac{0.5 + 0.5 \times \frac{tf}{\max\ \mathrm{tf}}}{\sqrt{\Sigma_{all\ terms}(0.5 + 0.5 \times \frac{tf}{\max\ \mathrm{tf}})^2}}$$

and the query term weights are,

$$(1 + \log(tf)) \times \log(\frac{N}{df})$$

and the inner product similarity measure is used to rank documents. We will use this term weight triple notation to characterize all our experiments in this work.

2.4 Evaluation

To determine if a technique is ranking the documents better than another technique, evaluating the effectiveness of our ranking techniques is very important in information retrieval. Objective evaluation across a set of queries and across different text collections facilitates a study of the general applicability of new techniques. Objective evaluations of IR techniques has been one of the biggest strengths of the field of information retrieval. [CM63,Sal65,CMK66,Lan79,Sal91] The two main objectives of an information retrieval system are:

1. **Find all relevant articles**: Obviously, given a query, we would like to present all useful articles to the user. The effectiveness with which an information retrieval system achieves this objective is measured by *recall* — the proportion of relevant articles retrieved by a system. If a system retrieves all relevant articles, then the recall value is 100% (or 1.0).

2. **Reject all non-relevant articles**: Once again, given a query, we would not like to present any useless article to the user. The effectiveness with which an IR system achieves this objective is measured in terms of *precision* — the proportion of the retrieved articles that are relevant. If the system does not retrieve any non-relevant article, then all the retrieved articles are relevant and the precision is 100% (or 1.0).

Traditionally, these two goal have been conflicting. Techniques that enhance search recall tend to hurt search precision and vice-versa. [SM83]

Most information retrieval experimentation heavily depends on the existence of a "test collection". A test collection is a collection of articles along with a set of test queries. The set of the relevant articles for each query is also known. To measure the recall and precision for a technique, documents are retrieved using that technique for the test queries from the collection. Since relevance for the queries is known, the recall and precision values can be measured. Usually the recall and precision values are averaged across all queries to estimate the average performance of a technique. A technique that works well for most of the queries is considered good. If a technique just works well only for a few queries, the evidence that this technique is in general applicable is not considered strong.

2.4.1 Average Precision

Recall and precision are defined based upon the notion of there being a retrieved set of documents and a non-retrieved set of documents. In ranked retrieval, however, there is no clear demarcation between the retrieved articles and the non-retrieved articles; all articles are simply ranked by their predicted goodness. One has to evaluate the quality of the ranking in the entire collection. *Average precision* is the most common evaluation technique used to compute the goodness of ranking for a system. [SM83]

Using a retrieval technique, articles in the test collection are ranked for a set of query. The precision values at 0% recall, 10% recall, ..., 100% recall are computed. As a query might not have well defined 10%, 20%, ...recall points, interpolation of precision values is used to find the precision values at various recall points. [Kee71] The precision value at each recall point is averaged across queries to compute the average precision value at that recall point for the entire set of queries. The average precision values at 11 recall points are further averaged to compute the 11-pt average precision for the entire set of queries.

To avoid any interpolation biases, Buckley has recently proposed using non-interpolated average precision which is currently being used in the TREC evaluation studies. [Har95] To obtain the average non-interpolated precision for a query, we average the precision value at each distinct recall point in the document ranking for the query. A new recall point is obtained when a new relevant document is encountered in the ranking. The average non-interpolated precision for all the queries is further averaged to obtain the average non-interpolated precision for the entire experiment.

2.4.2 The TREC Collections

Text REtrieval Conference, or TREC, is an ARPA and NIST co-sponsored effort that brings together information retrieval researchers from around the world to discuss their systems, and to develop a large test-bed for automatic retrieval systems in the process. [Har93,Har94,Har95,Har96] The fourth in this series of annual conferences, TREC–4, attracted thirty seven different participants from academic institutions, government organizations, and commercial organizations. With such a large participation of various IR researchers, large and varied collections of full-text documents, a large number of user queries, and a superior set of independent relevance judgments, TREC collections have rightfully become the standard test collections for current information retrieval research.

The common IR task of ranking documents for a new query is called the "ad-hoc" task in the TREC framework. The TREC data comes on CD-ROMs, called the TREC disks. The disks are numbered, and a

Table 2.2: TREC Document Statistics

Source	Size (Mb)	Number of articles	Median number of terms/article	Average number of terms/article
Disk 1				
WSJ	270	98,732	182	329
AP	259	84,678	353	375
ZIFF	245	75,180	181	412
FR	262	25,960	313	1017
DOE	186	226,087	82	89
Disk 2				
WSJ	247	74,520	218	377
AP	241	79,919	346	370
ZIFF	178	56,920	167	394
FR	211	19,860	315	1073
Disk 3				
SJMN	290	90,257	279	337
AP	242	78,321	358	379
ZIFF	349	161,021	119	263
PAT	245	6,711	2896	3543

combination of several disks can be used to form a text collection for experimentation. The characteristics of the data on various disks is listed in Table 2.2 (adapted from [Har95]). The sources for the data are:

- Disk 1
  - WSJ – Wall Street Journal (1987, 1988, 1989)
  - AP – AP Newswire (1989)
  - ZIFF – Articles from *Computer Select* disks, Ziff Davis Publishing
  - FR – Federal Register (1989)
  - DOE – Short abstracts from DOE publications
- Disk 2
  - WSJ – Wall Street Journal (1990, 1991, 1992)
  - AP – AP Newswire (1988)
  - ZIFF – Articles from *Computer Select* disks, Ziff Davis Publishing
  - FR – Federal Register (1988)
- Disk 3
  - SJMN – San Jose Mercury News (1991)
  - AP – AP Newswire (1990)
  - ZIFF – Articles from *Computer Select* disks, Ziff Davis Publishing
  - PAT – U.S. Patents (1993)

For the first three TREC conferences, TREC–1, TREC–2, and TREC–3, the collection of articles for the ad-hoc task was the same — all articles from disks 1 and 2. The document collection for the TREC–4 ad-hoc task was changed and disks 2 and 3 were used at TREC–4. Each year fifty new user queries are given to the participants in a blind test. Ranked lists from various participants are pooled and judged by the users for relevance. Once relevance is assessed, average precision values for various participants are computed.

Table 2.3: Query Statistics

Query Id.	Training 1–50	TREC–1 51–100	TREC–2 101–150	TREC–3 151–200	TREC–4 202–250
# of Queries	50	50	50	50	49
Median # terms / query	51	57	84	52	7
Average # terms / query	56.10	93.94	81.68	51.96	8.33

The queries have also varied widely from year to year. In the first two conferences, TREC–1 and TREC–2, the queries were quite long and represented long standing user information needs. It is well known that most users type very short queries for ad-hoc searching. Due to this, the queries for TREC–3 were considerably shorter and the queries for TREC–4 were just a sentence or two. The characteristics of the query sets are shown in Table 2.3. To train the systems for TREC–1, fifty training queries were also provided. It is clear that the TREC–4 queries are much closer to the realistic queries that users might enter into a search system. Short queries also give less information to a system about the users' information need; the quality of document ranking is usually poorer for short queries.

To verify the general applicability of our techniques, we will test our techniques on various queries. To make sure that our techniques are not collection dependent (effective only on one text collection), we will use the TREC corpus in various configurations (*e.g.*, disks 1 and 2, disks 2 and 3, WSJ only, AP only, ...). This ensures that we test our techniques on different set of documents with different collection characteristics. If a technique is effective for a large set of queries across multiple collections, there are strong chances that the technique will be generally effective.

Chapter 3

Study of Normalization

In this chapter, we introduce a new approach that we have developed to study the length normalization properties of retrieval techniques. Using our approach, we compare three successful term weighting formulations. Observations from our comparison suggest that there is a strong correlation between the length normalization properties and the retrieval effectiveness of a system.

3.1 Approach to Study Normalization

If all documents were the same length, information retrieval systems would not need to normalize term weights for document length variations. However, in most realistic text collections, there is a wide variation in the lengths of the documents and document length normalization of term weights is absolutely necessary. [SSB96] But how can we know if our normalization function is doing the right degree of length normalization? Document length normalization is necessary because longer documents are retrieved in preference over shorter documents because of their length, and not because they are more useful to the user.

If we can eliminate the competition between documents of different lengths, and make documents of a given length compete only with other documents of the same length to reach the top, then we will be doing the best possible normalization. For example, suppose we are retrieving (say) 100 articles for a query. If we somehow know that in an "ideal retrieval" for the query, seventeen of the 100 retrieved articles should be 2,000 bytes long, then we can force our system to retrieve exactly seventeen 2,000 bytes long documents in the top 100 ranks. Now the problems due to other documents being of different length (than 2000 bytes) will not affect us. In a way, all 2,000 bytes long documents are competing against each other to get in the top seventeen (documents of length 2,000 bytes) for the query.

At a more abstract level, if we somehow know that in an ideal retrieval run for a query retrieving P top documents, p_1 should be of length l_1, p_2 should be of length l_2, ..., and p_n should be of length l_n (where $p_1 + p_2 + \ldots + p_n = P$), then we can force our system to retrieve the best p_1 documents of length l_1, the best p_2 documents of length l_2, ..., and the best p_n documents of length l_n. Now all documents of length l_i will be competing against each other to make it to the top p_i documents of that length, and the competition between documents of different length to reach the top will not be there. This is the ultimate objective of a length normalization scheme.

But what is an ideal retrieval for a query? Ideally a retrieval system should retrieve all relevant articles for a query and reject all non-relevant articles for the query. However, we cannot use this criterion in practice because relevance for queries is not known in advance. But we can certainly use this criterion retrospectively to observe how close our retrieval was to the ideal retrieval for a set of test queries (relevance for which is known). For example if there are R known relevant articles for a test query, in an *ideal retrieval* for that query, we would like to retrieve these R relevant articles in the top R ranks. If r_1 of these relevant articles are of length l_1, r_2 are of length l_2, ..., and r_n are of length l_n, then we know that in an ideal retrieval, if we retrieve the top R articles, r_1 of those will be of length l_1, r_2 of length l_2, ..., and r_n of length l_n. The relative ranking of these relevant documents within the top R ranks is immaterial in terms of recall and precision.

In another view, if a system retrieves R documents for a query, and it retrieves exactly r_1 documents of length l_1, r_2 documents of length l_2, ..., and r_n documents of length l_n, then we know that the system is doing the best possible normalization. Even though the system might leave out some relevant articles of length l_i in the top r_i and retrieve some non-relevant articles, we know that the relevant articles of length l_i were not defeated by the non-relevant articles of another length. They were defeated due to some other aspect of our retrieval system, not document length. The normalization for this system is optimal. Based on this view of a good normalization scheme, we hypothesize that:

> Normalization schemes that retrieve documents of a certain length with the same chances as there are chances of finding a relevant document of that length, will have better retrieval effectiveness than normalization schemes which retrieve documents of various lengths with very different chances from their chances of relevance. [SSMB96]

One possible way, then, to study normalization properties of a retrieval system is to take a set of test queries, retrieve R top documents for a query (where R is the number of relevant documents from the query), pool the retrieved set and compare its length distribution to the length distribution of the relevant set (which is known for these test queries). If these two distributions differ noticeably, *i.e.*, we are retrieving the documents of a given length with chances different from their chances of relevance, then we are not normalizing well. On the other hand, if the length distribution of the retrieved set matches well with the length distribution of the relevant set, then we are doing good document length normalization. With good normalization, if a relevant article is ranked below a non-relevant article, it is so not due to poor normalization, but due to some other reason.

3.2 Implementation of Approach

In practice, however, we have observed that the length retrieval patterns of a retrieval scheme are similar irrespective of the number of top documents retrieved. For example, if the top 500 articles retrieved for a query have 50 articles of length l_1, 100 of length l_2, 150 of length l_3, and 200 of length l_4, then if we instead retrieve only 100 articles for this query, the system will retrieve about 10 articles of length l_1, 20 of length l_2, 30 of length l_3, and 40 of length l_4. On the other hand, if we retrieve 1,000 articles, the system will retrieve about 100 articles of length l_1, 200 of length l_2, 300 of length l_3, and 400 of length l_4. Basically, the ratio of the retrieved articles of length l_i to the total number of retrieved articles remains about the same at various rank cut-off levels.

This observation is illustrated in Figure 3.1 which shows the ratio of the retrieved articles of a given length to the total number of retrieved articles at three different rank cut-off levels, top 100 documents, top 500 documents and top 1,000 documents, averaged over 49 queries. The details of how we obtained this plot are discussed later in this Section. We used the TREC–4 ad hoc task data (documents from disks 2 and 3 and queries 202–250) to generate Figure 3.1. As we can observe, if we retrieve enough articles to give us stable ratios (top five or ten articles would probably not yield meaningful ratios), irrespective of how many top documents are retrieved, the ratio of the retrieved articles of a given length to the total number of retrieved articles remains approximately the same across the document length spectrum.

We observe that for a given query, if we limit ourselves to retrieving documents that can be even remotely useful to a user (say the best 30,000–50,000 documents for the query in the collection), then documents of a given length are distributed evenly across the similarity spectrum by a retrieval system. This fact is true for all document lengths. For this reason, the "length pattern" of the retrieved documents remains unchanged irrespective of how many documents are retrieved, as long as we retrieve a reasonable number of documents (say at least 100) and we do not retrieve unreasonably high number of documents (say more than 30,000).

With this uniformity at all rank cut-off levels, we can retrieve any reasonable number of articles and we will get the same length distribution for the retrieved articles. Many queries have very few relevant documents. If we only retrieve as many documents for a query as there are relevant documents for that query, we will retrieve very few articles for such queries and slight randomness in the retrieval process can change the length distribution noticeably. Retrieval of a higher number of articles for all queries yields more stable retrieval ratios. For this reason, instead of retrieving the same number of articles as there are relevant articles for a query, we retrieve the top 1,000 articles for every query in doing our length analysis.

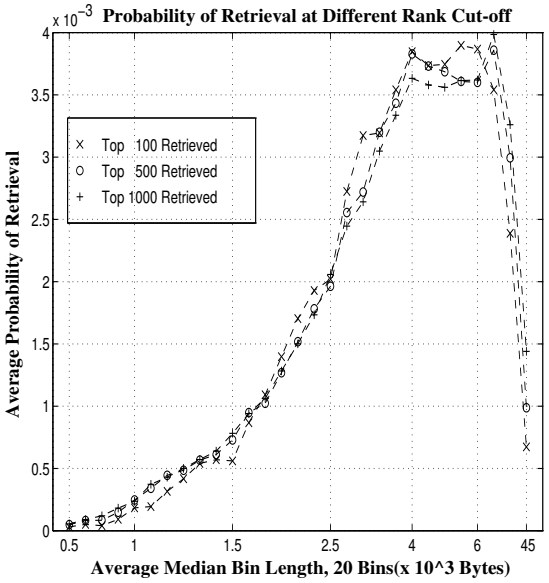


Figure 3.1: Probability of retrieval at various rank cut-offs

We also do a coarse length comparison, *i.e.*, if a relevant article is 2,000 bytes long and it is ranked below a non-relevant article which is "about 2,000 bytes" (say 1,996 or 2,010 bytes), then one cannot attribute this rank reversal to poor normalization. To develop a robust analysis technique, minor differences in the byte sizes of documents should not confuse our analysis. For this reason, we place the articles of similar lengths from our collection into bins (or buckets) and assume that all articles in a bin are of about the same length. We pick a representative bin length, typically the median document length in a bin. This bucketing allows us to do the coarse length comparison that we want to do.

With the retrieval of the top 1,000 articles for every query and the coarse length analysis, our final approach to study normalization is: [SSMB96,SBM96]

1. Sort the documents in the entire collection by their byte size and divide the documents into bins of N (about 1,000) documents each. In doing this, it is possible that the last bin gets a much lower number of documents. For example if the collection has 742,100 articles and we are using bins of 1,000 documents each, then the last bin will have only 100 articles. With so few articles, the relevance/retrieval ratio obtained for the last bin can be affected due to minor randomness in relevance/retrieval. If for an unusual combination of words, a query retrieves an article from the last bin, the retrieval ratio for the last bin will be an unusually high 1/100 or 1%. On the other hand if the query retrieves an article from a regular bin (by chance), the retrieval ratio is 1/1000, or 0.1%, an order of magnitude lower.

   This aspect becomes very important for us because the last bin usually contains the longest documents from the collection. These documents are usually an order of magnitude longer than most documents in the collection, and normalizing these well is a challenge for the normalization functions. To avoid getting random statistics for the last bin, we decrease our bin size (from 1,000 in the example) until the last bin has a size at least 80% of regular bins. Now we are sure that the statistics for the last bin are also reliable. In the above example, we would reduce our bin size from 1000 to 999 since the last bin will now contain 842 articles (742,100 = 742×999 + 842), 84.28% of the regular bins (which have 999 articles each).

2. Pool all the relevant articles for a set of queries and count how many of those (relevant) articles are from a particular bin. Retrieve the top 1,000 articles for each query and pool all the retrieved articles for the set of queries; count how many of those (retrieved) articles are from a particular bin. Since we are not retrieving exactly the same number of articles as there are relevant articles for a query (the number of relevant articles can be very different from 1,000, the number of articles retrieved), in order to compare these two numbers for any bin, we need to divide them by the total number of relevant

articles and the total number of retrieved articles, respectively. This yields the percentage, or the probability, of relevance/retrieval for a bin (as we can claim that if 100 articles are relevant/retrieved, X will be from the ith bin). In terms of conditional probability, given a document D, this ratio for the ith bin and the *relevant* documents can be represented by

$$P(D \in Bin_i \mid D \textit{ is Relevant})$$

Similarly this ratio for the ith bin and the *retrieved* documents can be represented by

$$P(D \in Bin_i \mid D \textit{ is Retrieved})$$

This way we get one probability of retrieval figure and one probability of relevance figure for every bin.

We also scale the figures for the last bin to be comparable to the rest of the bins. In the above example, the last bin had 842 articles and the rest of the bins had 999 articles each. If 100 relevant articles were from the last bin, keeping in mind that these 100 articles are from a total of 842 articles, and that if the last bin had the full 999 articles, this number would have been different, we scale this value to $100 \times \frac{999}{842}$ to make it compatible to other bins. The ratios are now comparable between relevance and retrieval, as well as across bins.

3. The bin length is assumed to be the median document length in the bin. The probability of relevance and retrieval figures for each bin can now be plotted on a graph against the corresponding bin length.

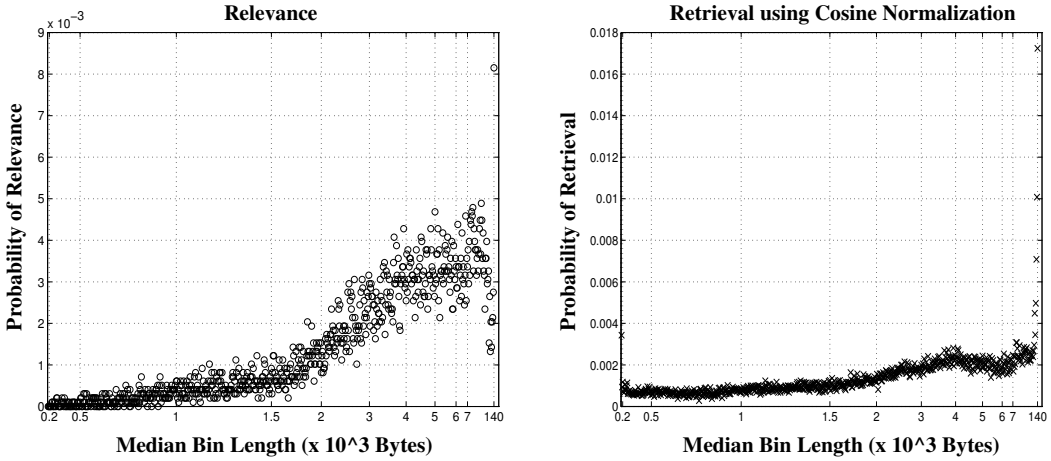
4. Finally for better visibility, we smooth the plots obtained from the above analysis. To smooth the plots, we take a sequence of around 20 bins and represent those by a single point. The representative point is obtained by averaging the median lengths and the probabilities for this group of (20 or so) bins. Once again, to get stable statistics for the last group, we select a number (around 20) so that the last group gets at least 80% of the bins as compared to all the other groups. For example, if we had 713 bins we would group 17 bins into one point which would yield 42 points, the first 41 made up of 17 bins each and the last one made up of 16 bins (94.11% of the regular points).

As an example, consider the analysis for the TREC–3 ad hoc task (documents from disks 1 and 2, and the 50 queries 151–200). To do this analysis, we sorted the 741,856 documents in order of increasing byte-length. We divided this sorted list into bins of one thousand documents each, yielding 742 different bins: the first 741 bins containing one thousand documents each, and the last bin containing the longest 856 documents. We selected the *median* document length in each bin to represent the bin on the graphs used in later analysis.

We took the 9,805 ⟨query, relevant-document⟩ pairs for the fifty test queries, and counted how many pairs had their document from the ith bin. We then computed the probability that a randomly selected relevant document belongs to the ith bin — the ratio of the number of pairs that have their document from the ith bin, and the total number of pairs (9,805). As shown above, this ratio for the ith bin is $P(D \in Bin_i \mid D \textit{ is Relevant})$. For example, in bin 710 there are 39 articles relevant to some query. (If some article is relevant to two different queries, it is counted twice.) Therefore, the relevance ratio for bin 710 is $\frac{39}{9805} = 3.9776 \times 10^{-3}$. Also, the last bin (bin number 742) has 69 relevant articles, but since it has a total of 856 articles (and not 1000), we scale up this number to $69 \times \frac{1000}{856} = 80.61$ and compute the ratio for the last bin ($\frac{80.61}{9805} = 8.2213 \times 10^{-3}$).

Similarly, by retrieving the top one thousand documents for each query (yielding 50,000 ⟨query, retrieved-document⟩ pairs) using a particular retrieval method, and computing the probability that a randomly selected retrieved document belongs to the ith bin — the ratio of the number of pairs that have their retrieved document from the ith bin, and the total number of pairs (50,000) we get the conditional probability of retrieval $P(D \in Bin_i \mid D \textit{ is Retrieved})$ for the normalization function used. (Figure 3.1 was generated similarly using such probability of retrieval computation at three different rank cut-off levels.)

Figures 3.2(a) and 3.2(b) show the plots of the probabilities obtained from the above analysis plotted against the median document length in a bin. Smart's *lnc.ltc* retrieval, which is based upon cosine normalization, was used to get the retrieval probabilities. [BASS95] In Figure 3.3, the smoothed plots for the relevance and the retrieval probabilities are graphed together. We generated smooth plots for various figures by representing a sequence of 24 bins by a single point and connecting these points by a curve. The 742 bins yielded 31 different points where the last point represented the longest 22 bins ($742 = 30 \times 24 + 1 \times 22$). The



(a) (b)

Figure 3.2: Probability of relevance (a) and retrieval (b) in a bin

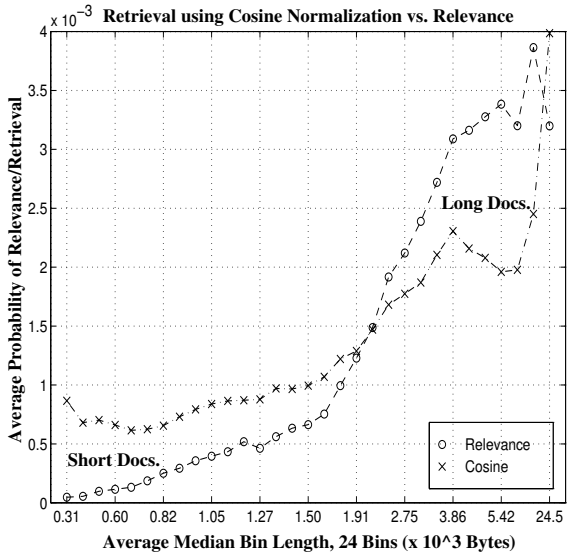


Figure 3.3: Smooth plot for probability of relevance/retrieval

representative point for a group of bins was obtained by taking averages of both the median lengths, and the probabilities of relevance/retrieval for the 24 (22 for the last point) consecutive bins.

Such a comparison reveals important information about the length normalization properties of a term weighting strategy. For example, we can observe from the smoothed plots in Figure 3.3 that *lnc.ltc* retrieval has a tendency to retrieve short documents with a higher probability than their probability of relevance; it is less likely to retrieve longer documents as compared to the likelihood of their relevance. This indicates that *lnc.ltc* is retrieving more short documents than it should and is retrieving fewer long documents. This observation reinforces the long held belief that *cosine normalization tends to favor short documents in retrieval.* When using *lnc.ltc* retrieval, we would like to (somehow) promote the retrieval of longer documents, and we would like to retrieve fewer short documents.

3.3 Comparison Between Systems

To test our hypothesis — a system that retrieves articles of all lengths with chances similar to the chances of their relevance will have a higher retrieval effectiveness than a system that retrieves articles of all lengths with very different chances from their chances of relevance — we compare the length retrieval patterns of several good term weighting formulations to the length pattern for the relevant documents in the TREC–3 ad hoc task (documents from disks 1 and 2, and fifty queries 151–200). [Har95]

Different term weighting approaches have shown promise in the TREC environment. In TREC–3, the three systems with the best base performance (when only term weights and query–document matching is used) were Okapi, INQUERY and Cornell's Smart[1]. These three systems are statistical systems whose term weighting approaches reduce to variations of the $tf \times idf$ weighting approach, each working from a very different basis. Okapi uses a probabilistic technique, based on approximations to the 2–Poisson model to get the document and the query term weights. [RWJ+95] INQUERY is based on the inference network model and uses another probabilistic technique to estimate the importance of a term in a document. [BCCN95] Smart, on the other hand, uses the classical $tf \times idf$ approach to term weighting. [BASS95]

In the TREC–3 ad hoc task, Okapi had the highest performance followed by INQUERY and then by Smart. Since all three systems use some variation of the $tf \times idf$ term weights, one asks what is causing the noticeable differences in their performance? In this study, we explore this question further by focusing on the most significant difference between Smart's term weighting scheme and the term weights used by Okapi and INQUERY, namely Smart's use of the cosine normalization function for document length normalization of term weights. Documents in TREC have large variation in their lengths, and length normalization in term weighting is extremely important in retrieval from the TREC collection. If no document length normalization is used, the average precision is very poor (in the range 0.0090 to 0.0413); whereas when any normalization is used, the average precision is much higher (in the range 0.2245 to 0.3047). [SSB96]

3.3.1 Document Length Dependence of Relevance

Figure 3.4 is the plot obtained from the length analysis of the relevant documents for the TREC–3 data. Figure 3.4(a)[2] shows all the 742 bins and Figure 3.4(b) is a zoomed-in view of the majority of the bins (obtained by removing the outlier bins). From these figure we observe that in TREC, the probability of relevance of a document increases as the documents grow longer. It is plausible that the longer documents contain more information, and are, thereby, potentially more useful to a user, but this phenomenon has never been observed earlier.

