รายงานวิจัย

A Graph-Based Information Retrieval Model

ผู้ช่วยศาสตราจารย์ ดร.โอม ศรนิล

กันยายน 2552

สถาบันบัณฑิตพัฒนบริหารศาสตร์

สถาบันบัณฑิตพัฒนบริหารศาสตร์
118 ถนนเสรีไทย คลองจั่น บางกะปิ
กรุงเทพมหานคร 10240
ประเทศไทย

โทร : 662-375-8972
โทรสาร: 662-374-2759
E-mail : rcadmin@nida.ac.th

A Graph-Based Information Retrieval Model

Ohm Sornil
School of Applied Statistics
National Institute of Development Administration
Bangkok, Thailand

Acknowledgement: This research was supported by Research Center, National Institute of Development Administration (NIDA).

ABTRACT

A Graph-Based Information Retrieval

The goal of information retrieval is to effectively retrieve documents relevant to users' queries. A variety of models and techniques have been proposed over the past 50 years. In this research, we introduce a novel, graph-based approach to information retrieval. Its computation is fast and scalable, and its structure is flexible to incorporate many performance enhancement techniques. The performance evaluation according to the HARD track of TREC 2003 is reported.

Keywords: information retrieval, graph, random walk with restart

TABLE OF CONTENTS

List of Tables

List of Figures

CHAPTER 1

INTRODUCTION

Information retrieval (IR) concerns with representations, storage, organization, and access to documents. The goal of an information retrieval system (generally called a search engine) is to retrieve documents relevant to users' information needs, typically presented to the system in form of queries. It has now been realized that IR plays a very important role in the desktop and World Wide Web applications.

A general view of the retrieval process is shown in Figure 1.1. Before a search system is in operation, words are recognized and extracted from documents in the collection, passed through the stopword elimination, stemming, and other processes according to the features of the system, such as phrase construction, term canonicalization, term expansion, etc. An index is then built from those terms with an indexing process.

Users employ a user interface provided by the system to specify their information needs in forms of queries. Queries are processed according to query operations, e.g., stopword elimination, stemming, and further transformed into the representation used by the system, normally in a similar format to that of the index. Then the searching process employs the index to identify a set of documents that potentially are relevant to the query. During the search, the system may give a matching score for each document. Once a set of documents has been identified and scored, they are ranked upon their scores and presented to the user through the interface. After, the result set is returned to the user, some systems provide methods for the user to refine his query by examining the results, marking those documents in the result set that are considered relevant, and resubmitting it into the system through the relevance feedback process. The system then calculates another result set and returns it to the user.

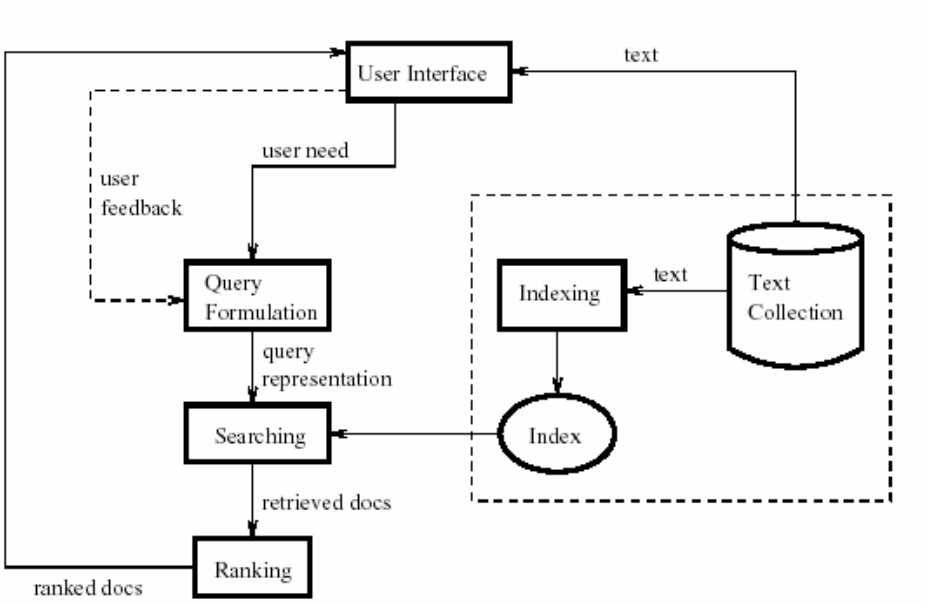

Figure 1.1 Information retrieval process

A major task of an information retrieval system is to predict which documents are relevant, and which are not, according to a user's information need. This decision usually depends upon similarity calculation intrinsic to the information retrieval model which attempts to establish an ordering of the documents retrieved. A number of models have been proposed over 50 years, for example:

Vector model: Both the query and each document are represented as vectors in the term space. A measure of the similarity between the two vectors is computed.

Probabilistic model: A probability based on the likelihood that a term will appear in a relevant document is computed for each term in the collection. For terms that match between a query and a document, the similarity measure is computed as the combination of the probabilities of each of the matching terms.

Language model: A language model is built for each document, and the likelihood that the document will generate the query is computed.

Inference Networks: A Bayesian network is used to infer the relevance of a document to a query. This is based on the evidence in a document that allows an inference to be made about the relevance of the document. The strength of this inference is used as the similarity coefficient.

Boolean Indexing: A score is assigned such that an initial Boolean query results in a ranking. This is done by associating a weight with each query term so that this weight is used to compute the similarity coefficient.

Latent Semantic Indexing: The occurrence of terms in documents is represented with a term-document matrix. The matrix is reduced via Singular Value Decomposition (SVD) to filter out the noise found in a document so that two documents which have the same semantics are located close to one another in a multi-dimensional space.

Neural Networks: A sequence of neurons in a network fire when activated by a query triggering links to documents. The strength of each link in the network is transmitted to the document and collected to form a similarity coefficient between the query and the document. Networks are trained by adjusting the weights on links in response to predetermined relevant and irrelevant documents.

Genetic Algorithms: An optimal query to find relevant documents can be generated by evolution. An initial query is used with either random or estimated term weights. New queries are generated by modifying these weights. A new query survives by being close to known relevant documents and queries with less fitness are removed from subsequent generations.

Fuzzy Set Retrieval: A document is mapped to a fuzzy set (a set that contains not only the elements but a number associated with each element that indicates the strength of membership). Boolean queries are mapped into fuzzy et intersection, union, and complement operations that result in a strength of membership associated with each document that is relevant to the query. This strength is used as a similarity coefficient.

Random walk algorithms on graphs have been used in citation analysis, social networks, and the analysis of the link structure of the Web. Generally, a graph-based ranking algorithm is a way of deciding on the importance of a vertex within a graph, by taking into account global information recursively computed from the entire graph, rather than relying only on local vertex-specific information. The basic idea implemented by a random walk algorithm is that, when one vertex links to another one, it is basically casting a vote for other vertex. The higher the number of votes that are casted for a vertex, the higher the importance of the vertex.

This research proposes a graph-based information retrieval system (GIR), employing a graph structure that captures occurrences of terms in documents as well as correlations among terms, and the similarity between a document and the query are calculated by a specific type of graph random walk algorithm, called Random Walk with Restart (RWR). The model can be extended to incorporate relevance feedback naturally. Its effectiveness is evaluated using TREC collections; its performance is measured and compared to the performance figures of systems participating in the TREC program.

The rest of this report is organized as follows. Chapter 2 reviews related work on retrieval models and graph random walk algorithms in information retrieval. Chapter 3 describes the GIR model. In Chapter 4, experimental evaluations and results are discussed. Finally, Chapter 5 provides concluding remarks for the research.

CHAPTER 2

RELATED WORK

2.1 Information Retrieval

A major task of an information retrieval system is to predict which documents are relevant, and which are not, to a user's query. This decision usually depends upon ranking strategies intrinsic to the engine which attempts to establish an ordering of the documents retrieved. Such a system can be characterized by three components: the document representation, the query representation, and the ranking algorithm.

As a background knowledge, we discuss three classical models: the Boolean, vector-space, and probabilistic models. In these models, a document is represented by a set of index terms, normally terms occurring in the document. Note, however, that these models actually are rather general. Concepts, citations, or other features can be used as index terms.

The Boolean model is a simple retrieval model based on Set theory and Boolean algebra. It was adopted by many early commercial bibliographic systems. In this model, a document d_i is represented by a set of binary digits $t_j \in \{0,1\}$, reflecting that the term t_j is absent or present in the document, that is $d_i = \{t_1, t_2, \ldots, t_n\}$. A query is represented as a Boolean expression of index terms. A document is considered relevant to the query if occurrences of terms in the document satisfy the query. Advantages of this model are its simplicity and theoretical foundation. A major disadvantage, in some sense lying in its simplicity, is that exact matching may overlook some documents, and its strict interpretation of Boolean operators.

