

การขยายตัวแบบภาษาโดยใช้น้ำหนักความใกล้ชิดของคำ
เพื่อหาความสัมพันธ์ของชุดคำในการค้นคืนสารสนเทศ



นาย สมพงษ์ กิตตินราคร

สถาบันวิทยบริการ

จุฬาลงกรณ์มหาวิทยาลัย
วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต
สาขาวิชาวิทยาศาสตร์คอมพิวเตอร์ ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย

ปีการศึกษา 2549

ลิขสิทธิ์ของจุฬาลงกรณ์มหาวิทยาลัย

**EXTENDING LANGUAGE MODELS WITH
TERM PROXIMITY WEIGHT TO UTILIZE TERM SET RELATION
IN INFORMATION RETRIEVAL**



Mr. Sompong Kittinaradorn

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

A Thesis Submitted in Partial Fulfillment of the Requirements
for the Degree of Master of Science Program in Computer Science
Department of Computer Engineering
Faculty of Engineering Chulalongkorn University
Academic Year 2006
Copyright of Chulalongkorn University

สมพงษ์ กิตตินราคร ชื่อวิทยานิพนธ์: การขยายตัวแบบภาษา โดยใช้น้ำหนักความ
ใกล้เคียงของคำเพื่อหาความสัมพันธ์ของชุดคำในการค้นคืนสารสนเทศ (EXTENDING
LANGUAGE MODELS WITH TERM PROXIMITY WEIGHT TO UTILIZE TERM SET
RELATION IN INFORMATION RETRIEVAL)

อ. ที่ปรึกษา: คร. อรรถสิทธิ์ สุรฤกษ์, อ. ที่ปรึกษาร่วม: นครทิพย์ พร้อมพูล, 93 หน้า.

งานวิทยานิพนธ์นี้มุ่งที่จะเพิ่มประสิทธิภาพในการค้นคืนสารสนเทศโดยใช้แนวทางใหม่ในการคำนวณน้ำหนักของคำในคำถาม(QUERY)ที่ส่งไปยังระบบค้นคืนสารสนเทศที่ใช้ตัวแบบภาษา (LANGUAGE MODEL) งานชิ้นนี้เสนอแนวคิดให้เพิ่มน้ำหนักของคำตามความสำคัญของคำนั้นที่มีต่อประเด็นหลักในคำถาม ตามแนวคิดดังกล่าว ข้อความหนึ่งจะประกอบด้วยชุดของลำดับคำ และแต่ละชุดคำจะประกอบกันเป็นประเด็นหนึ่งโดยมีคำที่มีน้ำหนักสูงสุดในกลุ่มเป็นตัวแทนของชุดคำนั้น ตัวแทนของชุดคำจะประกอบกันเพื่อเสริมประเด็นใหญ่ประเด็นเดียวของประโยคหรือข้อความ งานวิจัยนี้เสนอหลักเกณฑ์ในการแบ่งคำเป็นชุดคำโดยอาศัยกราฟเป็นเครื่องมือ กำหนดให้ลำดับของคำในข้อความเป็นค่าตามแกนนอนและน้ำหนักคำที่คำนวณแบบอินเวิร์ส (INVERSE DOCUMENT FREQUENCY: IDF) เป็นค่าตามแนวตั้ง คำที่เป็นจุดยอดบนเส้นกราฟและอยู่เหนือเส้นกำหนดค่าขั้นต่ำ (THRESHOLD) จะถือว่าเป็นคำตัวแทนประเด็น และแต่ละคำตัวแทนจะมีคำที่อยู่ข้างเคียงเป็นสมาชิกของชุดคำ คำที่มีน้ำหนักน้อยสุดระหว่างยอดสองยอดจะเป็นคำที่แบ่งชุดคำ งานวิจัยนี้นำเสนอสูตรในการคำนวณน้ำหนักของคำโดยมีขั้นตอนหลักสามขั้นตอน ขั้นตอนแรกคำนวณหาความสำคัญของคำตัวแทนของประเด็นที่มีต่อประเด็นใหญ่ ขั้นตอนต่อมาคำนวณหาความสำคัญของคำที่มีต่อคำตัวแทนของประเด็นในชุดคำเดียวกัน ขั้นตอนสุดท้ายใช้คำที่ได้จากการคำนวณในสองขั้นตอนแรกคำนวณหาความสำคัญของคำที่มีต่อประเด็นใหญ่ การทดลองนำสูตรดังกล่าวมาใช้กับฐานข้อมูลทดลองของเทร็ค(TEXT RETRIEVAL CONFERENCE: TREC) ให้ผลเป็นที่น่าพึงพอใจเมื่อเทียบกับผลการค้นคืนปกติ สูตรที่นำเสนอเพิ่มประสิทธิภาพในเชิงความแม่นยำเฉลี่ย (MEAN AVERAGE PRECISION) 16.12 และ 15.74 เปอร์เซนต์สำหรับชุดคำถามที่เจ็ดและแปดตามลำดับ (TREC 7, TREC 8)

ภาควิชา วิศวกรรมคอมพิวเตอร์

สาขาวิชา วิทยาศาสตร์คอมพิวเตอร์

ปีการศึกษา 2549

ลายมือชื่อนิสิท.....

ลายมือชื่ออาจารย์ที่ปรึกษา.....

ลายมือชื่ออาจารย์ที่ปรึกษาร่วม.....

4771452421 MAJOR: COMPUTER SCIENCE KEY WORD: TERMWEIGHTING / INFORMATION RETRIEVAL / TERM DEPENDENCY / LANGUAGE MODEL / TERM IMPORTANCE

SOMPONG KITTINARADORN, EXTENDING LANGUAGE MODELS WITH TERM PROXIMITY WEIGHT TO UTILIZE TERM SET RELATION IN INFORMATION RETRIEVAL THESIS ADVISOR: ATHASIT SURARERKS, PHD., THESIS CO-ADVISOR: NAKORNTHIP PROMPOON, 93 pp.

This research work is aimed at improving the performance of ad hoc information retrieval via a novel method to compute query term weights on the assumption that terms can be grouped by concepts, as against the conventional practice that terms are independent of one another. The new method is based on the approach that the importance of a term is determined by its contribution to the key concept term of the text. The research introduces a heuristics to group terms by concepts. To visualize it, a graph is plotted with the ordered term positions of a query on the x-axis and the well-known *idf* weights (Inverse Document Frequency) on the y-axis. Peak terms are classified as concept terms if their *idf* weights are above a threshold. The highest peak term is the key concept term. Each peak terms are supported by satellite terms on both sides. Between two adjacent peak terms, the term with the lowest *idf* weight is used to mark a boundary of term sets.

Computation is a three-stepped process: the first to compute the importance of the concept term to the distinct key concept term, the second to estimate the importance of a term in reference to the concept term of the same term set, and the last to compute the importance of the term to the key concept. The calculated weights differ from the *idf* weight in that the former reflects term importance in the context of a reference concept, i.e. it is a local property, whereas the *idf* weight is a global property derived from a document collection. In this way, the proposed method can be seen as a context-dependent or concept-determined importance.

To test the efficiency of the new term weighting scheme, an experimental design is devised on the hypothesis that a query with concept-dependent weights for its terms would yield better ad hoc information retrieval results. Experiments are conducted within the language modeling framework using query likelihood scoring method and Dirichlet prior smoothing technique. They produce convincing gains for the new approach compared to the baseline and the *idf*-based results. Improvements are significantly positive on all accounts and are particularly outstanding in the precision area. Using TREC 7 and TREC 8 query sets, the experiments report a 16.12% and 15.74% increases in mean average precision (MAP) respectively. The new method also outperforms the *idf*-based scheme by 9.10%, and 13.34% for TREC 7 and TREC 8 query sets respectively.

Department Computer Engineering

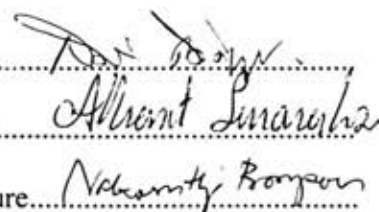
Student's signature.....

Field of study Computer Science.

Advisor's signature.....

Academic year 2006

Co-advisor's signature.....

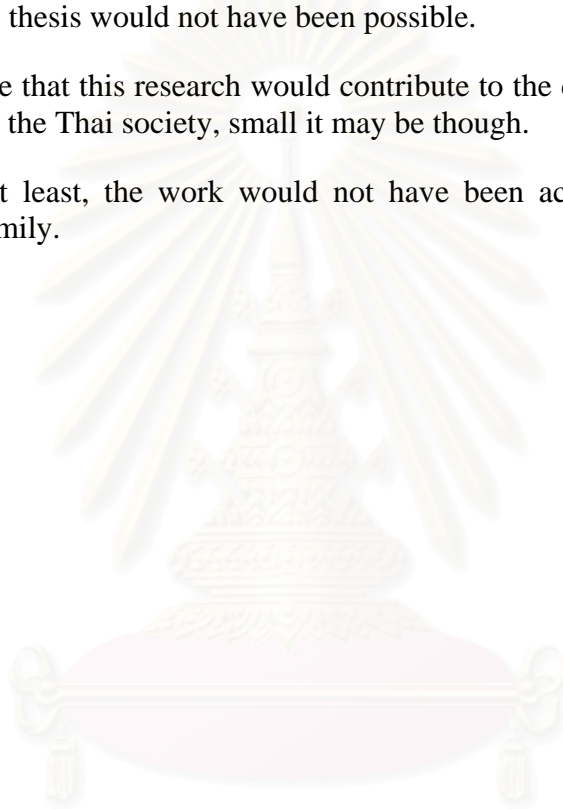


Acknowledgements

I would like to express sincere gratitude to my advisor Ajarn Athasit Surarerks, Ph.D. and co-advisor Ajarn Nakornthip Prompoon for their invaluable advices and their peer reviews of this research work. I also would like to express special thanks to Associate Professor Wanchai Rivepiboon, Ph.D, Associate Professor Somchai Prasitjutrakul for their sharp and thoughtful comments, and Assistant Professor Arnon Rungsawang, Ph.D. of Kasetsart University for reviewing and commenting on this work and for his kind provision of TREC database for the experiments. There is no doubt in my mind that without their kind assistance, this thesis would not have been possible.

It is my hope that this research would contribute to the development of information retrieval research in the Thai society, small it may be though.

Last but not least, the work would not have been accomplished without strong support from my family.



สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

Table of Content

THAI ABSTRACT.....	iv
ENGLISH ABSTRACT.....	v
ACKNOWLEDGEMENTS.....	vi
TABLE OF CONTENT.....	vii
TABLE OF TABLES	ix
TABLE OF FIGURES.....	x

CHAPTER 1..... 1

INTRODUCTION..... 1

1.1.	BACKGROUND AND IMPORTANCE.....	1
1.1.1.	Co-concurrency Frequency Approach.....	2
1.1.2.	N-gram Approach.....	2
1.1.3.	The WordNet Approach.....	3
1.1.4.	Association Rules Mining Approach.....	3
1.1.5.	Unsupervised Learning of Dependency.....	3
1.1.6.	Cognitive Approach.....	4
1.2.	RESEARCH OBJECTIVES.....	4
1.3.	RESEARCH SCOPE.....	5
1.4.	BASIC AGREEMENT.....	5
1.4.1.	Language Model/Query Likelihood Setting.....	5
1.4.2.	TREC 7-8 Experiment Environments.....	6
1.4.3.	Lemur Project Development Toolkit.....	6
1.4.4.	C++ Programming.....	6
1.5.	RESEARCH LIMITATIONS.....	6
1.5.1.	Thai Language Issue.....	6
1.5.2.	Comparison Issue.....	6
1.6.	DEFINITIONS.....	7
1.7.	EXPECTED BENEFITS.....	8
1.8.	RESEARCH METHODS.....	8
1.8.1.	Literature Survey.....	8
1.8.2.	Approach Formulation.....	9
1.8.3.	Preliminary Assumption Assessment.....	9
1.8.4.	Model Conceptualization & Theorization.....	9
1.8.5.	Experimental Planning.....	9
1.8.6.	Experimental Groundwork.....	9
1.8.7.	Run Experiments.....	9
1.8.8.	Evaluation Techniques.....	10
1.8.9.	Analysis & Conclusion.....	10
1.9.	RESEARCH PRESENTATION STEPS.....	10

CHAPTER 2..... 11

2.1.	CAPTURING TERM RELATIONS.....	11
2.1.1.	Hyperspace Analogue to Language (HAL) & Information Flow.....	11
2.1.2.	Unsupervised Learning of Term Dependencies.....	13
2.2.	MEASURING RETRIEVAL PERFORMANCES.....	18
2.2.1.	Relevant Document Return (<i>rel_ret</i>).....	18
2.2.2.	Interpolated Recall - Precision Averages.....	18
2.2.3.	Mean Average Precision (non-interpolated).....	19
2.2.4.	Precision After X Documents.....	19
2.2.5.	R-Precision.....	19

