ON THE EFFECT OF INQUIRY TERM-WEIGHTING SCHEME ON QUERY-SENSITIVE SIMILARITY MEASURES

A Thesis

by

ANANTH ULLAL KINI

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

December 2005

Major Subject: Computer Science

ON THE EFFECT OF INQUIRY TERM-WEIGHTING SCHEME ON QUERY-SENSITIVE SIMILARITY MEASURES

A Thesis

by

ANANTH ULLAL KINI

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Approved by:

Chair of Committee, Paul Nelson
Committee Members, John J. Leggett
William E. Burchill
Head of Department, Valerie E. Taylor

December 2005

Major Subject: Computer Science

ABSTRACT

On the Effect of INQUERY Term-Weighting Scheme on Query-Sensitive Similarity Measures. (December 2005)

Ananth Ullal Kini, B.E., University of Mysore, India
Chair of Advisory Committee: Dr. Paul Nelson

Cluster-based information retrieval systems often use a similarity measure to compute the association among text documents. In this thesis, we focus on a class of similarity measures named Query-Sensitive Similarity (QSS) measures. Recent studies have shown QSS measures to positively influence the outcome of a clustering procedure. These studies have used QSS measures in conjunction with the *ltc* term-weighting scheme. Several term-weighting schemes have superseded the *ltc* term-weighing scheme and demonstrated better retrieval performance relative to the latter. We test whether introducing one of these schemes, INQUERY, will offer any benefit over the *ltc* scheme when used in the context of QSS measures. The testing procedure uses the Nearest Neighbor (NN) test to quantify the clustering effectiveness of QSS measures and the corresponding term-weighting scheme.

The NN tests are applied on certain standard test document collections and the results are tested for statistical significance. On analyzing results of the NN test relative to those obtained for the *ltc* scheme, we find several instances where the INQUERY scheme improves the clustering effectiveness of QSS measures. To be able to apply the NN test, we designed a software test framework, Ferret, by complementing the features provided by dtSearch, a search engine. The test framework automates the generation of NN coefficients by processing standard test document collection data. We provide an insight into the construction and working of the Ferret test framework.

DEDICATION

To my dear Mother and Father
Those values I will forever cherish
To my dear Sister and Brother
Those childhood days that’ll never perish

ACKNOWLEDGEMENTS

I shall forever remain indebted to Dr. Paul Nelson for guiding me through the evolution of this thesis. His technical insight, multi-disciplinary knowledge and personality have presented me with precious pearls of wisdom. He demonstrated exemplary time management skills by ensuring ample time for our meetings in spite of his densely packed calendar. On countless occasions, he went out of his way to ensure a good work environment in which to carry out the research.

I am also grateful to Dr. John Leggett and Dr. William Burchill for serving on my committee and for placing faith in my work. The inception of the thesis would not have been possible without the support and suggestions from Ernest Kee (Ernie) from the Risk Management team at the South Texas Project Nuclear Operating Company (STPNOC). STPNOC financially backed the research position at the university, which I held during the course of this thesis. The funding from STPNOC also paved way for development of a software product that draws from aspects of this thesis. I thank my dear friends Rajiv Badi, Navendu Misra, and Manas Singh, from the department of computer science, for sharing their knowledge and time towards discussions surrounding the thesis. I would like to place my gratitude with Glenda Elledge, Wanda Allen, Elena Rodriguez, Bertha Martinez, Dave Cote, and Nolan Flowers for expedited handling of numerous administrative issues. I would like to thank Dr. Tassos Tombros (University of London), David Thede (dtSearch president), Steven Robertson, and various online Newsgroups for taking time in responding to my inquiring emails. Last, and by no means the least, I wholeheartedly appreciate the wonderful infrastructure made available to the student community by the Texas A&M University.

TABLE OF CONTENTS

LIST OF FIGURES

Page

LIST OF TABLES

1. INTRODUCTION

1.1. Motivation and Objective

Cluster analysis, as an information retrieval technique, was introduced several decades ago [22]. However, growth of the World Wide Web accompanied by information explosion has continued to fuel innovation in this area [29]. Typically, a document clustering method relies on a similarity/dissimilarity measure to compute the association between a pair of documents, and the choice of a measure may influence the outcome of the clustering procedure [16],[10]. Recently, a new class of similarity measures namely *Query Sensitive Similarity* (QSS) measures was tested successfully for the purpose of document clustering [28]. The usefulness of QSS measures is attributed to their ability to bias interdocument similarity towards pairs of documents that are jointly relevant to a user query.

Thus far, QSS measures have been implemented with SMART's *ltc* term-weighting scheme [28], [29] (See Section 2.2). However, subsequent introduction of other term-weighting schemes such as INQUERY [2] and Okapi [20] have shown to offer better retrieval performance in comparison to the *ltc* scheme (Section 2.2). The objective of the proposed thesis is to test whether introduction of an alternate term weighting scheme, INQUERY, improves QSS measure's clustering effectiveness. The testing will use variants of the NN test [30] that quantifies a test collection's degree of adherence to cluster hypothesis under QSS measures, thus measuring its clustering effectiveness. The results of the NN test will be discussed.

For the NN test to be executed successfully in this context, a system should at least support the following features – support for INQUERY term-weighting scheme, support for *ltc* term-weighting scheme, and the ability to compute similarity using QSS measures. Another desirable feature expected of the system is the automation of NN test result generation with minimal human intervention. We constructed such a system,

This thesis follows the style of *IEEE Transactions on Knowledge and Data Engineering*.

Ferret test framework, by complementing features of an existing search engine, dtSearch. The construction and working of this system will be elaborated (See Section 4).

1.2. Organization of the Thesis

This thesis will arrange the ensuing discussion under five main sections. Section 2 presents background material on information retrieval that helps place the thesis discussion in the proper context. This section contains definitions and notations as applicable to vector space representation of documents. We also briefly survey some of the popular term-weighting schemes, including the one central to the thesis namely INQUERY. The section further contains a discussion on some of the commonly used similarity measures, leading to the introduction of QSS measures and their utility in cluster-based IR systems. We also hypothesize how term-weighting schemes and similarity measures may jointly influence the outcome of a clustering procedure. Furthermore, we present formal definitions and notations pertaining to certain QSS measures we propose to test.

Section 3 describes testing procedures that have been successfully employed in the past for testing clustering effectiveness of a similarity measure. Previous research results obtained in this direction are also summarized. Section 4 elaborates on the implementation details of the test framework used in our experiments. Support for the NN testing procedure within the framework is also described. Section 5 presents the results obtained upon performing the NN test on the test collections and an associated discussion. Section 6 summarizes the results and states the conclusions that were derived as a result of our experimentation.

2. BACKGROUND

Boolean, vector and probabilistic models are the three basic models used in document retrieval [1]. That used in the proposed work is the vector space model. Section 2.1 is a quick review of the elements of this model. Section 2.2 is a summary of the literature on the term-weighting schemes that are fundamental to use of vector space models. Section 2.3 is similarly a brief overview of similarity and dissimilarity measures, as applicable in the field of document retrieval. Section 2.4 underscores the utility of QSS measures in cluster-based IR systems. Section 2.5 presents notations for the similarity measures, particularly QSS measures, whose clustering effectiveness will be tested in conjunction with *ltc* and INQUERY term-weighting schemes.

2.1. The Vector Space Model

A document can be decomposed into *tokens*, which are defined as a continuous string of characters delimited by spaces, punctuation or any other separating characters [31]. An *index term*, a *keyword* or, a *document term* is a token carrying a specific meaning with respect to a lingual, technical, specialty or other dictionary [31]. Due to historical reasons, for the purpose of information retrieval, a document collection is often viewed as a set of index terms [1]. A *vector space model* in information retrieval represents documents as *weighted* vectors in a *document space* defined by index terms [23].

More precisely, consider a document collection consisting of N documents, where each document is labeled D_i. Each document D_i contains one or more index terms T_j. Let t be the total number of unique index terms (*vocabulary* of size t) found across all documents in the collection. In a vector space model, each document D_i can be represented by a t-dimensional vector as shown in (1).

$$D_i = \left(d_{i1}, d_{i2}, \ldots, d_{it}\right) \qquad \mathbf{(1)}$$

where, d_{ij} represents the weight of the j^{th} index term for the i^{th} document D_i. The documents are said to be contained in a t-dimensional *document space*. Calculation of term-weight, d_{ij}, is an important step in construction of the vector space model. Further discussion on the topic of term-weighting can be found in Section 2.2.

A user's information need in the context of information retrieval is called a *query*. A query is essentially a subset of the vocabulary. The terms appearing in a query are called *query terms*. Furthermore, the index terms appearing in a query may also be weighted. If Q is a query, it can be represented as a vector in the document space as follows:

$$Q = (q_1, q_2, ..., q_t) \quad \textbf{(2)}$$

where, q_i is the weight assigned to the i^{th} *query* term. As before, t is the vocabulary size.

2.2. Term-Weighting Schemes

The main objective of using different term-weighting algorithms is to attempt to enhance retrieval effectiveness [24]. Qualitatively, retrieval effectiveness consists of two factors. Firstly, documents closer to a user's information need should be retrieved (*recall*); such documents are called *relevant* documents in the information retrieval nomenclature. Secondly, documents that are likely to be not useful to a user (nonrelevant) should be rejected by the system (*precision*). Ideally, term-weighting schemes are expected to maximize both recall and precision, but in reality, it is a tradeoff between the two.

Numerous term-weighting schemes have been proposed in the information retrieval literature. An extensive treatment of some of the established schemes can be found in [3][24]–[26]. Most weighting schemes are based on three proven principles[14],[18]:

1. The *term frequency* (*tf*) (i.e., number of occurrences of a term in a document) can be useful in enhancing recall. An underlying assumption here is that more the term frequency, more central is that term in describing the contents of the document.
2. The *inverse document frequency* (*idf*), or number of documents in a document collection where a given index term appears, can be useful to enhance

precision. In effect, the *idf* component assigns a lower weight to terms that occur too frequently in the document collection, while assigning a higher weight to those that occur infrequently.

3. *Normalization* of the document vectors, so that all document vectors pertaining to a given document collection have the same length, appears to enhance both recall and precision. Long documents are often verbose containing numerous terms and these terms may display large term frequencies. Thus, long documents may often obtain a better relevancy score over short documents. Normalization strives to overcome these effects.