The vector-space model [23] offers an ability to sort the retrieved documents, based upon their degrees of similarity to the query. The results produced usually are broader than those of the Boolean model. A document is represented as a vector $\vec{d}_j$ of term weights. The dimension of the vector space is the number of terms in the vocabulary or thesaurus. A query $\vec{q}$ is represented in the similar fashion as that of the document. A popular term weighting scheme is of the form *tf x idf* [24] where *tf* is a function of the

number of term occurrences in the document and *idf* is a function (inverse) of the number of documents containing this particular term. A popular one is $w_{ij} = (1+\log_e f_{ij}) \cdot \log_e\left(1+\frac{N}{df_i}\right)$ where f_{ij} is the frequency of occurrence of term *i* in document *j*, *N* is the number of documents in the collection, and df_i is the number of documents containing term *i*. The query term weight w_{iq} can be calculated similarly or differently, depending on the weighting scheme selected. Central to the model is the degree of similarity of a document, which is computed from the query and document term weights. One approach involves the cosine similarity measure which, as the name suggests, measures the cosine of the angle between the document and the query vector:

$$sim(d_j, q) = \frac{\vec{d}_j \cdot \vec{q}}{|\vec{d}_j| \times |\vec{q}|} = \frac{\sum_{i=1}^{t} w_{ij} \times w_{iq}}{\sqrt{\sum_{i=1}^{t} w_{ij}^2} \times \sqrt{\sum_{i=1}^{t} w_{iq}^2}}$$

where wij and wiq are document term weights and query term weights, respectively. Documents are then ranked decreasingly according to the similarity scores and presented to the user in that order.

The probabilistic model [22] attempts to capture the retrieval problem within a probabilistic framework. The algorithm typically ranks the documents according to the ratio of the probability that document *d* is relevant to the query *q* over the probability that the document is not relevant to the query. That is, *similarity(d, q) = P(relevant | d) / P(nonrelevant | d)*. After applying Bayes' rule, these probabilities can be estimated from the query and the collection statistics; and can be improved recursively at each step of the feedback process.

The three retrieval models discussed so far represent documents and queries by a set of index terms, and similarity between documents and queries is determined based upon the existence of query terms in the documents. This may lead to poor retrieval effectiveness because of two properties of natural language inherent in this type of representation. One property is that there are multiple meanings of a term in different contexts (known as polysemy), and the other is that multiple terms can be used to represent a particular meaning (known as synonymy). These effects cause nonrelevant

documents to be included and relevant documents to be excluded from the results, respectively.

Latent semantic indexing (LSI) [6] is regarded as a method to get around these problems. This approach represents documents and queries by a set of concepts, numerically derived from a set of index terms. The idea is to map the document-index term vector space to a lower dimensional space using Singular Value Decomposition. Every new dimension in this space is termed a concept. Queries are transformed into the same space, and similarity between documents and queries, such as the cosine similarity, is measured in this reduced space

A variety of alternative IR models have been proposed, such as the fuzzy set model [16], the extended Boolean model [25], the neural network model [31], the inference network model [30], the belief network model [21], and many more. For further information, those interested can refer to [2].

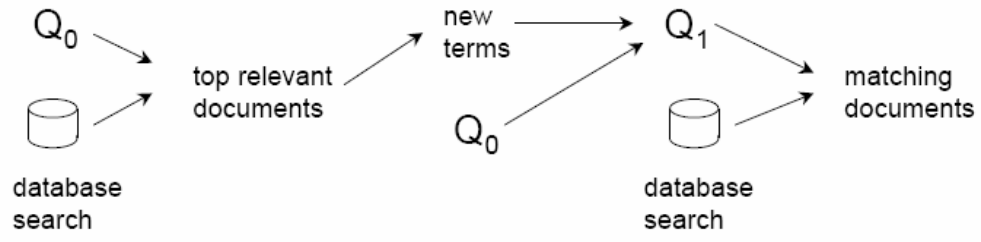

Figure 2.1 Relevance feedback process

Instead of relying solely on the retrieval models, a number of performance enhancements are proposed over the years. For relevance feedback, an initial search is carried out as described above. A user can identify documents they consider relevant to his query from the results set. The system will then take these judgments and recalculate scores for all documents. The process (shown in Figure 2.1) continue until the user finds needed information. The basic assumption is that a query term that occurs frequently in documents that are judged to be relevant to a particular query, and infrequently in documents that are judged to be nonrelevant is a good term for that query. In the vector model, such good query terms should thus be allocated greater weights than the other query terms.

Another technique that may help improve the performance of an IR system is thesaurus. A thesaurus is set of items (phrases or words) plus a set of relations between those items. Thesaurus can be constructed manually or automatically. An automatic thesaurus is usually collection-dependent and typically built from co-occurrence information, and relevance judgments are often used to estimate the probability that thesaurus terms are similar to query terms or the entire query. Thesaurus terms with sufficient similarity are then used to expand the query.

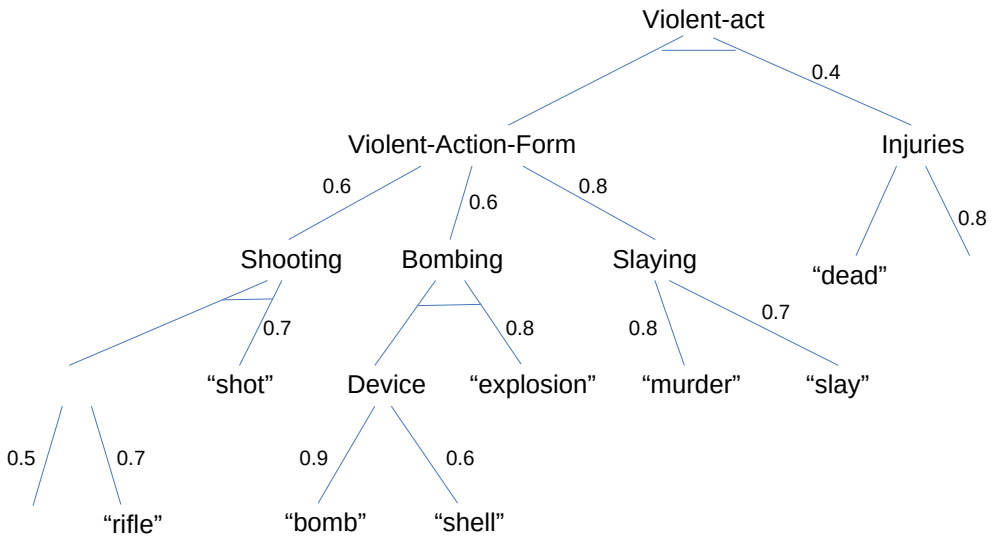

Figure 2.2 A rule tree in RUBRIC

2.2 Graph-Based Information Retrieval

There are some information retrieval systems utilizing graphs in some fashions. Rule Based Information Retrieval by Computer (RUBRIC) [28] uses production rules to capture user query concepts. A query is specified in a hierarchical fashion which can be viewed as a set of production rules. Each rule has an associated weight, assigned by the user. An example rule is shown in Figure 2.2 for the topic of "Violent-Act." RUBRIC determines relevant documents through a goal-driven search and creates a rule tree with the user's query topic at the root, subtopics are represented as branches, and the keywords to be matched with contents of documents are at the leaves. Score of each node is calculated by a fuzzy evaluation of the AND/OR operations.

Kazai, *et.al.* [11] present a retrieval model representing documents as graphs. A document is represented as a tree whose nodes are components of the documents and whose edges represent the relationships between those components. The model uses fuzzy aggregation, an approach based on the fuzzy representation of linguistic quantifiers, and ordered weighted averaging operators to assign scores to documents.

Ogilvie and Callan [17] introduce a document retrieval system using a tree-based generative language model. Nodes of the tree correspond to document components, such as titles, sections, and paragraphs. Each node in the tree is associated with a language model. The model for a leaf node is estimated directly from the text presented in the document components associated with the node. An internal node is estimated by a linear interpolation of the child nodes. This approach allows for structured queries and ranking of document components. One benefit of the model includes guidance from language modeling on how to estimate the probabilities used in ranking.

Quintana, *et.al.* [20] propose a Hypertext Retrieval System (HRS) that allows users to retrieve information from hypertext documents. This model is based on semantic contents of documents. Queries and information in hypertext documents are automatically converted to conceptual graphs. The information retrieval problem is turned into a graph matching problem. This technique is useful for the retrieval of information in large knowledge domains where a user needs to find specific information but does not know the organization of the hypertext documents or words used in the documents.