CHAPTER 3..... 20

3.1.	OVERVIEW OF RESEARCH METHODS	20
3.2.	APPROACH FORMULATION	21
3.2.1.	Flow of thoughts & flow of words.....	21
3.2.2.	Term Weight.....	24
3.2.3.	Term Direction	24
3.2.4.	Term Dependency/Term-Concept Contribution	25
3.2.5.	Heuristics Rules	25
3.2.6.	Methodology: Probabilistic Form of Local Term Importance	27
3.3.	EXPERIMENTATION OBJECTIVE	30
3.4.	HYPOTHESIS	30
3.5.	EXPERIMENTAL SETUP	30
3.5.1.	Document Collections	30
3.5.2.	Database	31
3.5.3.	Query Sets.....	31
3.5.4.	Toolkits.....	31
3.6.	EXPERIMENTATION STEPS	32
3.6.1.	Prepare Test Database:	32
3.6.2.	Prepare queries:	32
3.6.3.	Compute <i>idf-dd</i> weights for query terms:	32
3.6.4.	Pass weighted queries to retrieval engine:	34
3.6.5.	Evaluate retrieval results:	35
CHAPTER 4.....		36
4.1.	EVALUATION OF EXPERIMENTAL RESULTS	36
4.2.	COMPARISON WITH EXISTING STANDARDS	38
4.3.	RESULT ANALYSIS	44
CHAPTER 5		46
5.1.	CONCLUSION	46
5.2.	COMMENTS.....	47
5.3.	RECOMMENDATIONS	47
REFERENCES		48
APPENDICES		51
APPENDIX A		51
8.1.	A.1. BOOLEAN MODEL	51
8.2.	A.2. VECTOR SPACE MODEL	51
8.3.	A.3. PROBABILISTIC MODEL.....	52
8.4.	A.4. LANGUAGE MODEL.....	53
APPENDIX B.....		57
APPENDIX C		63
APPENDIX D		70
APPENDIX E.....		71
APPENDIX F.....		72

Table of Tables

Table 3.2 outlines the hypotheses and test strategies.....	30
Table 3.3 shows details of the three document collections	31
Table 3.4 shows the details of <i>threedb</i> database.....	31
Table 4.1 shows the evaluated results of experimental runs using the $f(idf, dd)$ weighting scheme with TREC 7 and TREC 8 query sets. The evaluation computation is done by <i>trec_eval.exe</i> program.....	36
Table 4.2 shows the break-down of the number of result counts on three measurement criteria and their combination for 100 query sets (ALL), TREC 7 query set (50), and TREC 8 query set (50).....	37
Table 4.3 shows information retrieval performance increase from using $f(idf, dd)$ weighting. The baseline in this case is the results from <i>idf</i> weighting scheme. Query set is from TREC 7.....	39
Table 4.4 shows information retrieval performance increase from using $f(idf, dd)$ weighting. The baseline in this case is the results from <i>idf</i> weighting scheme. Query set is from TREC 8.....	40
Table 4.5 shows comparative results on TREC 7.....	41
Table 4.6 shows retrieval performances from using <i>idf</i> -based and $f(idf, dd)$ weighting against the baseline result compared on TREC 8.....	42
Table 4.7 shows the R-precision after X number of returned documents for TREC 8 query set.....	44

Table of Figures

Figure 1.1: A query example with an established linkage. Stop words are in brackets.	3
Figure 2.1 illustrates an example of total inclusion that leads to maximum information flow from satellites to the combined concepts of space and program.	13
Figure 2.2 compares the three language models of unigram, bi-gram and dependency model. Terms in brackets are stop words.	14
Figure 2.3: (a) An example of a dependency cycle: given that $P(d_{23})$ is smaller than $P(d_{12})$ and $P(d_{13})$, d_{23} is removed (represented as dotted line). (b) An example of a dependency crossing: given that $P(d_{13})$ is smaller than $P(d_{24})$, d_{13} is removed.	18
Figure 2.4: Approximation algorithm of dependency parsing.	18
Figure 3.2 shows topic 402 of TREC 8. The description part is used in our example to plot a line chart.	22
Figure 3.3 shows the line chart of a porter-stemmed version of topic 402 from TREC 8. The circled areas cover terms in the same group as the head words which make the peaks of the graph. The labeled figures are the IDF weights of the respective terms.	22
Figure 3.4 illustrates a line chart of a sentence: What language and culture differences impede the integration of foreign minorities in Germany? The terms have been stemmed by Porter algorithm.	23
Figure 3.5 illustrates the line chart of peak words, removing satellite words from the graph to adjust the positions between the PEAK points.	28
Figure 3.6: Diagram shows the four steps in the experimentation of the new <i>idf-dd</i> weighting effects on ad hoc information retrievals.	33
Figure 3.7: The parameter file used by RetEval.exe. The <i>testrun8nostops.qry</i> input file contains query terms weighted by the new weighting scheme.	34
Figure 4.1: Pie chart showing that out of 100 query topics (TREC 7 & TREC 8 combined), 57 yield positive results, 27 negative and the rest 16 neutral.	38
Figure 4.2 (a), above, shows a graph comparing R-precisions of baseline, <i>idf</i> , and <i>f(idf,dd)</i> results, and (b), below, shows comparisons by Recall-Precision Averages. Queries used from TREC 8.	43
Figure F.1: A diagram showing an input query file in <i>BasicDocStream</i> class format going through an application <i>TestTermWeightApp.exe</i> , which invokes <i>TermweightQueryRep</i> class library and outputs a transformed query file structured to conform to the <i>WeightedDocStream</i> class.	72

CHAPTER 1

INTRODUCTION

1.1. Background and Importance

In Information Retrieval (IR), terms constitute fundamental building blocks. Terms carry different importance weights and how they are measured would directly influence the overall performance of a retrieval engine. The term weighting scheme $tf*idf$ (term frequency, inverse document frequency) has been used extensively to estimate the semblance of term importance. The tf usually represents a local property from a query or a document and the idf is a global property obtained statistically from a document collection. The assumption upon which the scheme is developed is that all terms are independent of one another. Relating term importance is no novel efforts. But to date, the twin issues of term weights and term relationship seem to have been addressed separately.

This study proposes a new approach that the importance of a term is determined by how much it contributes to the concepts that the text communicate to readers. Concepts are represented by emphasized terms and the degree of a term's contribution to a concept, i.e. a key word, justifies the former term's presence. Put it another way, terms depend on concepts.

Based on the approach, a model has to be developed to capture term dependencies from the local context. From the term dependency, together with the weight, i.e. the idf in this case, term importance is computed. In the conventional scheme, the function $tf*idf$ is used to compute term importance. In the approach proposed here, it is $f(idf, dd)$ where dd is dependency degree extracted and computed from the context as a local property. The idf remains a global property. The local property tf is implicitly retained. Furthermore, the function $f(idf, dd)$ is developed in probability form unlike the $tf*idf$, which has been formulated intuitively and empirically. Our research problem: How to define the function $f(idf, dd)$.

Two points have to be made clear from the outset:

Firstly, this model requires a mechanism to estimate the weights of individual terms as a global property. In the experiments conducted in this research, the idf is picked for the global term weighting. The efficiency of the model is therefore determined to a certain extent by the efficiency of the idf term weighting. The model, however, is flexible enough to support other term weighting schemes.

Secondly, this thesis does not propose improving ad hoc information retrieval by query weighting. The query weighting is used for the experimentation just to prove the efficiency of the new scheme. It is hoped that the work can be a step towards applying it to other IR areas.

Extracting term dependencies from a text and applying it in an IR task, has either been lacking or largely ineffective. This, despite the general acceptance that term dependency is an indispensable consequence of language use [3]. At the heart of most engines, some forms of idf or $tf*idf$ have been employed. The vector space

model [1] represents a document or query by a multi-dimensional vector of isolated terms. Okapi BM25 of the probability model also uses tf and document length (dl) from local text as local properties and employ idf and average document length statistics from a document collection as global properties [4].

The language modeling approaches to IR has the design ability to allow for seamless incorporation of term dependency, prompting a surge in the number of researches to incorporate term dependency properties. These proposed extensions to and/or enhancement of language models are evident in a broad range of IR domains, including ad hoc information retrieval and query expansion. Some of them have produced impressive results, testifying to the belief that term dependency has strong potentials to bring retrieval powers to a new level of sophistication.

Apart from that, a number of researches address the issue of term weighting together with term dependency. These recent researches propose alternatives to the use of idf (or $tf*idf$), contending that term relationship or context should be taken into account in measuring term weights. The alternatives include the use of association rules mining [4, 5] and the context term model [7].

That said, the fundamental issue of how to effectively and efficiently determine term relations remains largely unsettling. This point can be amplified by a quick survey of related IR researches. A list of researches, categorized by how they implement term dependency in support of their models, is summarized as follows:

1.1.1. Co-concurrency Frequency Approach

Term co-concurrency has also been used extensively to derive term relations in a number of studies [11]. Typically, the frequency of term concurrences observed from a certain text, which can be a whole document, a passage or a sentence or a window of a fixed length to provide an input to a function to calculate the strength of term relationship:

$$P_{co}(t_2 | t_1) = \frac{f(t_1, t_2)}{\sum f(t_1, t_i)}, \text{ where } f(t_1, t_2) \text{ is the frequency of the } t_1 \text{ and } t_2 \text{ co-}$$

concurrences.

1.1.2. N-gram Approach

In language modeling approaches to IR, a multinomial model $P(w/d)$ is estimated over terms in each document d in an indexed, searchable collection and used to assign the likelihood of a query terms $q=q_1, q_2 \dots q_n$. The likelihood that the query belongs to the language model of the document is estimated by $P(q | d) = \prod_{i=1}^n P(q_i | d)$. The posterior probability of the a document, $P(d | q) \propto P(q | d)P(d)$, is used to rank documents. The probability equation, $P(q | d) = \prod_{i=1}^n P(q_i | d)$, known as the unigram model, assumes that each q_i is independent from one another in the query. Initial efforts to introduce term dependency include the bi-gram [8] and bi-term models [9]. Dependency links are introduced to pairs of adjacent words in the bi-gram model and to pairs of adjacent or more distant terms in the bi-term model. Experimental results

showed no significant improvements, mainly because the links are arbitrarily introduced and also because they are restricted to pairs of words [10]. The assumption of the bi-gram model that dependency must occur between adjacent terms is also too rigid.

1.1.3. The WordNet Approach

This school exploits term links stored in a handcraft thesaurus, such as *WordNet*. Voorhees was the first to exploit *WordNet* for query expansion [12]. Liu et al. [13] also use *WordNet* to disambiguate word senses of query terms and from the determined word senses, its synonyms, hyponyms as well as their compound words are considered to be candidates for additions to the query. Mandela et al. [14] use both *WordNet* and automatically constructed thesauri, Cao et al. combine the strength of handcraft *WordNet* and co-concurrence information and neatly integrate the component into the language model [15].

1.1.4. Association Rules Mining Approach

Possas et al. proposed in [5] the Set-based Model, in which sets of correlated terms are computed from association rules mining using collection-wide co-concurrence frequency statistics. In a more recent work, [16] proposed Maximal Term-set which applies similar association rules techniques to restructure Web queries.

1.1.5. Unsupervised Learning of Dependency

Gao, et al. adopts the approach in their work on Dependency Language Model for Information Retrieval [10]. Fixing the pitfalls of bi-gram and bi-term language modeling approaches, Gao, et al. use dependency links between pairs of words for a linkage represented by an acyclic, planar undirected graph.

To find the strongest linkage in a query, the model requires a dependency link-annotated training data, which is lacking. They solve the problem via an EM-like technique to learn the parameters of their parsing model from an algorithm to remove the weaker of the cyclic and crossing links. The procedure has an $O(n^2)$ complexity, where n is the number of terms in a query.

Substantial experimental gains are reported for the approach. Unfortunately, it requires an unsupervised learning method to establish the most probable linkage [17].

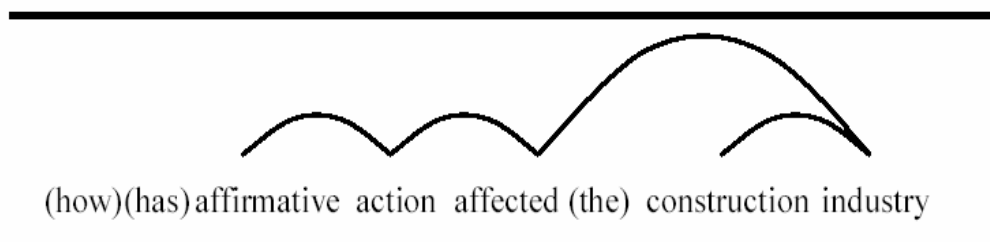


Figure 1.1: A query example with an established linkage. Stop words are in brackets.

1.1.6. Cognitive Approach

On a different track from classical IR approaches, Burgess et al. develop a representational model of semantics memory called Hyperspace Analogue to Languages (HAL), which automatically constructs a high dimensional space from a text corpus [18]. The HAL space contains a matrix of terms in the document collection. Each term has a high dimensional vector space of other terms that are found co-occurring with the term. Imitating human concept learning, HAL uses a technique of sliding a window frame of empirically-determined length l through a corpus. Moving the frame one word at a time, the co-occurrence of words in the same frame are recorded and accumulated. According to the approach, a term in the HAL space, together with its vector of other terms make a concept. Bruza et al. model information flows from the HAL-based matrix and integrate it into their query expansion model in IR [19]. The approach is distinguished from the others on its ability to combine concepts and infer a new term from concepts. Bai, et al., nicely incorporates the HAL-based information inference model into a language model to expand on queries [20].