It is possible that very small documents containing very few words have too little information to be useful to a user. A text should have some minimum length to be informative. This hypothesis is strongly supported by the study done by Callan on passages. [Cal94] In Table 7 of [Cal94], one notices that the average precision of a passage-based search increases as the size of passages used in the search increases from twenty five words to around three hundred words, suggesting that the twenty five or fifty word passages are not as informative as the two or three hundred word passages. Also, Kwok et al. segment large documents

[1] Various other TREC participants have also used the Smart system for their experiments. In the present study, the term *Smart* invariably refers to Cornell's participation in TREC.

[2] This is same as Figure3.2(a).

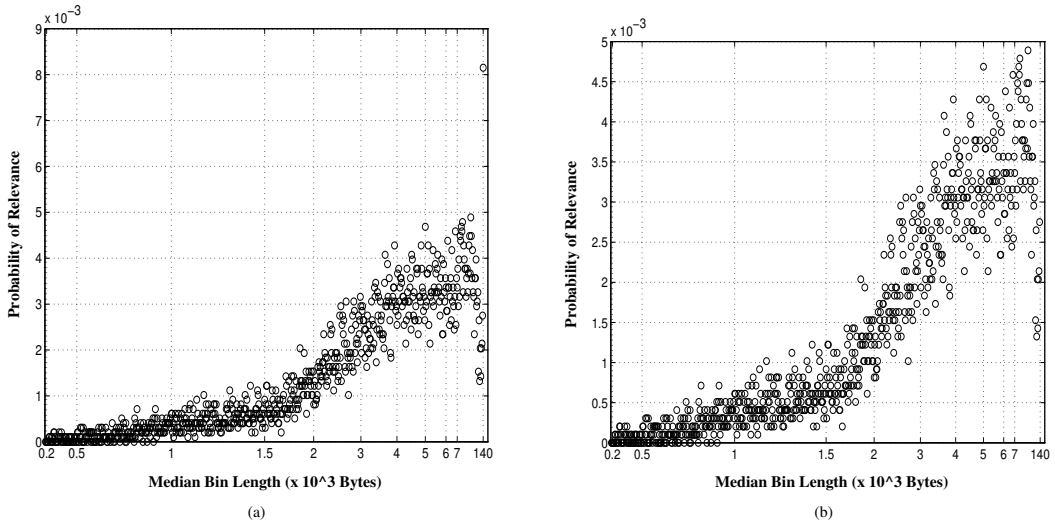


Figure 3.4: Probability of relevance in a bin

into chunks of five hundred and fifty words each to obtain good performance from their system in the TREC collection. [KGL95] Very small chunks did not work as well as the larger chunks. These results support our hypothesis that very small texts are not as informative as longer texts and could, therefore, be less useful to a user. The minimum amount of information needed to satisfy an information need can be termed as an "information unit". To be useful, a text should contain at least this much information.

Several researchers have pointed out that long documents consist of several information units (called *segments*, or *themes*, or *topics*, or *text–tiles*). [SS94,SA94,HP93] Since a single information unit can potentially satisfy a query, the chances of a document being useful to a user-query increase as the number of information units present in a document increases. This can explain the higher probability of relevance that we observe for the longer documents in the TREC collection. If a document addresses several topics or it addresses several aspects of the same topic, it is potentially useful for a large set of queries, as opposed to a small document that deals with just one topic and can be potentially useful to only the queries aimed at that particular topic. These observations suggest that long documents might be judged more useful by users in other full-text collections as well.

3.3.2 Retrieval by Smart

In TREC–3, the Smart group at Cornell used *lnc* weighting for the documents and *ltc* weighting for the queries. These weighting schemes are: [BSA93,BASS95]

Document term weights (W_{di}): $\frac{w_{di}}{\sqrt{w_{d1}{}^2+w_{d2}{}^2+\ldots+w_{dT}{}^2}}$

$w_{di} = 1 + log(tf_i)$

Query term weights (W_{qi}): $\frac{w_{qi}}{\sqrt{w_{q1}{}^2+w_{q2}{}^2+\ldots+w_{qT}{}^2}}$

$w_{qi} = (1 + log(tf_i)) \times log(\frac{N}{n})$

Query document similarity: $\sum_{matching\ terms} W_{dt} \times W_{qt}$

where,

tf_i	is the term frequency of the ith term in the document/query text,
T	is the number of unique terms in the document/query,
N	is the total number of documents in the collection,
n	is the number of documents within the collection in which the term under consideration is present,
W_{dt}	is the weight of matching term t in the document, and
W_{qt}	is the weight of matching term t in the query.

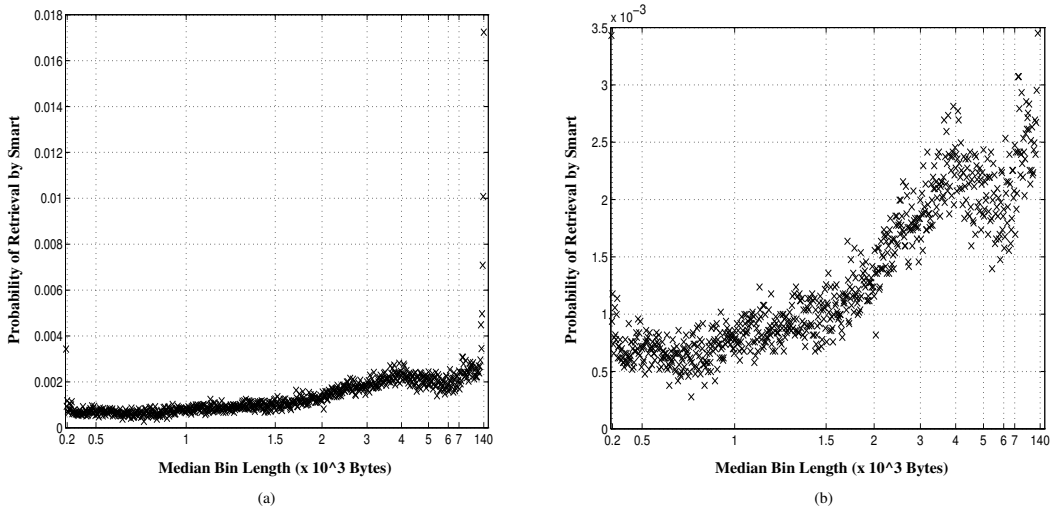


Figure 3.5: Probability of retrieval by Smart

Table 3.1: Our approximation to Okapi's term weighting

	Original Results (from [RWJ+95])	Our Approximation to Okapi	Our Approximation Compared to the Original Results
	Non-Interpolated Avg. Precision		
Smart	0.2842	0.2842	+ 2.6%
Okapi	0.3370	0.3456	
Improvement	+18.6%	+21.6%	

The expression $\sqrt{{w_1}^2 + {w_2}^2 + \ldots + {w_T}^2}$ is the cosine normalization factor.

To study the document length dependence of the probability of retrieval by the Smart system, we retrieved the top one thousand documents for every query and plotted the retrieval probability from a bin against the median length of the bin. Figure 3.5 is the plot obtained by this length analysis. Once again, Figure 3.5(a)[3] shows all the 742 bins and Figure 3.5(b) is a zoomed-in view of the majority of the bins. From Figures 3.4 and 3.5 we observe that with increasing document length, the increase in the probability of relevance is steeper than the increase in the probability of retrieval by the Smart system, which indicates that the length pattern for the retrieved documents does not match well with the relevance pattern. This observation was studied in Figure 3.3 where the smooth plots for the relevance and the retrieval probabilities were compared, and it was observed that cosine normalization retrieves too many short documents and too few long documents.

3.3.3 Term Weighting of Okapi and INQUERY

To test the document length retrieval pattern of the term weighting strategies of the Okapi system and the INQUERY system, we approximated their term weighting schemes within the Smart system[4]. We experimentally selected appropriate values for the parameters involved in Okapi's and INQUERY's term weighting schemes to yield good results. [SSMB96]

Our approximation to Okapi's term weighting scheme is: [RWJ+95]

[3] This is same as Figure3.2(b).

[4] In fact, the INQUERY system uses structured queries which are not directly implemented in the Smart system. Our *approximation* of the INQUERY system reduces to using only the *weighted sum* and the *phrase* operators in INQUERY.

Table 3.2: Our approximation to INQUERY's term weighting

	Original Results (from [BCCN95])	Our Approximation to INQUERY	Our Approximation Compared to the Original Results
	11-pt. Avg. Precision		
Smart	0.3057	0.3057	+ 4.7%
INQUERY	0.3180	0.3330	
Improvement	+ 4.0%	+ 8.9%	
	Non-Interpolated Avg. Precision		
Smart	0.2842	0.2842	
INQUERY	–	0.3118	
Improvement		+ 9.7%	

Document term weights (W_d): $\frac{tf \times log(\frac{N-n+0.5}{n+0.5})}{2 \times (0.25 + 0.75 \times \frac{dl}{avdl}) + tf}$

Query term weights (W_q): tf

Query document similarity: $\sum_{matching\ terms} W_{dt} \times W_{qt}$

where

tf	is the term frequency of a term in the document/query text,
N	is the total number of documents in the collection,
n	is the number of documents in the collection in which the term under consideration is present,
dl	is the length of the document (in bytes), and
avdl	is the average document length in the collection (in bytes).

Table 3.1 shows that this approximation to Okapi's term weighting scheme is good. It yields results that are 2.6% better than the original results reported by Robertson et al. [RWJ+95]

Our approximation to INQUERY's term weighting scheme is: [BCCN95]

Document term weights (W_{di}):

$$0.4 + 0.6 \times (0.4 \times H + 0.6 \times \frac{log(tf + 0.5)}{log(max_{tf} + 1.0)}) \times \frac{log(\frac{N}{n})}{log(N)}$$

Query term weights (W_q): tf

Query document similarity: $\sum_{matching\ terms} W_{dt} \times W_{qt}$

where,

tf	is the term frequency of a term in the document/query text,
max_{tf}	is the maximum term frequency for any term in that document,
N	is the total number of documents in the collection,
n	is the number of documents within the collection in which the term under consideration is present, and
H	= 1.0 if $max_{tf} \leq 25$ = $\frac{25}{max_{tf}}$ otherwise.

Table 3.2 shows that this approximation to INQUERY's term weighting is also reasonable. In fact, it yields results that are 4.7% better than the original results reported by Broglio et al. [BCCN95] As TREC evaluations use the non-interpolated average precision values, which were not reported for INQUERY's base run in [BCCN95], we have also listed this value for our approximation of INQUERY's weighting scheme in Table 3.2.

Table 3.3 compares the results from our approximations to the weighting schemes of the other two systems to the original results reported by Okapi and INQUERY. These results show that our results do follow the TREC ranking of the three systems — Okapi followed by INQUERY followed by Smart. The results from

Table 3.3: Comparison between our approximations and the original results

	Smart	INQUERY	Okapi
Original Results	0.2842	–	0.3370 (+18.6%)
Original Rank	III	II	I
Our Approximation	0.2842	0.3118 (+ 9.7%)	0.3456 (+21.6%)
Our Rank	III	II	I

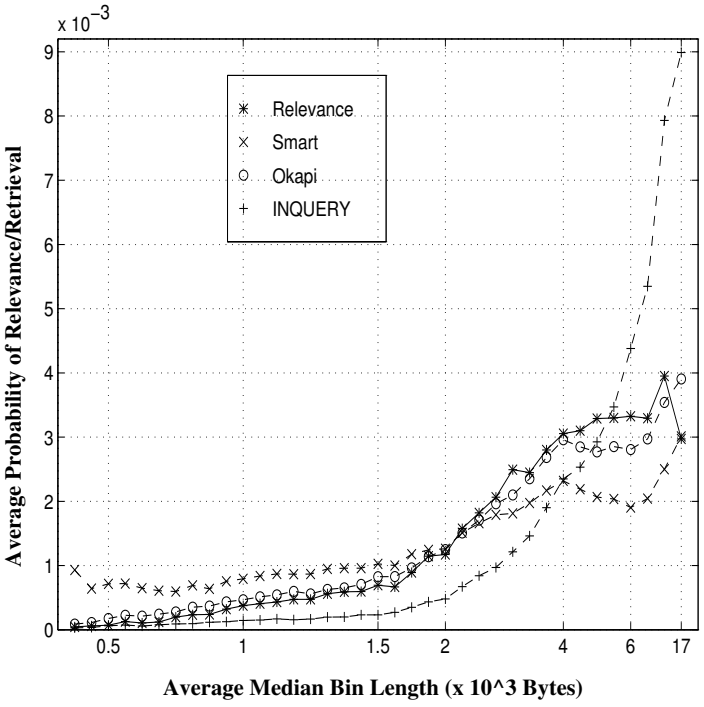
Tables 3.1, 3.2, and 3.3 confirm that the length analysis obtained from our approximations to Okapi's and INQUERY's term weighting will correctly reflect the document length retrieval patterns for the Okapi system and the INQUERY system.

3.3.4 Length Analysis

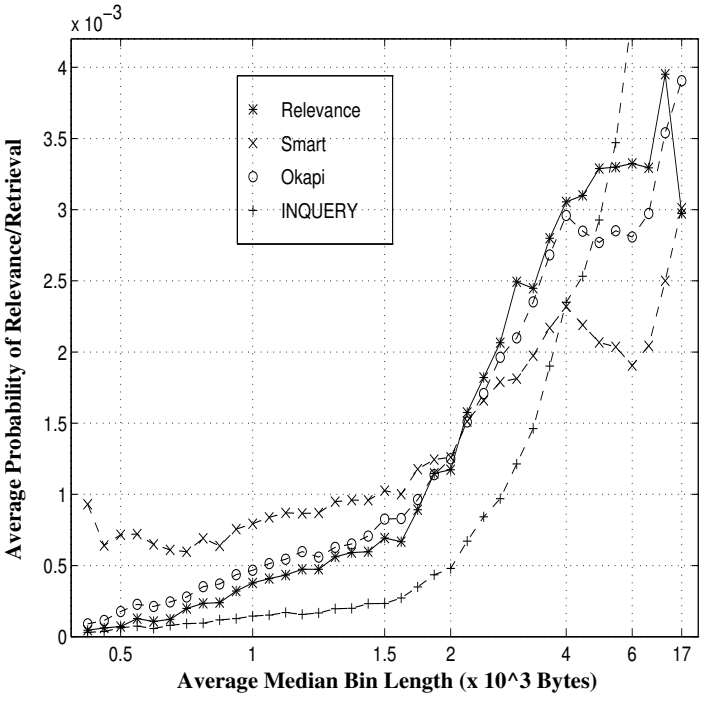
We retrieved the top one thousand documents for every query using our approximations to Okapi's term weighting and INQUERY's weighting scheme. We performed the probability analysis for these retrievals and generated the smooth plots for the probability of retrieval by the three systems and compared these plots to the smooth plot for the probability of relevance. We observe that for documents of a certain length, the probability of retrieval by Okapi and INQUERY was closer to the relevance probability, as compared to the probability of retrieval by Smart. This effect is illustrated in Figure 3.6, where the smoothed plots for retrieval using weighting schemes of all three systems are plotted together with the smoothed plot for relevance, obtained from Figure 3.4. Once again, Figure 3.6(a) shows all the smoothed points and Figure 3.6(b) shows the majority of those points.

Figure 3.6 shows that the probability of retrieving a document of a certain length using the weighting scheme of the Okapi system is very close to the probability of finding a relevant document of about the same length. We have also seen that the performance of Okapi's term weighting is significantly better than that of the other two weighting schemes. We believe that this strong correlation between Okapi's length retrieval pattern and the length pattern of the relevant documents is the main reason behind the superior retrieval effectiveness of Okapi's term weighting scheme.

Figure 3.6 shows that the system whose retrieval probabilities are close to the relevance probabilities for documents of all lengths has a higher retrieval effectiveness that the systems whose retrieval probabilities are different from the relevance probabilities. This is partial evidence in support of our hypothesis. The next chapter will show that if the Smart system is modified and its retrieval probabilities are matched to the relevance probabilities, then its retrieval effectiveness also increases and is comparable to the best system, *i.e.*, the Okapi system. This step will fully confirm our hypothesis that we should retrieve documents in accordance with their chances of relevance.



(a)



(b)

Figure 3.6: Smoothed plots for the three systems

Chapter 4

Pivoted Normalization

In Chapter 3 we hypothesize that normalization schemes that retrieve documents of all lengths with the same chances as their chances of relevance, will have better retrieval effectiveness than normalization schemes which retrieve documents of various lengths with very different chances from their chances of relevance. Based on this hypothesis, we would like to modify our retrieval strategy to retrieve documents of all lengths with the same chances as their chances of relevance.

This chapter introduces *pivoted normalization*, a technique that can be used to modify any normalization function and reduce the gap between the relevance and the retrieval probabilities. We show that the retrieval probabilities for a particular normalization method deviate systematically from the relevance probabilities, across different collections. Training pivoted normalization on one collection, we can successfully use it on other (new) text collections, yielding a robust, *collection independent* normalization technique. We use the idea of pivoting with the well known cosine normalization function, and compare the results to two other term weighting schemes from Chapter 3. We point out some shortcomings of the cosine function and present a new normalization function — *pivoted unique normalization*.

4.1 Pivoting

The length analysis technique developed in Chapter 3 can be used to compare the retrieval probabilities and the relevance probabilities for documents of all length. The result of such an analysis are smooth probability curves for probability of relevance and probability of retrieval of documents of different length, as shown in Figure 4.1 (which is just a reproduction of Figure 3.3). We observe from Figure 4.1 that the particular normalization scheme being used in this retrieval (which happens to be cosine normalization) is retrieving documents shorter than about 2,000 bytes with a higher probability than their probability of relevance. Also, it is retrieving documents longer than about 2,000 bytes with lower probability than their probability of relevance (except for the last point that represents the bins containing longest documents in the collection). For this normalization scheme, we would like to (somehow) promote the retrieval of longer documents, and we would like to retrieve fewer short documents to bring the retrieval curve closer to the relevance curve.

Using the inner product similarity formulation, the query–document similarity values are directly related to the individual term weights (see Section 2.1.1). If the individual term weights are high, then the similarity value will also be high; if the individual term weights are low, then the similarity value will also be low. The term weights in a document are inversely proportional to the document length normalization factor used for that document (the normalization factor is used in the denominator of the term weighting formula, see Section 2.2.3). The higher the value of the normalization factor for a document is, the lower are its term weights. Lower term weights, in turn, reduce the query–document similarity and, thus, the chances of retrieval for that document. In effect, the probability of retrieval of a document is inversely related to the normalization factor used in the term weight estimation for that document. This relationship suggests that to boost the chances of retrieval for documents of a certain length, we should lower the value of the normalization factor for those documents, and vice-versa. The pivoted normalization scheme is based on this principle.

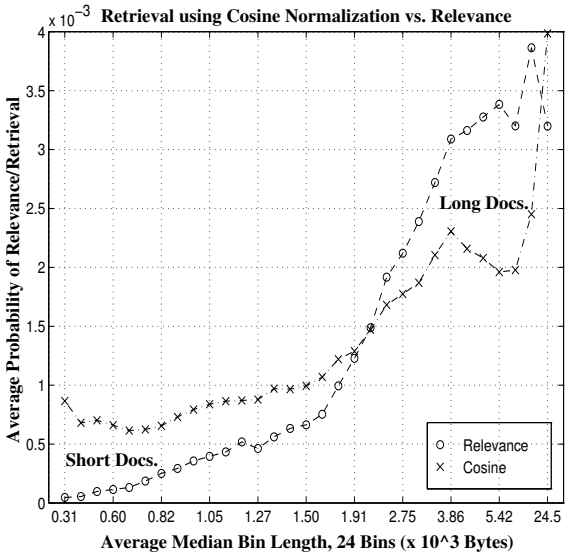


Figure 4.1: Smooth plot for probability of relevance/retrieval

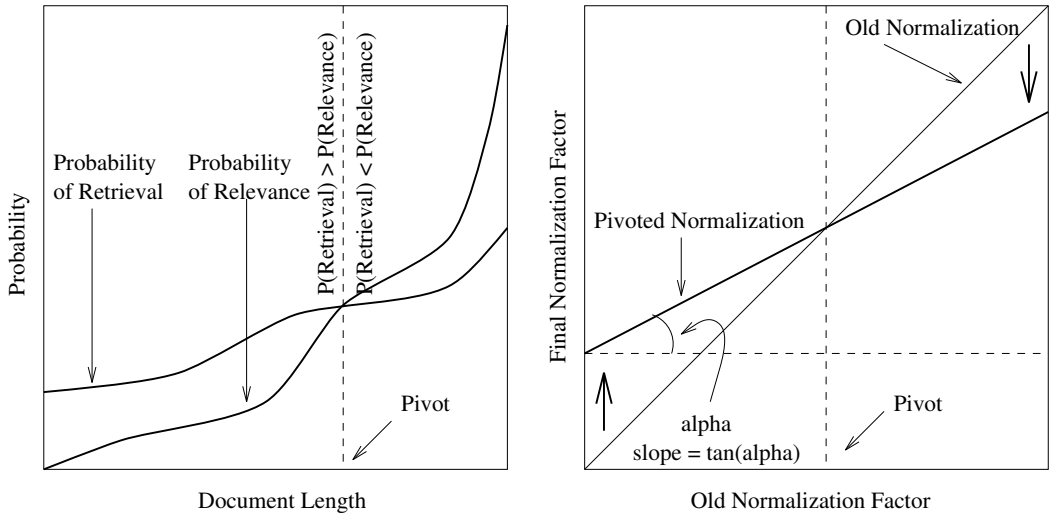


Figure 4.2: Pivoted Normalization

The basic idea of pivoted normalization is illustrated in Figure 4.2. Using a normalization function (like cosine, or byte-size), a set of documents is initially retrieved. As shown in Figure 4.1, the retrieval and the relevance curves are plotted. The point where these two curves cross each–other is called the *pivot*. The documents on one side of the pivot are generally retrieved with a higher probability than their relevance probability, and the documents on the other side of the pivot are retrieved with a lower probability than their probability of relevance. The normalization function can now be "pivoted" at the pivot and "tilted" to increase the value of the normalization factor, as compared to the original normalization factor, on one side of the pivot. This also decreases the value of the normalization factor on the other side of the pivot. The amount of "tilting" needed becomes a parameter of the weighting scheme, and is called the *slope*. With such pivoting and tilting, the pivoted normalization factor is represented by the equation for a line of gradient *slope* that intersects the line of unit gradient at the point *pivot*. [SSMB96]

$$pivoted\ \ normalization = (1.0 - slope) \times pivot + slope \times old\ \ normalization \tag{4.1}$$

The pivoted normalization scheme involves two parameters — the pivot and the slope. If the deviation of the retrieval pattern from the relevance pattern for a given normalization scheme is systematic across collections for the normalization function, the pivot and the slope values learned from one collection can be used effectively on another collection. [SBM96]

4.1.1 Removing One Parameter

Using pivoted normalization, the new weight of a document term can be written as:

$$\frac{tf \cdot idf\ \ weight}{(1.0 - slope) \times pivot + slope \times old\ \ normalization} \tag{4.2}$$

If we multiply every term weight for every document in the collection by a constant, the relative ranking of the documents under inner-product similarity measurement remains unchanged as individual document similarities for all documents are simply scaled by the constant. Multiplying each weight by the constant $(1.0 - slope) \times pivot$, we obtain the following term weighting formula:

$$\frac{tf \cdot idf\ \ weight \times (1.0 - slope) \times pivot}{(1.0 - slope) \times pivot + slope \times old\ \ normalization}$$

or

$$\frac{tf \cdot idf\ \ weight}{1 + \frac{slope}{(1.0 - slope) \times pivot} \times old\ \ normalization}$$

We observe that the form of the pivoted normalization function is $1 + c \times old\ \ normalization$, where the constant c equals $\frac{slope}{(1.0 - slope) \times pivot}$. If the pivot value in an optimal constant c is changed to $pivot'$, the slope value can be modified to $slope'$ to get back the optimal constant. If we fix the pivot value at some collection specific value, like the *average old normalization factor*, it is still possible to obtain an optimal slope value by training. Therefore, the number of parameters that we need to train for is reduced to just one instead of two.

Selecting the *average old normalization factor* as the pivot has a nice interpretation. If instead of multiplying every term weight by $(1.0 - slope) \times pivot$ in Equation 4.2, we multiply every weight by the constant *pivot* (which has the value *average old normalization*), the final normalization factor reduces to:

$$(1.0 - slope) + slope \times \frac{old\ \ normalization}{average\ \ old\ \ normalization} \tag{4.3}$$

From this expression, similar to Robertson's notion, [RW94] we can say that an *average length document* in a collection is of "appropriate length" and its weights should remain unchanged, *i.e.*, it should get *unit* (or no) normalization. Also, the slope can be interpreted as our "belief in normalization".

As the slope now is interpreted as our belief in normalization, the higher the slope value is, the stronger is the degree of normalization for a weighting scheme, and vice-versa. For example, if we use a slope value of 0.0, from Equation 4.3, we find that irrespective of the document length, all documents get unit (or no)

Table 4.1: Effectiveness of pivoted cosine normalization

	Cosine	Pivoted Cosine Normalization			
		Slope			
		0.65	0.70	**0.75**	0.80
# Relevant Retrieved (out of 9,805)	6,526	6,458	6,574	**6,629**	6,671
Average Precision	0.2842	0.3097	0.3144	**0.3171**	0.3162
Improvement	—	+ 9.0%	+10.7%	**+11.7%**	+11.3%

normalization. Therefore, a slope of 0.0 corresponds to the weakest possible normalization, *i.e.*, no length normalization at all. On the other hand, if we use a slope of 1.0, the final normalization factor is the same as the old normalization scheme being used (it is easier to observe this from Equation 4.2). If a slope value between 0.0 and 1.0 is used, the resulting pivoted normalization is a *weaker* form of normalization than the old normalization and will favor long documents as compared to the old normalization. Whereas if a slope value greater than 1.0 is used, the resulting pivoted normalization is *stronger* than the old normalization and will favor shorter documents as compared to the old normalization.

In Section 3.3.1 we discussed how very short texts might not be as informative as longer texts, and how a text should at least contain an "information unit" to be useful. The pivot in the above discussion might also reflect the presence of such an information unit. An alternative interpretation of the pivot can be: *the size of the text that might contain a full information unit.* Texts longer than the pivot size might contain more than one information units. In a given corpus (or a domain), it is possible that a text of average document size is of appropriate length and contains a single information unit (within that domain). This can explain the better performance obtained consistently across collections by the use of the average document length in a collection as the pivot.

4.2 Pivoted Cosine Normalization

Since cosine normalization is most commonly used in the vector space model, it is natural to test pivoting with the cosine function first. To study the effects of pivoting on cosine normalization, we will use the same data set that we used in Chapter 3 to compare systems — namely data for the TREC–3 ad-hoc task (documents from disks 1 and 2, and fifty queries 151–200). In our studies with the TREC collection, a *tf* factor of $1 + log(tf)$ works well for this collection. Also, the *idf* factor is only used in the query term weights and not in the document term weights. [BSA93,BAS94,BASS95] Fixing the pivot value at the average cosine factor for $1 + log(tf)$ weighted documents for TREC disks 1 and 2 (average = 13.36), we retrospectively learn the value of a good slope for TREC queries 151-200 (see Table 4.1). In Table 4.1, each entry shows the total number of relevant documents retrieved for the entire set of the fifty queries (out of a total of 9,805 relevant articles) when the top 1,000 documents are retrieved for every query, and the average non-interpolated precision value. Substantial improvements over cosine normalization — 9–12% improvement in average precision — are obtained using pivoted cosine normalization.