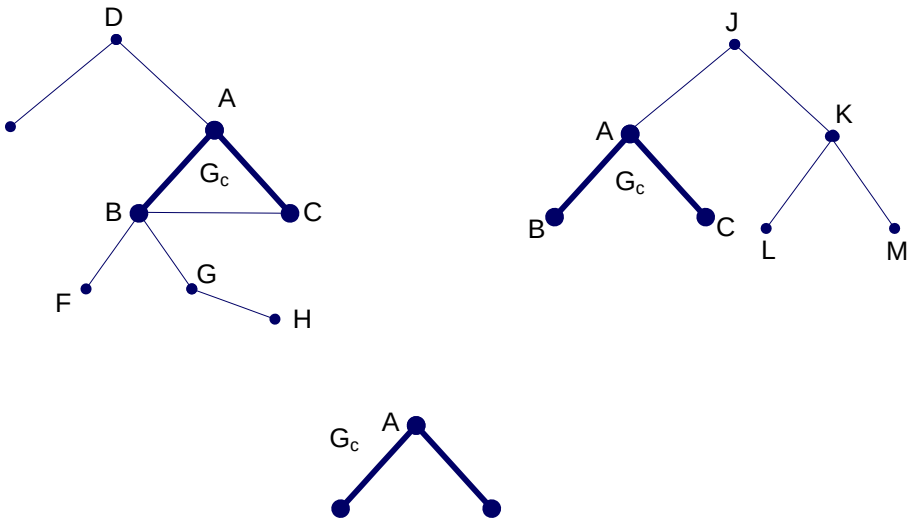

Figure 2.3 Intersection of two conceptual graphs

Montes-y-Gomez, *et.al.* [14] adapt the idea of conceptual graphs to represent the contents of documents and query. The model compares two conceptual graphs (document graph and query graph) by matching and measuring their similarity. The algorithm consists of two parts which are finding the intersection of the two graphs and measuring their similarity which corresponds to the size of their intersection graph, as shown in Figure 2.3. This measure is suitable for text comparison since it considers not only topical aspects of the phrases but also the relationships among elements in the texts.

2.3 Information Retrieval Based on Graph Random Walk Algorithms

In terms of graph ranking algorithms in information retrieval, their main uses are in ranking documents from link information in hypertext environments. Hypertext analysis is studied mostly to support Web search engines, in particular the ranking of the retrieved Web pages. A hyperlink from page A to page B is a recommendation of page B by the author of page A or the same topic being presented in both Web pages A and B. The hyperlink analysis supports the relevance evaluation of pages to a user query.

HITS (Hypertext Induced Topic Search) [12] is a well known algorithm introduced by Kleinburg. The algorithm computes two scores for each web page recursively which are: hub and authority scores. Pages with high authority scores are considered to have content relevant to the query, whereas pages with high hub scores are

considered to contain links to relevant content. The intuition is that a page which points to many other pages is a good hub, and a page that many pages point to is a good authority. Good authorities are those pointed to by good hubs, and good hubs are pointed to by good authorities. This mutually reinforce relationship is at the core of the algorithm.

PageRank [4] is a prominent technique for estimating importance scores for Web pages. The PageRank value is computed by weighting each hyperlink to the Web page proportionally to the quality of the Web page containing the hyperlink. The PageRank values are calculated recursively with arbitrary initial values. PageRank algorithm is used at the heart of the Google search engine to compute page importance for every Web pages based on linkages among pages. Google uses this technique to employ the global hyperlink structure of the Web and produce better rankings of search results compared to using the content of the page alone. Figure 2.4 demonstrates the propagation of ranks from pages A and B to others.

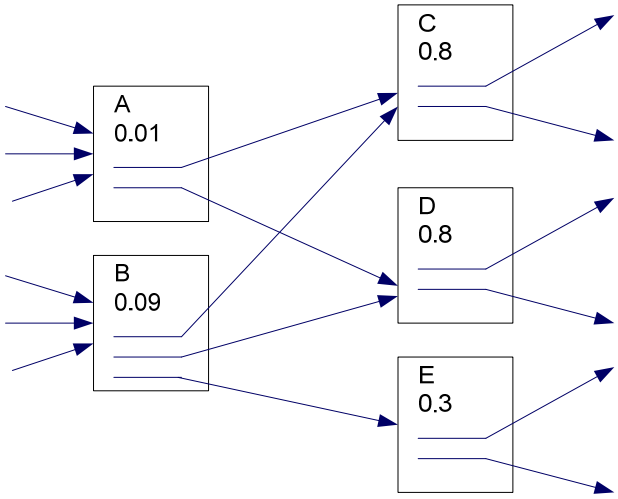

Figure 2.4 PageRank calculation

Yang and Chen [32] develop a modified multimedia PageRank algorithm for retrieving multimedia Web objects including Web pages, images and videos. The method analyzes hyperlinks and embedded links among multimedia objects together with weights for different types of links using the PageRank algorithm.

Mihalcea and Tarau [13] introduce the TextRank graph-based ranking model for keyword and sentence extraction from natural language texts. The algorithm takes into account edge weights while computing the score associated with a vertex in the graph.

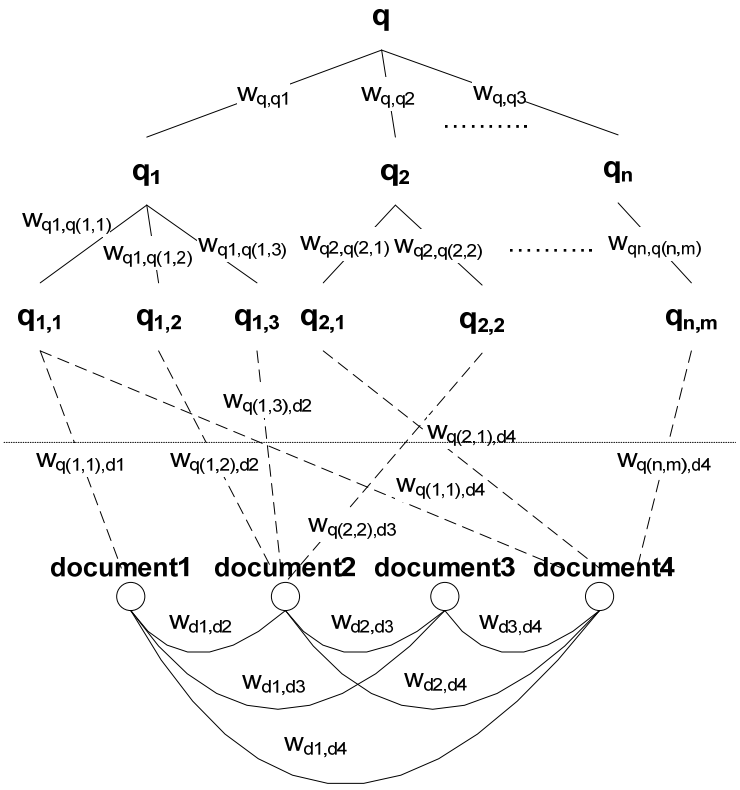

Figure 2.5 Graph structure in [27]

Graph-based ranking algorithms are used mostly in calculating term weights to represent the contribution of a term in its context. Blanco and Lioma [3] introduce the use of a random walk algorithm to weight terms in the tf-idf weighting scheme by adapting the TextRank algorithm. In this model, edge weights are not taken into account. Hassan, *et al.* [7] propose a random walk model considering term dependencies and their relations to the surrounding context for term weighing. Islam, *et al.* [9] exploit the relationship of local information of a vertex (term position) and global information (information gain) and term dependency to produce term weights.

For similarity ranking, Thammasut and Sornil [27] propose a graph-based IR system whose query can be expressed as a graph of topics/subtopics. Once the query graph is merged with the document relationship graph, a random walk algorithm is

applied to determine the score of each document node, as shown in Figure 2.5. The emphasis of the work is on expressing queries as graphs, and it does not incorporate term relationships in the structure.

CHAPTER 3

GIR: A GRAPH-BASED INFORMATION RETRIEVAL MODEL

This section introduces the proposed graph-based information retrieval model (GIR). The idea is to turn the information retrieval into a graph problem. The model creates a collection graph which represents content of documents and relationships among terms, built prior to the query time. Once a user supplies a query into the system, a query node is created and connected to query terms in the graph. A graph random walk algorithm is then applied to calculate the relevance of each document to the query.

3.1 Document Processing

After words are extracted from a document, they are preprocessed in three important steps: stopword elimination, stemming, and indexing.

3.1.1 Stopword Elimination

It is often found helpful to remove commonly used words (called stopwords) such as “it” and “can” from the index due to their very low discrimination power, thus not very helpful in the retrieval. The process of removing stopwords from text is called stopword elimination. This research uses the stopword list from [5], some of them are shown in Figure 3.1.

a	been	get	least	our	them	whether
about	before	getting	left	ourselves	then	which
after	being	go	less	out	there	while
again	between	goes	let	over	these	who
ago	but	going	like	per	they	whoever
all	by	gone	make	put	this	whom
almost	came	got	many	putting	those	whose
also	can	gotten	may	same	through	why
always	cannot	had	maybe	saw	till	will
am	come	has	me	see	to	with
an	could	have	mine	seen	too	within
and	did	having	more	shall	two	without
another	do	he	most	she	unless	won't
any	does	her	much	should	until	would
anybody	doing	here	my	so	up	wouldn't
anyhow	done	him	myself	some	upon	yet
anyone	down	his	never	somebody	us	you
anything	each	how	no	someone	very	your
anyway	else	i	none	something	was	
are	even	if	not	stand	we	
as	ever	in	now	such	went	
at	every	into	of	sure	were	
away	everyone	is	off	take	what	
back	everything	isn't	on	than	whatever	
be	for	it	one	that	what's	
became	from	just	onto	the	when	
because	front	last	or	their	where	

Figure 3.1 Stopword list used in this research

3.1.2 Stemming

The stemming process normalizes words by conflating a number of morphologically similar words to a single root form (or stem). For example, “connect,” “connected” and “connection” are all reduced to a single stem “connect.” Removing suffixes by automatic means is an operation which is especially useful in the field of information retrieval. In a typical IR environment, one has a collection of documents, each described by the words in the document title and possibly by words in the document

abstract. Ignoring the issue of precisely where the words originate, we can say that a document is represented by a vector of words, or terms. Terms with a common stem will usually have similar meanings, for example:

CONNECT
CONNECTED
CONNECTING
CONNECTION
CONNECTIONS

Frequently, the performance of an IR system will be improved if term groups such as this are conflated into a single term. This may be done by removal of the various suffixes -ED, -ING, -ION, IONS to leave the single term CONNECT. In addition, the suffix stripping process will reduce the total number of terms in the IR system, and hence reduce the size and complexity of the data in the system, which is always advantageous.