Hence, a term-weight can be defined as a triplet consisting of components that are functions of {*tf*}, {*idf*} and, {*normalization*} and are combined using appropriate mathematical operators, typically, multiplication. Some of the established weighting components are presented in Tables 1–3 [3],[24]. Moreover, the triplet used for index term-weighting may be different from that used for query term-weighting. If *ltc.lnc* is a term-weighting scheme used by an information retrieval system, the *ltc* triplet describes the index term-weighting scheme, while the *lnc* triplet describes the weighting scheme for the query terms.

Table 1: Term Frequency Component *tf*

Symbol	Formula	Description
t	tf	Normal term frequency – number of times a term occurs in a document
l	$1.0 + \ln(tf)$	Log
n	$0.5 + 0.5\left(\frac{tf}{Max \quad tf \quad in \quad vector}\right)$	Augmented Normalized Term Frequency
b	$1, 0$	Binary – One if term present in vector, zero if not

Table 2: Inverse Document Frequency Component *idf*

Symbol	Formula	Description
x	1.0	No inverse document frequency, no change in weight
f** or **t	$\log\left(\frac{N+1}{n}\right)^{♣}$	Traditional – inverse document frequency
p	$\log\left(\frac{N-n}{n}\right)$	Probabilistic Inverse Collection Frequency

♣ N – total number of documents in the collection; n – number of documents containing a given term

Table 3: Normalization Component

Symbol	Formula	Description
x	1.0	No normalization, no change in weight
c	$\frac{1}{\sqrt{\sum_{vector} w_i^2}}$	Cosine Normalization – each term-weight is divided by the vector's Euclidean length

Experiments were performed on TREC (Text REtrieval Conference, see Section 3.1) document collections with exhaustive combination of term-weighting schemes, the details of which can be found in [24]. The best document weighting schemes were found to be *tfc, nfc* (or *tpc, npc)* [24], whereas the best query weighting schemes were found to be *nfx*, *tfx*, *bfx* (or *npx*, *tpx*, *bpx*).

There are other term-weighting schemes that have proved effective within the TREC benchmark, prominent among which are Okapi and INQUERY. As regards the Okapi scheme, it suffices here to say that it uses approximations to 2-Poisson probabilistic model to obtain the term-weights for both documents and queries, it provided the best performance in TREC-3 [25], and its term-weighting formula contains additional parameters that need to be tuned after systematic experimentation with a document collection [21].

An instance of INQUERY term weighting formula [14] is given by

$$w_{ij} = \frac{tf_{ij}}{tf_{ij} + 0.5 + 1.5\,ndl_i} \frac{\log\left(\left(N+0.5\right)/n\right)}{\log\left(N+1\right)}. \quad \textbf{(3)}$$

Here

w_{ij} is the weight assigned to the j^{th} document term (T_j) present in i^{th} document in the collection

N is the total number of documents in the collection

n is the number of documents containing the document term (T_j)

tf_{ij} is the term frequency of the j^{th} document term (T_j) present in i^{th} document

$ndl_i = dl_i / (avdl)$

where, dl_i and *avdl* are respectively the document length (of the i^{th} document) and average document length measured in some suitable unit. For the purpose of this thesis, we used the number of unique terms in a document as a measure of its length.

In the above formula, one can notice the familiar term-weighting elements such as term frequency and inverse document frequency. However, there is also a dependence on document length (the term ndl_i), which is missing from the *ltc* term-weighting schemes discussed above. Because of the centrality of this term-weighting scheme to this thesis, we wish to discuss the central ideas behind the weighting scheme (3).

Equation (3) is one version of the term-weighting schemes that have been applied within the INQUERY information retrieval system [15]. All versions of the term-weighting schemes used within INQUERY appear to be efforts that approximately capture the hypothesis that documents can be viewed as “a random stream of term occurrences, each one having a fixed, small probability of being the term in question, this probability being constant over all elite documents, and also constant (but smaller) over all non-elite documents.” Here a document is said to be “elite” for a particular term if the document is (intuitively) “about” that term (i.e., the term is “central” to the document), otherwise it is said to be non-elite (for that term) [19].

More technically, probabilities of occurrence of terms are drawn from Poisson distributions in the case of both elite and non-elite documents, but with a smaller mean for the non-elite documents than for the elite documents. This “2-Poisson model is usually said to require that document length is constant” [19]. The term ndl_i in the denominator of equation (3) represents an effort to compensate for effects stemming from the fact that documents within a given collection differ in length. In this manner, term-weights based on the above INQUERY formula might more correctly capture the underlying characteristics of collections across which document lengths vary significantly.

Note that in contrast to the Okapi scheme, the above INQUERY formula is devoid of any parameter tuning. Its success with TREC collections and absence of parameter tuning contributed to our choice to employ INQUERY term-weighting scheme with our implementation of QSS measures.

2.3. Similarity/Dissimilarity Measures

Our ultimate objective is to test the clustering efficiency of QSS measures in conjunction with INQUERY term-weighted vector representation. For purposes of clustering, it is necessary to have some quantitative measure of the similarity or dissimilarity of two document vectors. This section is a review of the current state of knowledge of such measures.

Similarity/dissimilarity coefficients (or measures) are functions that assign a real number to a pair of objects in a collection, based on attributes that describe them, indicating the degree of similarity/dissimilarity [12]. For our purposes, those attributes will be the corresponding document vectors in some vector-space representation, principally that given by the INQUERY term-weighting scheme. For an extensive review on similarity/dissimilarity coefficients, the reader is referred to [10]. We will briefly discuss some of the commonly used coefficients in the following paragraphs.

Let D_1 and D_2 be two documents, whose interdocument relationship (similarity/dissimilarity) we are interested in finding. Using an appropriate term-weighting scheme, their vector representation is given by $D_1 = (d_{11}, d_{12}, \ldots, d_{1t})$ and $D_2 = (d_{21}, d_{22}, \ldots, d_{2t})$, where d_{ij} is the term-weight of the j^{th} term in the i^{th} document and t is the vocabulary size. The dot product of D_1 and D_2 can give a good indication of the interdocument similarity. It is defined as:

$$D_1 \cdot D_2 = \sum_{i=1}^{t} \left(d_{1i} \times d_{2i} \right) \quad \textbf{(4)}$$

The dot product is generally normalized so that the computed value lies between 0 (least similar) and 1 (most similar). Two commonly used normalized coefficients are cosine and Dice, which are defined respectively by

$$Cosine\left(D_1, D_2\right) = \frac{\sum_{i=1}^{t} \left(d_{1i} \times d_{2i} \right)}{\sqrt{\sum_{i=1}^{t} d_{1i}^2 \times \sum_{i=1}^{t} d_{2i}^2}}, \text{ and} \quad \textbf{(5)}$$

$$Dice\left(D_1, D_2\right) = \frac{2 \cdot \sum_{i=1}^{t} \left(d_{1i} \times d_{2i} \right)}{\sum_{i=1}^{t} d_{1i}^2 + \sum_{i=1}^{t} d_{2i}^2}. \quad \textbf{(6)}$$

Note that these are angle-based measures, in the sense that, they rely on finding the angle between two vectors [12]. On the other hand, one might employ distance-based dissimilarity measures e.g., Euclidean distance (7),

$$Euclidean\left(D_1, D_2\right) = \sqrt{\sum_{i=1}^{t} \left(d_{1i} - d_{2i} \right)^2}. \quad \textbf{(7)}$$

We choose to base our experiments on the cosine coefficient, as it is intuitive to use (1 – most similar, 0 – least similar) and due to its proven effectiveness in practice [12]. In fact, as we will later see in section 2.5, the measure we eventually use in the experiments, namely QSS, is a repeated application of cosine coefficient.

2.4. Utility of QSS measures in Document Clustering

Cluster analysis is a well-known statistical technique used to identify groups of similar objects represented in multi-dimensional space. Given that documents can be represented as multi-dimensional vectors, cluster analysis can be conveniently applied to cluster documents as well. Document clustering was introduced primarily to improve the effectiveness of serial search and to offer a user novel ways of exploring a document collection [5],[11]. As it is not the aim of this thesis to discuss various clustering schemes, we refer the reader to [27] and [4] for a discussion on clustering methods and the associated taxonomy. For the purpose of this thesis, it is sufficient to say that most clustering methods rely on measuring the association between pair of documents using a similarity/dissimilarity measure and that choice of such a measure may influence the outcome of the clustering procedure [16],[10].

The motivation for using cluster analysis in information retrieval arises out of *Cluster Hypothesis*. According to Cluster Hypothesis [11], documents relevant to a user query tend to be highly similar to each other, and hence, when subject to Cluster Analysis under some measure of similarity, are likely to appear in the same cluster. A clustering procedure introduced in [11] was solely driven by measures that calculated the association between a pair of document vectors without any consideration for the query vector. This implies that between any two documents D_i and D_j, the interdocument relationship Sim(D_i, D_j) remains static regardless of the query posed to the system. Hence, the aforementioned clustering technique is termed *static clustering*.

In [28] the idea of using Query-Sensitive Similarity (QSS) measures, which is to say a similarity measure that depends upon the query vector as well as the two document vectors, was suggested. Employing user query as a context favors appearance of documents jointly relevant to a query in the same cluster. As a result, a changing query dynamically alters interdocument relationships, tailoring the clusters to more closely match the query context. Hence, the clustering procedure is broadly termed *query-specific clustering*.

The presence of a query component in a QSS measure formula motivated us to experiment with term-weighting schemes. The term-weighting schemes, e.g. *ltc* and INQUERY, are designed to improve relevance matching of a document to a query. In the past, QSS measures have been implemented with the *ltc* term-weighting scheme [28], [29]. Since the INQUERY term-weighting scheme has proved a better performer in comparison to the *ltc* scheme [25], we expect the introduction of the former to wield a positive influence on the clustering effectiveness of QSS measures. In subsection 2.5, we introduce two QSS measures, M_1 and M_2, with the associated mathematical notations. In our experiments, we measure the clustering effectiveness of QSS measures M_1 and M_2 each implemented with *ltc* and INQUERY term-weighting schemes and compare their relative performance.