Most of the afore-mentioned approaches still rely on co-concurrency frequencies of terms to determine correlations of terms. Exceptions are the application of unsupervised learning which may not be convenient in practical terms, and the information inference mechanism which is represented by HAL-based vector space.

This research is inspired by previous works and the challenge of finding an alternative to relating term dependency in the overall scheme of concept importance. In the initial research efforts, we tried association rule mining techniques on the assumption that its efficiency should improve if relations are determined at the sentence level, rather than the document level. Although association rule mining is a powerful theory, the techniques used still rely to a large extent on global co-concurrency frequencies and simplistic proximity statistics. In our experiments, improving its efficiency by association rule mining also incurs substantial computational costs which we feel may not warrant the gains from it. We are also impressed by the Dependency Language Model but feel that the unsupervised learning of dependency links may not be appropriate in ad hoc IR. To our knowledge, the model proposed in this thesis is novel. It is significantly influenced by the idea of information concepts that prevails in the HAL space model. The combination of concepts and the information flow from concepts to another takes the general issue of text concepts to another higher level. For that aspect, this work is still way behind it.

1.2. Research Objectives

1. Achieve a better measurement of term importance in a context and prove by experiments that it can yield better results than conventional methods in IR tasks,
2. Provide a solution to the issue of how to incorporate term dependency into the overall scheme of term importance measurement,
3. Apply the model with a task of ad hoc information retrieval and prove

that it can increase the overall precision and recall of a standard retrieval engine in the language modeling approach to IR. The experiment is a means of validating the model of measuring term importance by term contributions to, i.e. dependencies on, concept terms.

1.3. Research Scope

1. Elaborate the concepts and theoretical support for the new model. The model must be complete with necessary functions or algorithms to compute the degrees of term contribution to key concepts of text.
2. Develop a software module to implement the functions and algorithms in conjunction with the approach, and use it for the experimentation to prove the value of the model.
3. Conduct a series of experimentation. The researcher chooses to apply the model with the task of re-weighting query terms for ad hoc information retrieval. The results of the experiments are measured against baseline results from using simple and *idf* weighting in a standard information retrieval engine. In the experimentation, the language modeling approach in the query likelihood scoring family with *Dirichlet prior* smoothing technique is used. Test databases are three TREC document collections (disk 4 and disk 5) provided by the National Institute of Standards and Technology (NIST) for IR researches. Queries used belong to the TREC 7 and TREC 8 query sets, also provide by NIST along with experts-picked relevant documents.
4. Report the results of the experiments in details, analyze the outcomes, and present observations as well as conclusions from the experiment results.

1.4. Basic Agreement

1.4.1. Language Model/Query Likelihood Setting

The baseline retrieval engine is in the query likelihood family of the language modeling approaches to IR. The query likelihood scoring method is adopted because

1. The multiple Bernoulli family of language modeling approaches to IR provides no model support for term dependencies as it considers terms as “bags of words” whereby ordering and positions are ignored. In this research, the more widely accepted multinomial family is adopted.
2. An alternative to query likelihood in the multinomial family is the KL-Divergence model proposed in [27]. This model differs from the query likelihood approach in that it transforms queries into a language model similar to the way language models are estimated from documents. Scoring is obtained by measuring the differences between the query language model and the document language model under consideration.

The query likelihood model, on the other hand, compares the query representation directly with the document language model and calculates the probability that the former is a sample of the document language model. We implement the query likelihood model simply to conform to the mainstream of IR researches.

1.4.2. TREC 7-8 Experiment Environments

Experiments are based on NIST-provided test document collections, TREC query sets, together with experts-picked results. The corpus of documents used is Financial Times news, Los Angeles Times news and Foreign Broadcasting Information Services (FBIS). The queries used are taken from TREC 7 and TREC 8 query sets.

1.4.3. Lemur Project Development Toolkit

The toolkit is provided charge-free to IR researchers by the Lemur project. The software used to compute the recall/precision and other measurement figures is obtained from NIST. The software is called *trec_eval.exe*.

1.4.4. C++ Programming

Coding is implemented as a standard C++ library as required by the Lemur Project Toolkit V 4.2.

1.5. Research Limitations

1.5.1. Thai Language Issue

The evaluation of the model can be experimented only in the English language because there is no available test collection in Thai that comes with answers. This does not mean that the proposed model is designed for only the English language. There is simply no complete set of test collections to experiment and evaluate Thai-language document collection.

1.5.2. Comparison Issue

Comparison Issue: Evaluation against other models. It is difficult and can possibly be biased to try to compare the proposed model in the context of other models that employ other approaches to obtain term dependency relations. As a result, the experiment has to be designed in a way that is considered sufficient to prove that the proposed model can improve efficiency of ad hoc information retrieval. The new model is tested against the baseline with normal queries and with queries weighted by the *idf* scheme.

1.6. Definitions

Ad Hoc Information Retrieval: An IR field which focuses on the retrieval of documents with respect to queries which are issued by users on a real-time basis. The retrieval engine has no prior knowledge of the queries.

Query Expansion: Another IR field which concentrates on expanding user queries to improve the efficiency of information retrieval.

Vector Space Model: Probably the first information retrieval model in modern IR history. The model sees terms of a query or documents as a multi-dimensional vector of terms. The vectors are combined to represent a document or query. Terms are assumed to be independent of one another. (Read formal definition and more explanation in Appendix A: Information Retrieval Models.)

Probabilistic Model: Another well-known information retrieval model developed on probability theory. The model assumes that when a user query is issued, documents are examined and ranked by the probability of relevancy between the query and documents. Documents are ranked by the probability calculations. (Read formal definition and more explanation in Appendix A: Information Retrieval Models.)

Language Model: The latest information retrieval model in modern IR, the language model was first introduced in the speech recognition and has been widely adopted in many disciplines related to natural languages, including natural language processing and machine translation. In IR, the language modeling approach is increasingly popular for encouraging experimental results and the strong theoretical foundation and its model flexibility to incorporate, among other things, word dependencies. The model is represented by the probability that a query is a sample of a language model estimated from a document in question. A derive alternative is the KL-divergence language model. (Read formal definition and more explanation in Appendix A: Information Retrieval Models.)

TF*IDF: Standard measurement of Term weighting. TF stands for term frequency and is usually counted from a user query. IDF stands for inverse document frequency which is computed from the logarithm of the number of documents in a collection divided by the number of documents which contain the term. The usual form of it is $tf*idf$. The tf in this form is normalized. (See more explanation in Section Approach, Chapter 3.)

Association Rule Mining: A field in Data Mining. Association rule mining originated from observations and analysis of the relations among shopping items. When it is applied in IR, terms are considered items and sets of terms (term-sets) are compared to baskets of shopped items. In IR, co-concurrency frequencies statistically collected from a corpus of document collection are used to measure the support and confidence levels of the relations.

Unsupervised learning: Unsupervised learning is a study field in Machine Learning discipline. It has been widely used in many areas of computer science. (See more explanation in Related Theories and Works, Chapter 2.)

Expectation Maximization (EM): A methodology to estimate unknown parameters of a function. The parameters are determined via iteration based on

assumed initial parameters. Learning algorithms and training data are required in the process. (See more in Related Theories and Works, Chapter 2.)

Cognitive Science: A discipline to study the process of human sensing and external environments and human learning to gain knowledge out of the process.

Concept-Dependent Term: A member of an ordered term set which makes a concept. The term differ from the other members for it highest *idf* weight.

Concept-Determined Term: Same a concept-dependent term.

Context-Dependent Term: Same a concept-dependent term.

Peak Term: Same a concept-dependent term.

Satellite Term: A member of an ordered term set which makes a concept.

1.7. Expected Benefits

1. This model presents itself as an alternative approach to compute term importance weighting which is required in a wide range of IR works. Citing an example, the sentence “*what languag and cultur differ impeded the integr of foreign minor in germani*” can be represented in Gao, et al.’s acyclic, planar and undirected graph of outstanding dependencies. This work is expected to be a step towards the application of the approach in other IR areas.

2. This model does not only reveal term associations and concepts but it also implies term importance measured probabilistically as contributions to the key concepts that form the heart of a text. From this perspective, it will hopefully introduce a new way to calculate context-sensitive term weights.

3. As it will be shown later, the computational costs of identifying term associations and calculating term importance is relatively low, compared to EM and association rule mining techniques.

1.8. Research Methods

1.8.1. Literature Survey

Conduct literature survey on IR models, including the Vector Space, the Probabilistic and the Language models, term weighting, term dependency, and related topics.

1.8.2. Approach Formulation

Formulate a novel approach to determine term dependency on the researcher's assumptions that the local context should reveal additional term properties. In the case of this study, the new properties explored are called *term directions* and *term dependency* (or term contributions to concepts).

1.8.3. Preliminary Assumption Assessment

Assess the assumptions by plotting graphs of written text. The description parts of Topics 351-400 of TREC 7 and Topics 401-450 of TREC 8 are used to plot the graph. The x-axis of the graph represents the positions of the ordered terms and the y-axis represents the global term weights. Examining the graphs lend substantial support to the approach.

1.8.4. Model Conceptualization & Theorization

Conceptualize a model from theoretical perspectives. Verify the model. Theories used include *idf*, Pythagoras triangle rule, interpolation of term weight difference and term distance, and the probability theory. All functions required are worked out in this stage.

1.8.5. Experimental Planning

Formulate an experiment plan to prove the model by way of testing the precision/recall of ad hoc information retrieval with queries refined by the new model. A standard language modeling approach is chosen as the experiment setting and the results of the experimentation of the new approach are evaluated by comparing them against baseline and *idf*-based results. The plan includes choices of test database, standard measures to evaluate the results and the choices of development toolkits.

1.8.6. Experimental Groundwork

Develop an application for an implementation of the model in consistency with the experiment plan. Preprocess three document collections: *Financial Times* news, *Los Angeles Times* news, and *Foreign Broadcasting Information Service* reports, to set up an inverted index database. Parse the description part of 100 query topics from TREC 7 and TREC 8. Porter stemming is applied to both the document side and query side preprocessing.

1.8.7. Run Experiments

In addition to run normal test queries on the baseline software as provided by Lemur project, the same queries weighted by *idf* is also run to provide benchmarks against which the results from using the new scheme is evaluated.

1.8.8. Evaluation Techniques

All the experimental results are evaluated by a standard evaluation program, *trec_eval.exe*, provided by US-based NIST for IR researches. Key

evaluation indicators are the mean average precision (map), R-precision and the overall recall as measured by the number of relevant documents returned. Evaluation is also measured by the interpolated recall – precision averages and the precision after X documents. Documents are judged relevant or irrelevant to a particular query by experts. The answers are part of the TREC database.

1.8.9. Analysis & Conclusion

Analyze the results for new findings and reach conclusions on the research topic.

1.9. Research Presentation Steps

1. Present the concept behind the proposed model, calculation methods with theoretical support.
2. Present the results of experiments in the forms of tabular data, charts and the transformed text. All are compared against the baseline and standard techniques.
3. Present the results, analysis and conclusions as integral parts of this thesis.



สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

CHAPTER 2

RELATED WORKS AND THEORIES

2.1. Capturing Term Relations

To our knowledge, there has been no work that takes up the same approach as this study's. Nonetheless, the approach is directly or indirectly influenced by a number of works that address the issue of term dependency. Some of the works are listed here:

2.1.1. Hyperspace Analogue to Language (HAL) & Information Flow

HAL is a cognitively motivated representation of the way human derives a new concept via an accumulation of experiences of contexts in which the concept appears. The accumulation is represented in a HAL matrix of words. Constructing a HAL matrix is done by sliding a fixed l -sized window through a corpus of text, one word at a time. Statistic counts of a term co-occurring with others are accumulated in the course of the scan, and recorded in the matrix. The co-occurrence strength is inversely proportionate to the distance between the two terms in the same window.

In this way, a term or concept has multi-dimensional vectors of terms, and their respective co-occurrence weights. If the weights are above a non-zero threshold, they are considered to be meaningful quality properties of the concept.

In HAL, concepts can be combined. In addition, information can flow, i.e. a concept can be inferred from another (or from combined concepts). The degree of the information flow can also be computed.

The approach is explained in details in [18, 19, and 20]. Here we summarized the key concepts in the subsequent subsections.

Combining Concepts

A simple way to combine concepts is to add together the vectors of their quality property weights. A more complicated way is to classify one concept by its higher $tf*idf$ as the dominant term, heuristically weigh up its property weights, and give additional weights to the intersecting properties of the two terms.

Let $c_1 = \langle w_{c_1 p_1}, w_{c_1 p_2}, \dots, w_{c_1 p_n} \rangle$ be the concept of the dominant term, $c_2 = \langle w_{c_2 p_1}, w_{c_2 p_2}, \dots, w_{c_2 p_n} \rangle$ the concept of the other term, where n is the dimensionality of the HAL space, and $w_{c_j p_i}$ is the weight of property i of term c_j . Let $QP(c)$ denote the quality properties of term c , and $c_1 \oplus c_2$ denote the resulting combined concept. Combining them takes the four following steps:

Step 1 Re-weigh c_1 and c_2 to assign more weights to the properties of the former concept

$$w_{c_1 p_i} = \ell_1 + \frac{\ell_1 * w_{c_1 p_i}}{\text{Max}(w_{c_1 p_k})}$$

k

and $w_{c_2 p_i} = \ell_2 + \frac{\ell_2 * w_{c_2 p_i}}{\text{Max}(w_{c_2 p_k})}$ where

k

$$\ell_1, \ell_2 \in (0.0, 1.0) \text{ and } \ell_1 > \ell_2$$

For example, if $\ell_1 = 0.5$ and $\ell_2 = 0.4$, property weights of c_1 are transferred to interval $[0.5, 1.0]$ and property weights of c_2 are transferred to interval $[0.4, 0.8]$, thus scaling the dimensions of the dominant concept higher.