The Porter Stemmer [19] is a conflation Stemmer developed by Martin Porter in 1980. The Stemmer is based on the idea that the suffixes in the English language (approximately 1,200) are mostly made up of a combination of smaller and simpler suffixes. This Stemmer is a linear step Stemmer. Specifically it has five steps applying rules within each step. Within each step, if a suffix rule matched to a word, then the conditions attached to that rule are tested on what would be the resulting stem, if that suffix was removed, in the way defined by the rule. For example such a condition may be, the number of vowel characters, which are followed be a consonant character in the stem, must be greater than one for the rule to be applied.

Once a Rule passes its conditions and is accepted, the rule fires, the suffix is removed, and control moves to the next step. If the rule is not accepted then the next rule in the sequence is tested, until either a rule from that step fires, and control passes to the next step, or there are no more rules in that step whence control moves to the next step. This process continues for all five steps, the resultant stem being returned by the Stemmer after control has been passed from step five.

3.1.3 Indexing

Indexing is the process of creating an inverted index. The index used in this research is inverted index, as shown in Figure 3.2. An inverted index is a word-oriented mechanism for indexing a text collection in order to speed up the search. The inverted index structure is composed of two elements: the vocabulary and the occurrences. The vocabulary is the set of all different terms in the collection. For each such term, a list of positions where the term appears is stored.

Document 1 = Information retrieval is searching and indexing
Document 2 = Indexing is building an index
Document 3 = An inverted file is an index
Document 4 = Building an inverted file is indexing

Vocabulary	Inverted List (document; position)
an	(2;4), (3;1), (3;5), (4;2)
and	(1;5)
building	(2;3), (4;1)
file	(3;3), (4;4)
index	(2;5), (3;6)
indexing	(1;6), (2;1), (4;6)
information	(1;1)
inverted	(3;2), (4;3)
is	(1;3), (2;2), (3;4), (4;5)
retrieval	(1;2)
searching	(1;4)

Figure 3.2 An inverted index

Creating an index is a nontrivial task due to the amount of computations and memory required. Well-thought-out algorithms with careful disk-memory interactions are needed.

We apply the multi-way merging algorithm [2] to create the index which can be described as follows:

Maintain a stack of segment indices
Create index for each incoming document
Push new indexes onto the stack
Let $b = 10$ be the merge factor; $M=\infty$
For ($size = 1$; $size < M$; $size$ *= b) {
  If (there are b indexes with size documents on top of the stack) {
    Pop them off the stack;
    Merge them into a single index;
    Push the merged index onto the stack;
  } else {
    Break;
  }
}

An example of the process is shown in Figure 3.3.

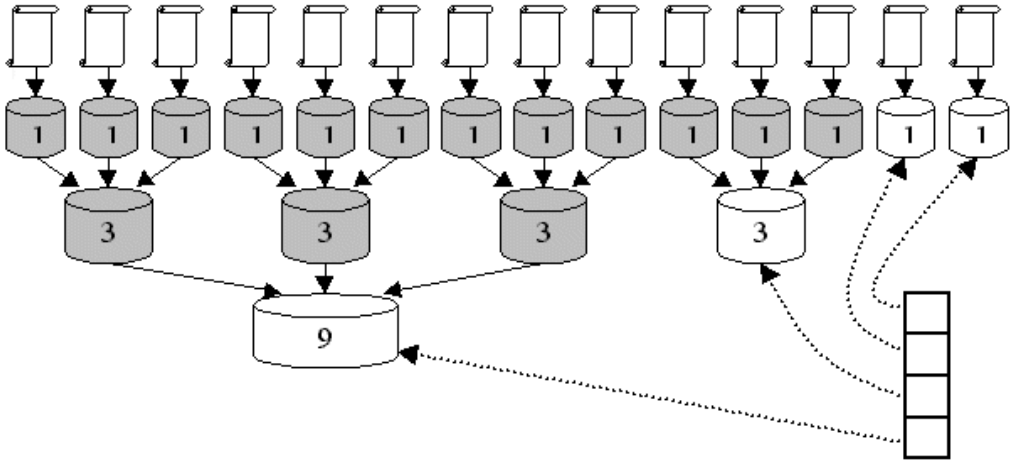

Figure 3.3. An inversion example (b = 3, 11 documents are indexed, the stack has four indexes, grayed indexes have been deleted, and 5 merges have occurred).

3.2 Collection Graph Construction

Collection graph $G = (V,E)$, shown in Figure 3.4, is a weighted, undirected graph whose vertices $V(x) \in V$ are organized in two layers: document and term layers. A document as well as a term is represented as a vertex or node in the graph. We denote $V(d_j)$ and $V(t_i)$ as the vertex of a document d_j and that of a unique term t_i in the vocabulary, respectively. $V(t_i)$ is connected to $V(d_j)$ when term t_i appears in document d_j. A term relationship connection $E(t_i, t_k) \in V \times V$ is placed between $V(t_i)$ and $V(t_k)$ if the bonding strength between the two terms is greater than a certain threshold. $V(t_i)$ has relationship connections placed to top k other term vertices that it has strong relationships with.

The weight w_{t_i,d_j} between $V(t_i)$ and $V(d_j)$ can be computed using one of the family of weighting algorithms known as *tf.idf* — these algorithms are based on the term frequency ($tf_{i,j}$) which is proportional to the frequency of t_i in d_j and inverse document frequency (idf_i) which is proportional to the inverse of the number of documents containing t_i (df_i). Empirical studies [24] indicate that *tf.idf* approaches, while simple, are very effective. The document term weight w_{t_i,t_j} and query term weight $w_{t_i,q}$ are defined as follows:

$$w_{t_i,t_j} = \frac{tf_{i,j} \cdot \log \frac{N}{df_i}}{\sum_{i=1}^{|V|} (tf_{i,j} \cdot \log \frac{N}{df_i})^2}, \text{ and}$$

$$w_{t_i,q} = \left(0.5 + \frac{0.5 \cdot tf_{i,q}}{\max_i tf_{i,q}} \right) \cdot \log \frac{N}{df_i}$$

where N is the total number of documents in the collection.

The graph structure includes the relationships/correlations among terms. To calculate these values, we adapt the association approach to automatic thesaurus construction [10]. Instead of extracting associations in the document level which may include lots of noises, in this research the calculations are carried out in the paragraph level. That is, paragraphs are regarded as the text units for association. The connection

weight c_{t_i,t_k} between $V(t_i)$ and $V(t_k)$ represents the strength of the relationship between term t_i and term t_k, calculated as:

$$c_{t_i,t_k} = \log_2 \frac{df_{i,k} / \sum_{i,k} df_{i,k}}{\{df_i / \sum_i df_i\} \cdot \{df_k / \sum_k df_k\}}$$

where df_i is the occurrence frequency of term t_i, and $df_{i,k}$ is the co-occurrence frequency of terms t_i and t_k.

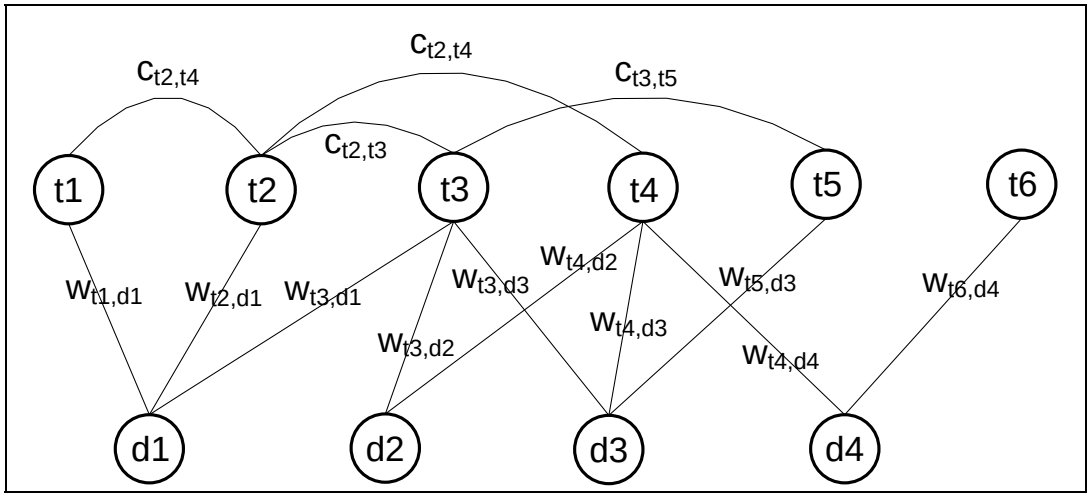

Figure 3.4 A collection graph

3.3 Similarity Calculation

After the collection graph is constructed, when a user supplies a query into the system, a query node $V(q)$ is added to the graph, connecting to nodes corresponding to terms in the query, as shown in Figure 3.5.