2.5. QSS Measure Definition

Cosine coefficient of similarity, $Sim(D_i, D_j)$, for any two documents D_i and D_j is given by

$$\underset{cosine}{Sim\left(D_i, D_j\right)} = \frac{\sum_{k=1}^{t}\left(d_{ik} \times d_{jk}\right)}{\sqrt{\sum_{k=1}^{t} d_{ik}^2 \times \sum_{k=1}^{t} d_{jk}^2}}, \qquad \textbf{(8)}$$

where, d_{ik} and d_{jk} are the term weights assigned to k^{th} term of documents D_i and D_j respectively and t stands for the vocabulary size. In the *cosine* measure (8), one can notice the absence of terms related to a user query Q, thus making the association between documents static in nature. As discussed previously, cluster analysis carried out using *cosine* measure is an instance of what is termed static clustering, as the document inter-relationship remains same regardless of the query used. The aforementioned argument can be extended in an analogous manner to other measures like Dice coefficient and Euclidean distance, when used in calculating the similarity, $Sim(D_i, D_j)$.

Without loss of generality, a QSS measure, $Sim(D_i, D_j \mid Q)$, can be expressed as a function $f(Sim(D_i, D_j), Sim(D_i, D_j, Q))$. This implies that QSS measures are made up of a static component $Sim(D_i, D_j)$ and a query-specific component $Sim(D_i, D_j, Q)$, making the measure as a whole a dynamic one. The $Sim(D_i, D_j)$ component captures the static similarity between the documents D_i and D_j. A possible definition for $Sim(D_i, D_j)$, namely *cosine*-coefficient, is provided by (8). The $Sim(D_i, D_j, Q)$, stands for the similarity jointly shared by the documents with the query. M_2, the shorthand used for $Sim(D_i, D_j, Q)$, can be defined [28] as

$$\underset{M_2}{Sim\left(D_i, D_j, Q\right)} = \frac{\sum_{k=1}^{m}\left(c_k \times q_k\right)}{\sqrt{\sum_{k=1}^{m} c_k^2 \times \sum_{k=1}^{m} q_k^2}}. \tag{9}$$

Here user query Q in the above equation is given by terms $\{q_1, q_2, q_3, \ldots, q_m\}$. The common terms between the two documents is given by $C = D_i \cap D_j = \{c_1, c_2, c_3, \ldots, c_{k,}, \ldots, c_m\}$. Each term c_k is equal to $(d_{ik} + d_{jk}) / 2$. This specific two-document term-weighting scheme is preferred by the authors [28] over other weighting schemes such as $\min(d_{ik,} d_{jk})$, $\max(d_{ik,} d_{jk})$, and $(d_{ik} \text{ x } d_{jk})$ due to consistent effectiveness shown by the former.

To fully qualify a QSS measure, we need a definition for $f(Sim(D_i, D_j), M_2)$, an example for which is shown in (3) [28]. A more recent definition for $f(Sim(D_i, D_j), M_2)$ is of the form $\alpha Sim(D_i, D_j) + \beta M_2$, where α determines the importance attached to the conventional measure $Sim(D_i, D_j)$, while β determines the weight offered to the query-sensitive component M_2 [29]. The disadvantage of the method is the tuning required for the parameters α and β, which has to be empirically determined. We choose to test the measure suggested in (10),

$$f\left(Sim\left(D_i,D_j\right),M_2\right)=Sim\left(D_i,D_j\right)\times M_2, \qquad \textbf{(10)}$$

as it is devoid of any parameter tuning [28].

Substituting (8) and (9) in (10), an expression for QSS measure, *Sim*(*D*$_i$, *D*$_j$ | *Q*), can be obtained as

$$\underset{M_1}{Sim\left(D_i,D_j \mid Q\right)}=\frac{\sum_{k=1}^{n}\left(d_{ik}\times d_{jk}\right)}{\sqrt{\sum_{k=1}^{n}d_{ik}^2\times\sum_{k=1}^{n}d_{jk}^2}}\times\frac{\sum_{k=1}^{m}\left(c_k\times q_k\right)}{\sqrt{\sum_{k=1}^{m}c_k^2\times\sum_{k=1}^{m}q_k^2}}. \qquad \textbf{(11)}$$

The QSS measure, also denoted as M_1 for sake of brevity, satisfies properties of symmetry, *Sim*(*D*$_i$, *D*$_j$ | *Q*) = *Sim*(*D*$_j$, *D*$_i$ | *Q*). It can also be verified that $0 \leq M_1 \leq 1$. As equation (11) on its own does not support the property of reflexivity, given by *Sim*(*D*$_i$, *D*$_i$ | *Q*) = 1, it can be introduced explicitly by the system. Thus, QSS measure M_1 agrees with other conventional similarity measures [17]. An extreme limiting case of measure M_1 would actually be the measure M_2 itself, as it takes into account only the terms common to two documents and the query. Hence, in principle, M_2 is also a QSS measure that we will test along with measures M_1 and *cosine*.

3. STANDARD TESTING PROCEDURES

The aim of this section is to provide an overview of the characteristics of the document collections used and the nature of the testing technique employed. We will also briefly discuss some of the important results previously obtained upon testing the measures M_1, M_2 and *cosine* on standard TREC collections. Most of the discussion revolves around results obtained in [28], unless otherwise stated.

3.1. Document Collections

In order to test the measures, five TREC (Text REtrieval Conference) document collections, CACM, CISI, LISA, Medline (MED) and WSJ were employed [28]. TREC conference, jointly sponsored by National Institute of Standards and Technology (NIST) and the Information technology office of Defense Advanced Research Projects Agency (DARPA), acts as a forum for researchers around the world to present and discuss their research on information retrieval [1]. Every test collection provides a set of standard queries. For each query, TREC provides a list of relevant documents as evaluated by the authors of the test collection. Performance of a given IR system can be measured by comparing the retrieved documents against this benchmark list.

The CACM collection consists of 3204 articles published in the Communications of the ACM. The CACM documents cover a host of topics in the area of computer science published between 1958 and 1979. The CISI collection, also called the ISI collection, is a collection assembled at the Institute of Scientific Information (ISI) about information sciences. LISA and MED collections were compiled at the Virginia Polytechnic Institute and State University in the fields of Library Science and Medicine respectively. LISA and MED offer a good setting for preliminary testing of information retrieval algorithm. While each of these document collections covered a major subject area, the WSJ (Wall Street Journal) collection displayed substantial diversity across articles. Each of the aforementioned collections has undergone thorough experimentation, making them the *reference* test collections used in most information retrieval experiments [1]. The statistics for these collections can be found in Table 4.

Table 4: TREC Document Collection Statistics [28]

Attributes	CACM	CISI	LISA	MED	WSJ
Number of docs.	3204	1460	6004	1033	74520
Mean terms per doc.	22.5	43.9	39.7	51.6	377
Number of queries	52	35	35	30	50
Mean terms per query	13	7.6	19.4	9.9	7.6
Mean relevant docs per query	15.3	49.8	10.8	23.2	71.4
Total relevant docs.	796	1742	379	696	3572

3.2. Testing Techniques

Two evaluation schemes are widely used in information retrieval to measure the *clustering tendency* or *classifiability* of a document collection under a similarity measure [14] – *Overlap Test* [16] and *Nearest Neighbor Test* [30]. These are discussed below.

The Overlap test or *Separation test* is based on the assumption that documents similar to each other are likely to be relevant to the same user queries, whereas dissimilar documents are unlikely to be relevant to the same queries. The test calculates all the *relevant-relevant* (RR) and *relevant-nonrelevant* (RNR) inter-document similarity coefficients. The test checks how much the average RR coefficient is larger than the average RNR coefficient. Additionally, coefficients can be summed over a set of test queries and plotted as a relative frequency histogram so as to check the overlap of the RR distribution against RNR distribution. The less the overlap between the two distributions, the better is the separation achieved between relevant and non-relevant documents by the measure under consideration.

The Nearest Neighbor test (NN test) aims at overcoming the limitation of the overlap test, wherein the relative frequency of non-relevant documents may far exceed that of relevant documents, thereby causing distortion in the generated results. The test consists of finding *N* most similar documents (nearest neighbors) for every relevant document of a query and subsequently, counting the number of relevant documents in that neighborhood. The higher the number of relevant documents in the neighborhood, higher is the separation between relevant and non-relevant documents, under a given measure. Furthermore, a single value (or a coefficient) for the entire test can be calculated by repeating the process for various other queries in the collection and computing the average number of relevant documents obtained in the neighborhood, *N*. Note that both Overlap and NN test are based on the concept of relevance, which is subjective in nature, but for the test sets endorsed by TREC.

Our testing procedure employs the NN test. For every test query accompanying a TREC collection, there is a predefined list of relevant documents, which makes it convenient to apply the NN test. It can also be noted that the NN test allows us to measure the clustering effectiveness of QSS measures without having to perform a complete clustering procedure. For the purpose of this thesis, we perform NN testing using measures M_1, M_2, and *cosine*. For each of the measures, we experiment with two term-weighting schemes, *ltc* and INQUERY. We particularly hope to observe a higher value of NN coefficients in case of QSS measures M_1 and M_2 implemented with INQUERY, as opposed to those obtained for the *ltc* scheme. It can be observed that higher the value of the NN coefficient better is the clustering effectiveness.

3.3. Previous Experimental Results

The SMART IR system [22] was used to identify the initial relevant documents in [28] using an *ltc* term-weighting scheme. Details of *ltc* term weighting are discussed in Section 2.2. The NN test, discussed in the previous subsection, was used to measure the degree of separation between relevant and non-relevant documents. Following an initial retrieval in SMART, the top-*n* ranked documents (as determined by SMART) were treated as a subcollection for further analysis using 5-Nearest Neighbor (5NN) test and 1NN test. The values five and one for NN test were adopted from [30] and [9]. The variable *n* assumed sample values of 100, 200, 350, 500, 750, 1000 and finally the number of documents in the entire collection. Testing for practical significance of the results used the *Wilcoxon signed-ranks* test [5],[9].

During a vast majority of the experimental conditions in [28], the measures M_1 and M_2 were found to outperform the conventional *cosine* measure with respect to the 5NN test. Moreover, the performance of *cosine* measure deteriorated with increasing values of *n* (used in denoting top-*n* documents). A similar study for measures M_1 and M_2 is open for further research [28]. With respect to the 5NN test, measure M_1 was shown to outperform $M_{2.}$ However, with respect to NN test, no such conclusion could be drawn. A query-by-query analysis could be more insightful in determining the conditions under which a measure is more effective than the other [28].