With a slope of 0.75, which is gentler than the full slope of 1.0, the pivoted cosine normalization function provides weaker normalization than the original cosine normalization. Long documents that had their cosine normalization factor greater than the pivot, *i.e.* 13.36, now get a normalization which is lower than their original normalization; and short documents that had their cosine normalization factor lower than the pivot now get a normalization which is higher than their original normalization. This should increase the term weights for the longer documents and decreases the term weights for the shorter documents (as compared to the original term weights). This weight modification, in turn, should increase the similarities for the longer documents and retrieve more long documents than before. It should also decrease the similarities for the shorter documents and retrieve fewer short documents than before. Based on Figure 4.1, this is exactly what we desire.

With the use of pivoting with cosine normalization, the improvements in the average precision are en-

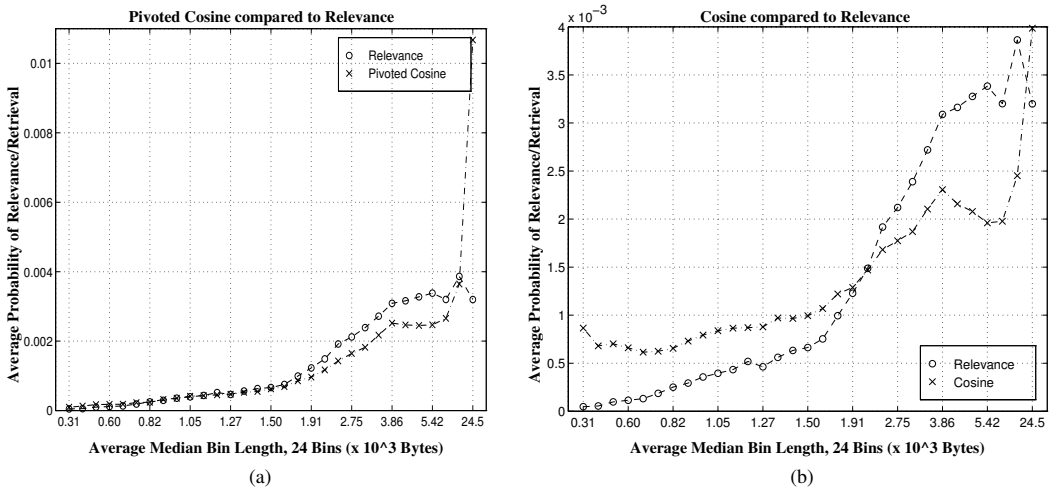


Figure 4.3: Comparison between pivoted cosine and cosine normalization

Table 4.2: Pivoted cosine normalization for TREC queries 1–150

	Cosine	Pivoted Cosine Normalization			
		Slope			
		0.65	**0.70**	0.75	0.80
# Relevant Retrieved (out of 46,555)	28,484	30,270	**30,407**	30,314	30,119
Average Precision	0.3063	0.3427	**0.3427**	0.3411	0.3375
Improvement	—	+11.9%	**+11.9%**	+11.4%	+10.2%

Table 4.3: Pivoted-cosine normalization compared to Okapi and INQUERY

	Cosine Normalization	Pivoted-Cosine Normalization
	Non-Interpolated Average Precision	
Smart	0.2842	0.3171
Okapi	0.3370	0.3370
Improvement	+18.6%	+ 6.2%
	11-pt. Average Precision	
Smart	0.3057	0.3386
INQUERY	0.3180	0.3180
Improvement	+ 4.0%	– 6.1%

couraging. To study the effects of pivoting on the retrieval probabilities of documents of different length, we performed the length analysis on documents retrieved using the pivoted cosine normalization. Figure 4.3(a) compares the retrieval pattern for pivoted cosine normalization to the relevance pattern for TREC queries 151–200 and documents from disks 1 and 2. For comparison with cosine normalization, Figure 4.1 has been reproduced here as Figure 4.3(b). We observe that the curve for the retrieval probabilities using pivoted cosine normalization is much closer to the curve for the relevance probabilities as compared to the curve for retrieval using cosine normalization. This indicates that in comparison to cosine normalization, pivoted cosine normalization retrieves documents of all lengths with chances much closer to their likelihood of relevance. This observation along with the **11.7%** improvement obtained over cosine normalization strongly supports our hypothesis that schemes that retrieve documents of different lengths with chances similar to their likelihood of relevance will have higher retrieval effectiveness.

To test the robustness of pivoted cosine normalization, we tested it on another 150 TREC queries (numbered 1–150). The training for slope for TREC queries 1–150 is shown in Table 4.2. We observe that a slope of 0.7 works well for queries 1–150, *i.e.*, once again we need a weaker normalization than cosine normalization. Also, with weaker degree of normalization by pivoted cosine normalization, once again we obtain 10–12% improvements in average precision over cosine normalization.

As relevance judgments are not available in an ad-hoc querying environment, to observe the variability in a good slope value across query sets, we also tested the optimal slope value obtained from a set of training queries (TREC queries 1–150) on a set of test queries (TREC queries 151–200). We observe from Table 4.2 that the best slope value for queries 1–150 is 0.70. If we use this slope value for queries 151–200, we would still achieve "near best" performance — 10.7% improvement in place of 11.7% (see Table 4.1). This indicates that it is possible to learn the slope value on one set of queries and successfully use it on another.

4.2.1 Comparison to Okapi and INQUERY

Continuing our study from Chapter 3, we compare the performance of the Smart system using pivoted-cosine normalization to the results reported by Okapi and INQUERY in their TREC–3 participation. Table 4.3 shows that when Smart's term weighting scheme is modified using pivoting to retrieve documents of all lengths with similar chances as their chances of relevance, its retrieval effectiveness improves substantially, and is now at par with these two systems (a bit better than INQUERY, and slightly worse than Okapi).

These results suggest that the earlier performance differences between the Smart system and the other two systems were mainly due to poor document length normalization by Smart's use of the cosine normalization function. The term weighting function used by the Okapi group in their TREC–3 participation was obtained by using four different parameter values, called k_1, k_2, k_3, and b, in their general parameterized term weighting formula BM25. The values 2.0, 0.0, ∞, and 0.75 were used for k_1, k_2, k_3, and b, respectively. This set of numbers was obtained by trying different values using the training queries available before TREC–3 (1–150), and selecting the set of parameters that yielded the best results. [RWJ+95] We believe that by learning the parameters by using the training queries, Okapi implicitly learned a good normalization function that would retrieve documents of all lengths with chances similar to their chances of relevance. This resulted in a

Table 4.4: Characteristics of six test collections

Coll.	# of	Document Length (Bytes)					# of	Average
	Docs.	Min.	Max.	Mean	Median	Std. Dev.	Queries	Rel/Qry
AP	164,597	189	16,495	3,130.65	2,961	1,535.15	196	99.79
DOE	226,087	59	2,187	852.43	792	384.99	80	29.40
FR	45,820	94	2,637,276	10,649.42	3,425	38,684.15	111	16.35
WSJ	173,252	67	136,041	3,077.58	1,845	2,970.12	200	104.91
ZIFF	132,100	312	367,687	3,315.68	1,585	5,332.86	122	95.51
TREC	741,856	59	2,637,276	2,924.89	1,499	10,287.41	200	281.80

Table 4.5: Statistics on cosine factors for test collections

Coll.	Cosine Normalization Factor 1+log(tf) weighted document vectors				
	Min.	Max.	Mean	Median	Std. Dev.
AP	1	36.12	15.89	16.02	4.38
DOE	1	15.89	8.51	8.43	2.29
FR	1.41	255.21	21.25	16.96	14.71
WSJ	1	74.12	14.59	12.52	7.14
ZIFF	2.62	138.32	13.98	10.53	7.98
TREC	1	255.21	13.36	11.14	7.44

superior performance by Okapi at TREC–3. When the Smart system is trained to incorporate this property in its retrieval, its performance becomes comparable with the other systems.

4.2.2 Pivoted Cosine on Six Different Collections

To test the effectiveness of pivoted cosine normalization on collections other than the full TREC collection, we experimented with all different sub-collections of TREC disks 1 and 2 (source dependent). Table 4.4 shows some characteristics of the entire TREC collection, as well as the various sub-collections — AP, DOE, FR, WSJ, and ZIFF. Table 4.4 also shows the number of queries that have at least one document in the sub-collection marked relevant for the query, and the average number of relevant documents per query.

It can be observed from Table 4.4 that the various sub-collections of TREC have very different characteristics. For example, the average document length for the FR collection is 10,649 bytes (quite long), whereas the average document length for DOE documents is only 852 bytes (quite short). We also notice that there is a significant variation in the lengths of the FR documents (94 bytes to 2,637,276 bytes). This is also reflected in the large standard deviation in the lengths of the FR documents (38,684.15 bytes). On the other hand, the DOE collection has opposite characteristics — little variation in the document lengths. Overall, these five sub-collections along with the full TREC collection form a varied group of test collections. If a technique works well on all the six collections, it should, in general, work well.

Table 4.5 shows some statistics on the cosine normalization factors for the $1 + log(tf)$ weighted document vectors for the six test collections. We observe that the cosine normalization factors for the documents show the same variability as the byte sizes in Table 4.4. *E.g.*, the high mean value for the FR collection indicates that FR documents are generally long; whereas the low mean cosine factor value for DOE indicates that DOE documents are generally short. Once again we can notice the high standard deviation for the FR collection and low standard deviation for DOE, indicating large length variation in FR and little length variation in DOE.

Fixing the pivot value at the average cosine factor for a collection, we retrospectively learn the best slope value for that collection. The best slope values and the corresponding improvements in the non-interpolated average precision are listed in Table 4.6. We observe that pivoted cosine normalization consistently performs

Table 4.6: Performance of pivoted cosine on six different collections

Collection	AP	DOE	FR	WSJ	ZIFF	TREC
# of Queries	196	80	111	200	122	200
Avg. Prec. Cosine	0.4000	0.3046	0.2931	0.3899	0.3441	0.3357
Best Slope	0.7	0.8	0.5	0.7	0.7	0.7
Pivoted-Cosine	0.4173	0.3336	0.2931	0.3899	0.3441	0.3357
(vs. Cosine)	**+ 4.3%**	+ 9.5%	+26.6%	**+10.6%**	**+21.7%**	**+11.6%**
Slope = 0.7	–	0.3211	0.2785	–	–	–
(vs. Cosine)	–	**+ 5.4%**	**+20.3%**	–	–	–
(vs. Best Slope)	–	– 3.7%	– 5.0%	–	–	–

better than cosine normalization yielding substantial improvements (4.3% to 26.6%) on all six collections. The best slope values support our earlier observation that cosine normalization favors short documents in retrieval. On all the test collections, the best slope value is anywhere between 0.5 and 0.8. As discussed in Section 4.1, with slope less than 1.0, pivoted cosine normalization is a weaker form of normalization than cosine normalization. With a weaker normalization, we favor long documents in retrieval, removing the deficiency of cosine normalization.

A deeper analysis of the results for various test collections is shown in Figures 4.4 and 4.5. Figure 4.4 shows the length analysis for the relevant documents, the comparison between the probability of retrieval using cosine and the probability of relevance, and the comparison between the probability of retrieval using pivoted-cosine and the probability of relevance for three test collections — AP, DOE, and FR. Figure 4.5 shows the same analysis for the WSJ, ZIFF, and TREC test collections. The six test collections have very different "length pattern" (probability distribution based on length) for the relevant documents. For example, the very high probabilities in the last few bins, that contain the longest documents for a collection, for the FR (Figure 4.4) and the ZIFF (Figure 4.5) collections indicate that very long documents in those collections have high chances of being relevant to a query, whereas the other (shorter) documents have a low likelihood of relevance.

The "Cosine vs. Relevance" plots in Figures 4.4 and 4.5 compare the probability of retrieval using cosine normalization to the probability of relevance for documents of various lengths. We observe from these plots that, irrespective of how relevance is distributed among documents of different lengths, cosine normalization always retrieves short documents with chances higher than their chances of relevance. It also retrieves long documents with chances lower than their likelihood of relevance. Once again, this analysis validates the fact that cosine normalization favors short documents in retrieval, and it is "too strong" for long documents.

The comparison between the relevance probabilities and the retrieval probabilities using pivoted cosine normalization in Figures 4.4 and 4.5 explains why using pivoted cosine normalization is always better than using cosine normalization. Using pivoted cosine normalization, the retrieval curves in Figures 4.4 and 4.5 are much closer to the relevance curves as compared to the retrieval curves for cosine normalization. *I.e.*, the probabilities of retrieval for documents of all length is much closer to the relevance probabilities for those documents. These observations combined with the improvements in Table 4.6 further show that if documents of all length are retrieved in accordance with their probability of relevance, the retrieval effectiveness improves.

With the use of the average cosine factor as the pivot, the only parameter involved in the pivoted cosine normalization scheme is the slope. Different collections have relevant documents with different length distributions. For example, we observe in Table 4.6 that, as compared to the rest of the collections, we need to use weaker normalization for the FR collection (a lower slope of 0.5 as opposed to 0.7 for most other collections) because very long documents are relevant to queries with a high chances. As training data is not always available to learn a good initial slope value based on the distribution of relevance among documents of various lengths, we would like to predict an initial slope value that should be globally used irrespective of the collection.

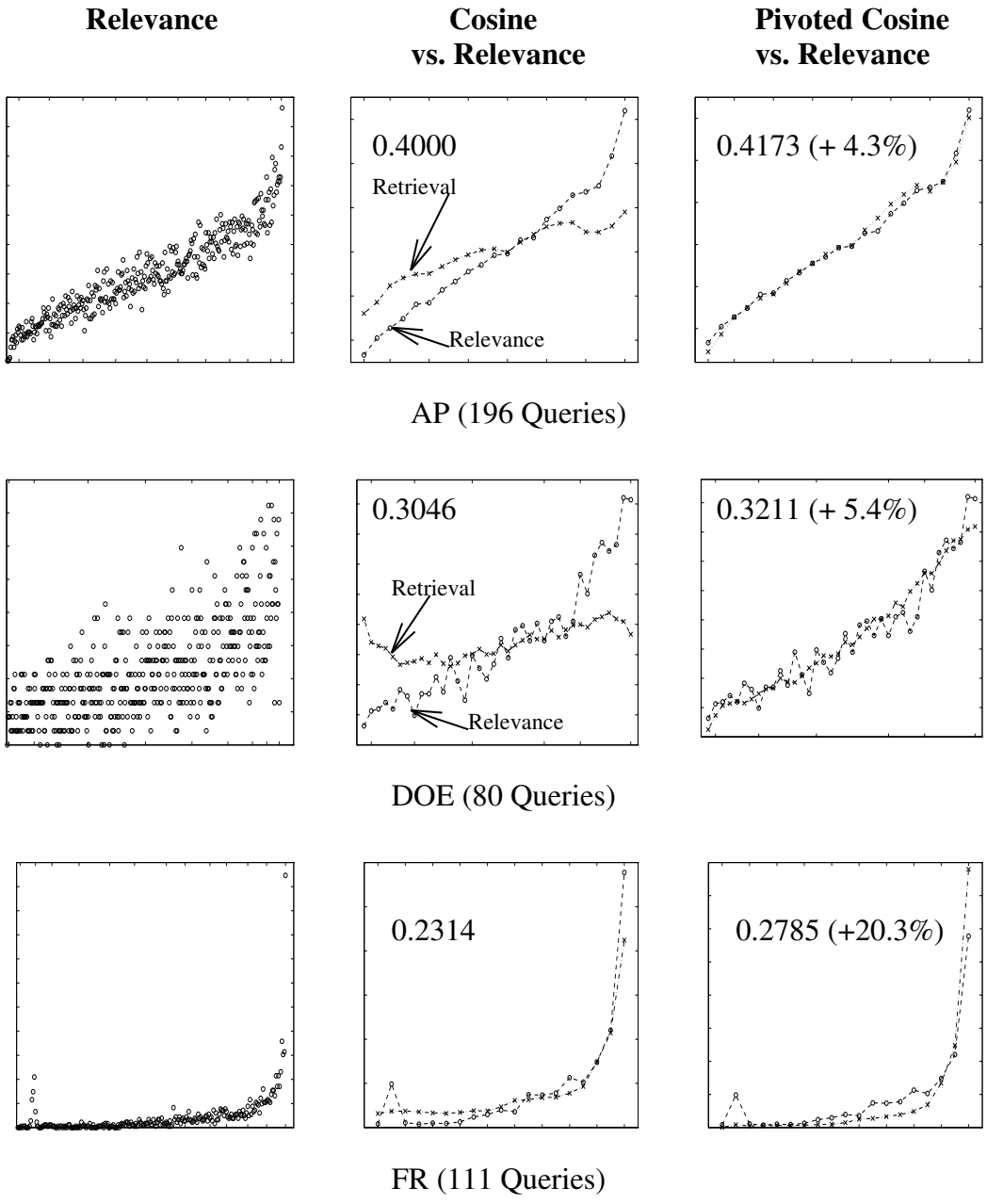


Figure 4.4: Probability of relevance/retrieval for AP, DOE, FR

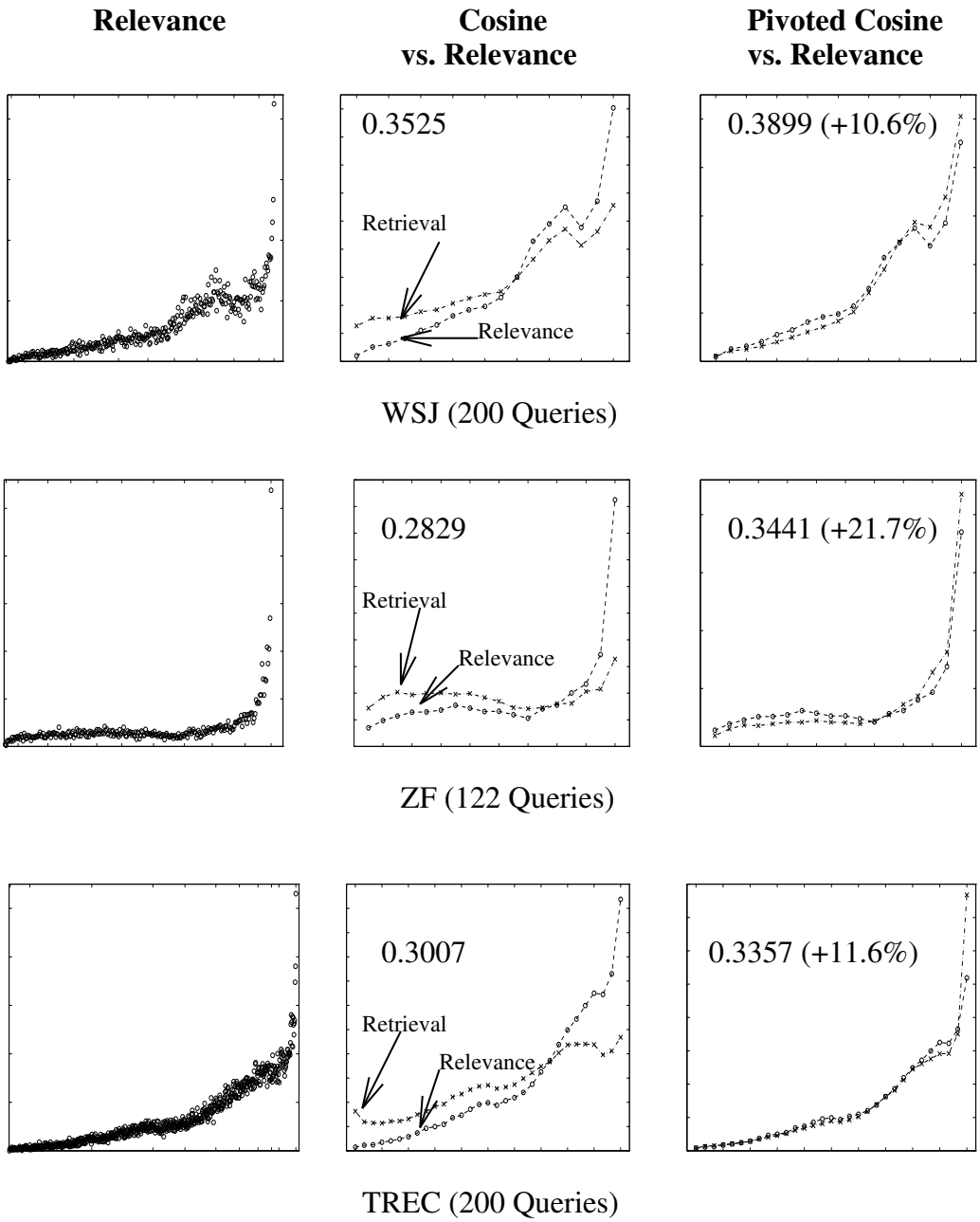


Figure 4.5: Probability of relevance/retrieval for WSJ, ZIFF, TREC

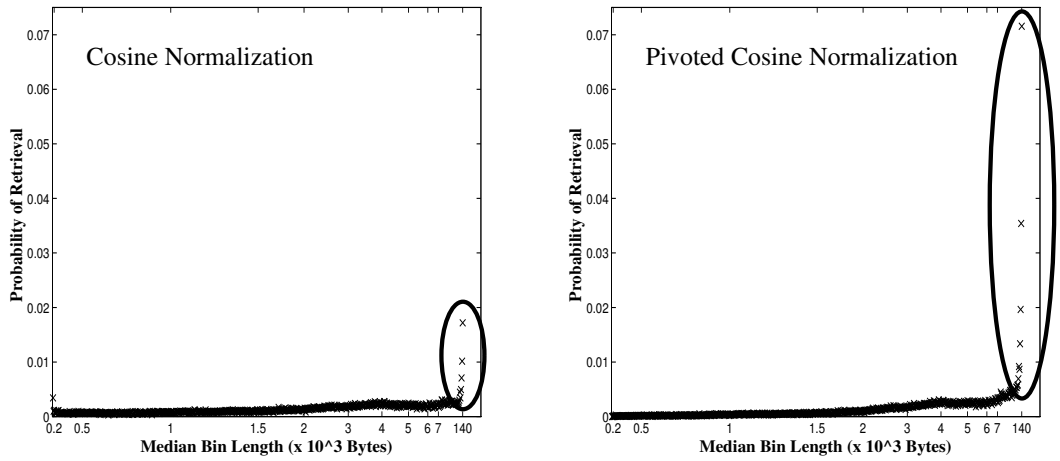


Figure 4.6: Retrieval probabilities using cosine and pivoted cosine normalization

Table 4.6 also shows the effect of using a slope value of 0.7 globally across all test collections. The value of 0.7 was selected based upon its high effectiveness on majority of the collections. We observe that for the two collections for which the value of 0.7 is non-optimal (DOE and FR), respectable improvements over cosine normalization (5.4% and 20.3%, respectively) are still obtained if a slope value of 0.7 was used. Also, the losses in retrieval effectiveness as compared to the best slope values (0.8 for DOE and 0.5 for FR) are marginal (3.7% and 5.0%, respectively). This shows that in absence of any training data, one can use a slope value of 0.7 and can still do "near best" normalization. As more training data becomes available, a better slope value with respect to the particular collection at hand can be learned.

The results from Tables 4.1, 4.2, and 4.6 also show that minor variations in the slope value do not decrease the retrieval effectiveness significantly. For all collections, changing the slope to other values near 0.70 works quite well. This indicates that pivoted normalization schemes are not sensitive to minor changes in the optimal parameter (slope) settings. Such robustness is required for these schemes to be applicable to new collections where it is not possible to "learn" a good slope value retrospectively.

The performance of pivoted cosine normalization scheme on multiple test collections is very encouraging. Pivoting the cosine normalization function substantially improves the retrieval effectiveness on all six test collection. Our probability curves indicate that the use of cosine normalization invariably favors short documents and retrieves fewer long documents. This behavior is a characteristics of the cosine functions and is consistent across test collections. Also, the degree to which we should weaken the cosine function (to get good normalization) is consistent across collections; a slope of 0.7 works well for all collections. This indicates that pivoted cosine normalization is removing the deficiency in the basic retrieval properties of the cosine normalization scheme. As the retrieval properties of a normalization scheme remain consistent across collections, the slope value for pivoted normalization learned from one test collection can be successfully used on other (new) collections for better document length normalization.

4.3 Analysis of the Cosine Function

Figure 4.6 shows the probability of retrieval from various bins for TREC queries 151–200 on documents from TREC disks 1 and 2 using cosine normalization (Figure 4.6(a)) and pivoted cosine normalization (Figure 4.6(b)), on the same scale. On close observation of Figure 4.6(a) we notice that when cosine normalization is used, the probability of retrieval for the documents in the last few bins (the "extremely" long documents) is substantially higher than the rest of the collection. The last few bins contain documents that are longer than 20,000 bytes, more than six times the average document size for the entire collection. This favoring of extremely long documents is more prevalent when pivoted cosine normalization is used — the last few bins in Figure 4.6(b) have very high retrieval probabilities.

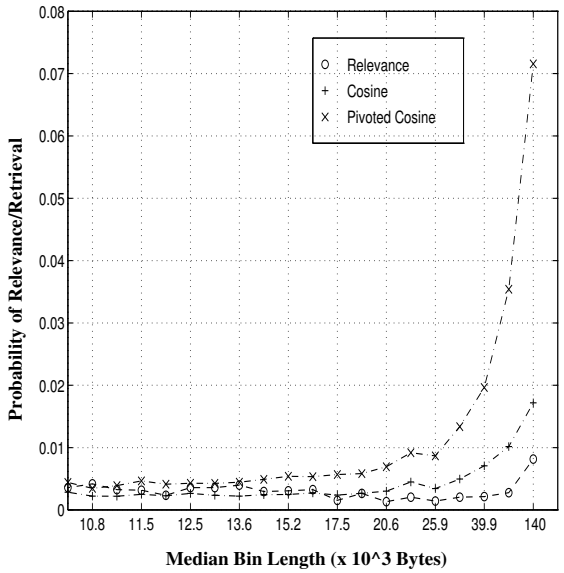


Figure 4.7: Probabilities in the longest twenty bins

This favoring of extremely long documents is further examined in Figure 4.7 which shows a magnified view of the long end of the document length spectrum (the last twenty bins). The last twenty bins contain the longest 19,856 documents (last bin has only 856 documents) from the collection. We notice that using cosine normalization, the retrieval probabilities for extremely long documents are marginally greater than their probability of relevance, *i.e.*, cosine normalization retrieves these documents with "slightly higher" chances than we would like. The very high values for the retrieval probabilities using pivoted cosine normalization for the last few bins (Figure 4.6(b)) show that when we use pivoted cosine normalization, which aims at favoring long documents, we end up "strongly favoring" extremely long documents. This effect causes excessive retrieval of such (possibly non-relevant) documents at high ranks, hurting the retrieval effectiveness.