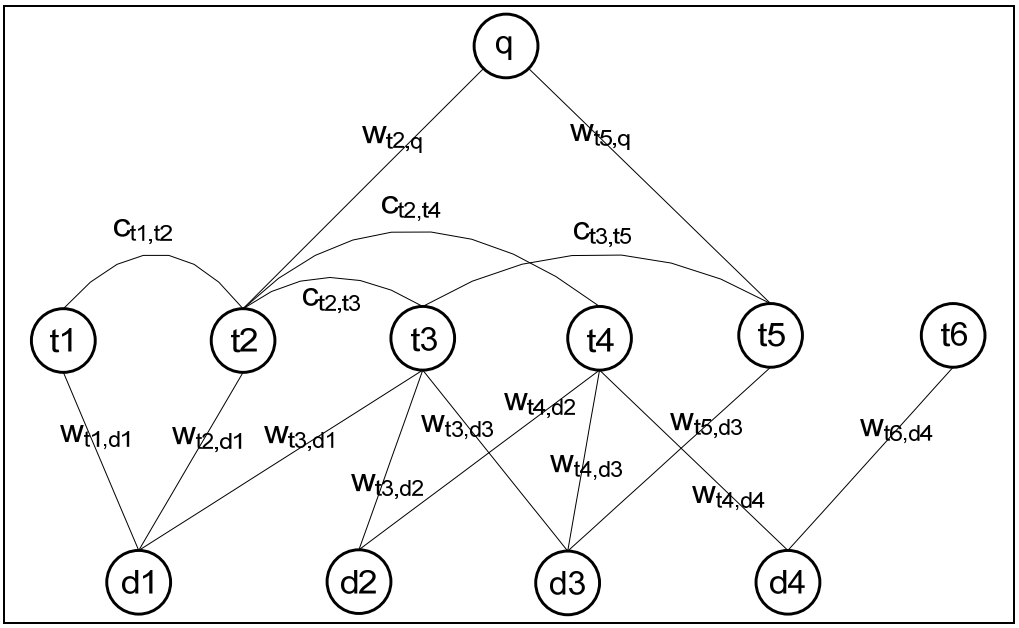

Figure 3.5 Structure of the graph after a query is supplied by a user

Then, the random walk with restarts (RWR) algorithm [18] is applied to calculate a similarity score between the query and each document. Documents are then ranked according to the scores, and the results are returned to the user in a descending order.

The RWR algorithm is selected since it is simple and able to bias toward the query node. The percentage of time the RWR walker spends on a document node is proportional to the closeness of the document node to the restart node. For information retrieval, we want to rank documents with respect to the query, thus the algorithm fits the problem naturally.

The algorithm operates as follows: to compute the similarity of vertex $V(d_j)$ to vertex $V(q)$, consider a random walker that starts from node $V(q)$. At each step, the walker chooses randomly among available edges, with one modification: before he makes a choice, he goes back to node q with probability c. Let $p(V(d_j))$ denotes the steady state probability that the random walker will find himself at node $V(dj)$, then $p(V(d_j))$ is the affinity (similarity/closeness) of $V(d_j)$ with respect to V(q). The system then ranks the scores in a descending fashion and returns the top M documents to the user.

The similarity scoring algorithm can be described as follows:

Initial State: The algorithm is initialized by

$U_i(0) = 0,\ 0 \le i \le n-1$ where $U_i(t)$ is the score of node i at iteration t.

$V_i(0) = 0,\ 0 \le i \le n-1$ where $V_i(t)$ is the restart of node i at iteration t.

$V_q(0) = 1$, where $V_q(0)$ is the restart of node q which is the query topic.

$U_i(0) = V_i(0), 0 \le i \le n-1$ where $U_i(t)$ is the score of node i at iteration t.

A is the adjacency matrix of the graph and is column-normalized.

c is the restart probability of restarting the random walk from the node which corresponding to the query topic.

Activation and Update State: Output of each node is calculated as follows :

$U_i(t+1) = (1-c)\, AU_i(t) + cV_i(t)$

Stable State: Repeat the iteration until convergence.

The stable state is achieved when sum of error at every node in the graph falls below a given threshold (e.g., 0.001).

Output State: After the graph converges, the resulting score for each node becomes the final similarity score of the corresponding document.

The RWR has an ability to bias toward the restart node. By setting the restart node as the query, RWR is able to rank documents according to the query topic node. The percentage of time the RWR walks spends on a document node is proportional to the closeness of the document node to the query node. This means ranking the document node with the respective similarity to the query. On the other hand, the popular PageRank method [4] may not be appropriate for our task, since the ranking it produces does not bias toward any particular node.

The RWR algorithm together with the graph structure represent a form of built-in thesaurus. It is unlike a conventional thesaurus however, in that terms are grouped on the basis of common documents rather than common meanings. This means that in a set of

documents concerning graphs, “node” and “edge” are more likely to be linked than “node” and “vertex”. The hope is that two terms that are common to a number of documents discuss the same subject.

3.4 Efficiency of the Algorithm

Let $\vec{u}_i$ and $\vec{v}_i$ be Nx1 probability vectors, we can see that according to RWR, the following equation holds:

$$\vec{u}_i(t) = (1-c)A\vec{u}_i(t-1) + c\vec{v}_i(t).$$

We can show that

$$\vec{u}_i(t) = c(I-(1-c)A)^{-1}\vec{v}_i(t)$$

where I is the N x N identify matrix.

It can be shown easily that the overall cost of retrieving documents in response to a query is linear on the number of edges, or

$T(n) = maxIteration \times O(E)$

$= O(E)$

From experiments, the number of iterations to converge (*maxIteration*) is relatively moderate (e.g., approximately 20), or it can be set to have an upper bound (e.g., 100) which is in the order of *O(1)*, thus the cost of retrieval is *O(E)*. Since the matrix is sparse, i.e., a small subset of terms appears in a document as well as the limited *k* number of connections from a term vertex. The algorithm is efficient. In addition, there are literatures on fast large-matrix inversion or fast approximations to solve the equation which makes this approach scalable.

3.5 Relevance Feedback

Relevance feedback can easily be incorporated into this architecture. In the manual relevance feedback, the user marks documents he considers relevant to his query from the initial/previous result set. In automatic relevance feedback, top *R* from the total of *M* documents returned in the previous run are assumed to be relevant and fed back into

the system to recalculate the ranking scores. In this research, *Ide_Dec-Hi* [8] approach to vector-based relevance feedback is modified, as follows:

$$w_{t_i,q}(new) = \max\left\{0, \min\left\{1, \left(\alpha \cdot w_{t_i,q}(old) + \beta \sum_{\forall d_j \in D_r} w_{t_i,d_j} - \gamma \cdot \max_{nonrelevant}(w_{t_i,d_j}) \right)\right\}\right\}$$

where $w_{t_i,q}(new)$ is the new query term weight recalculated from the relevance feedback, $w_{t_i,q}(old)$ is the original/previous query term weight, D_r is the set of documents marked as relevance by users or is the top k documents for the automatic method, and α, β, γ are constant. Figure 3.6 shows an example of the automatic relevance feedback when R = 2, i.e., d_2 and d_3 are the top two documents. *Ide_Dec_Hi* is chosen since it is found to give good performance while the calculation is simple.

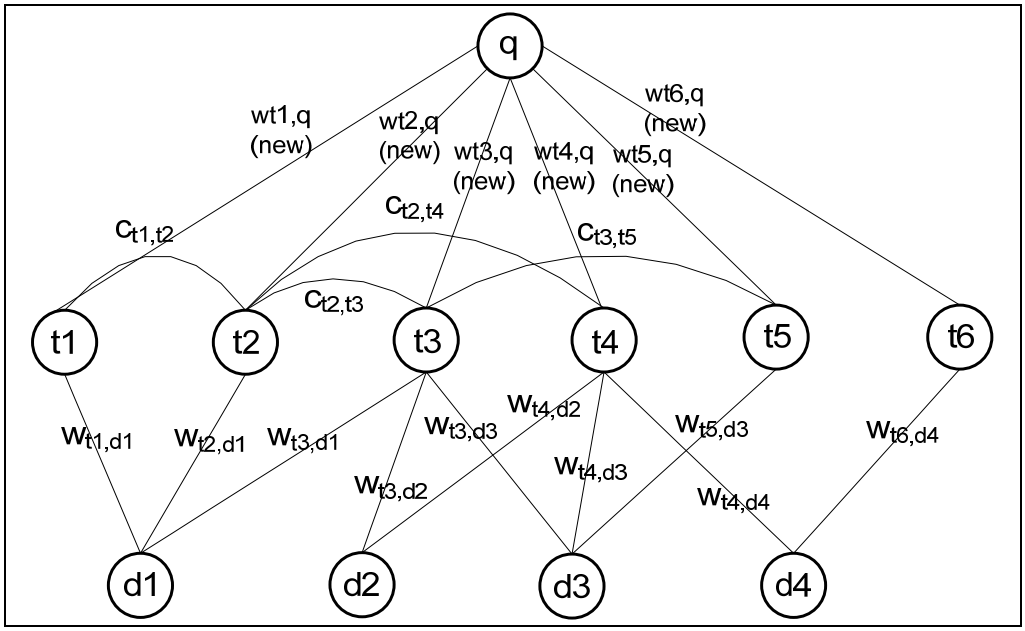

Figure 3.6 Graph structure when an automatic relevance feedback is incorporated

CHAPTER 4

EXPERIMENTAL EVALUATION

This chapter describes experiments carried out to measure the effectiveness of GIR. Four different configurations of the proposed technique are evaluated according to a standard evaluation approach on a standard information retrieval corpus, and the results are measured and analyzed.