The effect of query length on the performance of a measure was also studied in [28] using the NN test. It was observed that for longer queries, M_1 consistently outperformed *cosine*. In order to compare effect of query length on M_1 and M_2, the queries in WSJ collection were partitioned into two sets. One set contained shorter queries with an average length of 3.2 terms, while another set contained longer queries with an average length of 23.4. Measure M_1 was found to be stable over change in query length, while M_2's performance showed a significant drop in performance for shorter queries. M_2 can still be used in conjunction with short user queries, if the system uses a reliable means of query expansion. Nevertheless, correlation between query characteristics and choice of an appropriate measure warrants further research [28].

4. IMPLEMENTED PROCEDURE

In the previous section we discussed the improvements offered by the QSS measures M_1 and M_2 over the conventional similarity measure, *cosine*, for a cluster-based information retrieval system. Specifically, we are interested in exploring ways of improving the QSS measures, M_1 and M_2. In [28], SMART's *ltc* term-weighting scheme is used for implementing a QSS measure. However, term-weighting schemes such as INQUERY have shown to offer better performance in comparison to the *ltc* scheme (section 2.2). We hypothesize that introduction of INQUERY tends to improve QSS measure's clustering effectiveness.

In order to test the hypothesis, we run the NN test on standard TREC collections, namely CISI and MED. A portion of the WSJ collection was used in [28] to test a collection whose topics were heterogeneous. However, the details of choosing a document subset from the WSJ collection were not discussed. We will avoid running the NN test on the complete WSJ collection due to its large volume (74,250). We rather use another heterogeneous collection namely the TIME collection (see Table 5), which is a collection of general magazine articles [1].

Table 5: Tested Document Collection Statistics

Attributes	MED	CISI	TIME
Number of docs	1033	1460	425
Number of queries	30	35	83
Mean terms per query	9.9	7.6	13
Mean relevant docs per query	23.2	49.8	8
Total relevant docs	696	1742	664
Mean unique terms per doc♣	80	72	290
Min unique terms per doc♣	13	9	51
Max unique terms per doc♣	279	230	1589
Std Dev unique terms per doc♣	33.7	29.5	180.2

♣ As determined by Ferret indexed document vectors (See subsection 4.2.2)

Specific variants of the NN tests namely 5NN test and 1NN test are used to test the separation of relevant and non-relevant documents under QSS measure with INQUERY term-weighting (section 3.2). The same tests are carried out using QSS measure coupled with *ltc* term-weighting on the aforementioned document collections. The results obtained for the QSS measure with INQUERY term-weighting are compared with those obtained for QSS measure with *ltc* term-weighting. The Wilcoxon signed-ranks test is used to determine the statistical significance of these results.

By definition of the NN test, we are required to perform an initial retrieval operation on a chosen document collection. For purposes of initial retrieval, we use the dtSearch engine [8]. The dtSearch engine is an industry-scale search engine based on the vector space model. It supports indexing and subsequent retrieval of documents much similar to the SMART system used in [28]. In general, a tool such as SMART or dtSearch offers a means of computing query-document similarity, which is the primary goal of the initial retrieval operation. Additionally, the data structures used in these tools for storing index terms are optimized for computing query-document similarity.

Out of the three measures under examination, measures M_1 and *cosine* contain a document-document similarity component in their formula, in addition to the query-document similarity component. In order to make possible the calculation of document-document similarity, we design and implement a separate software component named *Ferret test framework*. An important goal of the system is also to automate the calculation of the NN test coefficients. This is largely possible by complementing and extending the features provided by dtSearch. Also, amongst other term-weighting schemes, the test framework offers the INQUERY scheme, which is central to our experiments.

In summary, a standard TREC query is first presented to dtSearch engine, which will use its vector space model to retrieve top-*n* documents (*n* could assume values of 100, 200, 350, 500, etc.). Ferret test framework receives the top-*n* documents from dtSearch and then proceeds to apply NN test using the QSS measure. Running the NN test for larger values of *n* can be potentially time consuming. In order to improve the speed of operation of Ferret, certain time and memory efficient data structures, such as one presented in [6] are utilized. A detailed description of Ferret test framework's construction and working is provided in the following sections, 4.1 and 4.2.

4.1. Ferret Software Components

The primary objective of the Ferret system is to provide for a test framework that enables testing of cluster hypothesis and detection of potential gains introduced by a given term weighting scheme on QSS measures. We realize this objective by complementing the features provided by dtSearch engine with software extensions of our own. The following three software components were used while developing Ferret:

1. Microsoft's C# language [13] based on the .NET platform as the programming language to develop the Ferret core class library. These classes implement text filtering, stop list processing, stemming, indexing, and methods for QSS computations, forming bulk of the functionality provided by Ferret.

2. Open-source object database DB4O [7] as a persistent data store for the document vectors and certain collection-wide statistics generated as a result of Ferret's indexing process.

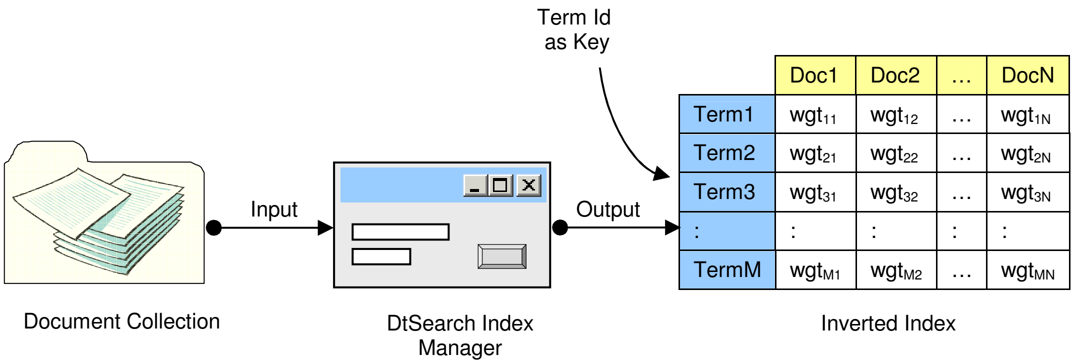
3. DtSearch engine's .NET Application Programming Interface (API) [8] is used for gaining programmatic access to results generated by dtSearch, which is crucial for Ferret's working.

4.2. Ferret Operation Details

The operation of Ferret test framework proceeds in multiple steps. As the first step, we index a given document collection using dtSearch indexing functionality. In the second step, we perform another indexing operation on the same collection, this time using Ferret indexing functionality. In the concluding step, we utilize the indices created in the previous steps along with the test query data to compute the NN test metric. In the subsections to follow, we will walk through the three steps in greater detail. Within them we will also describe the manner in which Ferret and dtSearch interact with each other to achieve the system objectives.

4.2.1. DtSearch Indexing

We are investigating QSS measure's utility in performing query-specific clustering on a document collection. Query-specific clustering involves generation of an initial relevancy list using the notion of query-document similarity. An *inverted index* has often been used in information retrieval as a fast and reliable technique for computation of query-document similarity [1]. DtSearch provides for an *index manager* application [8], which generates a compressed, inverted index, when submitted with a document collection. A given test document collection is initially indexed using the dtSearch index manager. A stop list recommended by the SMART IR system is used during the indexing operation. An illustration of the dtSearch indexing process is provided in Figure 1.

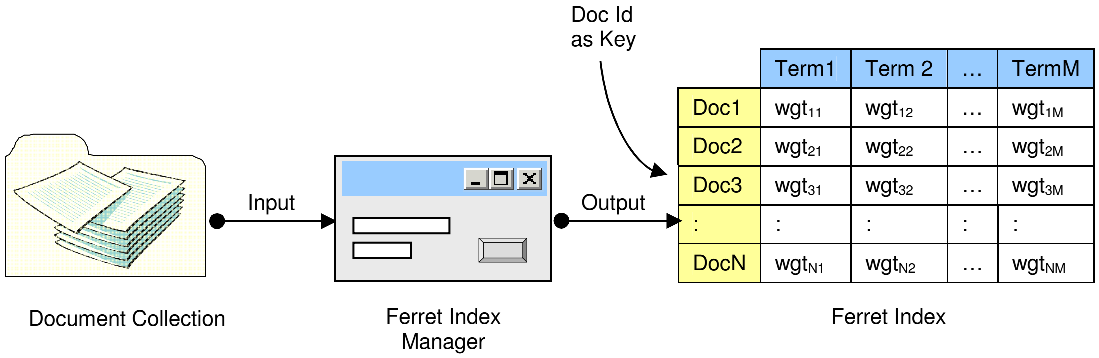


	Doc1	Doc2	…	DocN
Term1	wgt_{11}	wgt_{12}	…	wgt_{1N}
Term2	wgt_{21}	wgt_{22}	…	wgt_{2N}
Term3	wgt_{31}	wgt_{32}	…	wgt_{3N}
:	:	:	:	:
TermM	wgt_{M1}	wgt_{M2}	…	wgt_{MN}

Figure 1: Generation of Inverted Index Using dtSearch Index Manager

4.2.2. Ferret Indexing

QSS measure's definition contains two components – query-document similarity and document-document similarity. While query document similarity can be computed using the traditional inverted index, we use a complement of this data structure to compute the document-document similarity. More accurately, a Ferret index will contain vector representation of documents (document vectors), where given a unique document identifier (a fully qualified file path, in our case), we are able to access all non-noisy terms and their respective weights. This is the motivation behind creating a "non-inverted" index using the Ferret *index manager* application. A schematic diagram of this step is shown in Figure 2.



	Term1	Term 2	…	TermM
Doc1	wgt$_{11}$	wgt$_{12}$	…	wgt$_{1M}$
Doc2	wgt$_{21}$	wgt$_{22}$	…	wgt$_{2M}$
Doc3	wgt$_{31}$	wgt$_{32}$	…	wgt$_{3M}$
:	:	:	:	:
DocN	wgt$_{N1}$	wgt$_{N2}$	…	wgt$_{NM}$

Figure 2: Generation of Ferret Index Using Ferret Index Manager

To generate a document vector stored in a Ferret index, a series of processing steps are performed by the Ferret index manager on the document collection. This is depicted as flowcharts in Figure 3 and Figure 4. We avoid describing the flowchart, as it is largely self-explanatory. However, we would like to add a note about the collection vector entity. The collection vector records collection-wide frequency of terms (also known as document frequency for that term) and total vocabulary size. The collection-wide frequency of terms is used in term-weighting calculations involving both *ltc* and INQUERY formulae.