On deeper analysis of the cosine function, we observe that if all the terms appear just once in a document ($tf = 1$ for all terms), the cosine normalization factor for the document is (individual term weights are $1 + log(tf) = 1$, and we are not using the *idf* factor on documents):

$$\sqrt{1^2 + 1^2 + \ldots 1^2} = \sqrt{\#\ of\ unique\ terms}$$

In reality some terms occur more than once in a document, and the cosine factor can be higher than $\sqrt{\#\ of\ unique\ terms}$. In practice, however, the cosine normalization factors for documents are very close to the function $\sqrt{\#\ of\ unique\ terms}$ due to the following two facts:

- It is well known that the majority of the terms in a document occur only once. So there are only a few terms that have $tf > 1$.
- As we use $1 + log(tf)$ as the tf factor, for most of the terms with $tf > 1$, the tf factors are not too large. Due to the "dampening effect" of the *log* function, most of the tf factors, in practice, are close to 1.0.

We have observed that the cosine normalization factors for documents vary in close accordance with the function: $\#\ of\ unique\ terms^{0.6}$; a function with values quite close to $\sqrt{\#\ of\ unique\ terms}$.

Figure 4.8 is a study of the variation of the cosine factor for TREC documents in relation to the number of unique terms in a document. To generate Figure 4.8, we sampled every twentieth document from TREC disks 1 and 2. This gave us a varied sample of several documents from all the different sub-collections of TREC. We sorted this list of the sampled documents by the byte sizes of the documents, and grouped them into bins of ten documents each. We computed the median number of unique terms for documents in each bin. We also computed the cosine normalization factor for the documents and paired the median

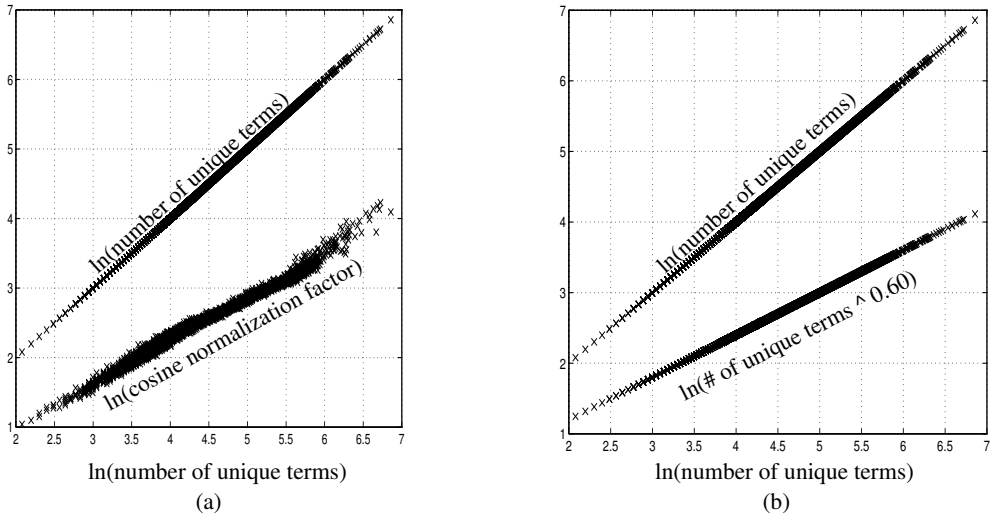


Figure 4.8: Comparison of cosine factors to number of unique terms

number of unique terms for each bin with the median cosine factor for the bin. Figure 4.8(a) is obtained by plotting the median cosine factors in a bin against the median number of unique terms for the bin (on a logarithmic scale). Figure 4.8(b) is obtained by plotting the *median number of unique terms*$^{0.6}$ function for a bin against the median number of unique terms for the bin (once again on a logarithmic scale). We observe from Figure 4.8 that the cosine factor for documents actually does vary in accordance with the function *# of unique terms*$^{0.6}$.

Further, with dampened *tf* factors, even with high raw *tf* values in individual documents, document retrieval is not strongly affected by the term frequency factors. The retrieval of documents is generally governed by the number of matches to the query. Assuming that the presence of a term is completely independent of the presence/absence of another term (the binary independence assumption made by most retrieval models), the probability of a match between a query and a document increases *linearly* in the number of different terms in a document. Suppose the vocabulary size is T, and document D has k different terms. The probability that a randomly selected term belongs to document D is $\frac{k}{T}$. This probability increases linearly in k. Assuming that a query term is randomly proposed by a user, the probability of a match between a query and the document also increases linearly in k. Therefore a good length normalization function should vary linearly in the number of unique terms in a document.

As documents grow longer, the number of unique terms present in a document increase, and the cosine function, with its variation as *# of unique terms*$^{0.6}$, becomes substantially weaker than a linear function in *# of unique terms*. For this reason, we observe that the use of cosine function can favor extremely long documents in retrieval. This problem is aggravated with the use of pivoted cosine normalization which further aids the retrieval of long documents. We propose that a function linear in the number of unique terms in a document be used for normalization. [SBM96]

4.4 Pivoted Unique Normalization

Based on the above observations, we use the number of unique terms in a document as the normalization function, and to match the likelihoods of relevance and retrieval, we use pivoting of this function to get the pivoted unique normalization function:

$$pivoted\ unique\ normalization = (1.0 - slope) \times pivot + slope \times \#\ of\ unique\ terms$$

The pivoted unique normalization function increases linearly in the number of different terms present in a document. This is the property we desire from a good normalization function.

Section 2.2.3 introduced the two main factors that necessitate the use of document length normalization:

1. higher term frequencies, and

2. more different terms.

By penalizing the documents when more unique terms are present in a document, pivoted unique normalization removes the advantage that the long documents have in retrieval over short documents because of their use of numerous different terms. Pivoted unique normalization does not compensate for the first reason that necessitates normalization — *higher tfs* in long documents. If used without any correction for *higher tfs* problem, pivoted unique normalization will still favor long documents in retrieval.

Since long documents generally have higher term frequency factors and thus higher individual term weights, we need a mechanism to penalize the individual term frequency factors as the documents grow longer. Normalization of *tf* weights by maximum *tf* in a document would force all tf factors in a document to be bounded between the values 0.0 and 1.0 (irrespective of a document's length) and would remove the advantage that long documents have due to higher tf factors (see Section 2.2.3). Since long documents have higher maximum *tf*, the penalty on the *tf* factors for long documents will be higher, but we believe that *max_tf* is not an optimal normalization scheme to fix the *higher tfs* problem. Consider a one word query and two documents D_1 and D_2 with the same number of unique terms (so the unique based normalization factors for both the documents are same). If the query term occurs five times in document D_1 in which all other terms occur just once, then D_1 is possibly more "about the query" than another document D_2 in which the query term occurs five times as well, but all other terms also occur five times each. D_2 is equally "about" all the words contained in it, but D_1 is more "about" the query word than any other word contained in it. If *max_tf* is used for normalization, D_1 has no advantage over D_2 since the query term will have the same weight in both the documents.

We believe that *average term frequency* in a document is a better representative of the higher term frequency problem. Judging term importance by term frequencies, if all terms were equally important in a document, each should occur the same number of times in that document with $tf = average\ tf$. For this reason, we would like a term that has $tf = average\ tf$ to have *unit importance* in a document. We use the function:

$$\frac{1 + log(tf)}{1 + log(average\ tf)} \tag{4.4}$$

as the term frequency factor for a term in a document. In experiments comparing *average term frequency* based normalization to *maximum term frequency* based normalization[1], in conjunction with pivoted unique normalization with retrospectively trained best slope value, we observed that average term frequency based normalization performed 5.7% better for 200 TREC queries on retrieval from documents of TREC disks 1 and 2.

Based on the *tf* factor from expression 4.4 (which we call the *L* factor in Smart's term weight triple notation) and pivoted unique normalization (which we call the *u* normalization factor), we obtain the final weighting strategy of the documents (called *Lnu* weighting in Smart):

$$\frac{\frac{1+log(tf)}{1+log(average\ tf)}}{(1.0 - slope) \times pivot + slope \times \#\ of\ unique\ terms}$$

Recall that the inverse document frequency factor is only used in the query term weights. Once again, we can use the *average number of unique terms* in a document (computed across the entire collection) as the pivot, and train for a good slope value.

The results of switching to pivoted unique normalization from pivoted cosine normalization for TREC queries 151–200 are listed in Table 4.7. We observe that the best slope in pivoted unique normalization yields another 6% improvement over the best pivoted cosine normalization, resulting in an overall **18.3%** improvement over cosine normalization. A comparison of retrieval using *Lnu* weighted documents to the relevance pattern (Figure 4.9(a)) reveals that in comparison to pivoted cosine normalization (Figure 4.9(b)), the probability of retrieval using pivoted unique normalization is, in fact, even closer to the probability of relevance for documents of all lengths.

[1] For the maximum term frequency based normalization we used the function $0.4 + 0.6 \times \frac{1+log(tf)}{1+log(max\_tf)}$, a function similar to the well tested and effective function of the INQUERY system. [BCCN95]

Table 4.7: Estimation of a good slope in pivoted unique normalization

	Cosine	Pivoted Unique Normalization			
		Slope			
		0.15	0.20	**0.25**	0.30
# Relevant Retrieved (out of 9,805)	6,526	6,688	6,841	**6,864**	6,852
Average Precision	0.2842	0.3268	0.3355	**0.3361**	0.3318
Improvement	—	+15.1%	+18.1%	**+18.3%**	+16.8%
Improvement over best (0.3171) Pivoted Cosine	—	+ 3.1%	+ 5.8%	**+ 6.0%**	+ 4.6%

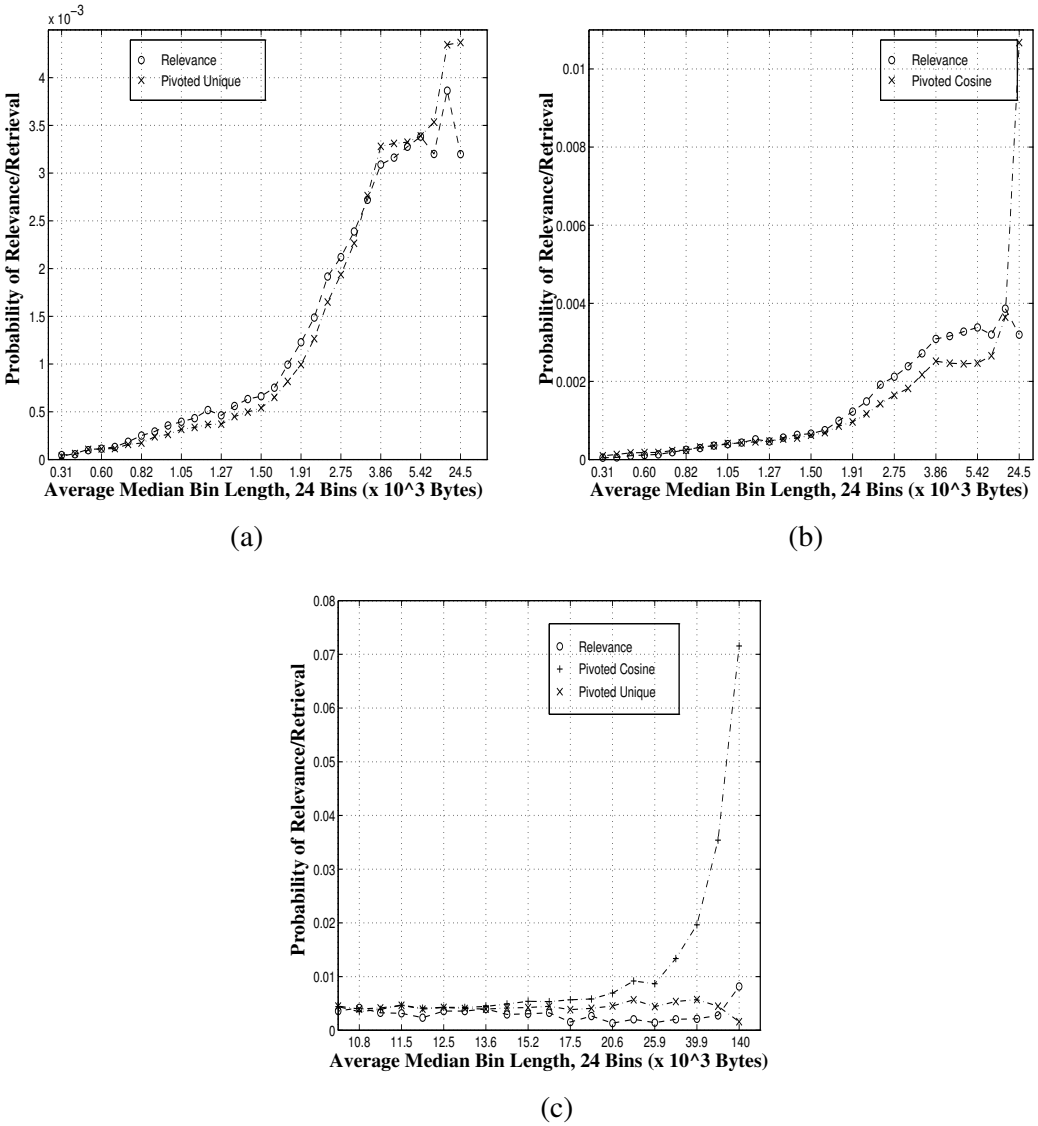


Figure 4.9: Pivoted unique compared to pivoted cosine

Table 4.8: Statistics on number of unique terms in documents

Coll.	Number of Unique Terms used by a document				
	Min.	Max.	Mean	Median	Std. Dev.
AP	1	690	143.71	134	70.87
DOE	1	143	46.17	44	18.96
FR	2	7,947	188.22	123	227.99
WSJ	1	1,733	130.69	81	115.63
ZIFF	3	3,570	111.23	62	124.00
TREC	1	7,947	107.89	67	110.68

Table 4.9: Performance of pivoted unique on six different collections

Collection	AP	DOE	FR	WSJ	ZIFF	TREC
# of Queries	196	80	111	200	122	200
Avg. Prec. Cosine	0.4000	0.3046	0.2931	0.3899	0.3441	0.3357
Best Slope	0.30	0.25	0.10	0.25	0.10	0.20
Pivoted-Cosine	0.4158	0.3341	0.2856	0.3895	0.3248	0.3395
(vs. Cosine)	+ 3.9%	+ 9.7%	+23.4%	+10.5%	+14.8%	+12.9%
(vs. Pivoted Cosine, Best Slope)	– 0.4%	– 0.2%	– 2.6%	– 0.1%	– 5.6%	+ 1.1%
Slope = 0.2	0.4100	0.3319	0.2435	0.3888	0.3115	–
(vs. Cosine)	+ 2.5%	+ 9.0%	+ 5.2%	+10.3%	+10.1%	–
(vs. Best Slope)	– 1.4%	– 0.7%	– 14.7%	– 0.2%	– 4.1%	–
(vs. Pivoted Cosine, Slope = 0.70)	– 1.7%	+ 3.4%	– 12.6%	– 0.3%	– 9.5%	–

As our main motivation for switching from pivoted cosine normalization to pivoted unique normalization was the excessive retrieval of extremely long documents by pivoted cosine normalization, we examine the effect of this switching on the retrieval probabilities for the extremely long documents. A magnified view of the retrieval/relevance probabilities in the last twenty bins (containing the longest 19,856 articles) is shown in Figure 4.9(c). We notice in Figure 4.9(c) that the advantage that very long documents had by the use of pivoted cosine normalization is removed by using pivoted unique normalization. The extremely long documents are not retrieved with exceptionally high probabilities anymore. The additional 6% improvement in Table 4.7 shows that as the retrieval probabilities come closer to the relevance probabilities, retrieval effectiveness increases. The closer the two curves are, the higher is the retrieval effectiveness.

4.4.1 Pivoted Unique on Six Different Collections

We test the general applicability of pivoted unique normalization by using it on the five different sub-collections of TREC disks 1 and 2, as well as the entire TREC collection. Some statistics on the number of unique terms in documents for all six test collections are shown in Table 4.8. Once again, the variations between the characteristics of the documents in different collections are evident in Table 4.8. The high average value for the number of unique terms used by a document and the high standard deviation for the FR collection, again show that FR documents are generally long and have a significant length variation. On the other hand, the low corresponding values for the DOE collection show that the DOE documents are short and have little length variation in their lengths.

The results for retrieval using pivoted unique normalization (Lnu weighted documents) are compared to

the results of using cosine normalization, and the results of using pivoted cosine normalization in Table 4.9. Once again we get impressive improvements — 3.9% to 23.4% — over cosine normalization across all test collections. We observe from Table 4.9 that with retrospectively learned slope values, the performance of the best pivoted unique normalization is close to that of the best pivoted cosine normalization — moderately worse for all (but ZIFF, for which it is noticeably worse) sub-collections, 1.1% better for the entire TREC collection. These results are not as encouraging as the results for TREC queries 151–200 from Table 4.7 where pivoted unique normalization yields 6% improvement over pivoted cosine normalization on the entire TREC disks 1 and 2. We were hoping to obtain similar improvements across collections by the use of the pivoted unique normalization function.

On further analysis of the AP, DOE, and WSJ collections we observe that these collections do not have many "extremely long" documents. Most of the very long documents in the AP collection are about twice the average document length (which, from Table 4.4, is 3,130.65 bytes). Similarly, for the DOE collection, the extremely long documents are about 2.25 times longer than an average document (which, from Table 4.4, is 852.43 bytes); and for the WSJ collection, the very long documents are about 3.5 times longer than the average document length (which, from Table 4.4, is 3,077.58 bytes).

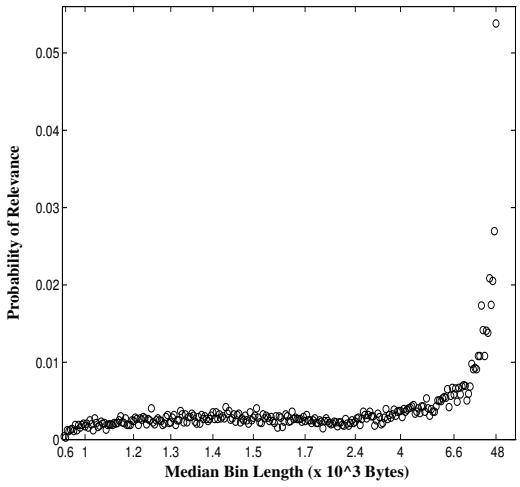
In contrast to AP, DOE, and WSJ, the very long documents in the FR collection are about 86.8 Kbytes long — about eight times as long as the average document length (10,649.42 bytes). Similarly, the longest documents for the ZIFF collection are about six times longer than the average document length. Similar variation also exists for the entire TREC collection for which the extremely long documents are about 24.6 Kbytes which is more than eight times the average document length (2924.89 bytes, from Table 4.4). The fact that the AP, DOE, and WSJ collections don't have too many extremely long documents is also independently verified by the lower standard deviations in their documents' lengths. The reverse is verified for the FR, ZIFF, and TREC collections by the higher standard deviations in their documents' lengths.

When a collection does not have too many extremely long documents, the deficiencies of the cosine normalization function do not affect the document ranking noticeably. The cosine function, which has a weaker form ($\#\ of\ unique\ terms^{0.6}$) than what is desired ($\#of\ unique\ terms$), normalizes equally well with a strong slope (0.7) as a good normalization function ($\#\ of\ unique\ terms$) would do with a reasonable slope (0.20). When the documents get extremely long, even a strong slope can not compensate for the deficiencies of the weak cosine function and the degree of normalization is weak. Since AP, DOE, and WSJ do not have too many extremely long documents, we do not expect to get very different performance from the pivoted cosine and the pivoted unique normalization functions. This is evident in the results for these three collections in Table 4.9. Switching from pivoted cosine to pivoted unique for these three collections has very little effect on the average precision.

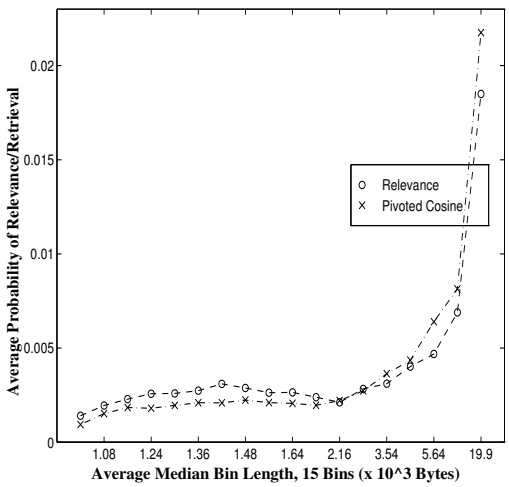
Due to the presence of many extremely long documents in the ZIFF collection, we would expect the deficiencies of the cosine function to affect ranking for ZIFF, and we expect pivoted unique normalization to perform better than pivoted cosine normalization on ZIFF. But, contrary to our expectations, we observe from Table 4.9 that the use of pivoted unique normalization is noticeably poor than the use of pivoted cosine normalization for ZIFF (the average precision using best slope for pivoted unique normalization is 5.6% worse than the average precision using best slope for pivoted cosine normalization). On a deeper analysis of the relevance for the ZIFF collection, we see that the distribution of relevance among documents of various lengths is strongly biased in favor of the extremely long documents (see Figure 4.10(a)). This is evident by the steep increase in the probability of relevance for extremely long articles as compared to the rest of the collection.

If a retrieval system has weak normalization and it favors extremely long articles, but the extremely long articles happen to be relevant with very high chances, the weakness of the retrieval system proves to be a boon for good document ranking, rather than a hindrance. This is what we observe in the ZIFF collection. The relevance pattern for the ZF collection is well suited for normalization schemes that excessively retrieve very long documents; schemes like pivoted cosine normalization. The smooth relevance pattern and the retrieval pattern for pivoted cosine normalization on the ZIFF collection are shown in Figure 4.10(b). It can be observed from Figure 4.10(b) that the favoring of extremely long documents by pivoted cosine is in strong agreement with the relevance properties in the ZIFF collection.

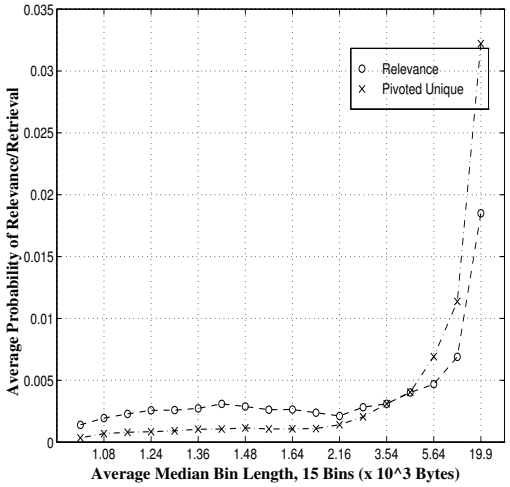
To obtain good retrieval performance from a function that does not preferentially retrieve extremely long documents, like the pivoted unique function, we have to force it to favor extremely long documents by using a very weak normalization (a small slope value) for the ZIFF collection. A slope value of 0.10 works best



(a)



(b)



(c)

Figure 4.10: Analysis for ZIFF collection

for retrieval from the ZIFF collection using the pivoted unique normalization function (for other collections, a slope of 0.20 to 0.30 works well). By making the normalization weak by using a low slope, even though the extremely long documents (which are mostly relevant) are retrieved higher in the ranking, the relative ranking of the documents of lengths other than extremely long documents is poor. This poor ranking for documents of lengths other than extremely long documents yields a lower average precision as compared to pivoted cosine normalization. The pivoted unique normalization function is not as well suited for the relevance pattern of the ZIFF collection as the pivoted cosine normalization function.

If we take the other possibility where extremely long documents are not very relevant to the queries (as for queries 151–200 on the entire TREC collection, Table 4.7), the pivoted cosine function turns out to be noticeably poor than the pivoted unique function. Since the entire TREC collection is a better representative of a realistic heterogeneous collection, even though it might not be optimal on some collections with peculiar relevance patterns, we suggest that pivoted unique normalization should be used in place of the pivoted cosine normalization. We will discuss some more deficiencies of the cosine function in the next chapter. These deficiencies further discourage the use of the cosine function.

In terms of stability of the slope, other than the FR collection, the slope value for the pivoted unique normalization function is quite stable. To deal with situations where training data is not available to learn a good slope, when the slope is fixed at 0.20, (other than FR) the loss in average precision compared to the best slope is small for all collections. For the FR collection, using a slope of 0.20 as opposed to the best slope (0.10) makes the average precision 14.7% worse. The foremost reason for this peculiar behavior is the low number of relevant documents present in the FR collections. Table 4.4 shows that in the FR collection, on an average there are only 16.35 relevant documents for a query. With such little relevance, poor ranking for even a single relevant document can change the average precision noticeably. For this reason, the FR collection is not a good test collection to do the slope stability analysis.

The problem in slope stability for the FR collection is compounded by the fact that it also, like the ZIFF collection, has a relevance pattern that is strongly biased in favor of the extremely long documents. With such a relevance pattern and a low relevance density per query, when we use a slope of 0.20, we lower the ranks of the extremely long documents. Since these documents are mostly relevant, we lower the ranks of many relevant documents. Due to the low number of total relevant documents, we obtain much poorer average precision. Like the ZIFF collection, the relevance in the FR collection is also distributed in a manner that encourages the use of weak normalization functions (like the pivoted cosine).

4.5 Discussion

The deficiencies of cosine normalization were evident to researchers before we developed pivoted normalization. Researchers have proposed ranking short passages from articles in conjunction will ranking full articles to obtain a final ranking for the documents. [SAB93,HP93,Cal94,Wil94,MS94] The main motivation behind the use of passages for document ranking was to "focus-in" on the relevant information if it is buried deep in a longer article. As a side effect, using passages for ranking document also alleviated some of the deficiencies of the document length normalization aspect of the retrieval systems.

When documents are broken into smaller pieces (or passages), the length variation in the small passages is much less than the rank variation between full text documents. Some researchers have even used fixed length passages which almost eliminates length biases across passages. [Cal94,BASS95] With little variation in the lengths of the passages, the affect of length variations on the query–passage similarity is minimal. Combining the query–document similarity with the query–passage similarity yields a final document similarity that is less sensitive to document length variations, as opposed to using only the query–document similarity. But query–passage similarity computation is expensive, especially when similar (or even better) results can be achieved by simply using a new, better document length normalization function.

Lee (in [Lee95]) shows that combining rankings from a strong document length normalization function (like cosine normalization) and a weak normalization function (like maximum-tf based normalization) yields better results than either normalization scheme used alone. When a strong normalization method, that favors retrieval of short documents, is combined with a weak normalization, that favors retrieval of long documents, the resulting retrieval tends to cancel out the deficiencies of either normalization used alone. However, doing two retrievals for a query and then merging the retrieved lists is once again more expensive

than using just one retrieval with a better length normalization function. Also, the results obtained by a good normalization function usually are better than the results obtained by combining two (one weak and one strong) poorer normalization functions.