4.1 Test Collections

Since 1993, The Text REtrieval Conference (TREC) [14] has defined a set of benchmark corpora to assess IR system performance. Each corpus consists of documents, queries, and their corresponding relevance judgments. The corpus used for our research is the High Accuracy Retrieval from Documents (HARD) track of 2003 since it fits the objective of the research. The main objective of the track is to explore methods for improving the accuracy of document retrieval systems. The collections used and their characteristics are shown in Table 4.1.

Table 4.1 Test collections and their characteristics

Collections	Number of Documents	Size (Mbs)
Agence France Press	226,777	497
Associated Press	236,735	644
Central News Agency	4,011	6
LA Times/Wash Post	34,145	107
New York Times	27,835	105
Salon.com	3,134	28
Ummah Press	2,557	5
Xinhua (English)	117,516	183
Total	652,710	1,575

The HARD track provides 20 training topics with 100 judged documents for each topic, and uses a set of 48 topics for evaluation. Two judgment results are distributed: the hard evaluation and the soft evaluation. In the hard evaluation, a document is relevant if not only the document is relevant to the topic but also the matches the metadata of the topic. To aid in this effort, the HARD track provides extra information for systems to utilize in their functions or for human to better form queries. In the soft evaluation, a document is considered relevant when it is relevant to the topic, regardless of the topic's metadata.

4.2 Performance Metrics

Performance metrics employed in this study follow what are used in the HARD track evaluation which consist of: mean average precision (MAP), precision at 10 relevant documents, R-precision, precision-recall graph, and non-interpolated average precision over all relevant documents for each query. These measures are based on two fundamental measures which are recall and precision.

Recall is a measure of the ability of a system to present all relevant documents, defined as:

$$Recall = \frac{number\ of\ relevant\ documents\ retrieved}{number\ of\ relevant\ documents\ in\ collection}.$$

Precision is a measure of the ability of a system to present only relevant documents, defined as:

$$Precision = \frac{number\ of\ relevant\ documents\ retrieved}{number\ of\ documents\ retrieved}.$$

4.2.1 Mean Average Precision (MAP)

Mean average precision takes average precisions over a number of queries that a system executes and averages them. MAP gives us the arithmetic mean of the average precisions, which is useful in many situations, but this measure does not differentiate between the relative improvements that a system brings to an individual query. MAP has been shown to have especially good discrimination and stability.

For a single query, an average precision is the average of the precision value obtained for the set of top *k* (say 100) documents existing after each relevant document is retrieved, and this value is then averaged over information needs. That is, if the set of relevant documents for an information need $q_i \in Q$ is $\{d_1, d_2, ..., d_{m_j}\}$ and R_{jk} is the set of ranked retrieval results from the top result until you get to document d_k, then

$$MAP(Q) = \frac{1}{|Q|}\sum_{j=1}^{|Q|}\frac{1}{m_j}\sum_{k=1}^{m_j}\Pr ecision(R_{jk})$$

When a relevant document is not retrieved at all, the precision value in the above equation is taken to be 0. For a single query, the average precision approximates the area under the uninterpolated precision-recall curve, and so the MAP is roughly the average area under the precision-recall curve for a set of queries.

4.2.2 Non-Interpolated Average Precision over All Relevant Documents

This is a single-valued measure that reflects the performance over all relevant documents for each query. It rewards systems that retrieve relevant documents quickly (highly ranked). It is the average of the precision value obtained after each relevant document is retrieved. As an example, consider a query that has 4 relevant documents which are retrieved at ranks 1, 2, 4, and 7. The actual precision obtained when each relevant document is retrieved is 1, 1, 0.75, and 0.57, respectively. The average precision over all relevant documents for this query is 0.83.

4.2.3 Precision at *k* Document Cutoff Value

The precision computed after a given number *k* of documents have been retrieved reflects the actual measured system performance as a user might see it, i.e., how many good results there are on the first few pages. Each document precision average is computed by summing the precisions at the specified document cutoff value and dividing by the number of queries.

4.2.4 Average R-Precision

R-Precision is the precision after *R* documents have been retrieved, where *R* is the number of relevant documents for the query. It deemphasizes the exact ranking of the retrieved relevant documents which can be partially useful where there are large numbers of relevant documents. The average R-precision for a system is computed by taking the mean of the R-precision of the individual queries. For example, given 2 queries one with 50 relevant documents and another with 10 relevant documents. If the retrieval system returns 17 relevant documents in the top 50 documents for the first topic, and 7 relevant documents in the top 10 for the second topic, then the system's R-Precision would be $\frac{\frac{17}{50}+\frac{7}{10}}{2}$ or 0.52.

4.2.5 Precision at 11 Standard Recall Levels

This measure is used to compare the performance of different systems. Each recall-precision average is computed by summing the interpolated precisions at the specified recall cutoff values and then dividing by the number of queries.

$\frac{\sum_{i=1}^{NUM} P_\lambda}{NUM}$ where $\lambda = \{0.0,0.1,0.2,...,1.0\}$, and NUM is the total number of queries.

This graph is the most commonly used method for comparing systems. The plots of different systems can be superimposed on the same graph to determine which system is superior. Curve closest to the upper right-hand corner of the graph (where recall and precision are maximized) indicate the best performance. Comparisons are best made in the three different recall ranges: 0 to 0.2, 0.2 to 0.8 and 0.8 to 1. These ranges characterize high precision, middle recall, and high recall performance, respectively.

4.3 System Configurations

Four configurations of GIR are studied, as shown in Table 4.2. The GIR-NoRelNoSig configuration uses binary link weights and does not include term relationship. The GIR-RelNoSig configuration includes term relationships with binary link weights. The GIR-RelSig configuration includes term relationship with real-valued connection weights between 0.0 and 1.0. The GIR-RF configuration is GIR-RelSig with

the automatic relevance feedback process as described in Chapter 3. In general, users will examine and thus mark (or rate) only a small number of documents from the first few pages in the results, thus in this study we assume that the top 10 documents from the initial run are automatically fed back into the model. All configurations of GIR are run and compared with the median and maximum performance values from the results of TREC 2003's HARD track.

Table 4.2 GIR configurations studied in this research

Configurations	**Term Relationships**	**Link Weights**	**Relevance Feedback**
GIR-NoRelNoSig	No	0 or 1	No
GIR-RelNoSig	Yes	0 or 1	No
GIR-RelSig	Yes	Between 0.0 and 1.0	No
GIR-RF (top 10)	Yes	Between 0.0 and 1.0	Yes

4.4 Experiments and Results

After a document is pre-processed by lexical analysis, stopword elimination according to [5], stemming by the Porter's algorithm [18], and indexing, as described in Chapter 3. Each GIR configuration is evaluated on the HARD track's corpus.

First, using the GIR-RelSig configuration we experiment to find out how different values of the parameters *c* and *k* affect the performance of the model by varying values of each parameter one at a time and measuring the value of the mean average precision on the training data set.

Pan, *et.al.* [17] conjecture that what determines a good value for the restart probability is the diameter of the graph. That is, we want our random walker to have a non-trivial chance to reach the outskirts of the whole graph. Thus, if the diameter of the graph is *d*, the probability that he will reach a point on the periphery is proportional to $(1-c)^d$.

For Web graph, the recommended value for *c* is typically 0.15 [1]. The diameter of the web graph is approximately *d = 19* [25] which implies that the probability *p* for the

random walker to reach a node in the periphery is roughly: $p = (1-0.15)^{19} = 0.045$. In our architecture, we have a three layered-graph, the diameter is roughly $d = 3$. If we assume the same value of p for our model, we have:

$$(1-0.15)^{19} = (1-c)^3$$
$$c \approx 0.65$$

We thus initially set the value of c at 0.65, vary the value of k, and measure the mean average precision. The results are shown in Figure 4.1 which suggest that the value of k between 15 and 25 yields the best performance, and after the value of k beyond 25 the effectiveness begins to drop. Next, we set the value of k at 20 and vary the value of c. The results are shown in Figure 4.2 which suggest that value of c after 0.5 does not give significantly different performance. Therefore, in further experiments we will use the following parameters: $c = 0.6$ and $k = 20$.