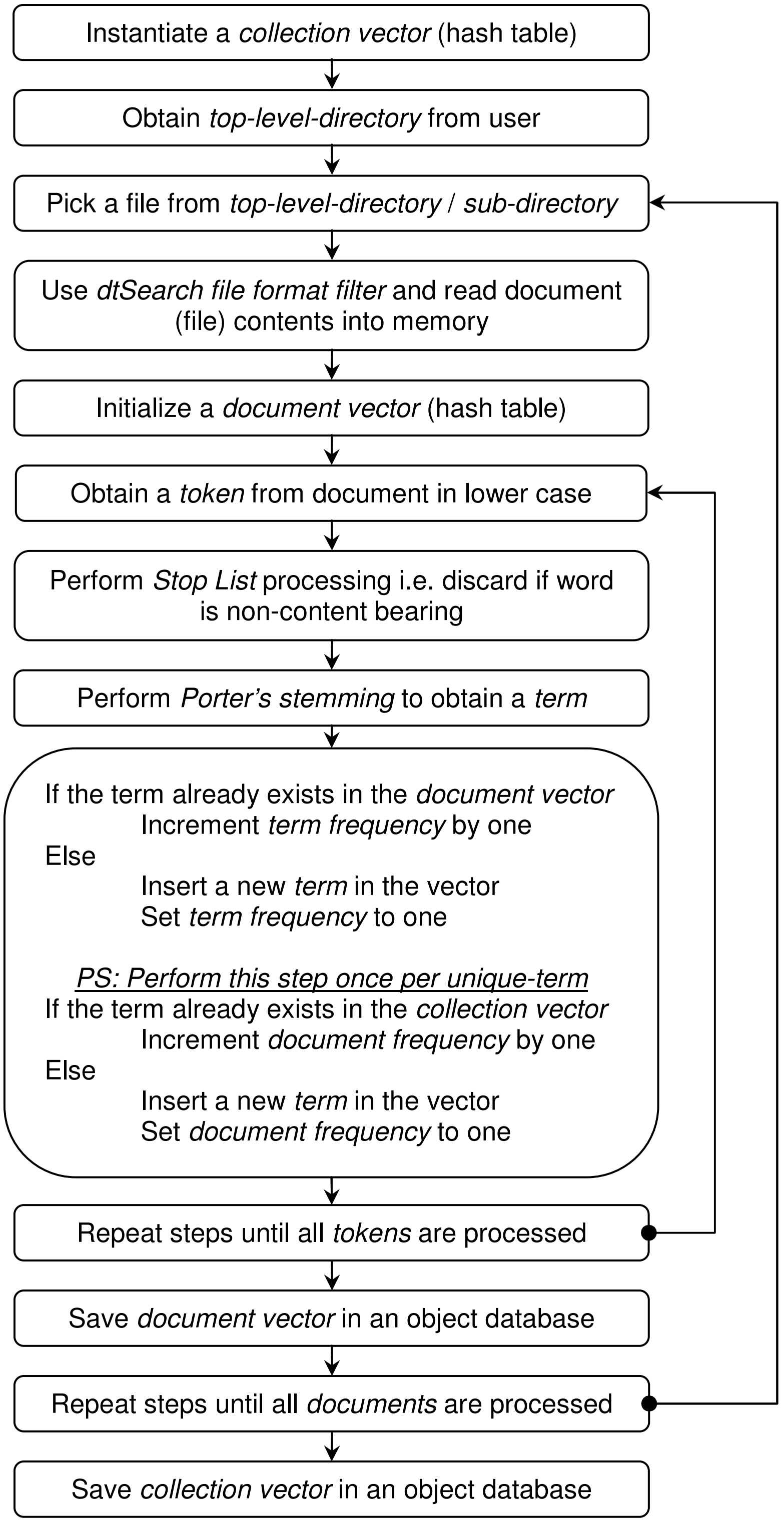

Instantiate a *collection vector* (hash table)
Obtain *top-level-directory* from user
Pick a file from *top-level-directory* / *sub-directory*
Use *dtSearch file format filter* and read document (file) contents into memory
Initialize a *document vector* (hash table)
Obtain a *token* from document in lower case
Perform *Stop List* processing i.e. discard if word is non-content bearing
Perform *Porter's stemming* to obtain a *term*
If the term already exists in the *document vector*
Increment *term frequency* by one
Else
Insert a new *term* in the vector
Set *term frequency* to one
PS: Perform this step once per unique-term
If the term already exists in the *collection vector*
Increment *document frequency* by one
Else
Insert a new *term* in the vector
Set *document frequency* to one
Repeat steps until all *tokens* are processed
Save *document vector* in an object database
Repeat steps until all *documents* are processed
Save *collection vector* in an object database

Figure 3: Algorithm for Generation of Ferret Index – Phase I: Recording Term Frequencies

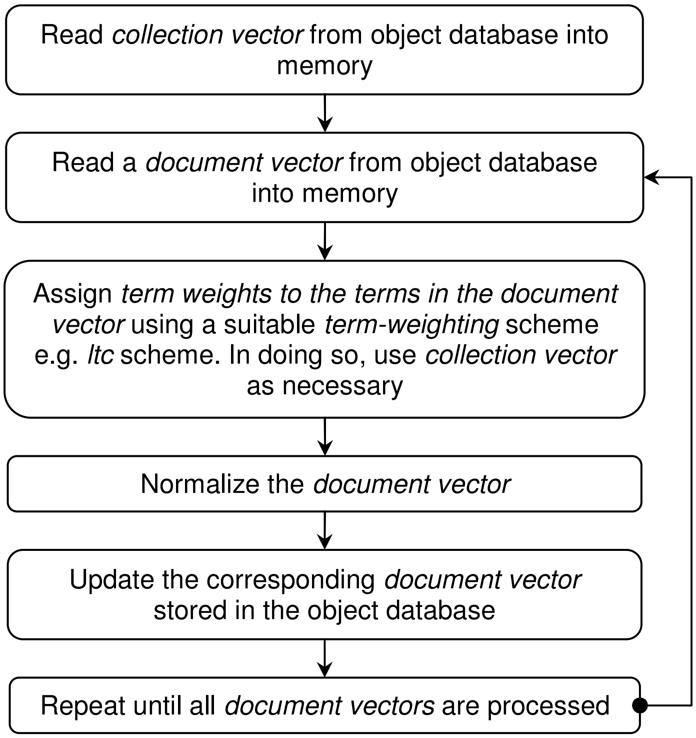

Read *collection vector* from object database into memory
Read a *document vector* from object database into memory
Assign *term weights to the terms in the document vector* using a suitable *term-weighting* scheme e.g. *ltc* scheme. In doing so, use *collection vector* as necessary
Normalize the *document vector*
Update the corresponding *document vector* stored in the object database
Repeat until all *document vectors* are processed

Figure 4: Algorithm for Generation of Ferret Index – Phase II: Updating Weights

4.2.3. Automated Testing

For convenience of the reader, in this section, each entity in Figure 5 is labeled with an alphabet, which will appear in the description next to the entity. The Ferret automated test framework (*G*) expects the availability of the following entities for its proper functioning:

1. An inverted index (*A*) created for a test document collection using dtSearch, the details of which were discussed in the section 4.2.1.

2. The dtSearch engine application (*B*). In fact, we utilize dtSearch engine APIs rather than an interactive version of the application. The utility of dtSearch APIs is twofold. Firstly, the search engine APIs can be used to process queries without actually invoking a user interface. Secondly, the search results generated by dtSearch are available as program objects that can be used readily by the Ferret programs for calculation of the NN metric. Both aspects are important from an automation perspective.

3. A standard test query list (*D*) that accompanies the test document collection.

4. A standard test query relevancy list that accompanies the test query set (*F*). This list specifies the relevant documents expected for each query in the list. The relevant documents in this case are determined by authors of the test document collection.

5. A Ferret index (*E*) also created for the same test document collection using Ferret index manager. Details pertaining to Ferret index were discussed in subsection 4.2.2.

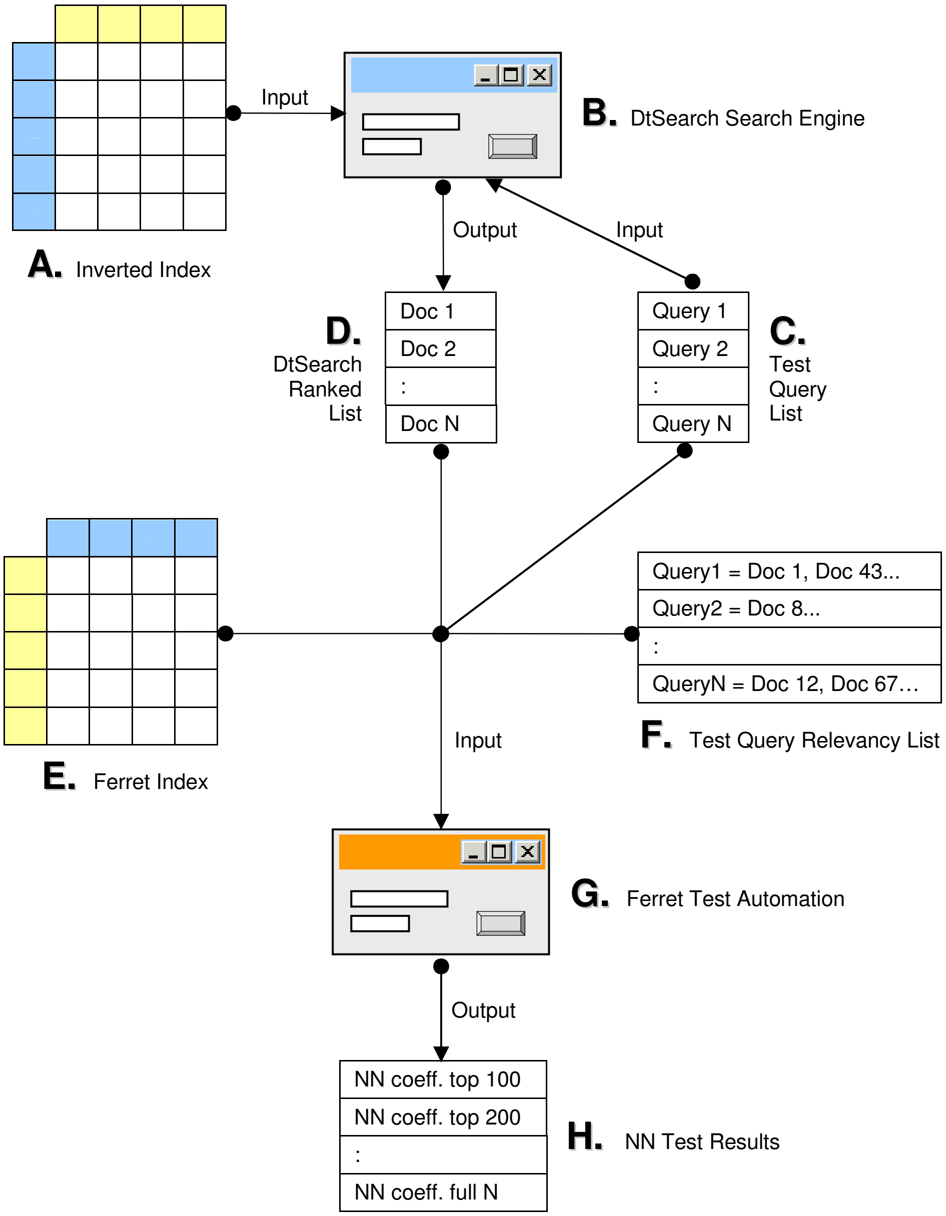


Figure 5: Schematic Diagram for Computation of NN Coefficients by Ferret Test Framework

Following is a high level algorithmic description of the steps performed by Ferret test framework to arrive at the NN coefficients:

1. Choose a query from the standard test query list (*D*) and pass it to the dtSearch engine API as a parameter.

2. Using the inverted index (*A*), dtSearch engine (*B*) generates a list of *N* ranked documents (*C*), which are considered relevant according to the vector space model in use by dtSearch.