Step 2 Strengthen the weights of the intersecting properties by a multiplier α .

$$\forall (p_i \in QP(c_1) \wedge p_i \in QP(c_2)) \mid w_{c_1 p_i} = \alpha * w_{c_1 p_i}, w_{c_2 p_i} = \alpha * w_{c_2 p_i},$$

where $\alpha > 1.0$

Step 3 Compute property weights in the combined concept $c_1 \oplus c_2$

$$w_{(c_1 \oplus c_2) p_i} = w_{c_1 p_i} + w_{c_2 p_i}, \text{ where } 1 \leq i \leq n$$

Step 4 Normalize the resulting vector of $c_1 \oplus c_2$, which, in turn, can be composed to another concept by the same heuristic. For the new concept, the degree of dominance is computed from the average of their $tf*idf$ figures.

Information Flow

Barwise and Seligman have proposed an account of information flow (IF) that lays down a theoretical foundation for establishing informational inferences between concepts [2]. For example, $space, program \mid -satellites$ denotes that “satellites” can be inferred from the combined concept “space program”. Bruza et al. [19] developed a heuristic way to compute the degree of information flows between concepts. The HAL space is used to represent the information states of concepts or combined concepts with respect to a given text corpus. The degree of information flow between “satellites” and the combined concept of “space” and “program” is directly related to the degree of inclusion between the respective information states. Inclusion is a relation

\subseteq over the concept space. Inclusion in the example can be denoted by $space \oplus program \subseteq satellites$

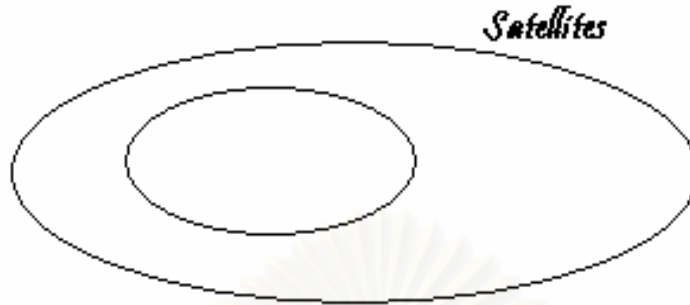


Figure 2.1 illustrates an example of total inclusion that leads to maximum information flow from satellites to the combined concepts of space and program.

Their work can be summarized as follows:

The initial HAL space is filtered to keep only the quality properties with strong co-occurring ties to the concept.

The degree of information flow is defined by

$$i_{1\dots, i_k} | - j \text{ iff } \text{degree}(\bigoplus_{1 \leq i \leq k} c_i \subseteq c_j) > \lambda \text{ where } \lambda \text{ is a threshold value.}$$

This is computed by the summation of the weights of all intersecting quality properties divided by the summation of the weights of all the quality properties of c_i . The function is denoted below.

$$\text{degree}(\bigoplus_{1 \leq i \leq k} c_i \subseteq c_j) = \frac{\sum_{p \in (QP(c_i) \wedge QP(c_j))} w_{c_i p}}{\sum_{p \in QP(c_i)} w_{c_i p}}$$

2.1.2. Unsupervised Learning of Term Dependencies

Unsupervised learning is an alternative way of capturing term dependencies at the sentence level. In their proposal for the Dependency Language Model [10], Gao et al. adopt the methodology to train a parsing model on a linkage L , a representation of term dependencies. The linkage, a set of links or pairs of words, is an acyclic, planar and undirected graph to conform to general linguistic characteristics. The Viberti iterative training procedure (an approximation of the EM training) is used for joint optimization of the parsing model and the training data.

Dependency Language Model

The Dependency Language Model is summarized here before we move to the central issue of unsupervised learning methodology:

The classic unigram model measures a query similarity to a document by the probability $P(Q/D)$ where D is the language model of a particular document and Q is a sample of the language model. Assuming that all terms are independent, we have the probability $P(Q | D) = \prod_{i=1,2,\dots,m} P(q_i | D)$, where q_i is a query term and i is term count. To handle term dependencies, N-gram models have been proposed on the assumption that a term depends on N terms preceding it. The bi-gram model, for an example, has the probability equation denoted by

$$P(Q | D) = P(q_1 | D) \prod_{j=2..m} P(q_j | q_{j-1}, D)$$

The model has failed to achieve consistent improvements in IR. In Dependency Language Model, dependencies are not restricted to adjacent terms. A dependency structure L is an acyclic, planar, undirected linkage comprised of links (or pairs of terms) and unlike N-gram models, the dependency structure is made explicit. Thus the original probability is extended into the probability $P(Q, L | D)$ or $P(L | D)P(Q | L, D)$ where L is the linkage. The probability $P(L | D)P(Q | L, D)$ is an approximation by maximum likelihood estimation, i.e. only the most probable link is used for further computation.

$$P(Q | D) = P(L | D)P(Q | L, D) \text{ such that } L = \arg \max_L P(L | Q)$$

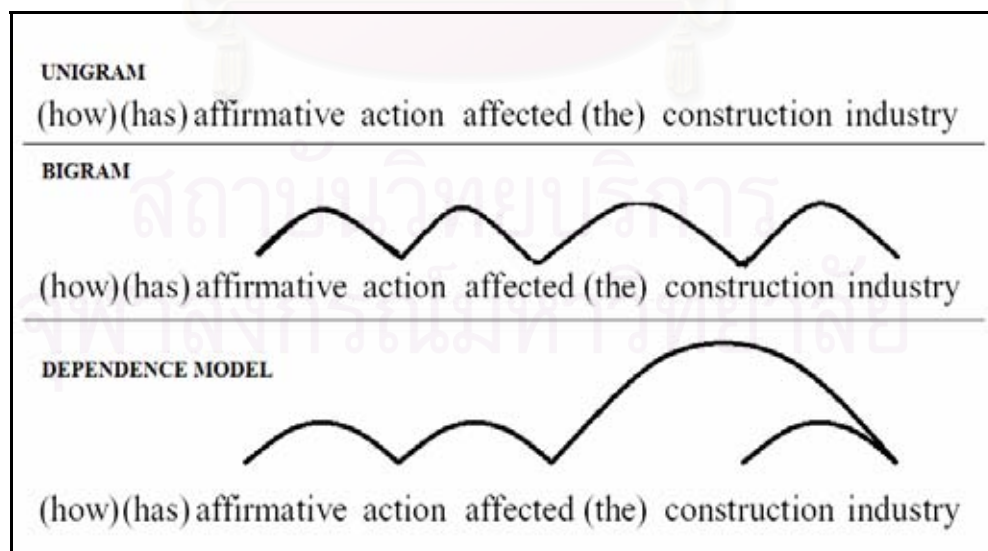


Figure 2.2 compares the three language models of unigram, bi-gram and dependency model. Terms in brackets are stop words.

The probability $P(Q | L, D)$ is then decomposed as follows:

$$P(Q | L, D) = P(q_h | D) \prod_{(i,j) \in L} P(q_j | q_i, L, D)$$

$$P(Q | L, D) = P(q_h | D) \prod_{(i,j) \in L} \frac{P(q_i, q_j | L, D)}{P(q_i | L, D)}$$

$$P(Q | L, D) = P(q_h | D) \prod_{(i,j) \in L} \frac{P(q_i, q_j | L, D) P(q_j | L, D)}{P(q_i | L, D) P(q_j | L, D)}$$

Moving the nominator term $P(q_j | L, D)$ out of the product operator and approximating it to $P(q_j | D)$, the equation can be written as follows:

$$P(Q | L, D) = P(q_h | D) \prod_{j \neq h} P(q_j | D) \prod_{(i,j) \in L} \frac{P(q_i, q_j | L, D)}{P(q_i | L, D) P(q_j | L, D)}$$

And

$$P(Q | L, D) = \prod_{i=1 \dots m} P(q_i | D) \prod_{(i,j) \in L} \frac{P(q_i, q_j | L, D)}{P(q_i | L, D) P(q_j | L, D)}$$

Substituting $P(Q | L, D)$ in $P(Q | D) = P(L | D) P(Q | L, D)$ and taking log, the equation has changed to

$$\log P(Q | D) = \log P(L | D) + \sum_{i=1 \dots m} \log P(q_i | D) + \sum_{(i,j) \in L} MI(q_i, q_j | L, D)$$

$$\text{where } MI(q_i, q_j | L, D) = \log \frac{P(q_i, q_j | L, D)}{P(q_i | D) P(q_j | D)}$$

Each of the three parts on the right-handed side of the equation can be interpreted as follows:

- $\log P(L | D)$ represents the linkage factor which is zero in the unigram and bi-gram model because in the two models $P(L | D)$ is the only event and equal to 1.
- The part $\sum_{i=1 \dots m} \log P(q_i | D)$ represents the probability of the language model generating each term q_i . This is the same as the unigram language model.
- $MI(q_i, q_j | L, D)$ represents dependence relations for term pairs. It can be mapped to $\prod_{j=2 \dots m} P(q_j | q_{j-1}, D)$ in the bi-gram probability

$$P(Q | D) = P(q_1 | D) \prod_{j=2 \dots m} P(q_j | q_{j-1}, D)$$

Gao et al. point out that the unigram and bi-gram models are special cases of their dependency language model.

Parsing Model

To compute the values for $P(L/D)$ in **II** and $MI(q_i, q_j/L, D)$ in **III**, Gao et al. have developed a parsing model to extract dependency links from query and documents. Unfortunately, the model requires unsupervised learning of L for the estimation of their parameters.

The development of such a parsing model is described in this subsection.

$P(L/D)$, which is actually $P(L/Q, D)$, can be approximated by a parsing model of $P(L/Q)$. Let L be a set of probabilistic dependencies, i.e. links, and l be a link, $l \in L$ and assumes that the dependencies are independent from one another, the parsing model can be developed from $P(L/D)$.

$$P(L | Q) = \prod_{l \in L} P(l | Q)$$

$P(l/Q)$ is estimated on a linkage-annotated training data, which is still lacking. Suppose such linkage-annotated training data is readily available, the estimation can be acquired from a pseudo probability function

$$F(R | q_i, q_j) = \frac{C(q_i, q_j, R)}{C(q_i, q_j)}$$

where $C(q_i, q_j, R)$ is the number of times that q_i

and q_j have links in the annotated training data and $C(q_i, q_j)$ is the number of times that the two terms appear in the same sentences. The normalization of the function to make it a real probability is ignored as it will have no effects on the ranking results. Thus,

$$P(l | Q) \propto F(R | q_i, q_j)$$

and.

$$P(L | D) = P(L | Q, D) = \prod_{l \in L} P(l | D) \propto \prod_{(i,j) \in L} F(R | q_i, q_j)$$

Due to data sparseness, interpolation and back-off techniques are also applied in the Dependency Language Model.

For $MI(q_i, q_j/L, D)$, if (q_i, q_j) is not seen in document D , the value is zero. The values of the seen dependency link is estimated as

$$\begin{aligned} MI(q_i, q_j | L, D) &= \log \frac{P(q_i, q_j | L, D)}{P(q_i | L, D)P(q_j | L, D)} \\ &= \log \frac{C_D(q_i, q_j, R) / N}{(C_D(q_i, *, R) / N) * (C_D(*, q_j, R) / N)} \end{aligned}$$

$$= \log \frac{C_D(q_i, q_j, R)N}{C_D(q_i, *, R)C_D(*, q_j, R)}$$

where $C_D(q_i, q_j, R)$ denotes the count of the link (q_i, q_j) in the document D , and $N = CD(*, *, R)$.

The obvious problem with the above model is that there is no ready availability of training documents which are annotated by dependency links. Gao et al. solves the problem by creating a link-annotated training corpus.

Creating a Dependency Link-Annotated Training Data

Gao et al. uses a *Viberti* iterative training procedure (an approximation of EM training) for joint optimization of the parsing model and the linkage of the training data. Three steps are needed for the implementation of the principle:

Step 1: Initialization

A N -sized window is used to determine the initial parameters. N is set at 3 in the Dependency Language Model experiments. Given a word trigram (w_1, w_2, w_3) , the initial links are arbitrarily set as l_{12} , l_{13} , and l_{23} . The links are used as inputs to equations $P(L|Q) = \prod_{l \in L} P(l|Q)$ and $F(R|q_i, q_j) = \frac{C(q_i, q_j, R)}{C(q_i, q_j)}$ to compute the linkage probability.

Step 2: (Re-)parsing the corpus

Use the Yuret algorithm [28] to select the most probable linkages from sentences in the training data. The parser successively eliminates the weaker of the conflicting links from the parsing model, resulting in an updated set of links

Step 3: Re-estimating parsing model parameters

With the updated set of links, re-estimate the parsing model parameters. Steps (2) and (3) are iterated until the improvement to the probability is less than a threshold. The algorithm does not guarantee an optimal outcome and its operating complexity is $O(n^2)$.