Therefore, even though other techniques could possibly solve the document length normalization problem to a certain degree, we suggest that we use a single, good normalization function. Using a good normalization function is more efficient (and often even more effective) than using other techniques. Also, a good normalization function would be useful in other IR tasks where the aim is not to produce just a good document ranking, instead we want to use the term weights for other purposes, like query reweighting and expansion in relevance feedback. [SB90a]

4.6 Conclusions

The pivoted unique normalization scheme has several advantages over the pivoted cosine scheme. The foremost advantage is that it does not excessively retrieve too many long documents in top ranks. Even though it yields poor precision if most of the relevant articles are long, but we expect it to be beneficial on general collections. Also, from a user's perspective, if too many extremely long articles are retrieved in the top ranks, the user has to sift through these huge documents to find the useful material. This overload might not be preferred by precision oriented users. Pivoted unique normalization is also free of another deficiency of the cosine function (discussed in the next chapter). Based on these observations, we recommend the use of pivoted unique normalization for retrieval systems.

Chapter 5

Length Normalization in Degraded Text Collections

Optical character recognition (OCR) is commonly used to convert printed material into electronic form. Using OCR, large repositories of machine readable text can be created in a short time. An information retrieval system can then be used to search through large information bases thus created. Modern information retrieval systems use sophisticated term weighting functions to improve the effectiveness of a search (see Chapter 2). Term weighting schemes can be highly sensitive to the errors in the input text introduced by the OCR process. This chapter examines the effects of OCR errors on cosine normalization, and proposes a new, more robust, normalization method. Experiments show that the new scheme is less sensitive to OCR errors and facilitates the use of more diverse unnormalized weighting schemes. This chapter also explains why the use of cosine normalization in presence of the inverse document frequency factor is not advisable in large document collections. Using ideas from Chapter 4, we develop a more general and robust normalization function — *pivoted byte size normalization*. [SSB96,SBM96]

5.1 Background

OCR is widely used to scan printed text, text not otherwise available in machine readable form, and convert it into electronic form. Advances in OCR technology have decreased the error rates of OCR devices, but due to the poor quality of some printed materials and other errors introduced by an OCR system, the error rate of an OCR process can still be substantial. [NR94] This poses a special challenge when scanned text collections are searched by automatic information retrieval systems. Most information retrieval systems use keywords (terms) from documents to match the documents to a user query. These keywords can be corrupted in the OCR process, hindering the successful retrieval of useful documents for queries. Modern information retrieval systems use sophisticated term weighting functions to assign importance to the individual words of a document. [SB88,RSJ76] These weighting functions use the occurrence statistics of words in the documents, and in the collection as a whole, to assign importance to different words. As the occurrence statistics of the spurious words generated due to OCR errors can be significantly different from the characteristics of the good words, weighting schemes are specially susceptible to degradation in the quality of the input text.

How do the OCR errors affect the retrieval effectiveness of an information retrieval system? This problem has been extensively studied by Taghva et al. at the Information Science Research Institution (ISRI) of University of Nevada, Las Vegas. In the various studies performed at ISRI [CHTB94,TBC94a,TBC94b,TBC94c, TBCE94,TBCI95], Taghva and his colleagues have noticed that in terms of recall–precision, the retrieval effectiveness of a retrieval system is not substantially affected by OCR errors. The main reason for this, they claim, is the excessive redundancy of information present in the full text documents. This redundancy compensates for the information lost in an OCR operation. They finally conclude that, in conjunction with an automatic information retrieval system, a degraded text collection produced by a reasonable OCR system can be searched almost as effectively as the correct text collection.

However, Taghva and his colleagues did notice several interesting facts when sophisticated term weighting

schemes were used in conjunction with degraded text. Specifically, whenever they used the cosine normalization function for document length normalization of term weights, the document ranking generated by a system for the degraded text collection was substantially different from the ranking generated by the same system for the correct text collection. In [TBC94a] they explain:

> ...from our variance test, cosine normalization is a significant factor. We know that "garbage strings" increase the length of the OCR document vectors. This is specially true for documents with many misrecognized terms and/or graphic text. We found that the OCR vector length was approximately three times the length of the corresponding corrected-text document vector for those documents with a high discrepancy in ranking. Aggravating this fact, "garbage strings" tend to have high term weights. When these terms are normalized, their significance to the document increases in relationship to other terms in the collection.

Due to OCR errors, numerous words are misrecognized and meaningless "garbage strings" are generated. Term weighting schemes are affected by the presence of such garbage strings, and assign misleading weights to different terms in a document.

It is evident from [TBC94a] that due to the wide corruption of the unnormalized term weights (especially the inverse document frequency factor), cosine normalization is not desirable in degraded text collections. Research in information retrieval has shown that document length normalization is important for effective retrieval of documents of all sizes. [SB88,SSMB96] Therefore, an alternative to cosine normalization is desirable in OCR environments. This study focuses on developing a new, robust, length normalization technique that is less sensitive to OCR errors.

5.2 Problems with Cosine Normalization

Recall that the cosine normalization factor for a text is computed by summing the squares of all the unnormalized term weights of the text–vector and then taking the square root of this sum (see Section 2.2.3).

$$\sqrt{{w_1}^2 + {w_2}^2 + \ldots + {w_n}^2}$$

where w_i is the unnormalized weight of term-i, and the text has n distinct terms. Due to the presence of all the unnormalized term weights in the normalization factor, when cosine normalization is used for length normalization, the final weight of a term depends on the individual unnormalized weights of all other terms in the text. *Cosine normalization introduces a mutual dependence between the weights of the terms in a text*. Due to such mutual dependence between term weights, an error in approximation of one term's weight can affect the weights of all other terms. This dependence is undesirable in environments where individual weights of the terms might be erroneous, and do not accurately represent the actual importance of the terms in a text. OCR environments are specially susceptible to such errors in term weight estimation.

When cosine normalization is used with text generated by an OCR system, the normalization factors for the degraded documents are significantly higher than the corresponding cosine normalization factors for the correct documents. Two main reasons are responsible for this increase:

1. Due to OCR errors, word separators (white spaces, punctuation, ...) might be inserted in one word, and a word in the correct text might get split into two or more words in the scanned text. The OCR system can even misinterpret some graphic material as one or more completely new words. This increases the number of distinct terms present in the vector of the degraded text in comparison to the number of distinct terms present in the vector of the corresponding correct text. When more distinct terms are present, the documents "seem" longer and the cosine normalization factor increases.

2. OCR errors create numerous garbage strings (senseless words), that occur in very few documents in the entire collection, usually just one or two. Such erroneous strings tend to have a very high inverse document frequency value and, therefore, a higher unnormalized $tf \times idf$ weight. When the idf factor is used in term weights, high individual term weights for such misrecognized terms can significantly increase the cosine normalization factor.

Table 5.1: Correct/corrupt ltc weighted vectors for DOE1-01-0942

Text: A specific solid system with first order phase transition in its surface is analysed. To accomplish this task, we use the Landau theory. (M.W.O.).		
	Correct Cosine Factor: 15.61	Corrupt Cosine Factor: 22.03
Term	Correct *ltc* weights	Corrupt *ltc* weights
surface	0.20609	0.14600
accomplish	0.26372	0.18682
task	0.22716	0.16092
solid	0.22955	0.16261
m.w.	0.58059	0.41129
landau	0.46977	0.33271
order	0.13781	0.09761
transition	0.23660	0.16757
theory	0.24174	0.17121
specific	0.16075	0.11385
analysed	0.11849	0.08392
phase	0.21261	0.15058
system/**systom**	0.08614	**0.61349**

The inflation in the cosine normalization factor due to OCR errors artificially depresses the weights of the correctly recognized terms, terms which are mainly responsible for predicting the usefulness of a document. When weights of the good terms in a useful document are depressed, the distinction between a good document and a poor document is blurred, decreasing the IR system's effectiveness in distinguishing between the good and the poor articles. When it is hard to distinguish the good documents from the poor documents, the retrieval of the useful documents for a user-query suffers, yielding poorer document ranking. This effect is quite severe when the *idf* factor is used in term weights (the second reason mentioned above).

For example, Table 5.1 shows the text for a short TREC [Har95] document — `DOE1-01-0942`, the correct *ltc* (logarithmic *tf*, *idf*, and cosine normalization; see Section 2.3) weighted document vector, and the document vector if the word `system` is misrecognized during the OCR process as `systom`. Assuming that this recognition error occurs only once, the corrupted term — `systom` — gets an unnormalized $tf \times idf$ weight of 13.52 ($tf = 1$, $N = 741,856$, $df = 1$, $idf = log(\frac{N}{df}) = 13.52$), as opposed to its initial unnormalized weight of 1.34 (the correct document frequency for the word `system` is 193,451). This single corruption in the source text elevates the cosine normalization factor for this document from 15.61 to 22.03, an increase of more than 40%, and unfairly depresses the weight of every other term in the document. For example, the weight of the useful term `Landau` has been reduced from 0.47 to 0.33. Such deviations can be much more severe in practice when documents have multiple misrecognized words.

5.3 New Normalization Technique

To remove the undesirable mutual dependence in term weights introduced by cosine normalization, we would like to replace the cosine function by another normalization scheme that does not introduce any mutual dependence between the term weights of a document. Using a normalization scheme based upon the number of distinct terms in a text is also not advisable in this environment because the OCR process inflates the number of unique terms present in a text. As a document grows longer, the number of bytes used by the document increase. This suggests that functions of the number of bytes in a document could possibly be used for document length normalization. Some researchers have successfully used the document size (in bytes) for length normalization. [RWJ+95] In OCR environments, the size in bytes of the documents are

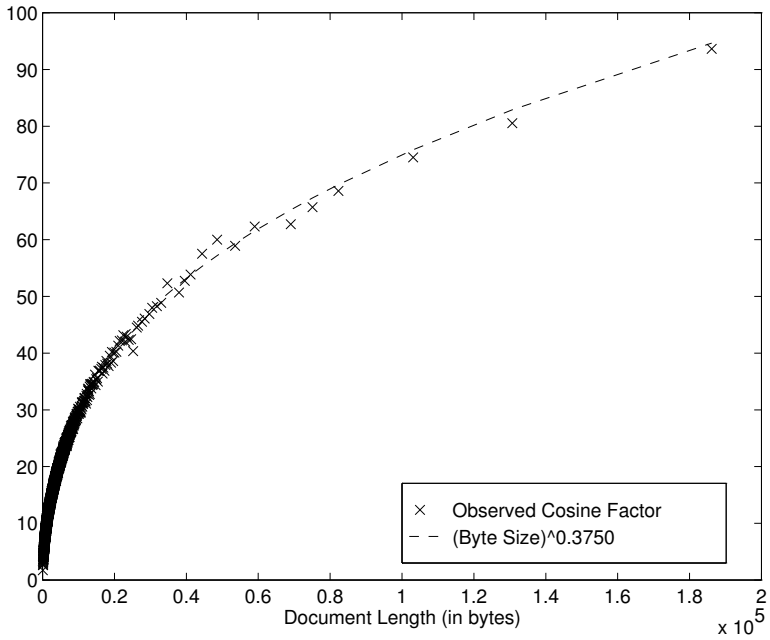


Figure 5.1: Learning a function of byte size for normalization

less distorted than the corruption in the cosine normalization factors. Using functions of byte size should be more stable than using cosine normalization and should, in principle, yield less deviant retrieval results.

To ensure that the "strength" of normalization performed by a new function of the byte size of the documents is comparable to cosine normalization, we want to learn a function that would have values similar to the values of the cosine normalization factor for documents of various length. To obtain such a function, we study the variation of the cosine normalization factors in relation to the byte size of the documents, and approximate that variation by a function of the byte size itself. We use the TREC collection [Har95] to do all our experiments in this study. To learn a function with above properties for the TREC collection, we sampled every twentieth document from correct[1] TREC disks one and two. This gave us a varied sample of several documents from all the different sub-collections of TREC. Our previous experiments with the TREC collection have shown that using *lnc* weights on document terms yields good results. [BSA93,BASS95] We computed the cosine normalization factors used in *lnc* weighting of the sampled set and plotted it against the byte size of the documents. The cosine normalization factor for a document with `t` terms (under *lnc* weighting) is given by:

$$\sqrt{(1+log(tf_1))^2 + (1+log(tf_2))^2 + \cdots + (1+log(tf_t))^2}$$

As different documents of similar (byte) lengths can have different cosine normalization factors, we need to smooth the plot obtained above to get an "approximate" value of the cosine factor the documents that are of "about the same length". In other words, we want a rough approximation of the cosine factors for documents of a certain length. To do the desired approximation, we use bucketing to smooth the plot obtained above. We sort the list of $\langle$*byte size, cosine normalization factor*$\rangle$ pairs by byte size and divide this sorted list into bins (buckets) of ten pairs each. Assuming that the ten documents in a bucket have very similar lengths, we select the median byte size in the bin as the byte size for the entire bucket. We select the median cosine normalization factor from every bin and represent that value on a graph against the median byte size in the bin (see Figure 5.1). A 'x' in Figure 5.1 denotes the observed median value for cosine normalization factor (for lnn weighted document vectors) in a bucket plotted against the median byte size in the bin.

[1] We do this learning on the correct text as we do not want the errors in the corrupted text to affect our normalization function. We hope to learn a robust function from one collection of text and use it on other (new) correct/corrupted collections.

Table 5.2: Characteristics of the correct/degraded collections

TREC Disks 1 and 2 — 741,856 Documents		
	Correct Text	Simulated OCR (5% degradation)
# Unique Terms in Collection	511,324	8,423,852
Avg. # of *Unique* Terms per Document	108.51	168.87
Avg. # of Terms per Document	210.52	239.49

The variation in the cosine factor with the byte length in Figure 5.1 suggests that a fractional power of the byte size ($byte\ size^{\alpha}$ where $\alpha < 1$) can be used to replace the cosine normalization factor. Figure 5.1 shows that a value of 0.3750 for α yields a function very similar to the actual cosine normalization factor. Therefore we use the function $byte\ size^{0.3750}$ for term weight normalization in our experiments. *We represent this new byte size normalization scheme by the letter* **b** *in the weighting triple notation of Smart.* This yields the final (*lnb*) term weights:

$$\frac{1 + log(tf)}{byte\ size^{0.3750}}$$

5.4 Test Collection

To study the true effects of OCR errors on an information retrieval system, we need two copies of the same collection, the correct copy and the corrupted copy. Given both the copies of a collection, it is possible to run an IR system on the correct text, and then on the corrupt collection, and study the differences in document ranking. It is feasible to scan small text collections and manually correct the OCR errors to get the correct copy. Unfortunately, scanning large text collections (like TREC) and correcting them manually is prohibitively expensive. For this reason, most of the real dual OCR collections (correct and corrupt) are small text collections. [TBC94c] Most IR experimentation for large degraded text collections is done by taking a correct collection available in electronic form, like the TREC collection, and simulating OCR degradation on the collection to obtain its degraded version. [CHTB94]

Information retrieval experiments performed on very small text collections do not necessarily reflect a retrieval system's performance on a larger, more diverse text collection. In the absence of a large collection of scanned text, we used simulated OCR degradation on disks one and two of the TREC collection, a total of 741,856 documents. [Har95] We used a simplified text degradation routine which randomly altered every character in the correct text with a given probability. We simulate a poor quality OCR process where every character was randomly altered with a probability 0.05 (5% degradation). For every character, we generate a random (real) number between zero and one, and if the random number is less than 0.05, we replace the correct character by a randomly generated character. Besides changing a word into an illegal string (*e.g.*, `system` to `systom`), this process can replace a legal letter in a word by a word separator (such as a blank character or a punctuation mark), resulting in two different words. Also, a work separator can be replaced by a legal letter, changing two different words into a single long word.

This simple stochastic simulation of OCR errors depicts the extreme corruption that an OCR process can cause in the input text. In reality, OCR errors tend to be repetitive and the erroneous characters are not independent of the input text sequence. For example, many OCR devices repeatedly confuse the letter sequence "ni" for the letter "m". Researchers have attempted a more guided degradation of a correct collection by learning the common errors caused by OCR devices. [TBCE94] In absence of a corruption

algorithm guided by observations from real OCR devices, we use our extreme form of stochastic corruption as described above. We believe that this difference from an actual OCR environment is not significant for the set of experiments described in the next section.

As the collection size is around 2 GB, and 5% of the characters are randomly changed, around 100,000,000 characters would be changed by the degradation process. Since many random changes produce a completely new sequence of letters, we expect the dictionary for the degraded collection to be much larger than the original dictionary. Some characteristics of the degraded collection and the correct collection are listed in Table 5.2. As expected, there is a large increase in the number of unique terms in the degraded collection (from 511,324 to 8,423,852). Since random degradations might corrupt one occurrence of a term and not change another occurrence of the same term, one unique term can generate several difference unique terms in a degraded document. Due to this effect, we notice an increase in the average number of unique terms per document (from 108.51 to 168.87). Also, due to possible insertion of random word terminators in the middle of a word, a word in the correct text might split into two or more words in the degraded text. This effect is reflected by an increase in the average number of (total) terms in a document (from 210.52 to 239.49).

Surprisingly, Croft et al. observed a reduction in the dictionary size in their experiments with simulated OCR degradation. [CHTB94] They degraded four different collections (CACM, NPL, WEST and WSJ) and the resulting dictionary size for their degraded collections was always smaller than the original dictionary size (see Table 2 in [CHTB94]). This is a result that we will need to reconcile. In that study, Croft, Taghva and their colleagues also reported a decrease in the total term occurrences (and thus the average number of terms in a document) in all the degraded collections (see Table 2 in [CHTB94]). This, again, does not agree with our observations. The characteristics of our degraded collection do, in fact, agree with the actual OCR collections used in earlier studies. [TBC94c] In experiments conducted with (small) real OCR collections, Taghva and others, like us, did notice a huge increase in the dictionary size for the degraded text collection (Table 2 in [TBC94b]).

We use the correct query text in our experiments. Since in an operative environment queries are typed by the users, it is meaningless to simulate OCR error on a query text. Having generated the corrupted version of the TREC database, we can now rank documents for the same queries using the correct TREC collection and its corrupted version. This will enable us to study the rank variations due to corruption in the input text. Based on these rank variations, we can observe if a given document length normalization strategy is robust against OCR errors or not.

5.5 Experiments

Changes in the average precision computed across a set of queries may not accurately reflect the effects of degradation on individual queries. In an OCR environment, one should measure how "close" the retrieval from the degraded collection is to the retrieval from the correct collection. The *variation between the document rankings produced for the correct text and the degraded text* is a more meaningful measure to study the effects of OCR errors on a retrieval system. [TBC94a] We analyze our results using the traditional recall–precision analysis, as well as by studying the variation in the document rankings produced by the different normalization methods (none, cosine, and byte size).

Using various normalization strategies, we retrieve the top one thousand documents for each of the fifty TREC queries (numbered 151–200) from the correct TREC collection and the corrupted copy of the collection. We compare the two ranked lists to study the variations in the document ranking produced by a given normalization scheme. A scheme for which the document ranking for a given query on the corrupt collection is closer to the document ranking obtained for the same query on the correct collection is less sensitive to OCR errors than another scheme for which the document ranking for the corrupt collection differs substantially from the document ranking for the correct collection.

5.5.1 Tests

As a first test, we compare the average number of common documents retrieved per query (within the top one thousand retrieved) from the correct collection and the degraded collection by the different normalization (weighting) schemes. A higher average number of common documents would signify a higher correlation be-

tween retrievals from the correct and the degraded collection, which, in turn, suggests that the normalization scheme used for retrieval is less susceptible to OCR errors during retrieval.

As the second test, we study the variations in the ranks of individual documents. We retrieved the top one thousand documents for every query from the correct collection and from the degraded collection using a particular weighting method. We then computed the difference in the ranks assigned to the *same document* in the two retrieval runs[2]. From this analysis, we obtained a list of rank differences ($| rank_1 - rank_2 |$) for the documents retrieved from the correct and/or the degraded collection. We computed the mean and the standard deviation for this list to estimate the variation in rankings generated by the weighting scheme for the two collections. A low mean would suggest that the variation in the rankings generated by a weighting scheme, for the correct text and for the degraded text, is low, and is indicative of the robustness of a weighting method against OCR errors. A low standard deviation would signify that the variance in the rank differences is low, and is also desirable from a weighting scheme.

5.5.2 Weighting Strategies

- **Document term tf:** Previous experiments with the TREC corpus have shown that the logarithmic term frequency weights provide superior retrieval effectiveness as compare to other *tf* weights. We therefore used the logarithmic *tf* factor in these experiments. [BASS95]

- **Document term idf:** Our experiments reveal that the use of cosine normalization in presence of the *idf* factor on the document weights tends to hurt retrieval effectiveness. To test this further, we ran experiments with both the absence, and the presence of the *idf* factor in document term weights.

  Since the function $byte\ size^{0.3750}$ was learned without using the *idf* factor on the document term weights, we needed to learn a similar function for document weights with *idf*. The function $4 \times byte\ size^{0.3750}$ had the desirable shape. As the constant 4 simply scales all the similarities in an inner-product similarity computation, we were able to use $byte\ size^{0.3750}$ to normalize the $tf \times idf$ weighted documents as well.

- **Document term weight normalization:** We experimented with the absence of any normalization, the use of cosine normalization, and the use of byte size normalization on document term weights.

- **Query term tf:** Previous experiments also indicate that the logarithmic term frequency weights work well on query terms too. [BASS95]

- **Query term idf:** As the *idf* factor should be used at least once in the similarity computation, we always used the *idf* factor on the query term weights.

- **Query term weight normalization:** Since normalization of query term weights just acts as a scaling factor for all the query–document similarities, and has no effect on the relative ranking of the documents, there was no need to vary the normalization factor for the query term weights.

With the above combination of the document and query term weights, we ran six different runs on the correct and the degraded collection — *lnn.ltc, ltn.ltc, ltc.ltc, ltb.ltc, lnc.ltc,* and *lnb.ltc.* All experiments were done with fifty TREC queries, numbered 151 to 200.

5.6 Results

The results, listed in Table 5.3, yet again, confirm the long known need for document length normalization of term weights. The two runs in which no normalization was used (document weights *lnn* and *ltn*, runs 1 and 2) are extremely poor (about 80–95% worse) in comparison to the other four runs in which document length normalization was used. For these two runs, the variation in the document rankings produced for the correct text and the degraded text, even though quite low, is not indicative of a realistic retrieval situation. In the absence of any length normalization, the poor retrieval effectiveness of these runs is due to the excessive

[2] If a document was retrieved in one run and not in the other, its rank was assumed to be 1,001 in the run in which it was not retrieved.

Table 5.3: Results on the correct and the degraded collection

Run		Correct Text	Degraded Text (5% degradation)			
		Average Precision	Average Precision	Average Common Documents/ Query	Rank Difference With Correct Text	
					Mean	Standard Deviation
1	lnn.ltc	0.0090	0.0068 (–24.9%)	936.30	58.90	64.12
2	ltn.ltc	0.0413	0.0287 (–30.5%)	910.12	83.54	92.17
3	ltc.ltc	0.2245	0.2046 (– 8.9%)	700.56	236.59	208.84
4	ltb.ltc	0.3071	0.2736 (–10.9%)	778.36	183.06	172.63
5	lnc.ltc	0.2730	0.2358 (–13.6%)	726.78	217.83	196.60
6	lnb.ltc	0.3047	0.2660 (–12.7%)	769.76	189.94	174.34

bias in favor of retrieval of the long documents. Since the TREC collection has few very long documents, when two different retrievals favor the excessive retrieval of very long documents, they are bound to retrieve a large number of common documents.

Table 5.3 shows that, with a reasonable (normalized) weighting scheme, the deterioration in retrieval effectiveness due to OCR degradation is in the range of 9–14% (runs 3 to 6). The fall in average precision in our results is somewhat larger than the deterioration reported by Croft, Taghva et al. in earlier studies (1–10%). [CHTB94,TBC94a] The difference in their results and the results from this study can be an artifact of the different collections that are being used in the two studies. Also, the text degradation schemes used in the two studies are substantially different. Overall, these results confirm the findings of Taghva and his colleagues that it is possible to use an automatic information retrieval system in conjunction with degraded text collections, without much loss in average recall-precision figures.

When cosine normalization is replaced by byte size normalization in the presence of the *idf* factor (runs 3 and 4), we observe that:

1. The average number of common documents retrieved, in the top one thousand from the correct and the degraded collection, increase from 700.56 per query to 778.36 per query, an increase of 11%.

   This indicates that when byte size normalization is used to retrieve documents from the degraded collection, it retrieves more documents in common with the documents it retrieves from the correct collection. On the other hand, when cosine normalization is used for retrieval from the corrupted collection, it retrieves fewer documents in common with the documents it retrieves from the correct collection. This indicates that corruption in the input text affects cosine normalization to a larger extent than its effects on byte size normalization. Byte size normalization is more robust again the degradation in the input text.

2. The mean rank difference between retrieval from the correct collection and the degraded collection reduces from 236.59 to 183.06.

   This indicates that (as compared to cosine normalization) not only is byte size normalization retrieving more documents from the degraded collection in common with the documents it retrieves from the correct collection, it also assigns similar ranks to the retrieved document (as the ranks it assigns in retrieval from the correct collection). This number, once again, shows that corruption of the input text has less effect on byte size normalization than its effect on cosine normalization.

3. The standard deviation in the rank difference reduces from 208.84 to 172.63.

   The lower standard deviation in the rank difference for byte size normalization indicates that — even when byte size normalization assigns ranks to the corrupted documents that are different than the ranks assigned to the corresponding documents in retrieval from the correct collection, not too many documents have a large variation in their ranks. The variance in rank differences is low. This again shows the robustness of byte size normalization in presence of errors.

Table 5.4: Effect of using *idf* with cosine/byte-size normalization

	Cosine Normalization		Byte Size Normalization	
	No *idf* *lnc.ltc*	With *idf* *ltc.ltc*	No *idf* *lnb.ltc*	With *idf* *ltb.ltc*
Avg. Precision	0.2730	0.2245 **(−17.8%)**	0.3047	0.3071 (+0.8%)
Number of Rel. Docs. Retrieved	6258	6002 (−256)	6294	6651 (+357)

The above three changes are desirable from a robust normalization scheme in the presence of OCR errors. This indicates that byte size normalization is better suited for OCR environments in comparison to cosine normalization.

It is possible that in the two effects discussed in Section 5.2 that increase the cosine normalization factors for the corrupt document vectors, the second effect (higher cosine factor due to high *idf* values of the "garbage strings") is more important than the first effect (higher cosine factor due to more distinct words in the degraded text). To check if the first effect is at all important, we removed the *idf* factor from the document term weights and repeated the above analysis. When no *idf* factor is used in the document term weights, and cosine normalization is replaced by byte size normalization (runs 5[3] and 6), we observe that:

1. The average number of common documents retrieved increases from 726.78 per query to 769.76 per query, an increase of 6%. This indicates that switching to byte size normalization is useful even when the *idf* factor is not used in the document term weights.