3. From the ranked list (*C*), pick top-*n* documents for running the NN test, where $n \leq N$. For example *n* can be assigned a value 100.

4. Apply the NN test. The test takes into account only documents that are relevant to the query. This information is provided by the test query relevancy set (*F*). For a relevant document, we find (let us say) five nearest neighbors using a given similarity measure (M_1, M_2, or *cosine*). We then count the number of relevant documents that appear in this neighborhood. Repeat the process for each relevant document appearing in the top-*n* set. Compute the average count to obtain NN coefficient for the current query.

   From our previous discussions about similarity measures and the role of Ferret index, we can convince ourselves that using Ferret index (*E*) and test query list (*F*) all calculations described in the aforementioned paragraph can be performed using either the *ltc* or INQUERY term-weighted vectors. Of course, the Ferret index (*E*), containing the document vectors, must be generated using one of the two term-weighting schemes and the queries from the test query list (*F*) must be treated as query vectors.

5. Repeat steps 1 through 4 for the remaining queries in the test query set. Compute the average NN coefficient over all the queries to obtain the NN coefficient for the top-*n* set.

6. Repeat steps 1 through 5 by varying the value of *n*. The typical values chosen for *n* are 100, 200, 300, 400, 500, 700, up until all *N* documents are picked. Record NN coefficients for each top-*n* set.

The NN coefficients appearing in the Tables 6–23 (Section 5) are derived using the aforementioned steps. We have generated NN coefficients for two different neighborhood sizes – one and five, which are respectively termed 1NN and 5NN tests.

5. EXPERIMENTAL RESULTS AND DISCUSSION

The NN test coefficients are computed for MED, CISI and TIME test collections using the procedure described in Section 4.2.3. The term-weighting schemes in use are one of *ltc* or INQUERY. The similarity measures used are one among M_1, M_2 and *cosine*. The results are bifurcated into two sets based on the neighborhood size used by the NN test. The first set contains results obtained for the 5NN test (e.g. Tables 6–8), whereas the second set contains those obtained for the 1NN test (e.g. Tables 9–11). Both sets are internally subdivided into results obtained for a combination of a given test collection and a similarity measure. For example, in the 5NN result set, NN coefficients for MED collection using measure M_1, M_2, and *cosine* and respectively appear in Table 6, Table 7, and Table 8. The remaining results are organized in a similar fashion.

A table of result consists of five columns. The first column (e.g. *top-n* in Table 6) indicates the number of documents presented to the Ferret test framework after the initial retrieval step performed by dtSearch. In other words, it is the number of relevant documents across which the NN coefficient is computed. The second column (e.g. M_1 *Ltc* in Table 6) contains the NN coefficient computed with *ltc* as the underlying term-weighting scheme. The third column (e.g. M_1 *Inq* in Table 6) contains the NN coefficient computed with INQUERY as the underlying term-weighting scheme. The fourth column (e.g. M_1 *Ltc %* in Table 6) displays the NN coefficients of the second column as a percentage of the neighborhood size searched. The fifth column (e.g. M_1 *Inq %* in Table 6) displays the NN coefficients of the third column as a percentage of the neighborhood size searched. It could be noted that for the 1NN test, the NN coefficient in percentage format or otherwise almost conveys the same information. We nevertheless retain both columns for the sake of completeness. For the convenience of the reader, we indicate in each table the term-weighting scheme and similarity measure combination, which performs significantly better by placing an asterisk mark in the corresponding column. The peak NN coefficient values attained by a term-weighting scheme and similarity measure combination is displayed in bold.

5.1. NN Test on MED Test Collection

Subsets of 100, 200, 300, 400, 500, 700 and 1033 (full size) documents are used for computation of NN coefficients for the MED test collection. In the 5NN case, on analyzing results for measure M_1 (Table 6), the *ltc* scheme is found to be offering better performance in comparison to the INQUERY scheme. The conclusion is drawn based on the Wilcoxon signed-ranks test at 95% and 97.5% confidence interval. The trend is also visible in the marginally higher NN coefficients produced by the *ltc* scheme. On the other hand, INQUERY scheme offers better performance in comparison to *ltc* scheme when we evaluated the results of the 1NN test (Table 9). The relative comparison of performance was also carried out using the aforementioned statistical significance test at 95% and 97.5% confidence intervals. In both variants of the NN test, the maximum percentage of nearest neighbors achieved by both *ltc* and INQUERY are comparable. However, in the 5NN case, the *ltc* scheme achieves its maximum NN coefficient of 58.69% much earlier in the experimentation i.e. at a smaller value for top-*n*. In the 1NN case, both schemes reach their peak NN coefficients at the same value for top-*n*.

The 5NN test results obtained for the *cosine* measure does not statistically favor either of the term-weighting schemes (Table 7). At a rather high confidence interval of 99%, the *ltc* term-weighting scheme demonstrates a slight edge over INQUERY. However, the peak NN coefficient attained by *ltc* is somewhat higher than that observed for M_1. In context of the 1NN test, statistical testing declares *ltc* scheme to be outperforming INQUERY at confidence intervals of 95% and 97.5% (Table 10). Nevertheless, the NN coefficients reach their peak values simultaneously.

The results of the 5NN test obtained for M_2 measure favors INQUERY term-weighting over *ltc* at 99% confidence interval (Table 8). The better performance of INQUERY in this case is hardly surprising, as M_2 is represents the traditional query-document similarity semantics. Recall that INQUERY was introduced as better alternative to traditional query-document similarity measures such as *ltc* (Section 2.2). In case of the

1NN test, the INQUERY scheme outperforms the *ltc* schemes at relatively convincing confidence intervals of 95% and 97.5% (Table 11).

Table 6: 5NN Test for M_1 Measure, MED Document Collection

Top-n	* M_1 Ltc	M_1 Inq	* M_1 Ltc %	M_1 Inq %
100	2.085690	2.055713	41.71%	41.11%
200	2.471918	2.467793	49.44%	49.36%
300	2.806847	2.808047	56.14%	56.16%
400	2.910682	2.902953	58.21%	58.06%
500	**2.984124**	2.972480	**59.68%**	59.45%
700	2.974297	2.978310	59.49%	59.57%
1033	2.983248	**2.983865**	59.66%	**59.68%**

Table 7: 5NN Test for M_2 Measure, MED Document Collection

Top-n	M_2 Ltc	* M_2 Inq	M_2 Ltc %	* M_2 Inq %
100	2.093241	2.091340	41.86%	41.83%
200	2.481522	2.484312	49.63%	49.69%
300	2.742126	2.758849	54.84%	55.18%
400	2.832290	2.898879	56.65%	57.98%
500	**2.900624**	**2.901376**	**58.01%**	**58.03%**
700	2.842378	2.879320	56.85%	57.59%
1033	2.842378	2.879320	56.85%	57.59%

Table 8: 5NN Test for cosine Measure, MED Document Collection

Top-n	* Cos Ltc	Cos Inq	* Cos Ltc %	Cos Inq %
100	1.814993	1.776424	36.30%	35.53%
200	2.172938	2.146123	43.46%	42.92%
300	2.463876	2.431742	49.28%	48.63%
400	2.562125	2.498521	51.24%	49.97%
500	2.617374	2.552471	52.35%	51.05%
700	2.611766	2.553530	52.24%	51.07%
1033	**2.622260**	**2.568036**	**52.45%**	**51.36%**

Table 9: 1NN Test for M_1 Measure, MED Document Collection

Top-n	M_1 Ltc	* M_1 Inq	M_1 Ltc %	* M_1 Inq %
100	0.645319	0.645157	64.53%	64.52%
200	0.687051	0.691897	68.71%	69.19%
300	0.779197	0.770147	77.92%	77.01%
400	0.771325	0.772178	77.13%	77.22%
500	0.783528	0.784010	78.35%	78.40%
700	**0.785433**	**0.785915**	**78.54%**	**78.59%**
1033	0.783581	0.784063	78.36%	78.41%

Table 10: 1NN Test for M_2 Measure, MED Document Collection

Top-n	M_2 Ltc	* M_2 Inq	M_2 Ltc %	* M_2 Inq %
100	0.675278	0.696528	67.53%	69.65%
200	0.711880	0.704103	71.19%	70.41%
300	0.782947	0.786280	78.29%	78.63%
400	0.789613	0.792947	78.96%	79.29%
500	**0.793620**	**0.796650**	**79.36%**	**79.67%**
700	0.760287	0.763317	76.03%	76.33%
1033	0.760287	0.763317	76.03%	76.33%

Table 11: 1NN Test for cosine Measure, MED Document Collection

Top-n	* Cos Ltc	Cos Inq	* Cos Ltc %	Cos Inq %
100	0.535856	0.504963	53.59%	50.50%
200	0.645435	0.645309	64.54%	64.53%
300	0.672284	0.680505	67.23%	68.05%
400	0.684663	0.687511	68.47%	68.75%
500	0.702369	0.700738	70.24%	70.07%
700	0.702691	0.701218	70.27%	70.12%
1033	**0.703000**	**0.702762**	**70.30%**	**70.28%**

5.2. NN Test on CISI Test Collection

Document subsets of size 100, 200, 300, 400, 500, 700, 1000 and 1460 (full size) are used during the NN test computations for the CISI test collection. On comparing 5NN coefficients for measure M_1 at 99.5% confidence interval, the *ltc* scheme outperforms the INQUERY scheme (Table 12). This improvement is not significant at 95%, 97.5% and 99% confidence intervals. Exactly same conclusions are drawn on comparing the 1NN coefficients for the two schemes (Table 15). In the 5NN case, *ltc* reaches the peak coefficient value at a top-*n* value of 400, whereas INQUERY does so at a top-*n* value of 1460. For the 1NN case, the peak values are comparable and are reached for same value of top-*n*, 400. At the same time, for every top-*n* value, the 1NN coefficient for *ltc* is a percentage point above that for INQUERY.