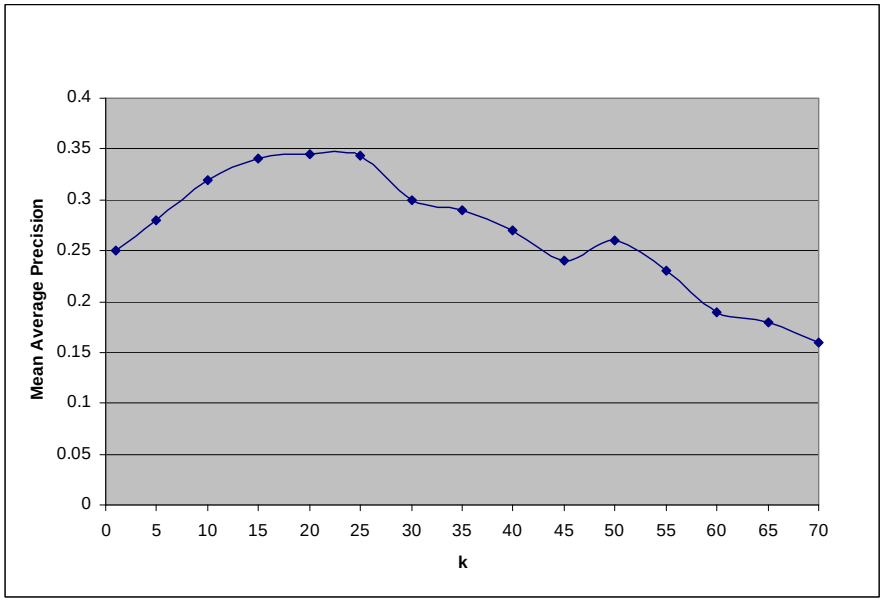

Figure 4.1 The results of varying values of k while fixing the value of c at 0.65

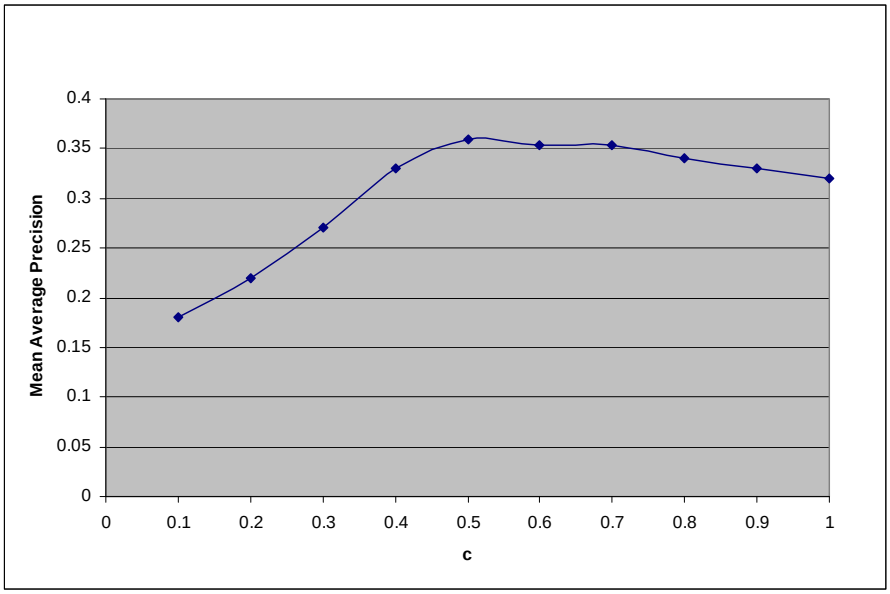

Figure 4.2 The results of varying values of *c* while fixing the value of *k* at 20

Next, all 4 configurations are evaluated according to the HARD track of the TREC 2003 program. Table 4.3 shows the results for all configurations. We can see that term relationships are helpful in the retrieval as the average precision increases from 0.2518 for GIR-NoRelNoSig to 0.3117 for GIR-RelNoSig (+23.79%), and term significance captured by weights is also found effective as we can see an increase from 0.3117 for GIR-RelNoSig to 0.3721 for GIR-RelSig (+19.38%).

Table 4.3 MAP, R-precision, and precision at 10 relevant documents for different configurations

Configuration	**Avg. Prec**	**R-Prec**	**Prec at 10**
GIR-NoRelNoSig	0.2518	0.2569	0.4318
GIR-RelNosig	0.3117	0.2961	0.4946
GIR-Full	0.3721	0.3229	0.5423
GIR-RF	0.3541	0.3112	0.5028
TREC Median	0.2841	0.2994	0.4729
TREC Max	0.4069	0.425	0.65

In terms of ranking ability, both precision at 10 relevant documents and R-precision show similar patterns of relative performance, ordered from the lowest to the highest performance in the following order: GIR-NoRelNoSig, GIR-RelNoSig, and GIR-RelSig.

Compared to the TREC median performance, the GIR-RelSig configuration performs better in all aspects. GIR-RelNoSig outperforms the median in the average precision, but there is no clear winner in terms of ranking ability. Though GIR-RelNoSig has better precision at 10 relevant documents, it has worse R-precision value than the median. Compared to the TREC maximum performance, none of the configurations studied performs better than it. However, the relative performance of GIR configurations and TREC performance must be used with care since in the TREC maximum performance the best results for different queries may come from different systems. That is, the maximum performance line does not come from a single system. In addition, some systems participating in TREC employ extra metadata information or extra human knowledge to assist during the search.

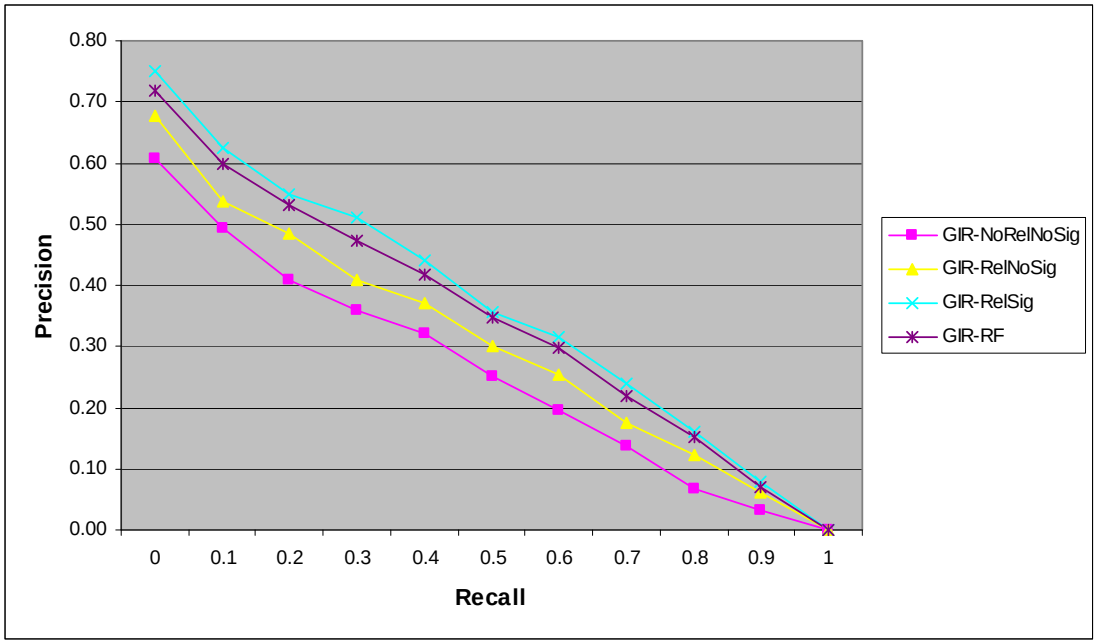

Figure 4.3 Precision-recall graph for different GIR configurations

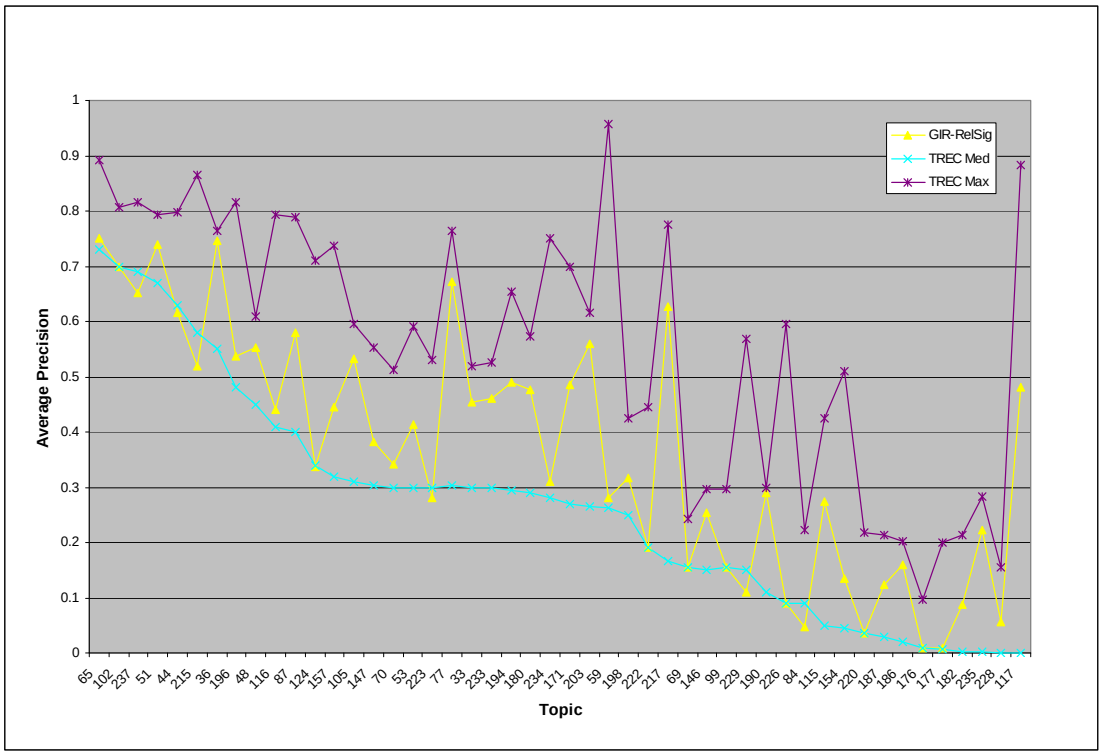

Figure 4.4 Comparisons of GIR-Full, TREC median, and TREC maximum (topic by topic)