2. The mean rank difference decreases from 217.83 to 189.94. This decreases is not quite as marked as the difference when the *idf* factor is used; however, using byte size normalization is still better.

3. The standard deviation in the rank difference reduces from 196.60 to 174.34. Again, using byte size normalization is slightly better than using cosine normalization.

These results show that even when the *idf* factor is not mixed with the cosine normalization function, using byte size normalization is more robust in the degraded text environments. The corruption in the cosine factor (due to more distinct words in the degraded text) is still more marked than the corruption in the byte size of the documents. The usefulness of byte size normalization is specially visible when the *idf* factor is present in the document term weights, indicating that the use of cosine normalization along with the *idf* factor is undesirable.

5.7 Effects on Correct Text

Table 5.4 shows the effects of using the *idf* factor in conjunction with the cosine normalization factor and with byte size normalization *on the correct text collection*. TREC queries 151–200 were used in this analysis and top 1,000 documents were retrieved for every query. Table 5.4 also shows the total number of relevant documents retrieved for all fifty queries (out of 9,805).

An interesting observation of using cosine normalization on the *correct text* is that the introduction of the *idf* factor in the document term weights produces significantly worse results (17.8% worse) than using no *idf*. This is in contrast to the earlier studies (done on small test collections) which showed that using or not using the *idf* factor in document term weights had no significant effect on the retrieval effectiveness. [SB88] Also, when the *idf* factor is used with cosine normalization, the system retrieves 256 fewer relevant documents

[3] This run is same as the base cosine run reported in the earlier chapters. The reason why the average precision for this run is 0.2730 instead of 0.2842 (as reported in the earlier chapters) is that we are not using phrases in our experiments with the degraded collection. Phrases were used in the experiments reported in the earlier chapters.

across all fifty queries. Whereas, when byte size normalization is used, introducing *idf* in document term weights not only yields a slightly better average precision (+0.8%), it also retrieves an extra 357 relevant documents in the top one thousand documents retrieved per query across all fifty queries. This shows that the use of the *idf* factor twice in document ranking is not inherently poor; instead, mixing the *idf* factor with the cosine normalization function generates poor document ranking.

The poor retrieval effectiveness when cosine normalization is used with the inverse document frequency factor in the document term weights can be attributed, once again, to the mutual dependence in term weights introduced by cosine normalization (see Section 5.2). Even in a correct collection, *if a document has one or more very rare terms, terms with very high* idf *values, similar to the misspelled term in Table 5.1, these terms unfairly depress the weights of the other document terms* when cosine normalization is used. This hinders the retrieval of good documents due to the presence of few rare terms, terms that will probably never occur in a user query. This effect is especially marked in large collections where the *idf* factors for the rare terms are usually very large.

These results show that the use of byte length normalization in place of cosine normalization opens up the possibility of using the *idf* factor in document term weights. In environments, such as the automatic generation of hypertext links [All95], or automatic query modification via relevance feedback [SB90a], where the use of the *idf* factor on document term weights is very important, using cosine normalization can hurt the effectiveness of the process. In such environments, byte size normalization has an added advantage over cosine normalization.

These results show that the *importance of a document term should not be largely affected by the importance of other terms from the same document.* Consider two documents (D_1 and D_2) of about equal lengths. If a term T occurs twice in a both the documents, but D_1 has few rare terms (usually assumed very important and assigned a high *idf* value), whereas, D_2 does not, with $tf \times idf$ weighting and cosine normalization, the weight for term T in D_1 will be smaller (due to the high *idf*s of the rare terms) than its weight in D_2. The results from Table 5.4 indicate that such strong mutual dependence between term weights is not useful. Intuitively, the importance of term T in documents D_1 and D_2 is "not too different". This observation advocates the use of normalization functions that do not depend on the individual weights of the terms in a document, *i.e.*, functions that do not introduce a strong mutual dependence between document term weights.

The pivoted unique normalization scheme introduced in Chapter 4 also has the desirable property that is does not introduce any mutual dependence between document term weights. The normalization factor in the pivoted unique normalization scheme does not use the individual term weights, it just uses the number of distinct terms used by a text. This is another reason why the pivoted unique function should be preferred over any form of the cosine function (*e.g.*, the pivoted cosine normalization).

5.8 Pivoted Byte Size Normalization

One drawback of the function learned in the above study is that the function $byte\ size^{0.3750}$ was obtained in a rather ad hoc manner with weak justification of the parameters involved (α). We did this work on the degraded TREC collection before we developed the pivoted normalization scheme. As pivoting is a more principled approach to learn a normalization function, combining pivoted normalization with byte size normalization should yield a better normalization function. With pivoted normalization, it is possible to use the raw byte size of a document for normalization (instead of the function byte size$^{0.3750}$) and we can pivot this function (raw byte size) to match the relevance and the retrieval patterns (as discussed in Chapter 4) to obtain a good normalization function. Since the byte size of a document increases with the multiple occurrences of a term as well as with the presence of more distinct terms, using byte size normalization compensates for both the reasons that necessitate the presence of document length normalization — higher term frequencies, and more distinct terms (see Section 2.2.3).

To test the applicability of a raw byte size based normalization function on retrieval effectiveness, we pivoted the raw byte size of documents and used it on the correct TREC collection. Using the average byte size as the pivot, we obtain the following new normalization function:
pivoted byte size normalization =

$$(1 - slope) \times average\ byte\ size + slope \times byte\ size$$

Table 5.5: Estimation of a good slope in pivoted byte size normalization

	Cosine	Pivoted Byte Size Normalization			
		Slope			
		0.25	**0.30**	0.35	0.40
# Relevant Retrieved (out of 9,805)	6,526	6,634	**6,678**	6,689	6,570
Average Precision	0.2840	0.3258	**0.3277**	0.3261	0.3088
Improvement	—	+14.7%	**+15.4%**	+14.8%	+8.7%
Improvement over best (0.3361) Pivoted Unique	—	- 3.1%	**- 2.5%**	- 3.0%	-8.1%

Table 5.5 shows the results of using the raw byte size of documents for normalization for TREC queries 151–200 and documents from correct TREC disks one and two. Each entry shows the total number of relevant documents retrieved (out of 9,805) for all fifty queries, and the non-interpolated average precision. The improvements in average precision over cosine normalization and over pivoted unique normalization are also shown. The pivot value was set to 2,730, which is the average number of indexable bytes in a document for TREC disks one and two.

Table 5.5 shows that pivoted byte size normalization also yields important improvements over cosine normalization. It is marginally worse than using the best pivoted unique normalization on the correct text. Overall, pivoted byte size normalization is an effective way to normalize. Since the pivoted byte size normalization does not introduce any mutual dependence between the term weights in a text, it will be especially useful in degraded text collections. This function has the goodness of the pivoted normalization functions (in terms of matching the relevance and the retrieval patterns), as well as the advantage that byte size based normalization has in corrupt text collections.

5.9 Conclusions

As a result of misrecognized words, an OCR system introduces meaningless garbage strings in a text collection. These garbage strings widely corrupt the cosine normalization factor, commonly used in term weighting. This corruption prohibits the use of cosine normalization in OCR environments. Byte size normalization is an attractive alternative due to its robustness to OCR errors. When used for the corrupted text, byte size normalization produces results more similar to the results from the correct text collection. Combining byte size normalization with pivoted normalization, we obtain a new normalization technique, the pivoted byte size normalization. The pivoted byte size normalization has retrieval effectiveness comparable to the pivoted unique normalization scheme and due to its use of the byte size of a document, it will be more robust in degraded text environments.

These results also show us that the use of cosine normalization is not advisable even on correct text collections, especially when the inverse document frequency factor is being used in term weights. Normalization schemes that do not depend upon the individual term weights to compute the normalization factor for a text are not only robust in the degraded text environments, they also facilitate the use of the inverse document frequency factor in document term weighting in correct text environments. This is another strong argument for replacing any cosine based normalization function, like pivoted-cosine normalization, by normalization schemes that do not introduce any mutual dependence between term weights of a text, for example uniue based normalization or byte-size based normalization. This advantage of pivoted unique normalization or pivoted byte-size normalization over pivoted cosine normalization is very important in situations where using the *idf* factor is necessary in term weights.

Chapter 6

The Inverse Document Frequency (IDF) Factor

Different words have different *content value* for retrieval. A match between a query and an article on a word with a specific meaning (for example, the word "telecommunications") makes the article potentially more interesting for a user as opposed to a match on a common word (for example, the word "common"). The inverse document frequency (*idf*) factor is used in term weighting to embody this *differential importance* of terms. For years, the formulation:

$$log(\frac{N}{df})$$

(where N is the total number of articles in the collection, and df is the number of articles that contain the word under consideration) has been used as the *idf* factor in term weights.

In a small study of the *idf* function, we investigate its properties and observe that the usefulness of the *idf* factor increases as we go down in document ranking. If the above *idf* formulation is replaced by a stronger *idf* formulation (say idf^2), ranks of low-ranked relevant documents improve noticeably. We observe that the query terms can generally be grouped into the *rare terms*, and the *non-rare terms*. We show that most of the improvements obtained by using a stronger formulation of *idf* (*e.g.*, idf^2) are due to the large increments that result in the weights of the rare terms. We believe that this behavior is possibly due to the fact that rare terms supplied by a user as part of a query are pivotal to the user's notion of relevance. The rare terms can also be called the *core query terms*, without which the potential relevance of a document is questionable.

We believe that the results of this study are very preliminary. The main objective of our work is to observe some properties of the *idf* function that have not been observed before. We believe that the *idf* function needs to be studied much more before anything conclusive could be said about the effects it has on document ranking.

6.1 Introduction

When a user types a natural language query into an IR system, certain keywords in the query are more pertinent to the user's information need than others. For example, consider the query:

What is the economic impact of recycling tires?

It is evident just by reading the query that the concept *recycling* is more relevant to the user's needs as compared to the concept *impact*; mainly because the concept *recycling* is more specific than the concept *impact*. An information retrieval system should incorporate such distinctions in its retrieval algorithms. In the above example, a match on the word *recycling* should contribute more towards a document's similarity than a match on the term *impact*.

Most modern IR systems incorporate such distinctions by using an inverse document frequency factor (commonly known as the *idf* factor) in term weighting. Motivated by Zipf's law, [Zip49] the *idf* factor is

based upon the hypothesis that the specificity of a term is inversely related to the number of different articles that use the term. [SJ72] If a term is used by numerous articles, then there is a good chance that this term is relatively common, and vice-versa. Based on this hypothesis, an inverse formulation of the number of articles that contain a term, often called the document frequency (or the *df*) of the term, is used as the *idf* factor.

As the number of articles that use a term increase, the importance of the term, as predicted by the *idf* factor, decreases. Based on Zipfian distribution, Sparck Jones proposes the use of the following function to estimate a term's importance:

$$K - log(df)$$

where K is a constant, and *df* is the number of articles that contain the index term. [SJ73] Since *df* is always less than or equal to the total number of articles in the collection, the constant K can be assigned a value of *log(N)*, where N is the total number of articles in the collection. Doing this we obtain positive or zero *idf* values for all term[1]. This function, over the years, has become the well-known inverse document frequency function. Very similar functions to the above *idf* function were later derived by Robertson and Sparck Jones, and Croft and Harper based on some simple (and reasonable) approximations for relevance weighting of index terms in the probabilistic model. [RSJ76,CH79]

The above formulation, or some formulation very closely resembling the above formulation, has been used in most IR systems for automatically computing the relative usefulness of terms. Even though this formulation has been criticized by some researchers, and other formulations (that measure how the real distribution of a term varies from expected distribution under some stochastic model of term distribution) have been proposed in place of *idf*, [Chu95] for ranking of documents, no other automatic measure has been shown to yield better term weights. The need to automatically assign good relative weights to terms is not severe in systems where users are allowed to change the relative importance of terms manually. For this reason, manual systems do not rely heavily upon the *idf* factor for good term weighting. [Nel96]

In this study, we investigate the properties of the *idf* function to better understand the relationship between the differential nature of term importance and the occurrence characteristics of terms. To summarize our results, we find that the usefulness of an inverse function of term *df* increases at low ranks (poorly matching documents). In other words, strong inverse functions of the document frequency (for example idf^2) do a better job at ranking documents at low ranks. We show that the function $idf^{1.5}$ works better than the traditional *idf* function on three different sets of queries, and propose that this function be used in place of the traditional *idf* function in term weighting.

6.2 Observations

As the basic purpose of *idf* weighting is to deem important the terms that occur in fewer documents, one question can be asked: how much should we believe in an inverse function of the document frequency? The standard *idf* functions says, for example, that a term that is present in one-tenth of the articles in a collection ($idf = log(10) = 2.3026$) is about 3.32 times more important than a term that is present in half the articles in the collection ($idf = log(2) = 0.6932$). The general shape of the *idf* function is shown in Figure 6.1. We use the TREC collection's disks one and two, with 741,856 articles as our canonical collection. So in all our illustrations, $N = 741,856$.

We want to investigate if the relative importance assigned to different terms by the standard *idf* function can be improved upon. *I.e.*, as shown in the above example, is a term that occurs in one-tenth of the collection actually 3.32 times more important than a term that is present in half the articles; or is it more than 3.32 times more important, or less? If we use a weaker inverse function of the document frequency, for example $idf^{0.7}$, then, instead of being 3.32 times more important, a term that is present in one-tenth of the articles will be about 2.32 times more important than a term that is present in half the articles. On the other hand, if we use a stronger inverse function of the document frequency, for example idf^2, then the less frequent term ($df = \frac{N}{10}$) will be about 11 times more important than the more frequent term ($df = \frac{N}{2}$).

[1] To avoid the problem that a term present in all the articles gets a zero *idf* weight, the Smart system uses the formulation $log(\frac{N+1}{df})$ as the *idf* factor.

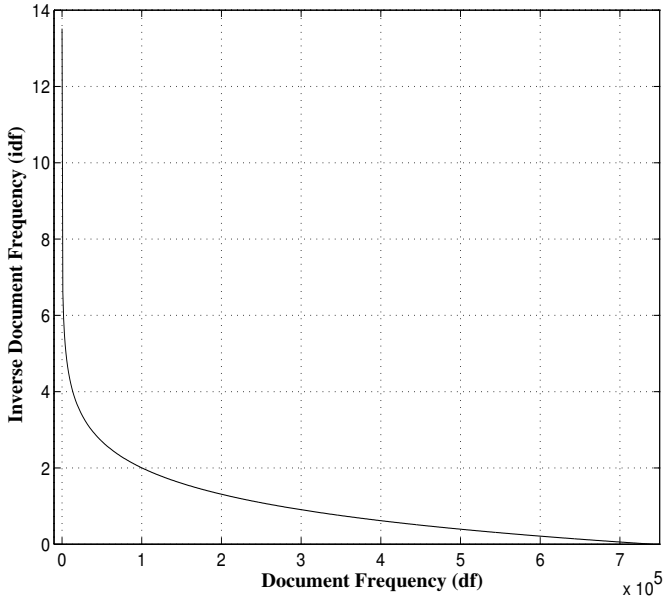


Figure 6.1: Shape of the *idf* function: $log(N/df)$

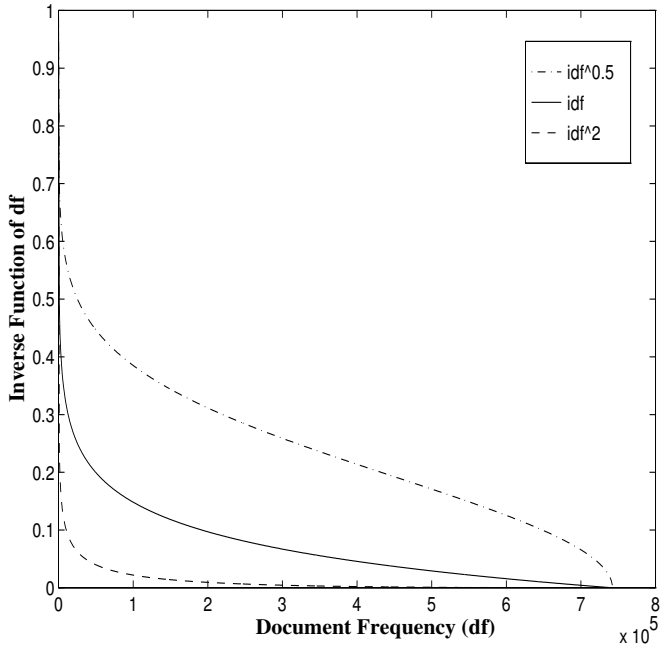


Figure 6.2: Shape of various powers of the *idf* function: $(log(N/df))^{\alpha}$

Table 6.1: TREC–3 ad-hoc task: Various powers of the *idf* function

TREC–3 ad-hoc task: Queries 151–200, disks 1 and 2. Total relevant: 9805				
α in idf^{α}	1.00	0.75	1.5	2.0
Rel. Ret.	6233	5610	7050	7090
Recall	Average Precision			
0.00	0.8832	0.8854(+0.2%)	0.8892(+0.7%)	0.8690(–1.6%)
0.10	0.6873	0.6733(–2.0%)	0.6618(–3.7%)	0.6306(–8.2%)
0.20	0.5674	0.5525(–2.6%)	0.5649(–0.4%)	0.5412(–4.6%)
0.30	0.4779	0.4640(–2.9%)	0.4805(+0.5%)	0.4641(–2.9%)
0.40	0.3956	0.3760(–5.0%)	0.4089(+3.4%)	0.4053(+2.4%)
0.50	0.3176	0.2976(–6.3%)	0.3425(+7.8%)	0.3383(+6.5%)
0.60	0.2466	0.2248(–8.8%)	0.2610(+5.8%)	0.2599(+5.4%)
0.70	0.1802	0.1536(–14.8%)	0.2066(+14.6%)	0.2145(+19.0%)
0.80	0.1122	0.0958(–14.7%)	0.1439(+28.2%)	0.1532(+36.5%)
0.90	0.0281	0.0271(–3.7%)	0.0324(+15.3%)	0.0466(+65.6%)
1.00	0.0000	0.0000(+0.0%)	0.0022(+Inf%)	0.0028(+Inf%)
Avg. Prec.	0.3356	0.3210	0.3456	0.3380
Improvement		–4.4%	+3.0%	+0.7%

After experimenting with various replacement functions for *idf*, we decided to further experiment with using various powers of the *idf* function in query term weights.[2] Shapes for some powers of the *idf* function are shown in Figure 6.2. We normalize all curves in Figure 6.2 to lie between zero and one for mutual comparison. Results of using various powers of the *idf* function on TREC queries 151–200 for retrieval from TREC disks one and two are shown in Table 6.1. We observe from Table 6.1 that using a weaker inverse function function of term document frequency is not better than using the standard *idf* function. We also observe that using a somewhat stronger function ($idf^{1.5}$) does yield improvements over the standard *idf* function.

We immediately observe in Table 6.1 that most of the improvements in average precision obtained by replacing the *idf* function by a higher power of the *idf* function ($idf^{1.5}$ and $idf^{2.0}$) are obtained at high recall levels (30% and above). A recall level of 30% or above occurs when 30% of the relevant document have been presented to the user. In a ranked system, this corresponds to a point where we have gone down in ranking enough so that at least 30% of the relevant document are ranked above this point. Usually this point does not occur at high ranks (top ranked document), instead it occurs in mid-ranks. In general, the low recall end corresponds to the top ranked documents and the high recall end corresponds to the low ranked documents.

In fact, by using a higher power of the *idf* function, the precision values at the low recall end are poorer than those for the standard *idf* function (recall level 10% and 20%). Also, this behavior becomes more and more pronounced as the power on the *idf* function is increased. For example, when we move from using $idf^{1.5}$ to using $idf^{2.0}$, the losses in precision at the low recall end increase (now we lose 8.2% in the average precision at 10% recall instead of losing just 3.7% for $idf^{1.5}$). Also, in moving from $idf^{1.5}$ to $idf^{2.0}$, the improvements obtained at the high recall levels are bigger. For example, the improvement in average precision at 80% recall increases from 28.2% to 36.5%.

It is well known that most users of IR systems provide very short queries to a system, usually just a few words. [CCW95] When queries are short, good relative weighting of the query terms becomes quite important for good retrieval. [Kwo96] Short queries provide an IR system with little information on the user's information need. Recognizing which terms out of the few terms given by the user are pivotal to relevance can boost the performance of a retrieval system significantly. [KG96] When queries are long and have many concepts that pertain to a user's information need, the need to do good relative weighting of query terms might not be that severe. The presence of numerous different concepts in a long query automatically boosts

[2] Notice that due to our use of the pivoted–unique normalization function, there is no mutual dependence between term weights, and using the *idf* factor on query term weights (and not on the document term weights) has the same effect as if it were used on the document term weights (and not on the query term weights).

Table 6.2: Number of distinct terms in short version of queries

Topic Numbers	TREC–2 101–150	TREC–3 151–200	TREC–4 202–250
Average number of distinct terms per query	9.42	11.26	7.96

Table 6.3: TREC–2 ad-hoc task: Powers of *idf* on short queries

TREC–2 ad-hoc task: Queries 101–150, disks 1 and 2. Total relevant: 11628				
α in idf^{α}	1.00	0.75	1.5	2.0
Rel. Ret.	6139	5794	6503	6503
Recall	Average Precision			
0.00	0.6589	0.6475(–1.7%)	0.6365(–3.4%)	0.6185(–6.1%)
0.10	0.4328	0.4264(–1.5%)	0.4394(+1.5%)	0.4235(–2.1%)
0.20	0.3703	0.3516(–5.0%)	0.3794(+2.5%)	0.3714(+0.3%)
0.30	0.3024	0.2773(–8.3%)	0.3231(+6.8%)	0.3214(+6.3%)
0.40	0.2481	0.2199(–11.4%)	0.2730(+10.0%)	0.2729(+10.0%)
0.50	0.1958	0.1564(–20.1%)	0.2250(+14.9%)	0.2197(+12.2%)
0.60	0.1291	0.1077(–16.6%)	0.1531(+18.6%)	0.1533(+18.7%)
0.70	0.0679	0.0508(–25.2%)	0.0959(+41.2%)	0.1009(+48.6%)
0.80	0.0347	0.0271(–22.0%)	0.0538(+54.8%)	0.0674(+94.1%)
0.90	0.0108	0.0084(–22.1%)	0.0180(+66.1%)	0.0239(+121.2%)
1.00	0.0000	0.0000(+0.0%)	0.0027(+Inf%)	0.0032(+Inf%)
Avg. Prec. Improvement	0.2026	0.1852 –8.6%	0.2174 +7.3%	0.2150 +6.1%

the matching performance of a retrieval system.

We test the effectiveness of the relative weights assigned to the terms of short queries by various powers of the *idf* function. The TREC queries have a clearly demarked "description" section which is a one sentence description of the user's information need. Using only the description section of the queries for retrieval mimics what real users might give to an IR system as a statement of their information need. Motivated by this, in the most recent TREC conference (TREC–4), the participants were only provided with these one-sentence *description only* queries. [Har96] The number of different terms used in three different sets of these short queries is shown in Table 6.2[3]. The full length TREC–3 queries (queries 151–200) have an average of 33.34 terms per query[4]. It is clear that the short queries we use in our experiments are much smaller, and are closer to (actually, still a little longer than) what a user might type as a query for an IR system.

Tables 6.3–6.5 show the results of using various powers of the *idf* function on short TREC queries. The observations from these three sets of queries are strikingly similar. We observe from Tables 6.3–6.5 that:

1. $idf^{0.75}$: Using a weaker inverse function of document frequency ($idf^{0.75}$) than the standard *idf* function for assigning importance to the query terms is never useful. There is a loss in average precision at every recall level across query sets. We also have poorer performance in terms of the overall non-interpolated average precision. But we do observe that in all the three query sets, the losses in average precision are more marked at the high recall levels (30% and above). This indicates that using a weaker inverse function of document frequency than the standard *idf* function is more hurtful at low ranks, reinforcing

[3] We only consider single terms, not phrases, to compute the averages.

[4] Once again, we only consider single terms, not phrases, to compute the average.

Table 6.4: TREC–3 ad-hoc task: Powers of *idf* on short queries

TREC–3 ad-hoc task: Queries 151–200, disks 1 and 2. Total relevant: 9805				
α in idf^{α}	1.00	0.75	1.5	2.0
Rel. Ret.	5387	5144	5684	5678
Recall	Average Precision			
0.00	0.8054	0.7981(–0.9%)	0.7997(–0.7%)	0.7650(–5.0%)
0.10	0.5308	0.5201(–2.0%)	0.5162(–2.8%)	0.4825(–9.1%)
0.20	0.4123	0.3962(–3.9%)	0.4250(+3.1%)	0.4076(–1.1%)
0.30	0.3462	0.3247(–6.2%)	0.3688(+6.5%)	0.3563(+2.9%)
0.40	0.2706	0.2527(–6.6%)	0.2871(+6.1%)	0.2939(+8.6%)
0.50	0.2032	0.1859(–8.5%)	0.2332(+14.8%)	0.2358(+16.1%)
0.60	0.1412	0.1298(–8.0%)	0.1696(+20.1%)	0.1742(+23.4%)
0.70	0.1070	0.0935(–12.6%)	0.1188(+11.0%)	0.1275(+19.2%)
0.80	0.0681	0.0559(–17.8%)	0.0764(+12.2%)	0.0798(+17.3%)
0.90	0.0229	0.0215(–6.4%)	0.0200(–12.8%)	0.0248(+8.1%)
1.00	0.0000	0.0000(+0.0%)	0.0000(+0.0%)	0.0000(+0.0%)
Avg. Prec.	0.2380	0.2251	0.2499	0.2449
Improvement		–5.4%	+5.0%	+2.9%

Table 6.5: TREC–4 ad-hoc task: Powers of *idf* in query term weights

TREC–4 ad-hoc task: Queries 202–250, disks 2 and 3. Total relevant: 6501				
α in idf^{α}	1.00	0.75	1.5	2.0
Rel. Ret.	3709	3591	3805	3782
Recall	Average Precision			
0.00	0.7150	0.7089(–0.9%)	0.7030(–1.7%)	0.6746(–5.6%)
0.10	0.4794	0.4784(–0.2%)	0.4755(–0.8%)	0.4634(–3.3%)
0.20	0.4100	0.4054(–1.1%)	0.4108(+0.2%)	0.3929(–4.2%)
0.30	0.3457	0.3279(–5.2%)	0.3565(+3.1%)	0.3419(–1.1%)
0.40	0.2735	0.2628(–3.9%)	0.2907(+6.3%)	0.2903(+6.1%)
0.50	0.2223	0.2003(–9.9%)	0.2377(+6.9%)	0.2413(+8.5%)
0.60	0.1591	0.1474(–7.4%)	0.1738(+9.2%)	0.1810(+13.8%)
0.70	0.0964	0.0847(–12.2%)	0.1106(+14.7%)	0.1113(+15.4%)
0.80	0.0507	0.0396(–21.9%)	0.0631(+24.5%)	0.0700(+38.1%)
0.90	0.0189	0.0152(–19.3%)	0.0321(+70.2%)	0.0361(+91.4%)
1.00	0.0016	0.0013(–20.6%)	0.0030(+81.7%)	0.0024(+47.9%)
Avg. Prec.	0.2326	0.2233	0.2388	0.2332
Improvement		–4.0%	+2.6%	+0.3%

our observation that the use of *idf* is more effective at low ranks (poorly matching documents). Using a weaker inverse function in place of the standard *idf* function, the losses at low recall levels, *i.e.*, the high precision end or the highly ranked documents (recall levels 10% and 20%), are minimal.