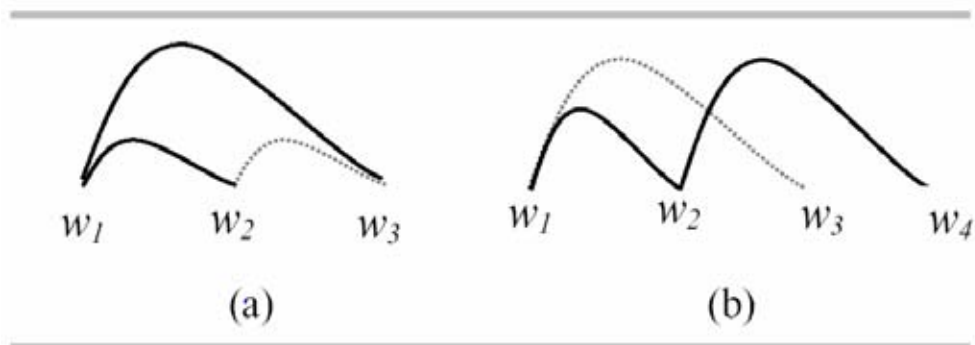


Figure 2.3: (a) An example of a dependency cycle: given that $P(d_{23})$ is smaller than $P(d_{12})$ and $P(d_{13})$, d_{23} is removed (represented as dotted line). (b) An example of a dependency crossing: given that $P(d_{13})$ is smaller than $P(d_{24})$, d_{13} is removed.

```

DEPENDENCY-PARSING( $W$ )
1   for  $j \leftarrow 1$  to LENGTH( $W$ )
2   for  $i \leftarrow j-1$  downto 1
3     PUSH  $d_{ij} = (w_i, w_j)$  into the stack  $D_j$ 
4     if a dependency cycle (CY) is detected in  $D_j$ 
      (see Figure 2.7(a))
5       REMOVE  $d$ , where  $d = \min_{d \in CY} \arg d P(d)$ 
6     while a dependency crossing (CR) is detected
      in  $D_j$  (see Figure 2.7(b)) do
7       REMOVE  $d$ , where  $d = \min_{d \in CR} \arg d P(d)$ 
8   OUTPUT ( $D$ )

```

Figure 2.4: Approximation algorithm of dependency parsing

2.2. Measuring Retrieval Performances

Following is a brief explanation of the key measures adopted by trec_eval.exe for the experiments:

2.2.1. Relevant Document Return (rel_ret)

This is a general measure of how many relevant documents are returned. It is interpreted with figures on the number of retrieved documents, and the number of relevant documents (num_rel) in the collection. **Recall rate** = rel_ret/num_rel . All values are totals over all queries being evaluated.

2.2.2. Interpolated Recall - Precision Averages

Interpolated Recall - Precision Averages at 0.00, at 0.10 ..., at 1.00 measures precision (percent of retrieved documents that are relevant) at various recall levels (after a certain percentage of all the relevant documents for that query have been retrieved). "Interpolated" means that, for example,

precision at recall 0.10 (i.e., after 10% of relevant documents for a query have been retrieved) is taken to be MAXIMUM of precision at all recall points that is greater than or equal to 0.10. Values are averaged over all queries (for each of the 11 recall levels). These values are used for Recall-Precision graphs.

2.2.3. Mean Average Precision (non-interpolated)

MAP over all relevant docs. The precision is calculated after each relevant doc is retrieved. If a relevant doc is not retrieved, its precision is 0.0. All precision values are then averaged together to get a single number for the performance of a query. Conceptually this is the area underneath the recall-precision graph for the query. The values are then averaged over all queries.

2.2.4. Precision After X Documents

Precisions measured at 5 docs, at 10 docs ..., at 1000 docs. The precision (percent of retrieved docs that are relevant) after X documents (whether relevant or non-relevant) have been retrieved. Values averaged over all queries. If X docs were not retrieved for a query, then all missing docs are assumed to be non-relevant.

2.2.5. R-Precision

Precision after R measures precision after R documents have been retrieved, where R is the total number of relevant docs for a query, i.e. $R = num_rel$. Thus if a query has 40 relevant documents, then precision is measured after 40 documents, while if it has 600 relevant documents, precision is measured after 600 docs. This avoids some of the averaging problems of the “precision at X docs” values in (2.2.4) above. If R is greater than the number of docs retrieved for a query, then the non-retrieved documents are all assumed to be non-relevant.

CHAPTER 3

RESEARCH METHODS

3.1. Overview of Research Methods

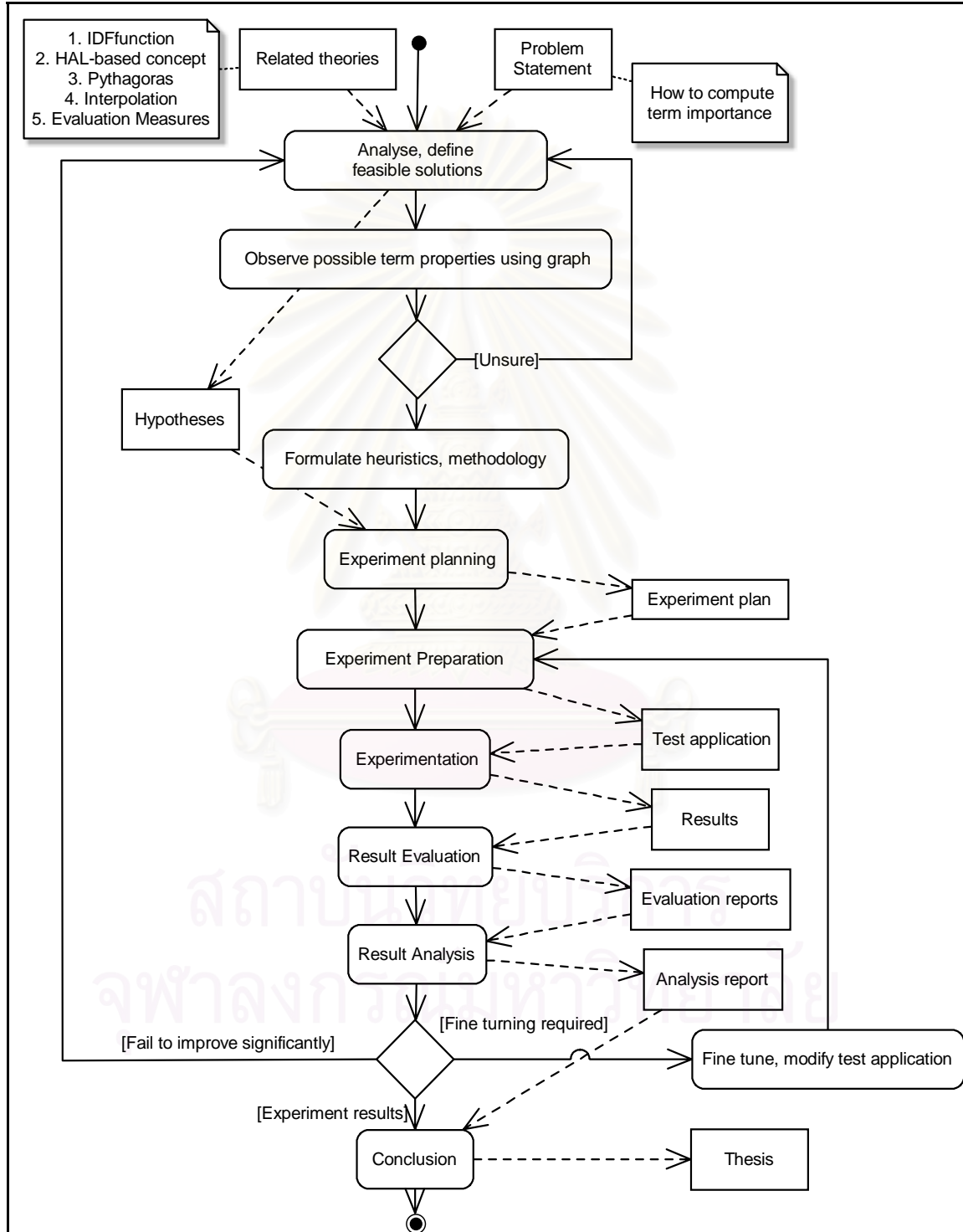


Figure 3.1 shows the overall picture of the research methods.

Formulating, testing and verifying a new approach is an iterative process. As shown in the above figure the research method starts with a problem statement and preliminary survey of related theories. In our case, the new approach is intuitively inspired by the general characteristics of language expression and refined through observations of charts of text. The thinking is that ranking term importance by sorting the *idf* weights of terms is one dimensional, ignoring another dimension of term positions in an expression of concepts. The charts, coupled with some imagination and background geometric theories, have been helpful in the formulation and reformulation process. Once the approach stabilizes, the next steps are to design and plan IR experiments, prepare the experimentation, conduct the experiments, evaluate and analyze the results. The steps continue to iterate until experimental results confirm the soundness of the approach.

The research methods are explained in more details in the following subsections.

3.2. Approach Formulation

3.2.1. Flow of thoughts & flow of words

Intuitively, a flow of ordered terms in a written sentence reflects the writer's flow of thoughts. When a writer wants to make a point, the chance is that he would choose a more elegant, less common word to emphasize his idea. Syntactical rules of a language also forces him to position supportive words in the neighborhood of the key word and when he pauses to make another important point, a different type of terms would be chosen to join two groups of words that represent two ideas.

If the semantic importance of each term can be somehow measured, the weights of the key terms will be highest, followed by those of the supportive terms and the pause words. In this study, the *idf* term weighting is adopted as an indirect way of measuring term importance.

The presence of the supportive terms depends on the existence of the key words that the former support. Hence, the key terms are "dependable words", the satellite and supportive terms are "dependant words". The pause words are actually the "stop words" in the information retrieval discipline.

Understanding a message is a reversed process. A reader notices the dependable words, the dependant terms and their relationship, and the least important pause words, to help him grasp the points conveyed.

It is this generalized way of communications with which this research approach the problem of capturing term dependency and measuring its importance in term of its contribution to the key concept.

Note that the *idf* term weighting is a global property of individual terms as averaged statistically from a document collection. The implications: it is static with respect to a corpus of documents; each term is independent of one another and so is its importance. The assumptions may not be realistic. A term is the basic unit of a language and a concept is explained by either a term or a

combination of terms. Thus the importance of a term is determined by the different roles that it plays in different contexts. In fact, its importance can be described by the degree to which the term contributes to the concept of a text message. It is this kind of term importance that is the subject of our study.

Taking the description part of topic 402 from TREC 8 as a sample, the flow of thoughts can be represented in the figure 3.3.

```

<top>
<num> Number: 402
<title> behavioral genetics
<desc> Description:
What is happening in the field of behavioral genetics, the study of the relative influence of genetic and environmental factors on an individual's behavior or personality?
<narr> Narrative:
Documents describing genetic or environmental factors relating to understanding and preventing substance abuse and addictions are relevant. Documents pertaining to attention deficit disorders tied in with genetics are also relevant, as are genetic disorders affecting hearing or muscles. The genome project is relevant when tied in with behavior disorders (i.e., mood disorders, Alzheimer's disease).
</top>

```

Figure 3.2 shows topic 402 of TREC 8. The description part is used in our example to plot a line chart.

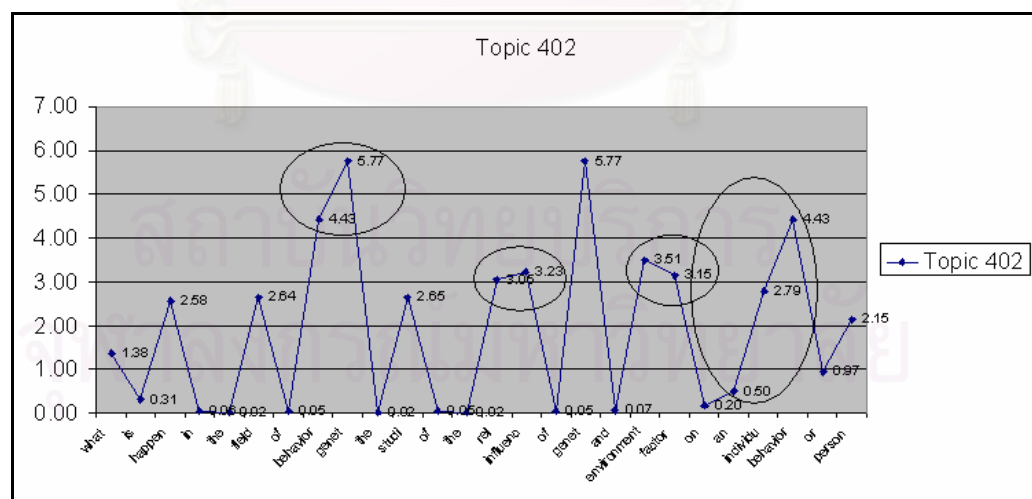


Figure 3.3 shows the line chart of a porter-stemmed version of topic 402 from TREC 8. The circled areas cover terms in the same group as the head words which make the peaks of the graph. The labeled figures are the IDF weights of the respective terms.