Next, we consider the effect of incorporating automatic relevance feedback function in GIR. To our surprise, the results show that automatic relevance feedback where top 10 documents from the previous search is fed back into system to recalculate similarity scores (GIR-RF) does not improve but instead hurts the performance. This may be due to the fact that using automatic relevance feedback makes the model overly focus on documents surrounding those top documents while the test corpus contains mostly news documents which vary in word selections and styles of writing, thus decreases the recall. However, it is still better than the TREC median performance.

Overall, the experiments show that query expansion through term relationship works well, and taking into account differences in term significances through weights benefits the retrieval.

Figure 4.3 shows the precision-recall graphs for the GIR configurations. We can see that in the high precision region, performance of different GIR configurations clearly reflect what appear in Table 4.3. However, in the high recall region, those differences are

less clear, for example, GIR-RelSig, GIR-FB, and GIR-RelNoSig nearly overlap. Finally, Figure 4.4 shows the topic-by-topic comparison of TREC median and maximum to GIR-RelSig.

CHAPTER 5

CONCLUSION

In this research, a novel information retrieval model based on graph random walk (GIR) is proposed. The architecture of the model comprises three layers of nodes. In the first layer, each node represents a single document in the collection. In the second layer, a node represents a term extracted from documents in the collection. Term relationship connections are placed between terms with sufficient relationship strengths, calculated from passage-level excerpts. GIR is evaluated according to TREC 2003's HARD track. The results show that including term relationships and term significance weightings are useful for retrieval purpose. GIR is found to perform far better than the TREC median performance, and overall it performs close to the TREC maximum performance. However, incorporating automatic query expansion to the model is found not much helpful. The proposed architecture for information retrieval can be extended in various ways. It can directly be applied to text summarization by instead of ranking excerpts based solely on document similarity, the term layer also is considered, a random walk algorithm is then used to assign important scores for excerpts, and the top *n* excerpts are extracted as summary. It also can be applied to Web searching by seamlessly adding directed edges between documents to incorporate hyperlinks into the model. The effect of relevance feedback and other performance enhancements should be studied further. In addition, a few techniques for efficient implementation of the random walk with restart algorithm have been proposed [28], and their effectiveness for information retrieval should be investigated.

BIBLIOGRAPHY

[1] A. Albert, H. Jeong, and A.-L. Barabasi. Diameter of the World Wide Web. *Nature*, 401:130–131, 1999.

[2] R. Baeza-Yates and B. Ribeiro-Neto. Modeling (chapter 2). In *R. Baeza-Yates and B. Ribeiro-Neto, editors, Modern Information Retrieval*. AWL England, 1999.

[3] R. Blanco and C. Lioma. Random Walk Term Weighting for Information Retrieval. *SIGIR'07*, Amsterdam, Netherland, July 23-27, 2007.

[4] S. Brin and L. Page. The Anatomy of A Large-Scale Hypertextual Web Search Engine. *Computer Networks and ISDN Systems*, 30(1-7).

[5] W. B. Frakes and R. Baeza-Yates. *Information Retrieval: Data Structures and Algorithms*, Prentice Hall, Englewood Cliffs, NJ, 1992.

[6] G. W. Furnas, S. Deerwester, S. T. Dumais, T. K. Landauer, R. A. Harshman, L. A. Streeter, and K. E. Lochbaum. Information Retrieval Using a Singular Value Decomposition Model of Latent Semantic Structure. In *Proceedings of the 14th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 465--480, Grenoble France, 1988.

[7] S. Hassan, R. Mihalcea, and C. Banea. Random Walk Term Weighting for Improved Text Classification. In *Proceedings of TextGraps: 2nd Workshop on Graph Based Methods for Natural Language Processing, A*CL, 2006, 53-60.

[8] E. Ide. New Experiments in Relevance Feedback. *The SMART Retrieval System*, pages 337-354. Prentice-Hall, 1971.

[9] M. R. Islam, B. D. Sarker, and M. R. Islam. An Effective Term Weighting Method Using Random Walk Model for Information Retrieval. In *Proceedings of the International Conference on Computer and Communication Engineering*, Malaysia, 2008.

[10] H. Kaji, Y. Morimoto, T.Aizono, and N. Yamasaki. Corpus-Dependent Association Thesauri for Information Retrieval. In *Proceedings of the 18th Conference on Computational Linguistics*, Germany, 2000.

[11] G. Kasai, M. Lalmas, and T. Roelleke. Focused Structured Document Retrieval. In *Proceedings of the 9th International Symposium on String Processing and Information Retrieva*l, 2000.

[12] J. M. Kleinberg. Authoritative Sources in A Hyperlinked Environment. *Journal of the ACM*, 46(5): 604-632.

[13] R. Mihalcea and P. Tarau. TextRank: Bringing Order into Texts. In *Proceedings of Empirical Methods in Natural Language Processing. ACL*, 2006, 404-411.

[14] M. Montes-y-Gomez, A. López-López, and A. Gelbukh. Information retrieval with conceptual graph matching. In the *Proceedings of the 12th International Conference of Database and Expert Systems Applications*, 2001.

[15] National Institute of Standards and Technology. *Text REtrieval Conference*. http://trec.nist.gov

[16] Y. Ogawa, T. Morita, and K. Kobayashi. A Fuzzy Document Retrieval System Using the Keyword Connection Matrix and a Learning Method. *Fuzzy Sets and Systems,* 39:163--179, 1991.

[17] P. Ogilvie and J. Callan. Language Models and Structured Document Retrieval. In *Proceedings of the Initiative for the Evaluation of XML Retrieval Workshop*, 2002.

[18] J. Y. Pan, H. J. Yang, C. Faloutsos, and P. Duygulu. GCap: Graph-based Automatic Image Captioning. In *Proceedings of the 4th International Workshop on Multimedia Data and Document Engineering (MDDE'04)*, Washington, DC, USA, 2004.

[19] M. F. Porter. An Algorithm for Suffix Stripping. *Program*, 14(30) pp.130-137, 1980.

[20] Y. Quintana, M. Kamel, and A. Lo. Graph-Based Retrieval of Information in Hypertext Systems. *In Proceedings of the 10th annual international conference on Systems documentation*, ACM Press, 1992, pp.157-168.

[21] B. Ribeiro-Neto and R. Muntz. A Belief Network Model for IR. In *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 253--260, Zurich, Switzerland, 1996.

[22] S. E. Robertson and K. Sparck Jones. Relevance Weighting of Search Terms. *Journal of the American Society for Information Sciences*, 27(3):129--146, 1976.

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

[24] G. Salton and C. Buckley. Term-Weighting Approaches in Automatic Retrieval. *Information Processing & Management*, 24(5):513--523, 1988.

[25] G. Salton, E. A. Fox, and H. Wu. Extended Boolean Information Retrieval. *Communications of the ACM*, 26(11):1022--1036, 1983.

[26] S. K. Taher Haveliwala and G. Jeh. An analytical Comparison of Approaches to Personalizing Pagerank. *Technical report, Stanford University*, 2003.

[27] D. Thammasut and O. Sornil. A Graph-Based Information Retrieval System. In *Proceedings of International Symposium on Communications and Information Technologies*, 2006, pp. 743-748.

[28] R. M. Tong, L. A. Appelbaum, V. N. Askman, and J. F. Cunningham. Conceptual Information Retrieval Using RUBRIC. In *Proceedings of ACM Conference on Research and Development in Information Retrieval*, 1987, pp.247-253.

[29] H. Tong, C. Faloutsos, and J.-Y. Pan. Fast Random Walk with Restart and Its Applications. In *Proceedings of International Conference of Data Mining*, 2006.

[30] H. Turtle and W. B. Croft. Evaluation of an Inference Network-Based Retrieval Model. *ACM Transactions on Information Systems*, 9(3):187--222, July 1991.

[31] R. Wilkinson and P. Hingston. Using the Cosine Measure in a Neural Network for document retrieval. In *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 202--210, Chicago, IL USA, October 1991.

[32] C. C. Yang and K. Y. Chan. Retrieving Multimedia Web Objects Based on PageRank Algorithm. In *Proceedings of the 14th International Conference of World Wide Web*, 2005.