2. $idf^{1.5}$: Using a stronger inverse function of document frequency ($idf^{1.5}$) results in better retrieval effectiveness than using the standard *idf* function on all three query sets. The improvements obtained in retrieval effectiveness are more marked for the TREC–2 and the TREC–3 queries (7.3% and 5.0%, respectively) than the improvements obtained for the TREC–4 queries (2.6%). Still, the pattern of improvements at various recall level is similar — marginal losses at low recall end, and noticeable improvements at high recall levels. Once again, these results show that an *idf* type function is more effective at low ranks. Using a strong *idf* type function can actually result in loss of precision at high ranks (low recall or high precision) end.

3. $idf^{2.0}$: Using an even stronger inverse function of *df* ($idf^{2.0}$) than $idf^{1.5}$ further emphasizes our observation that an *idf* type function is more effective at low ranks. When we switch to using $idf^{2.0}$ in place of $idf^{1.5}$, for all three query sets, the improvements in average precision at high recall end (50% and above) are further boosted. But the losses at the low recall end also increase, yielding a poorer overall performance than $idf^{1.5}$ in terms of average precision.

6.3 Analysis

It is true that the relative importance of query terms depends largely upon the user's notion of relevance, and is not always optimally assessed by an inverse function of the document frequency. For example, consider the TREC–4 query (number 244):

Status of trade balance with Japan – deficit problem.

For this query, it is most likely that a document not related to *Japan* or *trade* will not be relevant. But both these words are more prevalent in the collection (have higher *df*, thus lower *idf*) than the words *status*, *balance*, and *deficit*. Therefore, for this query, emphasizing *idf* by using $idf^{1.5}$ (instead of just the standard *idf*) amounts to saying that the terms *status*, *balance*, and *deficit* are much more important than the terms *Japan* and *trade*; which, of course, is not true, and is hurtful (the average precision falls from 0.4112 to 0.3188, a loss of 22.5%).

On the other hand, consider the query (number 216):

What research is ongoing to reduce the effects of osteoporosis in existing patients as well as prevent the disease occurring in those unafflicted at this time?

For this query, the two high *idf* terms are *osteoporosis* and *unafflicted*, and both are quite important concepts for relevance. For this query, the user's notion of term importance agrees very well with an inverse function of document frequency, and using stronger powers of *idf* are beneficial for this query (switching from *idf* to $idf^{1.5}$ changes the average precision from 0.2953 to 0.4207, an improvement of 42.4%; moving to $idf^{2.0}$ from $idf^{1.5}$ further boosts the average precision to 0.5089, an overall improvement of 72.3% over standard *idf*).

When a large collection of queries is used for analysis, biases from individual queries are averaged out and stable patterns, that hold in general across queries, emerge in the retrieval results. Tables 6.3–6.5 show that, in general, the usefulness of an inverse function of document frequency is pronounced at high recall levels. The question than arises: *why is using a strong inverse function of term df than (in place of the standard* idf *function) resulting in improved ranking at high recall levels?*.

One main effect of using the function $idf^{1.5}$ in place of the standard *idf* function is that the function $idf^{1.5}$ tremendously boosts the *idf* weights of the very rare terms. (Since we are using $idf^{1.5}$, *idf* values that are initially high get a very high increase due to the extra multiplication by $\sqrt{idf}$.) A rare term that occurs in only 2% of the documents gets a 97.79% increase in its *idf* based importance, whereas the importance of a common term that occurs in 20% of the articles increases by only 26.86%. This boost, along with the fact that the rare terms initially have quite high *idf* weights, results in a much wider gap between the *idf* weights for the rare terms, and the non-rare terms. On the other hand, the relative weights of two non-rare

Table 6.6: TREC–2 ad-hoc task: Weights of rare terms boosted by 50%

TREC–2 ad-hoc task: Queries 101–150, disks 1 and 2. Total relevant: 11628 (Average: 9.42 terms/query)						
	idf	$idf^{1.5}$	*rare*=1.0%	*rare*=1.5%	*rare*=2.0%	*rare*=2.5%
Avg. # of Terms Boosted			1.2 terms/q	1.8 terms/q	2.1 terms/q	2.4 terms/q
	6139	6503	6416	6540	6517	6523
Rcl.	Average Precision					
0.00	0.6589	0.6365(−3.4%)	0.6460(−2.0%)	0.6560(−0.4%)	0.6531(−0.9%)	0.6578(−0.2%)
0.10	0.4328	0.4394(+1.5%)	0.4387(+1.4%)	0.4460(+3.1%)	0.4389(+1.4%)	0.4412(+1.9%)
0.20	0.3703	0.3794(+2.5%)	0.3737(+0.9%)	0.3774(+1.9%)	0.3775(+1.9%)	0.3796(+2.5%)
0.30	0.3024	0.3231(+6.8%)	0.3193(+5.6%)	0.3234(+6.9%)	0.3244(+7.3%)	0.3244(+7.3%)
0.40	0.2481	0.2730(+10.0%)	0.2669(+7.6%)	0.2718(+9.6%)	0.2728(+10.0%)	0.2762(+11.3%)
0.50	0.1958	0.2250(+14.9%)	0.2194(+12.1%)	0.2260(+15.4%)	0.2242(+14.5%)	0.2157(+10.2%)
0.60	0.1291	0.1531(+18.6%)	0.1527(+18.3%)	0.1633(+26.4%)	0.1616(+25.1%)	0.1571(+21.6%)
0.70	0.0679	0.0959(+41.2%)	0.0946(+39.4%)	0.0937(+38.0%)	0.0924(+36.0%)	0.0915(+34.7%)
0.80	0.0347	0.0538(+54.8%)	0.0544(+56.5%)	0.0474(+36.3%)	0.0433(+24.5%)	0.0426(+22.5%)
0.90	0.0108	0.0180(+66.1%)	0.0165(+52.1%)	0.0162(+49.4%)	0.0156(+44.7%)	0.0162(+49.6%)
1.00	0.0000	0.0027(+Inf%)	0.0023(+Inf%)	0.0023(+Inf%)	0.0023(+Inf%)	0.0023(+Inf%)
	0.2026	0.2174 +7.3%	0.2157 +6.4%	0.2184 +7.8%	0.2174 +7.3%	0.2172 +7.2%

Table 6.7: TREC–3 ad-hoc task: Weights of rare terms boosted by 50%

TREC–3 ad-hoc task: Queries 151–200, disks 1 and 2. Total relevant: 9805 (Average: 11.26 terms/query)						
	idf	$idf^{1.5}$	*rare*=1.0%	*rare*=1.5%	*rare*=2.0%	*rare*=2.5%
Avg. # of Terms Boosted			1.80 terms/q	2.56 terms/q	3.26 terms/q	3.70 terms/q
	5387	5684	5497	5603	5774	5736
Rcl.	Average Precision					
0.00	0.8054	0.7997(−0.7%)	0.7912(−1.8%)	0.8069(+0.2%)	0.8091(+0.5%)	0.8089(+0.4%)
0.10	0.5308	0.5162(−2.8%)	0.5044(−5.0%)	0.5154(−2.9%)	0.5332(+0.4%)	0.5292(−0.3%)
0.20	0.4123	0.4250(+3.1%)	0.4074(−1.2%)	0.4256(+3.2%)	0.4358(+5.7%)	0.4346(+5.4%)
0.30	0.3462	0.3688(+6.5%)	0.3535(+2.1%)	0.3602(+4.1%)	0.3693(+6.7%)	0.3696(+6.8%)
0.40	0.2706	0.2871(+6.1%)	0.2758(+1.9%)	0.2808(+3.8%)	0.2860(+5.7%)	0.2865(+5.9%)
0.50	0.2032	0.2332(+14.8%)	0.2222(+9.4%)	0.2237(+10.1%)	0.2312(+13.8%)	0.2258(+11.1%)
0.60	0.1412	0.1696(+20.1%)	0.1583(+12.1%)	0.1650(+16.9%)	0.1658(+17.4%)	0.1578(+11.8%)
0.70	0.1070	0.1188(+11.0%)	0.1168(+9.1%)	0.1229(+14.8%)	0.1220(+14.1%)	0.1210(+13.1%)
0.80	0.0681	0.0764(+12.2%)	0.0710(+4.3%)	0.0734(+7.8%)	0.0712(+4.6%)	0.0689(+1.2%)
0.90	0.0229	0.0200(−12.8%)	0.0197(−14.2%)	0.0210(−8.4%)	0.0213(−7.0%)	0.0204(−10.8%)
1.00	0.0000	0.0000(+0.0%)	0.0000(+0.0%)	0.0000(+0.0%)	0.0000(+0.0%)	0.0000(+0.0%)
	0.2380	0.2499 +5.0%	0.2415 +1.5%	0.2481 +4.2%	0.2534 +6.4%	0.2505 +5.2%

terms do not change substantially enough to make the impressive difference in the recall–precision values we observe in Tables 6.3–6.5.

It is possible that by using a higher power of the *idf* function, we are simply obtaining the benefits from a widened gap between the weights of the rare terms and the weights of the non-rare terms. To test this hypothesis, instead of using the function $idf^{1.5}$, we experiment with simply boosting the weights of the rare term. We leave the weights of the non-rare terms unchanged. We call this technique "boosting". Boosting, like $idf^{1.5}$, also widens the gap between the weights of the rare terms and the weights of the non-rare terms in a query. We boost the weights of the rare terms by 50% (by multiplying their weights by 1.5). A term is considered *rare*, if it occurs in fewer that (say) 2% of the articles in the collection.

The results from boosting the weights of the rare query terms by 50% are shown in Tables 6.6–6.8. A term is considered to be rare if it occurs in fewer than X% of the articles in the collection. Results for four different cut-off values (X) are shown in Tables 6.6–6.8: 1%, 1.5%, 2%, and 2.5%. These results are remarkably similar to the results obtained by using the function $idf^{1.5}$ in query term weights (see the third column in Tables 6.6–6.8). For example, for all three query sets, if we define a term to be rare if it occurs in fewer than 1.5% of the articles (fewer than 11,128 articles out of a total of 741,856 articles for TREC–2 and TREC–3; and fewer than 8,511 articles out of a total of 567,419 articles for TREC–4), then the improvements obtained by boosting the weights of the rare query terms by 50% are comparable to the improvements obtained by using the function $idf^{1.5}$. Also, the pattern of improvements across various recall

Table 6.8: TREC–4 ad-hoc task: Weights of rare terms boosted by 50%

TREC–4 ad-hoc task: Queries 202–250, disks 2 and 3. Total relevant: 6501 (Average: 7.96 terms/query)						
	idf	$idf^{1.5}$	*rare*=1.0%	*rare*=1.5%	*rare*=2.0%	*rare*=2.5%
Avg. # of Terms Boosted			1.27 terms/q	1.57 terms/q	2.06 terms/q	2.37 terms/q
	3709	3805	3746	3832	3796	3800
Rcl.	Average Precision					
0.00	0.7150	0.7030(−1.7%)	0.6963(−2.6%)	0.6994(−2.2%)	0.7033(−1.6%)	0.6985(−2.3%)
0.10	0.4794	0.4755(−0.8%)	0.4733(−1.3%)	0.4828(+0.7%)	0.4783(−0.2%)	0.4793(−0.0%)
0.20	0.4100	0.4108(+0.2%)	0.4069(−0.7%)	0.4124(+0.6%)	0.4070(−0.7%)	0.4139(+1.0%)
0.30	0.3457	0.3565(+3.1%)	0.3547(+2.6%)	0.3559(+2.9%)	0.3499(+1.2%)	0.3538(+2.3%)
0.40	0.2735	0.2907(+6.3%)	0.2908(+6.3%)	0.2918(+6.7%)	0.2804(+2.5%)	0.2857(+4.5%)
0.50	0.2223	0.2377(+6.9%)	0.2350(+5.7%)	0.2370(+6.6%)	0.2274(+2.3%)	0.2297(+3.3%)
0.60	0.1591	0.1738(+9.2%)	0.1783(+12.1%)	0.1721(+8.2%)	0.1636(+2.8%)	0.1642(+3.2%)
0.70	0.0964	0.1106(+14.7%)	0.1100(+14.0%)	0.1041(+7.9%)	0.0913(−5.3%)	0.0874(−9.4%)
0.80	0.0507	0.0631(+24.5%)	0.0633(+24.8%)	0.0586(+15.7%)	0.0554(+9.2%)	0.0560(+10.4%)
0.90	0.0189	0.0321(+70.2%)	0.0314(+66.7%)	0.0290(+54.1%)	0.0273(+45.0%)	0.0273(+45.0%)
1.00	0.0016	0.0030(+81.7%)	0.0032(+97.0%)	0.0032(+97.0%)	0.0015(−7.7%)	0.0015(−7.7%)
	0.2326	0.2388 +2.6%	0.2392 +2.8%	0.2392 +2.8%	0.2336 +0.4%	0.2341 +0.4%

levels is the same — some losses at low recall levels but large improvements at high recall level.

Not only are the overall average improvements obtained over standard idf by using $idf^{1.5}$ and by boosting the weights of the rare terms by 50% comparable, the magnitude of the improvements at various recall levels is also amazingly similar. For example, at 40% recall level for TREC–2 queries, switching from standard idf to $idf^{1.5}$ yields a 10% improvement. For the same data point, the improvements from boosting the weights of the rare terms are between 7.6% and 11.3%. Such closeness of improvements is observed in all three query sets across recall levels. Also, the total number of relevant documents retrieved by using $idf^{1.5}$ are very similar to the total number of relevant documents retrieved by boosting the weights of the rare terms. For example, $idf^{1.5}$ retrieves a total of 5,684 relevant articles for TREC–3 queries, and the best boosting result retrieves a total of 5,774 relevant articles. Overall, the results obtained from boosting the weights of the rare terms closely resemble the results obtained by using $idf^{1.5}$.

On doing a query by query comparison of boosting and $idf^{1.5}$, we find that on an individual query basis, the results are also very similar. We also observe that if we consider terms that occur in fewer than 1.5% of the collection as the rare terms, and boost their query weights, on an average only about two terms per query get their weights boosted. By simply incrementing the weights of the (on an average) two most rare terms in a query by 50% we get very similar results as the results obtained by using $idf^{1.5}$ in place of the standard idf function.

On a detailed query by query analysis we observe:

- Thirty four out of the fifty TREC–2 queries benefit by the use of $idf^{1.5}$ instead of the standard idf function (see Table 6.9). Thirty of these thirty four queries are also helped by boosting weights of the rare terms. The corresponding improvements are also quite similar. Four queries that benefit by using $idf^{1.5}$ did lose in terms of average precision by boosting (queries 101, 102, 114, and 143), but the differences for three of these four queries are very small. Only query 143 shows a reasonable improvement by the use of $idf^{1.5}$, but loses noticeably by boosting.

  Similarly, sixteen queries lose in terms of average precision by the use of $idf^{1.5}$. Twelve out of these sixteen also post similar losses by boosting. The four queries that do not (queries 127, 139, 141, and 142) have very little difference in their average precisions for $idf^{1.5}$ and boosting.

- Thirty out of the fifty TREC–3 queries benefit by the use of $idf^{1.5}$ instead of the standard idf function (see Table 6.10). Twenty three of these thirty are also helped by boosting weights of the rare terms. Out of the twenty queries that lose by the use of $idf^{1.5}$, sixteen also lose due to boosting.

- Out of the forty nine TREC–4 queries, twenty seven benefit by the use of $idf^{1.5}$ in place of the standard idf function (see Table 6.11). Twenty two out of these twenty seven queries also benefit by boosting. The remaining five queries (210, 220, 223, 234, and 241) gain by using $idf^{1.5}$, but lose by boosting the

Table 6.9: TREC–2: Query by query comparison of idf, $idf^{1.5}$, and boosting

qid	idf	$idf^{1.5}$	Boosting	qid	idf	$idf^{1.5}$	Boosting
101	0.051	0.053 (+3.2%)	0.049 (−3.4%)	102	0.068	0.080 (+18.2%)	0.067 (−0.9%)
103	0.115	0.187 (+62.3%)	0.187 (+62.5%)	104	0.139	0.173 (+23.9%)	0.179 (+28.8%)
105	0.049	0.042 (−12.5%)	0.045 (−6.6%)	106	0.256	0.255 (−0.5%)	0.251 (−2.0%)
107	0.309	0.297 (−3.6%)	0.284 (−7.9%)	108	0.174	0.132 (−24.0%)	0.138 (−20.4%)
109	0.397	0.429 (+8.0%)	0.421 (+5.8%)	110	0.328	0.330 (+0.6%)	0.337 (+2.7%)
111	0.272	0.283 (+4.2%)	0.301 (+10.9%)	112	0.049	0.136 (+178.8%)	0.126 (+158.7%)
113	0.044	0.063 (+42.1%)	0.070 (+57.5%)	114	0.149	0.150 (+0.6%)	0.149 (−0.4%)
115	0.271	0.283 (+4.4%)	0.293 (+8.1%)	116	0.134	0.136 (+2.1%)	0.147 (+9.7%)
117	0.227	0.250 (+10.2%)	0.265 (+16.7%)	118	0.176	0.230 (+30.3%)	0.260 (+47.3%)
119	0.131	0.173 (+32.0%)	0.199 (+52.1%)	120	0.007	0.010 (+45.5%)	0.016 (+117.3%)
121	0.004	0.004 (+16.1%)	0.004 (+16.6%)	122	0.158	0.171 (+8.2%)	0.179 (+13.0%)
123	0.357	0.380 (+6.3%)	0.405 (+13.4%)	124	0.156	0.126 (−19.3%)	0.142 (−9.4%)
125	0.079	0.118 (+48.8%)	0.128 (+61.4%)	126	0.130	0.124 (−5.1%)	0.117 (−10.2%)
127	0.115	0.111 (−3.2%)	0.115 (+0.3%)	128	0.059	0.038 (−35.2%)	0.047 (−20.3%)
129	0.125	0.091 (−27.3%)	0.104 (−17.0%)	130	0.507	0.532 (+4.9%)	0.550 (+8.4%)
131	0.079	0.063 (−19.6%)	0.056 (−28.9%)	132	0.380	0.513 (+34.8%)	0.499 (+31.3%)
133	0.693	0.706 (+1.9%)	0.701 (+1.3%)	134	0.642	0.677 (+5.4%)	0.663 (+3.2%)
135	0.500	0.529 (+5.8%)	0.530 (+5.9%)	136	0.296	0.347 (+17.1%)	0.324 (+9.5%)
137	0.410	0.473 (+15.4%)	0.442 (+8.0%)	138	0.070	0.070 (+0.8%)	0.074 (+5.5%)
139	0.143	0.137 (−4.0%)	0.143 (+0.3%)	140	0.100	0.108 (+7.8%)	0.101 (+0.8%)
141	0.044	0.043 (−2.2%)	0.045 (+1.1%)	142	0.006	0.004 (−26.9%)	0.007 (+18.9%)
143	0.104	0.123 (+18.5%)	0.075 (−27.9%)	144	0.021	0.010 (−51.2%)	0.018 (−14.2%)
145	0.293	0.328 (+12.0%)	0.321 (+9.3%)	146	0.191	0.211 (+10.4%)	0.222 (+16.1%)
147	0.017	0.009 (−47.0%)	0.014 (−17.6%)	148	0.696	0.716 (+2.9%)	0.701 (+0.7%)
149	0.081	0.077 (−4.6%)	0.080 (−1.4%)	150	0.326	0.334 (+2.6%)	0.330 (+1.1%)

Table 6.10: TREC–3: Query by query comparison of idf, $idf^{1.5}$, and boosting

qid	idf	$idf^{1.5}$	Boosting	qid	idf	$idf^{1.5}$	Boosting
151	0.593	0.623 (+5.1%)	0.610 (+2.9%)	152	0.023	0.014 (−39.7%)	0.013 (−42.8%)
153	0.205	0.201 (−2.0%)	0.193 (−5.7%)	154	0.474	0.478 (+0.9%)	0.483 (+1.8%)
155	0.088	0.083 (−4.7%)	0.077 (−12.1%)	156	0.300	0.441 (+47.0%)	0.514 (+71.4%)
157	0.237	0.282 (+18.9%)	0.234 (−1.6%)	158	0.229	0.175 (−23.6%)	0.200 (−12.7%)
159	0.271	0.284 (+4.7%)	0.266 (−2.0%)	160	0.157	0.221 (+40.7%)	0.219 (+39.4%)
161	0.493	0.511 (+3.6%)	0.503 (+1.8%)	162	0.070	0.097 (+37.9%)	0.037 (−47.3%)
163	0.807	0.825 (+2.2%)	0.813 (+0.8%)	164	0.305	0.308 (+1.2%)	0.307 (+0.9%)
165	0.160	0.157 (−1.9%)	0.145 (−9.6%)	166	0.356	0.277 (−22.4%)	0.273 (−23.3%)
167	0.123	0.114 (−7.6%)	0.098 (−20.2%)	168	0.134	0.150 (+11.9%)	0.140 (+4.6%)
169	0.105	0.120 (+13.9%)	0.125 (+18.6%)	170	0.697	0.689 (−1.0%)	0.700 (+0.4%)
171	0.054	0.050 (−8.9%)	0.043 (−20.5%)	172	0.065	0.068 (+4.8%)	0.065 (−0.1%)
173	0.498	0.599 (+20.3%)	0.600 (+20.4%)	174	0.302	0.368 (+22.0%)	0.358 (+18.8%)
175	0.234	0.249 (+6.6%)	0.260 (+11.2%)	176	0.313	0.380 (+21.5%)	0.360 (+15.2%)
177	0.221	0.195 (−11.9%)	0.254 (+14.9%)	178	0.364	0.386 (+6.1%)	0.343 (−5.9%)
179	0.093	0.078 (−16.4%)	0.082 (−11.3%)	180	0.143	0.163 (+14.0%)	0.160 (+11.8%)
181	0.053	0.054 (+1.8%)	0.059 (+10.7%)	182	0.123	0.152 (+23.4%)	0.138 (+11.7%)
183	0.169	0.201 (+19.1%)	0.161 (−4.4%)	184	0.263	0.257 (−2.3%)	0.264 (+0.4%)
185	0.323	0.422 (+30.8%)	0.430 (+33.1%)	186	0.124	0.111 (−10.2%)	0.113 (−8.3%)
187	0.049	0.086 (+73.9%)	0.048 (−2.3%)	188	0.241	0.255 (+5.6%)	0.257 (+6.3%)
189	0.177	0.203 (+14.9%)	0.214 (+20.6%)	190	0.054	0.038 (−28.9%)	0.071 (+30.7%)
191	0.177	0.158 (−11.1%)	0.141 (−20.3%)	192	0.303	0.271 (−10.5%)	0.270 (−11.0%)
193	0.440	0.408 (−7.4%)	0.402 (−8.8%)	194	0.009	0.007 (−25.2%)	0.009 (−9.6%)
195	0.083	0.057 (−31.0%)	0.082 (−1.5%)	196	0.442	0.458 (+3.5%)	0.457 (+3.3%)
197	0.211	0.236 (+11.6%)	0.219 (+3.7%)	198	0.224	0.244 (+9.0%)	0.308 (+37.6%)
199	0.140	0.150 (+7.2%)	0.142 (+2.0%)	200	0.178	0.139 (−22.3%)	0.144 (−19.1%)

Table 6.11: TREC–4: Query by query comparison of idf, $idf^{1.5}$, and boosting

qid	idf	$idf^{1.5}$	Boosting	qid	idf	$idf^{1.5}$	Boosting
202	0.554	0.601 (+8.4%)	0.607 (+9.5%)	203	0.109	0.125 (+14.3%)	0.129 (+18.2%)
204	0.211	0.204 (–3.7%)	0.198 (–6.2%)	205	0.002	0.003 (+36.1%)	0.003 (+35.3%)
206	0.007	0.003 (–63.4%)	0.003 (–55.5%)	207	0.535	0.525 (–1.8%)	0.526 (–1.7%)
208	0.176	0.163 (–7.2%)	0.154 (–12.4%)	209	0.216	0.221 (+2.4%)	0.223 (+3.3%)
210	0.605	0.639 (+5.7%)	0.603 (–0.3%)	211	0.115	0.090 (–21.6%)	0.115 (–0.4%)
212	0.245	0.215 (–12.4%)	0.218 (–11.3%)	213	0.298	0.248 (–16.8%)	0.290 (–2.8%)
214	0.611	0.640 (+4.8%)	0.641 (+4.9%)	215	0.511	0.540 (+5.8%)	0.536 (+4.9%)
216	0.295	0.421 (+42.4%)	0.421 (+42.6%)	217	0.391	0.472 (+20.5%)	0.472 (+20.7%)
218	0.122	0.113 (–7.1%)	0.114 (–6.4%)	219	0.152	0.142 (–6.5%)	0.144 (–5.4%)
220	0.514	0.531 (+3.2%)	0.511 (–0.6%)	221	0.302	0.322 (+6.6%)	0.337 (+11.5%)
222	0.306	0.328 (+7.3%)	0.329 (+7.5%)	223	0.078	0.086 (+10.1%)	0.059 (–25.1%)
224	0.305	0.302 (–0.8%)	0.304 (–0.2%)	225	0.643	0.664 (+3.2%)	0.656 (+2.0%)
226	0.017	0.019 (+7.7%)	0.025 (+44.6%)	227	0.163	0.130 (–20.2%)	0.129 (–21.0%)
228	0.078	0.074 (–5.6%)	0.088 (+12.8%)	229	0.467	0.465 (–0.4%)	0.468 (+0.2%)
230	0.131	0.101 (–22.7%)	0.057 (–56.3%)	231	0.086	0.072 (–16.1%)	0.076 (–11.0%)
232	0.006	0.007 (+31.2%)	0.006 (+0.0%)	233	0.040	0.042 (+6.7%)	0.042 (+6.1%)
234	0.551	0.578 (+4.8%)	0.490 (–11.1%)	235	0.283	0.322 (+13.7%)	0.298 (+5.2%)
236	0.008	0.006 (–25.7%)	0.002 (–73.8%)	237	0.232	0.228 (–1.9%)	0.254 (+9.4%)
238	0.304	0.328 (+7.9%)	0.356 (+16.9%)	239	0.013	0.036 (+170.6%)	0.044 (+228.2%)
240	0.061	0.112 (+82.0%)	0.137 (+123.0%)	241	0.011	0.011 (+1.7%)	0.009 (–15.8%)
242	0.130	0.109 (–16.3%)	0.102 (–21.6%)	243	0.127	0.143 (+12.4%)	0.134 (+5.3%)
244	0.411	0.319 (–22.5%)	0.356 (–13.4%)	245	0.181	0.179 (–1.1%)	0.193 (+6.7%)
246	0.175	0.171 (–2.5%)	0.184 (+4.6%)	247	0.316	0.328 (+3.8%)	0.319 (+1.2%)
248	0.036	0.062 (+71.5%)	0.084 (+131.7%)	249	0.188	0.192 (+2.5%)	0.206 (+9.8%)
250	0.076	0.067 (–12.6%)	0.066 (–13.3%)				

weights of the rare terms. The differences for four (210, 220, 223, and 241) out of these five queries are small.