On testing for the *cosine* measure using the 5NN coefficients, the *ltc* scheme displays better adherence to cluster hypothesis at a somewhat high confidence interval of 99.5% (Table 13). The results of the 1NN test yield precisely the same results (Table 16). Both schemes attain peak NN coefficients at the same top-*n* value for the 5NN and 1NN tests. The peak coefficients obtained for *ltc* appears to be marginally higher than that obtained for INQUERY.

The result for measure M_2 was somewhat anomalous. In the 5NN case, INQUERY emerged as a superior scheme (Table 14), whereas in the 1NN case (Table 17), *ltc* held the upper hand. These results were statistically significant at a high confidence interval of 99.5%. In a rather unusual observation for the 1NN case, the *ltc* NN coefficients were higher than those of INQUERY by a margin of approximately 4.4%.

Table 12: 5NN Test for M_1 Measure, CISI Document Collection

Top-n	* M_1 Ltc	M_1 Inq	* M_1 Ltc %	M_1 Inq %
100	1.041902	1.063106	20.84%	21.26%
200	1.224077	1.217391	24.48%	24.35%
300	1.259389	1.231504	25.19%	24.63%
400	**1.277357**	1.246867	**25.55%**	24.94%
500	1.259369	1.226175	25.19%	24.52%
700	1.236281	1.214805	24.73%	24.30%
1000	1.251345	1.237665	25.03%	24.75%
1460	1.269168	**1.249964**	25.38%	**25.00%**

Table 13: 5NN Test for M_2 Measure, CISI Document Collection

Top-n	M_2 Ltc	* M_2 Inq	M_2 Ltc %	* M_2 Inq %
100	0.947475	1.006222	18.95%	20.12%
200	1.238875	1.258032	24.78%	25.16%
300	1.300129	1.334016	26.00%	26.68%
400	1.309914	**1.387360**	26.20%	**27.75%**
500	**1.321122**	1.339856	**26.42%**	26.80%
700	1.268840	1.313490	25.38%	26.27%
1000	1.213269	1.288562	24.27%	25.77%
1460	1.213607	1.288927	24.27%	25.78%

Table 14: 5NN Test for cosine Measure, CISI Document Collection

Top-n	* Cos Ltc	Cos Inq	* Cos Ltc %	Cos Inq %
100	0.897963	0.878131	17.96%	17.56%
200	0.954714	0.920684	19.09%	18.41%
300	0.942494	0.919382	18.85%	18.39%
400	**0.958163**	**0.933884**	**19.16%**	**18.68%**
500	0.903175	0.872565	18.06%	17.45%
700	0.868580	0.842632	17.37%	16.85%
1000	0.906606	0.881112	18.13%	17.62%
1460	0.927394	0.900792	18.55%	18.02%

Table 15: 1NN Test for M_1 Measure, CISI Document Collection

Top-n	* M_1 Ltc	M_1 Inq	* M_1 Ltc %	M_1 Inq %
100	0.365836	0.365255	36.58%	36.53%
200	0.370059	0.361716	37.01%	36.17%
300	0.384628	0.366083	38.46%	36.61%
400	**0.387915**	**0.381577**	**38.79%**	**38.16%**
500	0.361834	0.356351	36.18%	35.64%
700	0.360107	0.355430	36.01%	35.54%
1000	0.367028	0.355961	36.70%	35.60%
1460	0.369612	0.358114	36.96%	35.81%

Table 16: 1NN Test for M_2 Measure, CISI Document Collection

Top-n	* M_2 Ltc	M_2 Inq	* M_2 Ltc %	M_2 Inq %
100	0.273174	0.275048	27.32%	27.50%
200	**0.407730**	**0.355195**	**40.77%**	**35.52%**
300	0.390664	0.336811	39.07%	33.68%
400	0.365092	0.310333	36.51%	31.03%
500	0.365666	0.310786	36.57%	31.08%
700	0.366604	0.339937	36.66%	33.99%
1000	0.367792	0.312422	36.78%	31.24%
1460	0.367819	0.312422	36.78%	31.24%

Table 17: 1NN Test for cosine Measure, CISI Document Collection

Top-n	* Cos Ltc	Cos Inq	* Cos Ltc %	Cos Inq %
100	**0.324447**	**0.314380**	**32.44%**	**31.44%**
200	0.293499	0.288947	29.35%	28.89%
300	0.299436	0.293388	29.94%	29.34%
400	0.316586	0.309163	31.66%	30.92%
500	0.290002	0.282519	29.00%	28.25%
700	0.281124	0.265384	28.11%	26.54%
1000	0.291972	0.284407	29.20%	28.44%
1460	0.296324	0.289336	29.63%	28.93%

5.3. NN Test on TIME Test Collection

In case of the TIME test collection, document subsets of size 100, 200, 300, 400, and 426 (full size) are used for NN test computations. Comparison of the 5NN coefficients for measure M_1 statistically shows measure *ltc* as being a superior scheme (Table 18). A confidence interval of 99.5% applies in this case. However, the peak coefficients and the top-*n* value at which the two schemes peak are comparable. In case of the 1NN test, the INQUERY term-weighting scheme emerges as a better system over the *ltc* scheme (Table 21). The statistical test at 95% confidence interval confirms this conclusion. The peak value of the NN coefficient reached by INQUERY is higher than that of *ltc* by approximately 1.8%.

In case of the *cosine* similarity measure for the 5NN test, we conclude that the *ltc* scheme outperforms the INQUERY scheme at 99.5% confidence interval (Table 19). The peak coefficient value of *ltc* scheme is marginally higher than that of INQUERY, although they both reach their peaks for approximately the same top-*n* value. In the 1NN case, the INQUERY scheme outperforms the *ltc* scheme, also at 99.5% confidence interval (Table 22). The peak coefficient of the INQUERY scheme is 1.6% higher than that of *ltc* scheme, which appears to be a reasonable improvement.

Surprisingly, on observing the 5NN results for measure M_2, we find the *ltc* term-weighting scheme outdoing INQUERY scheme at 95% confidence interval (Table 20). The peak coefficient of *ltc* exceeds that of INQUERY by just 0.44%. The trend is exactly the opposite in the context of the 1NN test for the same confidence interval (Table 23). The peak coefficient of INQUERY exceeds that of *ltc* by a convincing 2.33%.

Table 18: 5NN Test for M_1 Measure, TIME Document Collection

Top-n	* M_1 Ltc	M_1 Inq	* M_1 Ltc %	M_1 Inq %
100	1.130532	1.107677	22.61%	22.15%
200	1.208314	1.215407	24.17%	24.31%
300	1.277144	1.267900	25.54%	25.36%
400	**1.283799**	**1.275576**	**25.68%**	**25.51%**
426	1.283799	1.275576	25.68%	25.51%

Table 19: 5NN Test for M_2 Measure, TIME Document Collection

Top-n	* M_2 Ltc	M_2 Inq	* M_2 Ltc %	M_2 Inq %
100	1.139950	1.126506	22.80%	22.53%
200	1.238531	1.218876	24.77%	24.38%
300	1.310619	1.281124	26.21%	25.62%
400	**1.318651**	**1.296185**	**26.37%**	**25.92%**
426	1.318651	1.296185	26.37%	25.92%

Table 20: 5NN Test for cosine Measure, TIME Document Collection

Top-n	* Cos Ltc	Cos Inq	* Cos Ltc %	Cos Inq %
100	1.027862	1.013764	20.56%	20.28%
200	1.112645	1.093736	22.25%	21.87%
300	1.160607	**1.143577**	23.21%	**22.87%**
400	**1.160619**	1.143204	**23.21%**	22.86%
426	1.160619	1.143204	23.21%	22.86%

Table 21: 1NN Test for M_1 Measure, TIME Document Collection

Top-n	M_1 Ltc	* M_1 Inq	M_1 Ltc %	* M_1 Inq %
100	0.400979	0.385825	40.10%	38.58%
200	0.402305	0.393644	40.23%	39.36%
300	**0.419574**	0.428483	**41.96%**	42.85%
400	0.417897	**0.430029**	41.79%	**43.00%**
426	0.417897	0.430029	41.79%	43.00%

Table 22: 1NN Test for M_2 Measure, TIME Document Collection

Top-n	M_2 Ltc	* M_2 Inq	M_2 Ltc %	* M_2 Inq %
100	0.382081	0.401759	38.21%	40.18%
200	0.390113	0.425053	39.01%	42.51%
300	0.431278	0.457583	43.13%	45.76%
400	**0.440557**	**0.46385**	**44.06%**	**46.39%**
426	0.440557	0.46385	44.06%	46.39%

Table 23: 1NN Test for cosine Measure, TIME Document Collection

Top-n	Cos Ltc	* Cos Inq	Cos Ltc %	* Cos Inq %
100	0.351428	0.355511	35.14%	35.55%
200	0.350827	0.364248	35.08%	36.42%
300	**0.373016**	0.386832	**37.30%**	38.68%
400	0.371344	**0.389037**	37.13%	**38.90%**
426	0.371344	0.389037	37.13%	38.90%

5.4. Comparison among Measures M_1, M_2 and Cosine

As observed in [28], for a given test collection, we find that measure M_1 and M_2, both of which are query-sensitive measures, performs significantly better than the *cosine* measure. This observation applies equally to the *ltc* case and to the INQUERY case. Moreover, the observation holds true for both 1NN and 5NN tests. This reiterates the fact that QSS measures are an improvement over the traditional *cosine* measure [28]. No such unilateral trend exists when we compare relative performance of M_1 versus M_2. For example, for the 5NN tests on the MED collection, M_1 performs better than M_2 regardless of the term-weighting scheme used, whereas for the 5NN test on the TIME collection, M_2 performs better than M_1.