Without any knowledge on the term semantics, the following can be observed and derived from the graph:

Dependable words are represented by peaks. If a line is drawn across the graph, the obvious peaks in the upper part will be “genetics”, “influence”, “genetics”, “environment”, and “behavior”. The less obvious are “happen”, “field”, “study”, and “person”.

The circled areas show the dependant terms of some of the dependable words; “genetics”:{“behavioral”, “genetics”}, “influence”:{“relative”, “influence”}, “environment”:{“environment”, “factor”}, “behavior”:{“an”, “individual”, “behavior”}.

Stop words are in the set {“is”, “in”, “the”, “of”, “to”, “of”, “the”, “of”, “an”, “on”, “or”}.

Citing a porter-stemmed version of topic 401 as another shorter example, we plot the following graph. Similar conclusions can be drawn from it.

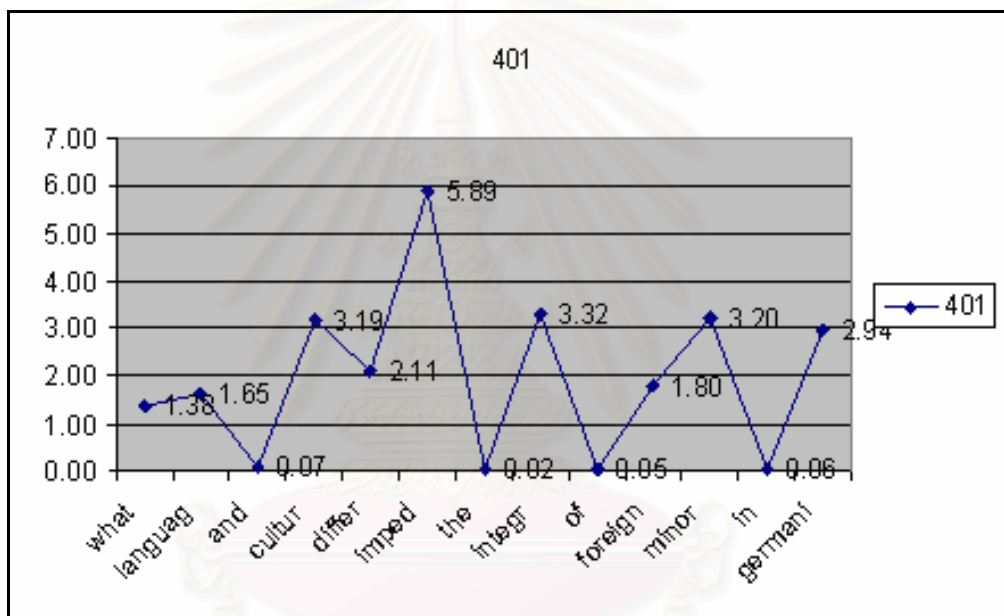


Figure 3.4 illustrates a line chart of a sentence: **What language and culture differences impede the integration of foreign minorities in Germany?** The terms have been stemmed by Porter algorithm.

Noticeably, the term “differ” (stemmed from “difference”) should not be classified as a stop word despite its bottom shape because its weight is too high for that. The criteria to determine a stop word is therefore multiple ones, the bottom shape and the weight of the term. In the case of the term differ, we may consider it a fluctuation and assume that it belongs to the peak word set “impede”:{“language”, “culture”, “difference”}.

In the following text, the study uses dependable word, dependable term, peak word, peak term, head word and head term interchangeably. Interchangeable words for dependant word are supportive words, supportive terms, satellite words and satellite terms. And stop words are sometimes called pause words in this thesis.

The above illustrations reveal three possible properties of a term: term weight, term dependency and term direction.

3.2.2. Term Weight

This study adopts the *idf* term-weighting scheme in measuring term importance. Other methods include the use of entropy (the Signal Noise-Ratio), and the Term Discrimination Value [1].

Let w be the *idf* weight of a term, we have $w = \log \frac{N}{n_i}$, where N is the number of all documents in a corpus and n_i is the number of documents where term i appears.

Since the weight is a property value derived from document collections, it is considered background knowledge in analyzing term importance in the local context.

3.2.3. Term Direction

Table 3.1 illustrates a table showing the IDF weights, the length of term links, and the direction property values of ordered terms in our example.

<Topic 401- TREC 8>	IDF Term Weight	Dependency Distance	Dependency Direction
"what"	1.38	6.73	3
"language" ("language")	1.65	5.82	4
"and"	0.07	6.54	3
"culture" ("culture")	3.19	3.36	4
"differ"("differences")	2.11	3.90	3
"imped" ("impede")	5.89	0.00	6
"the"	0.02	5.94	3
"integr"("integration")	3.32	2.57	6
"of"	0.05	6.17	3
"foreign"	1.80	4.21	1
"minor"("minorities")	3.20	2.68	6
"in"	0.06	5.91	3
"germani"("Germany")	2.94	3.56	4

A new local property is assigned to the link between two vertexes to denote the directions of terms.

The direction properties are:

1. UP shows that the vertex is heading up in its link to the next one. The enumerated value is assigned integer 1.
2. DOWN shows that the vertex is heading down. Its integer value is 2.
3. DOWNUP shows it is reversing its direction from down to up. Simply put, it is a bottom. Integer value is 3.
4. UPDOWN shows that it is a peak word, Integer value is 4.

5. PEAK is a special property value for the UPDOWN-tagged vertexes that stay above a threshold. They are considered to be the more prominent concepts. Integer value is 6.

3.2.4. Term Dependency/Term-Concept Contribution

In this study, the notion of term dependency is parallel to its contribution to the related concept. Visually, it can be thought to be the link between a peak at coordinating point (x, y) and that of the satellite term at coordinating point (a, b) on a graph. The dependency degree is inversely proportional to the distance, i.e. the shorter the link, the more important the dependant term is to the head word of the same set. The length of the link is influenced by the vertical weights of the two vertexes as well as their horizontal proximity.

The length is referred to as dependency distance. In consistency with our geometric representation, dependency distance calculation is motivated by the Pythagoras triangle rule.

Let w be the dependency distance, Δidf the difference of *idf* weights of two joining vertexes, and Δpos the horizontal distance between their positions, we have

$$w = \sqrt{\Delta idf^2 + \Delta pos^2} \quad (1)$$

Alternatively, the distance can be estimated by a function that linearly interpolates Δidf and Δpos :

$$w = \alpha \times \Delta pos + (1 - \alpha) \times \Delta idf \quad (2)$$

Where α is an adaptive parameter. Let *maxhorzlen* be the maximum x-axis distance between the peak vertex and the farthest term within the sphere of the peak influence, and *topweight* be the *idf* weight of the peak point, we have

$$\alpha = \text{maxhorzlen} / (\text{topweight} + \text{maxhorzlen}) \quad (3)$$

How to substitute *maxhorzlen* and *topweight* will be basically determined by the frame set on the terms under consideration. Implementation of (3) will be elaborated in *subsection 3.2.6*.

Equation (1) and equation (2) yield slightly different outputs. In our experimentation, we adopt (2) on the ground that we do not know the real relations between the term proximity and term weight factors. As a rule of thumb, the linear equation is assumed.

3.2.5. Heuristics Rules

This intuition-motivated graph representation has given rise to the following set of heuristic rules:

Rule 1 A sentence or phase is made of one or more set of terms. Each term set is represented by the most important term, the head word.

Rule 2 The term sets are separated from one another by stop words. Members of the same term set are associated to the head word. Each has its degree of dependency on the head word, measurable by the length of the distance from the head word to the member word. Four possible cases for a stop word:

Case I: It serves as words that join two concepts together,

Case II: It is just a pause word, an intermission in an expressed flow of thoughts,

Case III: It is a syntactical word required by grammatical rules but carries with it little or no semantics, and

Case IV: It is a member of a term set. This can happen in a broken expression or in cases where stop words are removed in a filtering mechanism.

For the first three cases, we can estimate that the boundary terms are negligible as we are interested in only their roles to separate one concept from another. Uncertainty arises in the fourth case, what can be assumed is that it can belong to either of the preceding or the next concept term set.

One criteria that can be used to guess (the very same manner a reader makes his guess) if the boundary term is a genuine one which is negligible (case I, case II, and case III in rule 4) or a term member of a concept group (case IV) is to see the weight of the term. For that, a threshold is required. The basis idea is that if the DOWNUP vertex is deep enough it should serve as a bound of a term group; otherwise it is a group member. A rule based on term weighting is required to handle the case.

Rule 3 A concept group may contain a number of smaller concept groups. Visualizing this scenario, one can notice that for a high hill in a graph, there are smaller hills on both sides of the sides of the highest one. These smaller hills with their respective peaks and bottoms can be considered errors or smaller concepts. In this study, they will be smoothed and are considered as just group members of the umbrella concept.

Applying the rules to our example, the results are as follows:

There are six hill tops, indicating six concepts. They are represented by “language”, “culture”, “impede”, “integration”, “minority” and “Germany”. By setting a threshold for the higher peaks, i.e. the more important concepts, one may obtain “impede”, “integration”, and “minority” as the peak terms of the outstanding concepts. The terms “language” and “culture” have their status approximated to simple members of the concept represented by “impede”. As it can be seen from the graph representation, all the other terms except boundary terms (stop words) belong to one term set or another.

The boundary words appear to be “what”, “and”, “differ”, “the”, “of” and “in”. Through an algorithm based on the weights of these terms, “differ” is reclassified as an ordinary group member. The list is thus shortened to {what, and, the, of, in}.

Now, we have three outstanding concepts represented by “impede”, “integration”, and “minorities”: “impede”: {“language”, “culture”, “difference”, “impede”}; “integration”: {“impede”}; and “minorities”: {“foreign”, “minorities”, “germany”}.

If the threshold of peaks is moved up to the effect that there is only one peak represented by “impede”, there will be two satellite groups, one on the left side and the other on the right. On the left side the set is comprised of {“language”, “culture”, “difference”}, and on the right side it is {“integration”, [“of”], “foreign”, “minorities”, [“in”], “germany”}.

3.2.6. Methodology: Probabilistic Form of Local Term Importance

We apply a probabilistic approach to the problem of measuring the contribution that a term makes to the key concept of the text. Here are the implementation steps:

Step 1: Determine Global Peak

Determine the key concept term, also referred to as global peak in the following text. The global peak is the term with the highest *idf* weight.

Step 2: Determine Local Peaks

Set a threshold to determine the peak terms or terms with PEAK direction state. In our implementation the threshold is set at $0.6 * \text{top } idf \text{ term weight}$ as the threshold for peak terms should weigh above $0.5 * \text{top } idf \text{ term weight}$. The UPDOWN terms above the threshold are re-classified as the PEAK ones. Except for the global peak, all the other PEAK terms are called local peaks.

Step 3: Compute Local Peak Contribution Strengths

We first set the window frame for all the PEAK terms, global or local. The baseline on the y-axis is moved up from zero to the minimum *idf* weight of the PEAK terms. The positions of the terms whose *idf* weights are below the baseline are dropped to the effect that the frame horizontal length shrinks from text length to text length minus the counts of non-PEAK terms.

For each local peak, calculate the probability of its importance (contribution strength) to the global peak, denoted by p . Since the probability p cannot be directly computed, we resort to computing its opposite probability

q , standing for the probability of its negligibility by the global peak. Then, we have

$$p = 1 - q \quad (4)$$

We resort to (2) to compute the dependency link. We also normalize the output w from (2) to ensure it falls in $[0,1]$. Hence, we have

$$p = 1 - \frac{w}{norm} \quad \text{where } norm \text{ is the normalization factor} \quad (5)$$

To compute w in (5), we define α in (2) as follows:

$$\alpha = \frac{orderedpeaks.size}{orderedpeaks.size + maxpeakweight - minpeakweight}, \quad \text{where}$$

$orderedpeaks.size$ is the number of peak terms, $maxpeakweight$ the idf weight of the global peak term and $minpeakweight$ the minimum local peak weight term.

We also substitute Δidf with the difference of idf weights between the global peak and the local peak under consideration, and substitute Δpos with the adjusted positional distance between the two terms. By adjustment, the positions of non-peak terms are not counted.

To compute $norm$ in (5), we use (2) and substitute Δidf and Δpos with the count of all the peak terms ($orderedpeaks.size$) and the difference between $max_peakweight$ and $min_peakweight$, i.e.

$$norm = \alpha * orderedpeaks.size() + (1 - \alpha) * (maxpeakweight - minpeakweight) \quad (6)$$

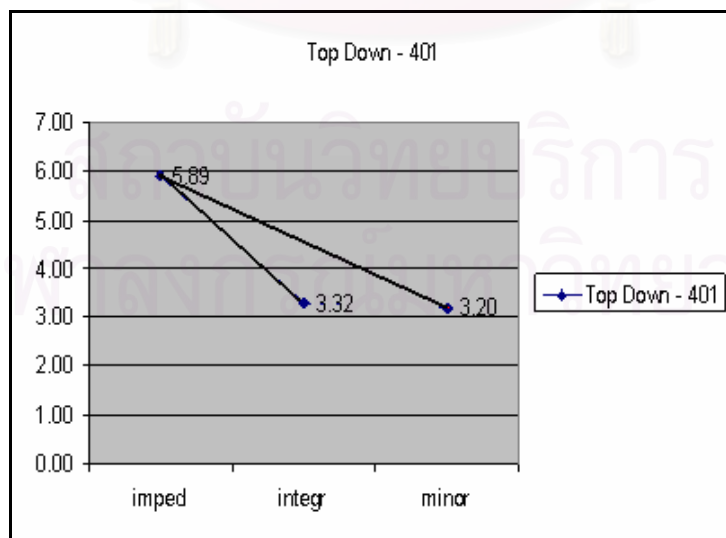


Figure 3.5 illustrates the line chart of peak words, removing satellite words from the graph to adjust the positions between the PEAK points.