Twenty two TREC–4 queries have poorer average precision as a result of using $idf^{1.5}$ in place of the standard idf function. Seventeen out of these twenty two queries post similar losses in average precision by boosting. Five queries (228, 229, 237, 245, and 246) do improve due to boosting.

These results show that the improvements obtained by the use of a stronger inverse function of document frequency over the use of the standard idf function are mainly due to the significant increase in the weights of the rare terms. On an average, increasing the weights of about two rare terms in the query yields improvements very similar to the improvements obtained by the use of the function $idf^{1.5}$. If we boost the weights of the rare terms too much (say by 100% or more instead of 50%), the effects are similar to those of using a much stronger inverse function of document frequency (for example, idf^2 or idf^3). All these results indicate that by using a higher power of the standard idf function, the main advantages are obtained due to the large increase in the term weights of the rare term, which, in turn, results in a wider gap between the weights of the rare and the non-rare terms.

6.4 Discussion

All the retrieval results in the previous sections show that when there is not a very strong match (*e.g.*, all query terms being present in a document would constitute a strong match) between a query and the documents, the rare terms are a more reliable indicator of relevance than the non-rare terms. When a user types a rare term as part of his/her query, then the presence of the rare term in an article is often quite significant in determining relevance. We believe that the rare terms form the *main topic* of query, and can be called the *core query terms*. All other terms, which can be called the *supporting terms*, form the context or the *sub-topic* within the main topic that the user is interested in. Chances that an article is relevant if the main topic of the query is absent from the article are quite poor. On the other hand, there is a good chance that an article might interest a user even when some of the supporting terms are missing.

If most of the supporting terms are present alongside the core terms, then promoting documents that have high weights for the core terms but are missing the supporting terms is hurtful. This is what we observe

at the low recall end of our recall-precision results. At the low recall end (or the high precision end), most articles are well related to the theme of the query, *i.e.*, they have most of the query terms (both core terms and the supporting terms) occurring together. If we increase the weights of the rare terms in the query, then some articles that have high term frequency for the core query terms but are missing the supporting terms will out-rank articles that have somewhat lower weights for the core terms but do have the supporting terms. This reversal of ranks replaces articles that are relevant to a query with a high probability with articles that are relevant with lower chances, and hurts precision at low recall levels.

When we consider articles that are not highly related to the query, *i.e.*, they are missing some query terms, then it is more important that the core query terms be present in an article. This can be observed in the large improvements obtained at the high recall end by boosting the weights of the rare terms. At the high recall end, the articles have a weak match to the query, and some query terms are missing from the articles. If we boost the weights of the rare terms, articles that are missing the supporting terms but do contain the core terms are promoted above the articles that contain almost all the supporting terms but are missing the core terms. Since the chances of relevance of an article that contains the core terms are higher than the chances of relevance for an article that is missing the core terms, this promotion of articles (containing the rare query terms) is beneficial at the lower ranks.

Recently Kwok has suggested that term repetition in long queries is a good source to estimate the relative importance of the query terms. As short queries seldom repeat terms, and do not give enough information to a retrieval system to reasonably estimate the relative importance of the query terms, human judgment should be used to selectively repeat the important query terms. [KG96] Kwok has also proposed a scheme for automatically selecting the query terms that should be repeated. [Kwo96] When we boost the weight of a query term using our techniques, it has a similar effect as repeating the query term. For if we had repeated the query term in the query, the query term frequency would be 2, and its weight would become $(1 + log(2)) \times idf = 1.69 \times idf$, which is the same as boosting the query term weight by about 70% (instead of the 50% that we use). We just use a much simpler algorithm (boost the weights of the rare terms) than the term selection algorithm presented in [Kwo96]. It will be interesting to see if we can apply Kwok's proposed scheme for term importance estimation in conjunction with a stronger inverse function of document frequency that we propose in this study.

Overall, we suggest that we replace the standard *idf* function with the function $idf^{1.5}$ for general term weight estimation. The function $idf^{1.5}$ yields important improvements over the use of the standard *idf* function in retrieval. We could also have used boosting to attain the same effect, but with boosting we have to experimentally estimate two parameters, the cut-off level at which a term should be considered rare, and the amount of boosting needed (we have arbitrarily used 50% in our experiments). Whereas with $idf^{1.5}$, we get the same effect by the use of just one parameter — the power of the *idf* function (1.5 seems to work well across queries).

These observations also suggest that a modified retrieval strategy can be used to obtain good ranking at all recall levels. If we rank articles using the standard *idf* function as long as all rare terms are present in an article, then switch to ranking by a stronger inverse function of the document frequency (for example $idf^{1.5}$), then we should we able to prevent the small losses incurred at the low recall end due to use of a stronger function. Results from preliminary investigation of this approach have not yielded an improvement, but we plan to pursue this further before we can conclude anything.

6.5 Conclusions

Different terms in a user query have different importance in conveying the user's information need to an information retrieval system. More specific terms are typically more useful for ranking documents. The inverse document frequency function has been used in term weights to incorporate this differential importance of terms. Experiments show that the usefulness of a strong inverse function of document frequency is high at low ranks. The main reason for this effect is the widened gap between the weights of the rare terms and the non-rare query terms. We believe that the rare query terms are the core query terms without the presence of which, the chances of relevance of an article are low. But if an article contains most of the query terms (the core terms as well as the supporting terms), it is still relevant with higher chances than another article that does contain the core terms with very high weights but is missing the supporting terms.

We recommend that the standard *idf* function be replaced by the function $idf^{1.5}$. The function $idf^{1.5}$ yields important improvements over the standard *idf* function (up to 7% improvement in average precision). The general applicability of this function looks promising, especially in light of the fact that $idf^{1.5}$ works better than the standard *idf* function on all three sets of queries used in this study.

Chapter 7

Conclusions

In this study we have learned how the chances of relevance for a document are related to the document's length, and how this information can be exploited to design better term weighting functions. In a brief study of the *idf* function, we show that the *idf* function can be improved upon with relatively simple changes.

This work demonstrates the tremendous value that the TREC effort adds to the field of experimental information retrieval. The presence of a large text corpus, a large set of user queries, and a superior set of relevance judgments opens up the possibility to learn more statistical characteristics of text, and how these characteristics can be effectively used to improve the information retrieval task.

7.1 Results

There are three main factors that determine a term's importance in a text — the term frequency factor, the inverse document frequency factor, and document length normalization. Researchers have defined good functions for the term frequency factor. Buckley et al. have proposed the use of a logarithmic term frequency factor. [BSA93,BAS94] Robertson et al. have proposed using the function $\frac{tf}{2+tf}$ for use as the term frequency factor. [RW94,RWJ+95] As compared to a linear function of term frequency, both these functions down-weight multiple occurrences of a term in a text. The advantage with the logarithmic term frequency factor is that it does not involve an empirically determined constant (the constant 2 involved in the function $\frac{tf}{2+tf}$). These functions are very effective for retrieval from large full-text collections.

In this work, we have studied the remaining two factors that determine term importance — the inverse document frequency factor, and document length normalization. We first show that document length normalization schemes that retrieve articles of all lengths with the same chances as their chances of relevance will be better than other schemes that retrieve documents of various lengths with chances very different from their chances of relevance.

The retrieval properties of a normalization function deviate systematically from the length distribution of relevance, across different text collections. For example, cosine normalization invariably retrieves more short documents than there are short relevant documents. Using a set of queries for which relevance is known, we can learn how the length retrieval properties of a normalization function deviate from the length distribution of relevance, and modify the normalization function to change its retrieval pattern to match the length distribution of relevance. The modified normalization function can then be used on new queries and a completely different set of documents in a real retrieval situation.

We find that linear modification of normalization functions (if we were using a normalization function $f(x)$, then using a modified normalization function $f'(x) = m \times f(x) + c$) suffices in general and yields significant retrieval improvements across queries and across text collections. We learned non-linear modified normalization functions that were better suited for the learning set of queries, but observed that even though the performance improves on the learning data, in a realistic situation, when the modified function is used on new queries and new data, the improvements are marginal. We observed that by stronger learning of a modified normalization function, we over-fit our function to the learning data and the performance doesn't improve on other text collections.

An important observation of our studies is that very short documents are not as useful to a user as the longer documents. The chances of a very short document being judged relevant to a query are much lower than the chances of a long document being judged relevant to a query. The main reason for this, we believe, is the fact that to be useful, a text must contain some minimum amount of information; we call this an "information unit". Very short texts often do not contain a full information unit and, therefore, are possibly less useful to a user. This observation has been substantiated by other researchers who have studied replacing the full-text of a document by a shorter text-segment from the document for computing the document's score. Researchers have noticed that using very short text segments for retrieval is not as effective as using somewhat longer text segments, but using long text segments works almost as well as using the entire document text. [Cal94,KGL95]

Our hypothesis about the presence of an information unit is further supported by the presence of a single pivot in our normalization experiments. Use of more than one pivot in the pivoted normalization scheme was no better than using just one pivot. This single pivot can be interpreted as the amount of text needed to convey a coherent idea in a particular domain. Texts shorter than the pivot contain less information than needed to satisfy an information need. Text longer than the pivot might contain multiple information units and can potentially satisfy a more diverse set of information needs. Based on our tools to study the relationship of document length to relevance and retrieval, we show that cosine similarity is not the appropriate similarity measure for the IR task. We show that using pivoted-unique normalization with inner product similarity is more effective for ranking articles.

This work also suggests that the inverse document frequency factor and cosine normalization should not be used in the same formulation, especially for large text collections. Cosine normalization introduces a mutual dependence between the weights of the terms in a text. Such mutual dependence is not desirable when the individual weights of terms might be inaccurate, as in the degraded text collections. When optical character recognition is used to scan printed material, the text collection thus created has numerous "garbage strings". These spurious strings have different occurrence characteristics in a collection than regular words. If weights of the correctly recognized terms are made dependent upon the weights of these spurious strings, then the retrieval effectiveness of a system for a degraded collection is poor.

Finally, in our studies with the inverse document frequency factor we find that an inverse function of the document frequency which is stronger than the traditional *idf* function works better than the traditional *idf* function. The main reason behind this effect is the increased gap between the weights of the rare query terms and the non-rare query terms. When a user types a rare term as part of the query, the usefulness of that term in predicting relevance is quite high. This evidence is specially useful when there is a weak match between the query and a document. After having tried various formulations as a replacement for *idf*, we found that the function $idf^{1.5}$ consistently works better than the *idf* function. We tested it on three different sets of queries in the TREC collection.

7.2 Recommendations

We recommend that the following functions be used in term weighting for ad-hoc information retrieval:

- Document term weights:
  - in correct text collection

$$\frac{\frac{1+log(tf)}{1+log(avg.\ tf\ in\ document)}}{0.8\ +\ 0.2 \times \frac{\#\ of\ unique\ terms\ in\ document}{avg.\ \#\ of\ unique\ term\ per\ document}}$$

  - in degraded text collections

$$\frac{1 + log(tf)}{0.7\ +\ 0.3 \times \frac{document\ length\ in\ bytes}{avg.\ document\ length\ in\ bytes}}$$

- Query term weights: $(1 + log(tf)) \times (log(\frac{N}{df}))^{1.5}$

Where *tf* is the term frequency, and *df* is the document frequency of a term. N is the number of articles in the collection.

7.3 Future Directions

The two central tasks that the field of information retrieval has been concerned with for years are: [BC92]

1. The *ad-hoc retrieval* task: a user comes up to an information retrieval system looking for a specific piece of information, types a query and expects that a ranked list of articles be presented to him/her.

2. The *routing* task: This task is the information filtering task. A user has a long standing information need. Over a period of time, the user tells us which articles were useful to his/her information need and which were not. Now the user expects the system to send (route) to him/her any new articles that could be relevant to his/her information need.

Having demonstrated the usefulness of term weighting in the ad-hoc retrieval scenario, we believe that the niceties of term weighting in the routing task deserve a lot more work.

When a user types a few terms for an IR system as a query, we understand, relatively well, how those query terms should be weighted. For example, we have shown in our work that the rare query terms provided by a user are important for predicting relevance if there is a weak match between the query and a document. But when the query terms are not provided by a user and are learned automatically from a set of full-text documents marked relevant by the user, as in the routing scenario, then these query terms, which actually are document terms, should have a much different notion of importance. A simple example would be the weights of the rare terms. If a rare term is present in a document marked relevant by a user, then this rare term does not carry the same semantics as a rare term typed by a user as part of an ad-hoc query. It is very possible that a rare term present in a relevant article is not a very important term for determining relevance of other articles. Therefore, the traditional *idf* function might not be an ideal term importance predictor in the routing task.

In summary, we believe that term weighting functions which are more suited for the routing task deserve more work. Also, the inverse document frequency formulation should be studied further, especially when used to estimate term importance in the routing task.

Bibliography

[All95] James Allan. *Automatic Hypertext Construction.* Ph.D. dissertation, Department of Computer Science, Cornell University, February 1995. Also Available as Technical Report TR96-1484, Department of Computer Science, Cornell University.

[BAS94] Chris Buckley, James Allan, and Gerard Salton. Automatic routing and ad-hoc retrieval using SMART : TREC 2. In D. K. Harman, editor, *Proceedings of the Second Text REtrieval Conference (TREC-2)*, pages 45–56. NIST Special Publication 500-215, March 1994.

[BASS95] Chris Buckley, James Allan, Gerard Salton, and Amit Singhal. Automatic query expansion using SMART : TREC 3. In D. K. Harman, editor, *Proceedings of the Third Text REtrieval Conference (TREC-3)*, pages 69–80. NIST Special Publication 500-225, April 1995.

[BC92] Nicholas Belkin and W. Bruce Croft. Information filtering and information retrieval: two sides of the same coin? *Communication of the ACM*, 35(12):29–38, December 1992.

[BCCN95] J. Broglio, J.P. Callan, W.B. Croft, and D.W. Nachbar. Document retrieval and routing using the INQUERY system. In D. K. Harman, editor, *Proceedings of the Third Text REtrieval Conference (TREC-3)*, pages 29–38. NIST Special Publication 500-225, April 1995.

[BS75] A. Bookstein and D. Swanson. A decision theoretic foundation for indexing. *Journal of the American Society for Information Science*, 26(1):45–50, January-Feburary 1975.

[BSA93] Chris Buckley, Gerard Salton, and James Allan. Automatic retrieval with locality information using SMART. In D. K. Harman, editor, *Proceedings of the First Text REtrieval Conference (TREC-1)*, pages 59–72. NIST Special Publication 500-207, March 1993.

[Buc93] Chris Buckley. The importance of proper weighting methods. In M. Bates, editor, *Human Language Technology.* Morgan Kaufman, 1993.

[Cal94] J.P. Callan. Passage–level evidence in document retrieval. In W. Bruce Croft and C.J. van Rijsbergen, editors, *Proceedings of the Seventeenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 302–310. Springer-Verlag, New York, July 1994.

[CCH92] J.P. Callan, W.B. Croft, and S.M. Harding. The INQUERY retrieval system. In *Proceedings of the Third International Conference on Database and Expert Systems Applications*, pages 78–83. Springer-Verlag, 1992.

[CCW95] W.B. Croft, R. Cook, and D. Wilder. Providing government information on the internet: Experiences with THOMAS. In *Proceedings of the Digital Libraries Conference DL'95*, pages 19–24, Austin, TX, June 1995.

[CH79] W.B. Croft and D.J. Harper. Using probabilistic models of information retrieval without relevance information. *Journal of Documentation*, 35(4):285–295, December 1979.

[CHTB94] W. B. Croft, S. Harding, K. Taghva, and J. Borsack. An evaluation of information retrieval accuracy with simulated OCR output. In *Proceedings of the Third Annual Symposium on Document Analysis and Information Retrieval*, pages 115–126, Las Vegas, NV, April 1994.

[Chu95] K.W. Church. One term or two? In Edward Fox, Peter Ingwersen, and Raya Fidel, editors, *Proceedings of the Eighteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 310–318, July 1995.

[CM63] C.W. Cleverdon and J. Mills. The testing of index language devices. *Aslib Proceedings*, 15(4):106–130, April 1963.

[CMK66] C.W. Cleverdon, J. Mills, and E.M. Keen. Factors determining the performance of indexing systems. In *Design*, volume 1. Aslib-Cranfield Research Project, Cranfield, England, 1966.

[Fag87] Joel Fagan. *Experiments in Automatic Phrase Indexing for Document Retrieval: A Comparison of Syntactic and Non-Syntactic Methods.* Ph.D. dissertation, Department of Computer Science, Cornell University, September 1987.

[Har93] D. K. Harman. Overview of the first Text REtrieval Conference (TREC-1). In D. K. Harman, editor, *Proceedings of the First Text REtrieval Conference (TREC-1)*, pages 1–20. NIST Special Publication 500-207, March 1993.

[Har94] D. K. Harman. Overview of the second Text REtrieval Conference (TREC-2). In D. K. Harman, editor, *Proceedings of the Second Text REtrieval Conference (TREC-2)*, pages 1–20. NIST Special Publication 500-215, March 1994.

[Har95] D. K. Harman. Overview of the third Text REtrieval Conference (TREC-3). In D. K. Harman, editor, *Proceedings of the Third Text REtrieval Conference (TREC-3)*, pages 1–19. NIST Special Publication 500-225, April 1995.

[Har96] D. K. Harman. Overview of the fourth Text REtrieval Conference (TREC-4). In D. K. Harman, editor, *Proceedings of the Fourth Text REtrieval Conference (TREC-4)*, 1996. To appear.

[HP93] Marti A. Hearst and Christian Plaunt. Subtopic structuring for full-length document access. In *Proceedings of the Sixteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 59–68, June 1993.

[Kee71] E.M. Keen. Evaluation parameters. In Gerard Salton, editor, *The SMART Retrieval System—Experiments in Automatic Document Processing.* Prentice Hall, Englewood Cliffs, NJ, 1971.

[KG96] K.L. Kwok and L. Grunfeld. TREC-4 ad-hoc, routing retrieval and filtering experiments using PIRCS. In D. K. Harman, editor, *Proceedings of the Fourth Text REtrieval Conference (TREC-4)*, 1996. To appear.

[KGL95] K.L. Kwok, L. Grunfeld, and D.D. Lewis. TREC-3 ad-hoc, routing retrieval and thresholding experiments using PIRCS. In D. K. Harman, editor, *Proceedings of the Third Text REtrieval Conference (TREC-3)*, pages 247–255. NIST Special Publication 500-225, April 1995.

[Kwo96] K.L. Kwok. A new method for weighting query terms for ad-hoc retrieval. In *Proceedings of the Nineteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 187–195, August 1996.

[Lan79] F.W. Lancaster. Criteria by which information retrieval systems may be evaluated. In *Information Retrieval Systems—Characteristics, Testing and Evaluation, 2nd Edition.* John Wiley and Sons, New York, 1979.

[Lee95] J.H. Lee. Combining multiple evidence from different properties of weighting schemes. In Edward Fox, Peter Ingwersen, and Raya Fidel, editors, *Proceedings of the Eighteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 180–188, July 1995.

[Lov68] J.B. Lovins. Development of a stemming algorithm. *Mechanical Translation and Computational Linguistics*, 1-2(11):11–31, March and June 1968.

[Luh57] H.P. Luhn. A statistical approach to mechanized encoding and searching of literary information. *IBM Journal of Research and Development*, 1(4):309–317, October 1957.

[MK60] M.E. Maron and J.L. Kuhns. On relevance, probabilistic indexing and information retrieval. *Journal of the ACM*, 7(3):216–243, July 1960.

[MS94] Elke Mittendorf and Peter Schauble. Document and passage retrieval based on hidden markov models. In W. Bruce Croft and C.J. van Rijsbergen, editors, *Proceedings of the Seventeenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 318–327. Springer-Verlag, New York, July 1994.

[Nel96] P.E. Nelson. The Excalibur TREC-4 system, preparations, and results. In D. K. Harman, editor, *Proceedings of the Fourth Text REtrieval Conference (TREC-4)*, 1996. To appear.

[NR94] Thomas A. Nartker and Stephen V. Rice. OCR accuracy: UNLV's third annual test. *INFORM*, 8(8):30–36, September 1994.

[RSJ76] S.E. Robertson and K. Sparck-Jones. Relevance weighting of search terms. *Journal of the American Society for Information Science*, 27(3):129–146, May-June 1976.

[RW94] S.E. Robertson and S. Walker. Some simple effective approximations to the 2–poisson model for probabilistic weighted retrieval. In W. Bruce Croft and C.J. van Rijsbergen, editors, *Proceedings of the Seventeenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 232–241. Springer-Verlag, New York, July 1994.

[RWJ+95] S.E. Robertson, S. Walker, S. Jones, M.M. Hancock-Beaulieu, and M. Gatford. Okapi at TREC–3. In D. K. Harman, editor, *Proceedings of the Third Text REtrieval Conference (TREC-3)*, pages 109–126. NIST Special Publication 500-225, April 1995.

[SA94] Gerard Salton and James Allan. Automatic text decomposition and structuring. In *RIAO 94 Conference Proceedings*, pages 6–20, October 1994.

[SAB93] Gerard Salton, James Allan, and Chris Buckley. Approaches to passage retrieval in full text information systems. In *Proceedings of the Sixteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 49–58, June 1993.

[Sal65] Gerard Salton. The evaluation of automatic retrieval procedures—selected test results using the SMART system. *American Documentation*, 16(3):209–222, July 1965.

[Sal71] Gerard Salton, editor. *The SMART Retrieval System—Experiments in Automatic Document Retrieval*. Prentice Hall Inc., Englewood Cliffs, NJ, 1971.

[Sal75] Gerard Salton. A theory of indexing. In *Regional Conference Series in Applied Mathematics, No. 18*, Philadelphia, PA, 1975. Society for Industrial and Applied Mathematics.

[Sal81] Gerard Salton. A blueprint for automatic indexing. *ACM SIGIR Forum*, 2(16):22–38, Fall 1981.

[Sal89] Gerard Salton. *Automatic text processing—the transformation, analysis and retrieval of information by computer*. Addison-Wesley Publishing Co., Reading, MA, 1989.

[Sal91] Gerard Salton. The state of retrieval system evaluation. Technical Report 91-1206, Cornell University, May 1991.

[SB88] Gerard Salton and Chris Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing and Management*, 24(5):513–523, 1988.

[SB90a] Gerard Salton and Chris Buckley. Improving retrieval performance by relevance feedback. *Journal of the American Society for Information Science*, 41(4):288–297, 1990.

[SB90b] Gerard Salton and Chris Buckley. A note on term weighting and text matching. Technical Report 90-1166, Cornell University, October 1990.

[SBM96] Amit Singhal, Chris Buckley, and Mandar Mitra. Pivoted document length normalization. In *Proceedings of the Nineteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 21–29, August 1996. Also Technical Report TR95-1560, Department of Computer Science, Cornell University, Ithaca, NY 14853, November 1995.

[SJ72] K. Sparck-Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, 28(1):11–21, March 1972.

[SJ73] K. Sparck-Jones. Index term weighting. *Information Storage and Retrieval*, 9(11):619–633, November 1973.

[SM83] Gerard Salton and M.J. McGill. *Introduction to Modern Information Retrieval*. McGraw Hill Book Co., New York, 1983.

[SS94] Gerard Salton and Amit Singhal. Automatic text theme generation and the analysis of text structure. Technical Report 94-1438, Department of Computer Science, Cornell University, Ithaca, NY 14853, July 1994.

[SSB96] Amit Singhal, Gerard Salton, and Chris Buckley. Length normalization in degraded text collections. In *Fifth Annual Symposium on Document Analysis and Information Retrieval*, pages 149–162, April 1996. Also Technical Report TR95-1507, Department of Computer Science, Cornell University, Ithaca, NY 14853, April 1995.

[SSMB96] Amit Singhal, Gerard Salton, Mandar Mitra, and Chris Buckley. Document length normalization. *Information Processing and Management* (to appear), 1996. Also Technical Report TR95-1529, Department of Computer Science, Cornell University, Ithaca, NY 14853, July 1995.

[SWY75] Gerard Salton, A. Wong, and C.S. Yang. A vector space model for information retrieval. *Journal of the American Society for Information Science*, 18(11):613–620, November 1975.

[SY73] Gerard Salton and C.S. Yang. On the specification of term value in automatic indexing. *Journal of Documentation*, 29(4):351–372, December 1973.

[SYY75] Gerard Salton, C.S. Yang, and C.T. Yu. A theory of term importance in automatic text analysis. *Journal of the American Society for Information Science*, 26(1):33–44, January–February 1975.

[TBC94a] Kazem Taghva, Julie Borsack, and Allen Condit. Effects of OCR errors on ranking and feedback using the vector space model. Technical Report 94-06, Information Science Research Institute, University of Nevada, Las Vegas, August 1994.

[TBC94b] Kazem Taghva, Julie Borsack, and Allen Condit. Evaluation of model-based retrieval effectiveness with OCR text. Technical Report 94-09, Information Science Research Institute, University of Nevada, Las Vegas, October 1994.

[TBC94c] Kazem Taghva, Julie Borsack, and Allen Condit. Results of applying probabilistic IR to OCR text. In *Proceedings of the Seventeenth Annual International ACM/SIGIR Conference on Research and Development in Information Retrieval*, pages 202–211, Dublin, Ireland, July 1994.

[TBCE94] Kazem Taghva, Julie Borsack, Allen Condit, and Srinivas Erva. The effects of noisy data on text retrieval. *Journal of the American Society for Information Science*, 45(1):50–58, January 1994.

[TBCI95] Kazem Taghva, Julie Borsack, Allen Condit, and Padma Inaparthy. Querying short OCR'd documents. Technical Report 94-10, Information Science Research Institute, University of Nevada, Las Vegas, February 1995.

[Tur90] Howard Turtle. *Inference Networks for Document Retrieval*. Ph.D. dissertation, Department of Computer Science, University of Massachusetts, Amherst, MA 01003, 1990. Available as COINS Technical Report 90-92.

[Wil94] Ross Wilkinson. Effective retrieval of structured documents. In W. Bruce Croft and C.J. van Rijsbergen, editors, *Proceedings of the Seventeenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 311–317. Springer-Verlag, New York, July 1994.

[YBLS83] C.T. Yu, C. Buckley, K. Lam, and G. Salton. A generalized term dependence model in information retrieval. *Information Technology: Research and Development*, 2(4):129–154, October 1983.

[Zip49] G.K. Zipf. *Human Behavior and the Principle of Least Effort.* Addison-Wesley Publishing Co., Reading, MA, 1949.