6. CONCLUSIONS AND FUTURE WORK

In this thesis, we have indexed test collections using two term-weighting schemes, INQUERY and *ltc*. We have applied QSS measures, M_1 and M_2, and traditional inter-document similarity measure, *cosine*, in conjunction with both the term-weighting schemes. On running the NN test and testing for statistical significance of the results, we observed reasonably encouraging results, the summary of which is presented in Table 24. For each test collection, the table displays a check mark against the term-weighting scheme and the measure for which the results were significantly better than the other combination.

Table 24: Summary of Relative Performance of the Term-Weighting Schemes

	M_1		M_2		Cosine	
	Inq	*ltc*	*Inq*	*ltc*	*Inq*	*ltc*
MED 5NN		✓	✓			✓
MED 1NN	✓		✓			✓
CISI 5NN		✓	✓			✓
CISI 1NN		✓		✓		✓
TIME 5NN		✓		✓		✓
TIME 1NN	✓		✓		✓	

In the context of QSS measure M_1, the *ltc* scheme was found to perform better than the INQUERY scheme. A possible reason for INQUERY not performing as well in this case could be due to the fact that equal weights were offered to the two components comprising M_1 namely, M_2 and *cosine*. By design, term-weighting schemes have a better chance of strongly influencing the query-sensitive similarity measure, M_2. However, it may be noted that INQUERY scheme was not lagging behind by too large a margin. Moreover, in the case of 1NN tests for MED and TIME collections, INQUERY scheme actually outperforms the *ltc* scheme.

When we observe the results for QSS measure M_2, INQUERY scheme holds the upper hand over the *ltc* scheme for most part of the experimentation. This result is expected, as INQUERY weighting was originally suggested as an improved method of computing query-document similarity measure, such as M_2. Observing results for the *cosine* measure, *ltc* scheme holds an edge over the INQUERY scheme. Rounding up our discussion, we can state that the INQUERY term-weighting scheme wields a positive influence on QSS measures, M_1 and M_2. However, the results obtained for measure M_1 were far less conclusive than it was for the M_2 measure. In the following two paragraphs, we discuss some of the potential reasons for the aforementioned observation.

The Ferret test framework uses dtSearch for initial retrieval operation. The initial retrieval operation completely depends on the vector space model (and term-weighting) in use within dtSearch engine, and not on the term-weighting scheme under test. Hence, the recall and precision characteristics of INQUERY or *ltc* scheme may have been overshadowed by use of dtSearch engine. This is a limitation of the test framework in use.

By definition, measure M_1 (Section 2.5) is a function of two components M_2 and *cosine*. In our experimentation, we have provided equal weighing to both these components. Providing appropriate weighting to these components taking into account the characteristics of the document collection and the nature of queries can significantly improve the retrieval performance [29]. For example, in a system where query-document similarity is considered more important than the static *cosine* similarity between documents, the M_2 component will be given more weighting than *cosine* component. Since we have observed that INQUERY weighting-based performance is noticeably good with measure M_2, this system might offer better results over an *ltc* weighting-based system.

Following up on our discussion in section 2.2, we expected the INQUERY term-weighting formula (and the corresponding QSS measure) used in our experiments to display superior clustering performance when used for document collections with significant variations in document length. The TIME collection (Table 5) is an example of one such document collection. Although we noticed this trend in case of the 1NN test for measures M_1 and M_2, it was not visible in case of the 5NN test involving the same measures. Further experimentation with document collections resembling TIME may be worth performing in the future.

In this thesis, we have not analyzed whether query characteristics can influence the choice of a QSS measure and a term-weighting scheme. A query-by-query analysis of the clustering effectiveness may reveal dependence of QSS measures on query characteristics such as length, query term-weighting and the average number of terms common with a query and its relevant documents. Another possible future work would involve constructing a practical clustering system based on variants of QSS measures and term-weighting schemes. By evaluating and comparing end-user experience, we might have a clearer understanding of the benefits of one system over the other.

Finally, we have explained the construction and working of a QSS measure testing system from readily available software components. The design principles used in construction of the framework may prove as a good starting point for similar systems attempted in the future.

REFERENCES

[1] R. Baeza-Yates and B. Ribiero-Neto. *Modern Information Retrieval*. New York: ACM Press, 1999.

[2] J. Broglio, J. Callan, W.B. Croft, and D. Nachbar, "Document Retrieval and Routing Using the INQUERY System," In D. K. Harman, editor, *Proceedings of the Third Text Retrieval Conference* (*TREC-3*), NIST, 1995.

[3] C. Buckley, "The Importance of Proper Weighting Methods," *Proceedings of the DARPA Workshop on Human Language Technology*, pp. 349-352, Princeton, NJ, 1993.

[4] F. Can and E.A. Ozkarahan, "Concepts and Effectiveness of the Cover-Coefficient-Based Clustering Methodology for Text Databases," *ACM Transactions on Database Systems*, vol. 15, no. 4, pp. 483-517, December 1990.

[5] W.B. Croft. *Organizing and Searching Large Files of Document Descriptions*. Ph.D. Thesis, United Kingdom, University of Cambridge, 1978.

[6] I.S. Dhillon, Y. Guan, and J. Fan, "Efficient Clustering of Very Large Document Collections," Invited book chapter in *Data Mining for Scientific and Engineering Applications*, pp. 357-381, Boston: Kluwer Academic Publishers, 2001.

[7] DB4O (Database for objects) online documentation, db4objects Inc., San Mateo, CA 2005. Available: http://www.db4o.com/community/ (Accessed: June 2005)

[8] dtSearch online documentation, dtSearch Corp., Bethesda, MD 2005. Available: http://www.dtsearch.com/ (Accessed: May 2005)

[9] A. El-Hamdouchi. *Using Inter-Document Relationships in Information Retrieval*. Ph.D. Thesis, United Kingdom, University of Sheffield, 1987.

[10] D. Ellis, J. Furner-Hines, and P. Willett, "Measuring the Degree of Similarity Between Objects in Text Retrieval Systems," *Perspectives in Information Management*, vol. 3, no. 2, pp. 128-149, 1993.

[11] N. Jardine and C. J. van Rijsbergen, "The Use of Hierarchic Clustering in Information Retrieval," Information Storage and Retrieval, vol. 7, no. 5, pp. 217-240, December 1971.

[12] R.R. Korfhage. *Information Storage and Retrieval*. New York: Wiley Computer Publishing, 1997.

[13] Micrsoft Visual C# online documentation, Microsoft Corp., Redmond, WA 2005. Available: http://msdn.microsoft.com/vcsharp/ (Accessed: July 2005)

[14] G. Muresan, *Using Document Clustering and Language Modelling in Mediated Information Retrieval*. Ph.D. thesis, School of Computing, Robert Gordon University, Aberdeen, Scotland, United Kingdom, January 2002.

[15] J.M. Ponte and W.B. Croft, “A Language Modeling Approach to Information Retrieval,” *Proceedings of ACM SIGIR ’98*, pp. 275–281, Melbourne, 1998.

[16] C.J. van Rijsbergen and K.S. Jones, “A Test for the Separation of Relevant and Non-Relevant Documents in Experimental Retrieval Collections,” *Journal of Documentation*, vol. 29, no. 3, pp. 251-257, 1973.

[17] C.J. van Rijsbergen, *Information Retrieval*. 2nd Edition, London: Butterworths, 1979.

[18] S.E. Robertson and K. Sparck Jones. *Simple, Proven Approaches to Text Retrieval.* Technical Report 356, Computer Laboratory, Cambridge University, May 1997.

[19] S.E. Robertson and S. Walker, “Some Simple Effective Approximations to the 2-Poisson Model for Probabilistic Weighted Retrieval,” *Proceedings of ACM SIGIR ’98*, pp. 232-241, 1994.

[20] 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*), NIST, 1995.

[21] S.E. Robertson, S. Walker, and M. Beaulieu, “Experimentation as a way of life: Okapi at TREC,” Information Processing & Management, vol. 36, no. 1, pp. 95-108, January 2000.

[22] G. Salton, *The SMART Retrieval System - Experiments in Automatic Document Retrieval*. Englewood Cliffs, NJ: Prentice Hall Inc., 1971.

[23] G. Salton, A. Wong, and C. S. Yang, “A Vector Space Model for Automatic Indexing,” *Communications of the ACM*, vol. 18, no. 11, pp. 613-620, 1975.

[24] G. Salton and C. Buckley, “Term Weighting Approaches in Automatic Text Retrieval,” *Information Processing and Management*, vol. 24, no. 5, pp. 513-523, 1988.

[25] A. Singhal, G. Salton, M. Mitra, and C. Buckley, “Document Length Normalization,” *Information Processing & Management*, vol. 32, no. 5, pp. 619-633, September 1996

[26] A. Singhal, C. Buckley, and M. Mitra, "Pivoted Document Length Normalization," *SIGIR '96*: *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 21–29, 1996.

[27] P.H.A. Sneath, R.R. Sokal, D. Kennedy, and R.B. Park. *Numerical Taxonomy*. San Francisco: W.H. Freeman and Company, 1973.

[28] A. Tombros and C. J. van Rijsbergen, "Query-Sensitive Similarity Measures for the Calculation of Interdocument Relationships," *CIKM '01*: *Proceedings of the Tenth International Conference on Information and Knowledge Management*, pp.17-24, November 2001.

[29] A. Tombros and C. J. van Rijsbergen, "Query-Sensitive Similarity Measures for Information Retrieval," *Knowledge and Information Systems*, vol. 6, no. 5, pp. 617-642, September 2004.

[30] E.M. Voorhees. *The Effectiveness and Efficiency of Agglomerative Hierarchic Clustering in Document Retrieval*. Ph.D. dissertation, Department of Computing Science, Cornell University, 1985.

[31] T. Woodfield. *Text Mining Using SAS Software*. SAS Course Notes, Cary, NC: SAS Institute Inc., 2003.

VITA

Contact Details Ananth Ullal Kini

Permanent Address 46, Dena Bank Colony, 2nd Main, 3rd Cross, Ganganagar,
Bangalore, India 560032

Email Address kini@tamu.edu, kiniananth@gmail.com

Education M.S., Computer Science Aug 2003 – Dec 2005
Texas A&M University (TAMU)
College Station, TX

B.E., Computer Science Nov 1997 – Aug 2001
University of Mysore, SJCE
Mysore, India