Step 4: Group Terms into Frames

Terms are grouped by their proximate concept terms, i.e. local peaks. In our implementation, we identify the term with the lowest *idf* weight between two adjacent local peaks as the demarcation between term sets. The lower bound of a frame is the *idf* weight of local peak of the term set and the lower bound of all frames is set to be 1 in *idf* unit. Terms below the lower threshold is considered to be stop words and has zero probability. The horizontal lines of a frame are discounted by the number of the terms with zero-probability. The upper bound of a frame is determined by the maximum local peak weights which is equal to the global peak weight.

Step 5: Compute Term Contributions To Local Peak

By the same token as term weight computation in *Step 3*, (2), (4), and (5) are used to calculate member term contribution to the local peak of the same set.

The parameter α for a frame is substituted in (2) by *falpha* as follows:

$$falpha = \frac{winlen}{maxpeakweight - pausethreshold + winlen}, \text{ where } winlen \text{ is the}$$

count of terms in the term set minus the number of the members with zero probability, i.e. those below the lower bound, *maxpeakweight* is the top *idf* weight, and *pausethreshold* is the lower-bound threshold which is set to 1.

Substitution of Δidf is straightforward. Like the substitution of Δpos in *Step 3*, the distance has to be discounted. In this case, the positional distance is discounted by the count of zero-probability terms lying between the term and the local peak.

The normalization factor in this case is defined by the following equation:
 $f_{norm} = falpha * winlen + (1 - falpha) * (maxpeakweight - pausethreshold)$ (7)

We obtain the probability p by applying (5) and (6).

Step 6: Get Conditional Probability

The probability output from (4) and (5) are used as inputs to compute the probability of term importance in relation to the key concept term as follows:

Let p_{global} and p_{local} be the probability outputs from (4) and (5), we have the final probability p_{final} as follows:

$$p_{final} = p_{global} * p_{local} \quad (8)$$

In the subsequent sections, we describe experimentation objective, strategies, methods and environments to test the new term weighting scheme proposed in this research. They make crucial, integral parts of the research.

3.3. Experimentation Objective

The objective of the experiments is to show that queries for an ad hoc information retrieval would produce better retrieval results if they are re-weighted under the new approach towards dependency-related term importance.

3.4. Hypothesis

Query terms are normally weighted by the counts of each (the tf concept) by a retrieval engine. A retrieval engine assumes that the counts reflect the importance the user attaches to the respective terms. Our hypotheses are listed in the table below.

Table 3.2 outlines the hypotheses and test strategies.

	Hypothesis	Test Strategy
1	Each term carries with it different importance to the query concept and if estimated by the proposed $f(idf,dd)$ function, such term weights could yield a better results in the same ad hoc information retrieval experiment environments.	Compare retrieval results by unaltered, original queries with those by queries whose terms are weighted via the proposed scheme. In the former case, the weight is $tf*1$. In the latter case, the weight is $tf*computed\ weight$.
2	The term weights computed by our scheme is more effective in improving retrieval results than the idf term weighting because the former is a local property reflecting the strength of each term's contribution to the query concept while the latter is a global property from document collections.	Compare ad hoc information retrieval results by queries whose terms are tagged with the idf global weight, with those by the queries whose term weights are computed by the $f(idf,dd)$ function.

3.5. Experimental Setup

3.5.1. Document Collections

Three TREC collections are used for the experiments. They are combined to make a large database. The collections are:

- a. The Financial Times (FT) collection on disk 4,

- b. Foreign Broadcasting Information Services (FBIS) on disk 5,
- c. The Los Angeles Times news collection on disk 5

Details of the collections are shown in the following table.

Table 3.3 shows details of the three document collections

Collection	Number of Documents	Avg Doc Length (word)	Max Doc Length (word)
FBIS	130,471	516	139,709
FT	209,097	394	16,021
LA	131,896	505	24,653

3.5.2. Database

The three document collections are used to build index of a combined database. The index type is inverted index (term to document index). The database is also comprised of document-to-term index (dt). The indices support term positions. The database is built by *BuildIndex.exe* application which is bundled with *Lemur Toolkit*. The following table shows essential details of the test database.

Table 3.4 shows the details of *threedb* database.

Database Name	Threedb.ifp
Database Type	Inverted index
Document-term index size	0.98KB
Inverted index size	9.11MB
Stemming	Porter
Stop list	None
# of documents	348,503
# of terms	165,744,766
# of unique terms	407,659
Average document length	475

3.5.3. Query Sets

Only the medium-sized description part of the TREC 7 & TREC 8 query sets is used. TREC 7 query topics ranges from topic 351 to 400 and TREC 8 sets cover another 50 queries from topic 401 to 500. Each query is comprised of three parts: title, description and narration. The narration part is the longest of the three.

3.5.4. Toolkits

IR Development Toolkit

Lemur Toolkit Version 4.2 (please refer to *Appendix E*) is used for use and development for the experiment. The toolkit provides development platform on both Linux and Windows. This project uses

the Windows-platform version of its codes, the console application development options using C++, and the standalone application as against another option of experimenting it on Web platform.

Evaluation

Evaluation is done by `trec_eval.exe`, a standalone application distributed free of charge by NIST to IR research community. The application fully supports TREC formats.

3.6. Experimentation Steps

Five steps are involved in the experimentation of the *idf-dd* weighting scheme:

3.6.1. Prepare Test Database:

Objective: To build test database

Procedures: Parse document collections to build an inverted index files for the database. Porter stemming is used to reduce the database index files. No stop word list is used.

Input: Three document collections, Financial Times, LA Times, and Foreign Broadcasting Service (FBIS), from *disk 4*, *disk 5*.

Output: *threedb* test database

Application: `BuildIndex.exe`, an application provided under the Lemur Toolkit.

3.6.2. Prepare queries:

Objective: To prepare queries for ad hoc information retrieval operations

Procedures: Parse query topics 401-450 with Porter stemming. A TREC query topic is divided into three parts: topic, description and narration. Only the description part is used in the experiments. The parsed queries are saved to a file.

Input: Query topics (TREC 8) items 401-450

Output: A file saved with Porter-stemmed queries.

Application: A parsing application developed within the Lemur development framework, for this research.

3.6.3. Compute *idf-dd* weights for query terms:

Objective: To compute and assign query term weights.

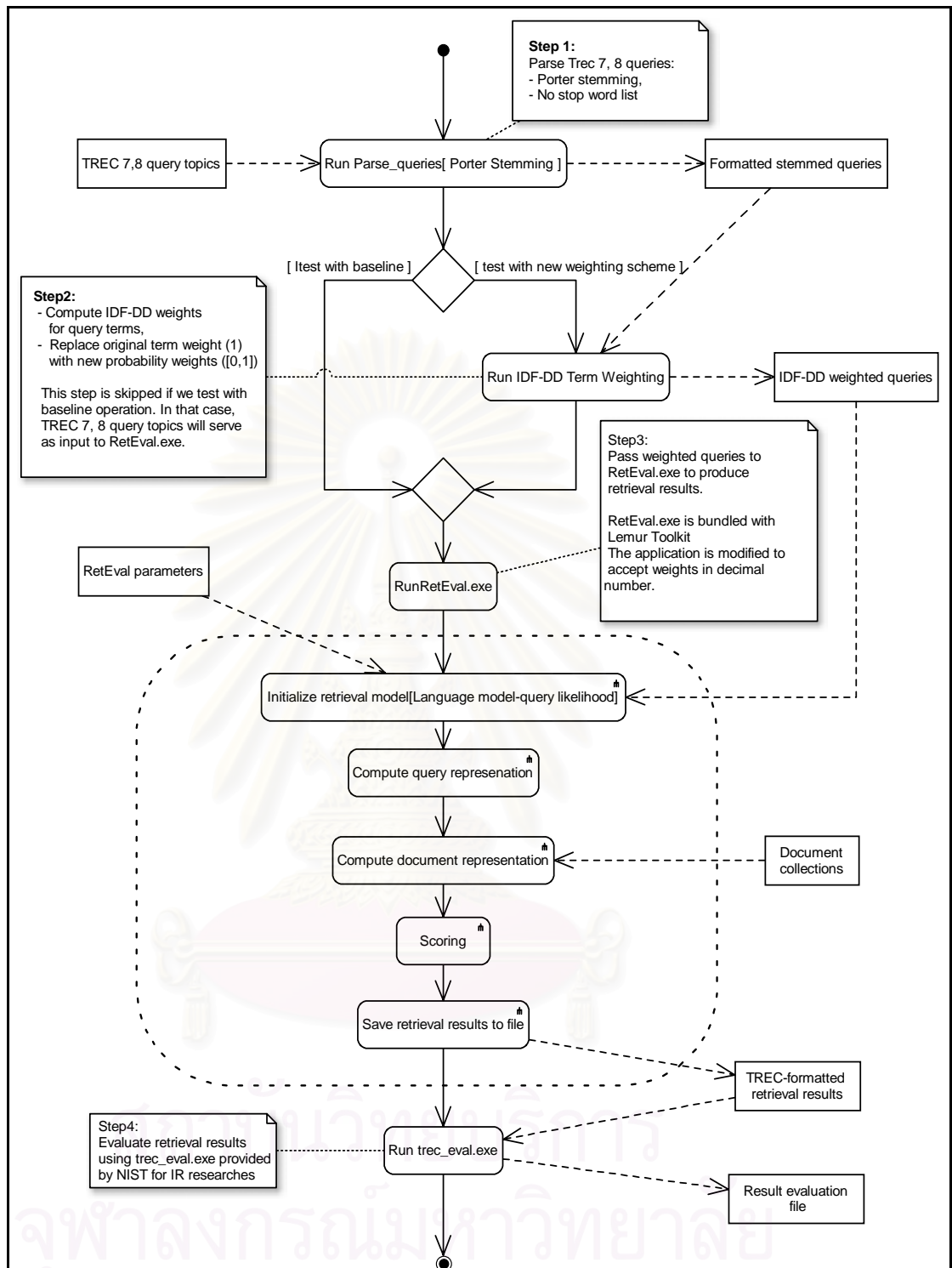


Figure 3.6: Diagram shows the four steps in the experimentation of the new *idf-dd* weighting effects on ad hoc information retrievals.

Procedures: Run *TestTermWeightApp.exe*, an application developed in the research to compute $f(idf,dd)$ term weights for query terms. The input is the parsed queries from Step 2. The probability weights replace the value 1 for each term. Term order is retained. (See *Appendix F* for more implementation details.)

Input: Formatted Porter-stemmed queries

Output: A file of queries with terms weights computed by $f(idf, dd)$

Application: *TestTermWeightApp.exe* developed for this research

3.6.4. Pass weighted queries to retrieval engine:

Objective: To retrieve documents from the retrieval engine.

Procedures: Run *RetEval.exe* provided by the Lemur Toolkit. The application is modified to accept weights in double. The application will call Lemur libraries to initialize the retrieval method class, and the specific language model according to a user-defined parameter file.

Input: A file of queries with *idf-dd* term weights, a parameter file specifying that language model retrieval engine with query likelihood scoring, *Dirichlet prior* smoothing technique in interpolation mode will be used. Result set is pre-set at 1,000, Dirichlet Prior parameter set at default 1,000. In Lemur implementation, the query likelihood scoring method is an option of the KL-Divergence retrieval engine.

Output: A file containing retrieval results in TREC format

Application: A modified version of *RetEval.exe*

```
<parameters>
<retModel>kl</retModel>
<index>C:\TrecDatabase\ThreeDB\threedb.ifp</index>
<textQuery>C:\TrecDatabase\Bin\workspace\testrun8nostops.qry</textQuery>
<resultFile>C:\TrecDatabase\Bin\RESULTTESTRUN8</resultFile>
<resultFormat>trec</resultFormat>
<resultCount>1000</resultCount>
<useWorkingSet>0</useWorkingSet>
<workingSetFile>D:\TrecD4\FT\worksetfile.txt</workingSetFile>
<feedbackDocCount>0</feedbackDocCount>
<feedbackTermCount>0</feedbackTermCount>
<smoothSupportFile>C:\TrecDatabase\Bin\smoothtext.txt</smoothSupportFile>
<smoothMethod>dir</smoothMethod>
<smoothStrategy>interpolate</smoothStrategy>
<adjustedScoreMethod>q1</adjustedScoreMethod>
<JelinekMercerLambda>0.5</JelinekMercerLambda>
<DirichletPrior>1000</DirichletPrior>
<discountDelta>0.7</discountDelta>
<queryUpdateMethod>rm2</queryUpdateMethod>
<feedbackCoefficient>0</feedbackCoefficient>
<feedbackTermCount>10</feedbackTermCount>
<feedbackProbThresh>0.001</feedbackProbThresh>
<feedbackProbSumThresh>1</feedbackProbSumThresh>
<feedbackMixtureNoise>0.5</feedbackMixtureNoise>
<emIterations>0</emIterations>
</parameters>
```

Figure 3.7: The parameter file used by *RetEval.exe*. The *testrun8nostops.qry* input file contains query terms weighted by the new weighting scheme.

3.6.5. Evaluate retrieval results:

Objective: To evaluate retrieval results by comparing outputs to expert choices.

Procedures: Run the *trec_eval.exe* application provided by NIST to IR research community. The application will compute key measurement criteria including the mean average precision, R-precision and the number of relevant documents returned by the queries.

Input: A file containing retrieval results in TREC format

Output: A file detailing the evaluation of retrieval results, also in TREC format.

Application: *trec_eval.exe*



CHAPTER 4

EXPERIMENTAL RESULTS AND EVALUATION

4.1. Evaluation of Experimental Results

Evaluation by *trec_eval.exe* program of experimental runs shows significant performance improvements of the ad hoc information retrieval conducted within the framework of the query likelihood ranking method of the language model when the proposed $f(idf, dd)$ weighting scheme is applied to the input queries from TREC 7 and TREC 8 sets of queries. The interpolation version of weighting functions is adopted in the experimentation (equation (2) in *Chapter 3*).

The overall retrieval results with the new weighting scheme beat the baseline results by all key measurements, namely the number of relevant documents returned, the mean average precision (MAP), the R-precision, the interpolated recall-precision averages, and the precision after X documents.

For TREC 7 and TREC 8 query sets respectively, map increases strongly by 16.12% and 15.74%, R-precision 12.52% and 8.40%, number of relevant document return -1.62% and 9.14%. Measurements by Interpolated recall-averages precision and precision after X docs are also in the black. The results are shown in the following table.

Table 4.1 shows the evaluated results of experimental runs using the $f(idf, dd)$ weighting scheme with TREC 7 and TREC 8 query sets. The evaluation computation is done by *trec_eval.exe* program.

Measurements	TREC 7	TREC 8
Relevant doc return	-1.62	9.14
Mean average precision	16.12	15.74
R-precision	12.52	8.40
ircl_prn.0.00	0.04	1.87
ircl_prn.0.10	17.04	4.61
ircl_prn.0.20	7.30	17.80
ircl_prn.0.30	8.97	22.03
ircl_prn.0.40	24.06	14.53
ircl_prn.0.50	37.21	19.70
ircl_prn.0.60	41.04	50.25
ircl_prn.0.70	100.77	54.62
ircl_prn.0.80	59.38	32.41
ircl_prn.0.90	-25.93	33.51
ircl_prn.1.00	N.A.	20.83
P5	6.06	12.50
P10	7.69	9.29
P15	1.68	4.43
P20	6.45	4.32
P30	7.34	4.72
P100	4.12	2.19
P200	0.41	3.52
P500	-0.56	10.30
P1000	-1.56	9.19

By way of counting the number of positive, negative and neutral results by query, it can also be shown that the new scheme weighting has produced a higher percentage of gains vis-à-vis loss. The break-down of the counts are shown in the table in figure 15. Note that combined counts are based on the following criteria:

1. Positive Counts: It is so defined if the result is positive by at least one of the three measurements and there must be no negative measurements at all,
2. Negative Counts: If the result is negative by at least one measurement standard and there must be no positive measurements at all,
3. Neutral Counts: If the results are mixed or neutral by any measurements or both.

The pie chart in figure 4.1 illustrates the break-down of the combined results for TREC 7 and TREC 8 query sets. The total number of queries used is 100.

Table 4.2 shows the break-down of the number of result counts on three measurement criteria and their combination for 100 query sets (ALL), TREC 7 query set (50), and TREC 8 query set (50).

ALL			
<i>Measurement</i>	<i>Positive</i>	<i>Negative</i>	<i>Neutral</i>
Relevant doc return	54	30	16
Mean average precision	62	38	0
R-precision	51	31	18
Combined Counts	57	27	16
TREC 7			
Relevant doc return	27	18	5
Mean average precision	30	20	0
R-precision	25	16	9
Combined Counts	26	14	10
TREC 8			
Relevant doc return	27	12	11
Mean average precision	32	18	0
R-precision	26	15	9
Combined Counts	31	13	6

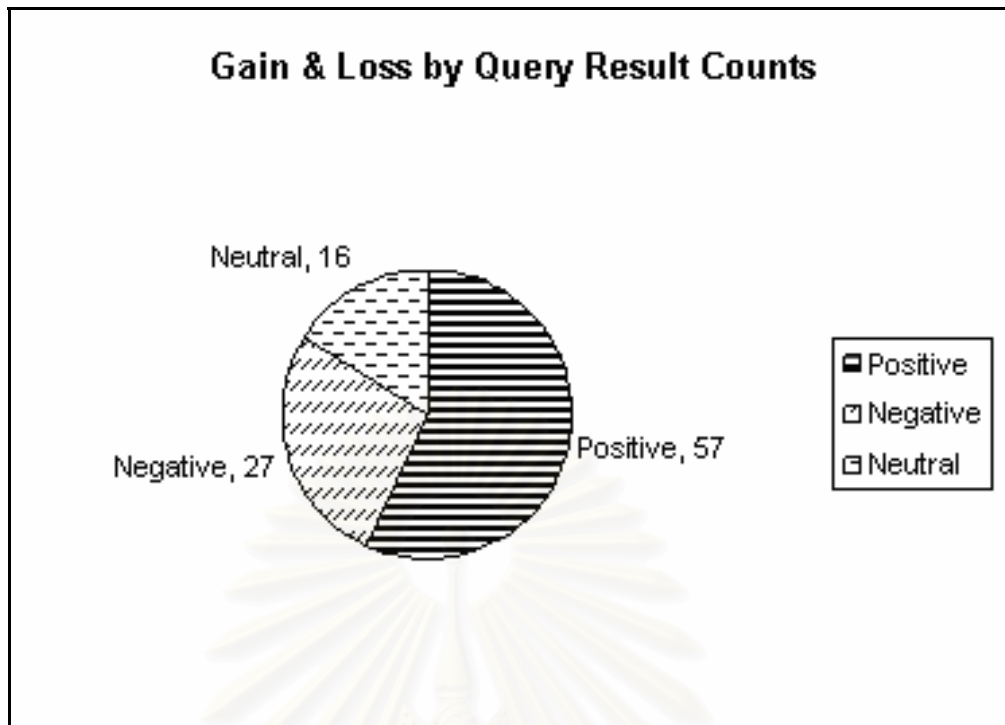


Figure 4.1: Pie chart showing that out of 100 query topics (TREC 7 & TREC 8 combined), 57 yield positive results, 27 negative and the rest 16 neutral.

4.2. Comparison with Existing Standards

As it is known, the *idf* weight computing scheme is the current standard in IR. In language modeling approach to IR, however, the *idf* computation gives way to probabilistic computation that makes use of only the term counts (as an input to the calculation of probability). Given that, the $f(idf,dd)$ implementation is run against the baseline with each query term weight equal to 1 unit for a term, and the *idf* weighted implementation which re-weights the terms in accordance with the global property derived from document collection.

As expected, the comparison shows that the $f(idf,dd)$ weighting scheme outperforms both the baseline and the *idf*-weighting significantly, confirming the experimental hypothesis (refer to *Chapter 3 on Research Methods*). The results are shown in table 4.3, 4.4, 4.5 and 4.6.

Table 4.3 compares the $f(idf,dd)$ results to those of the *idf*-based results for 50 queries from TREC 7 set. Table 4.4 makes the same comparison for TREC 8 query set. Table 4.5 and table 4.6 compares the *idf*-based and the $f(idf,dd)$ results against the baseline results with term weight unit equal to 1 for each term. The comparison in table 4.5 is based on TREC 7 query set while table 4.6 is based on TREC 8 query set.

Figure 4.2 (a) compares the interpolated recall-precision averages of the baseline, the *idf*-based and the $f(idf,dd)$ retrieval results for TREC 8, and (b) which makes comparisons by R-precision standard.

Table 4.3 shows information retrieval performance increase from using $f(idf,dd)$ weighting. The baseline in this case is the results from idf weighting scheme. Query set is from TREC 7.

Standard Measure	TREC 7 Query	IDF-based Probability	IDF-DD Interpolation	
		Scores	Scores	%Change
Num_q	351-400	50.00	50.00	0.00
Num_ret	351-400	50,000.00	50,000.00	0.00
Num_rel	351-400	4,226.00	4,226.00	0.00
Num_rel_ret	351-400	1,420.00	1,576.00	10.98592
Map	351-400	0.14	0.15	9.10
R-prec	351-400	0.19	0.20	7.21
ircl_prn.0.00	351-400	0.63	0.67	7.79
ircl_prn.0.10	351-400	0.35	0.38	11.35
ircl_prn.0.20	351-400	0.24	0.26	7.75
ircl_prn.0.30	351-400	0.17	0.18	7.25
ircl_prn.0.40	351-400	0.13	0.15	8.72
ircl_prn.0.50	351-400	0.09	0.11	12.29
ircl_prn.0.60	351-400	0.08	0.08	6.29
ircl_prn.0.70	351-400	0.05	0.05	-0.19
ircl_prn.0.80	351-400	0.03	0.03	-6.93
ircl_prn.0.90	351-400	0.01	0.01	0.00
ircl_prn.1.00	351-400	0.00	0.00	N.A.
P5	351-400	0.35	0.42	20.69
P10	351-400	0.32	0.36	13.75
P15	351-400	0.29	0.32	10.08
P20	351-400	0.27	0.30	8.39
P30	351-400	0.23	0.26	12.52
P100	351-400	0.13	0.15	9.57
P200	351-400	0.09	0.10	6.37
P500	351-400	0.05	0.05	6.65
P1000	351-400	0.03	0.03	10.92

Table 4.4 shows information retrieval performance increase from using $f(idf,dd)$ weighting. The baseline in this case is the results from idf weighting scheme. Query set is from TREC 8.

Standard Measure	TREC 8	IDF-based Probability	IDF-DD Interpolation	
	Query Topic	Scores	Scores	%Chg
Num_q	401-450	50.00	50.00	0.00
Num_ret	401-450	49,127.00	50,000.00	1.78
Num_rel	401-450	4,522.00	4,522.00	0.00
Num_rel_ret	401-450	1,701.00	1,959.00	15.17
Map	401-450	0.17	0.20	13.34
R-prec	401-450	0.22	0.24	10.86
ircl_prn.0.00	401-450	0.59	0.71	19.93
ircl_prn.0.10	401-450	0.40	0.46	12.89
ircl_prn.0.20	401-450	0.31	0.35	14.10
ircl_prn.0.30	401-450	0.25	0.29	16.40
ircl_prn.0.40	401-450	0.18	0.20	12.98
ircl_prn.0.50	401-450	0.14	0.16	11.51
ircl_prn.0.60	401-450	0.11	0.12	8.52
ircl_prn.0.70	401-450	0.07	0.08	3.77
ircl_prn.0.80	401-450	0.05	0.05	5.44
ircl_prn.0.90	401-450	0.02	0.03	40.11
ircl_prn.1.00	401-450	0.02	0.02	9.43
P5	401-450	0.38	0.47	24.47
P10	401-450	0.34	0.40	18.34
P15	401-450	0.29	0.35	17.65
P20	401-450	0.28	0.31	12.14
P30	401-450	0.23	0.27	13.34
P100	401-450	0.14	0.16	13.73
P200	401-450	0.10	0.11	12.14
P500	401-450	0.06	0.07	15.37
P1000	401-450	0.03	0.04	15.29

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

Table 4.5 shows comparative results on TREC 7

Standard Measure	T 7	Baseline	IDF Probability		IDF-DD Interpolation	
	Query	Scores	Scores	%Chg	Scores	%Chg
num_q	351-400	50	50.00	0.00	50.00	0.00
num_ret	351-400	50,000	50,000.00	0.00	50,000.00	0.00
num_rel	351-400	4,226	4,226.00	0.00	4,226.00	0.00
num_rel_ret	351-400	1,602	1,420.00	-11.36	1,576.00	-1.62
map	351-400	0.13	0.14	6.43	0.15	16.12
R-prec	351-400	0.18	0.19	4.95	0.20	12.52
ircl_prn.0.00	351-400	0.67	0.63	-7.19	0.67	0.04
ircl_prn.0.10	351-400	0.33	0.35	5.11	0.38	17.04
ircl_prn.0.20	351-400	0.24	0.24	-0.41	0.26	7.30
ircl_prn.0.30	351-400	0.17	0.17	1.60	0.18	8.97
ircl_prn.0.40	351-400	0.12	0.13	14.12	0.15	24.06
ircl_prn.0.50	351-400	0.08	0.09	22.19	0.11	37.21
ircl_prn.0.60	351-400	0.06	0.08	32.70	0.08	41.04
ircl_prn.0.70	351-400	0.03	0.05	101.16	0.05	100.77
ircl_prn.0.80	351-400	0.02	0.03	71.25	0.03	59.38
ircl_prn.0.90	351-400	0.01	0.01	-25.93	0.01	-25.93
ircl_prn.1.00	351-400	0.00	0.00	N.A.	0.00	N.A.
P5	351-400	0.40	0.35	-12.12	0.42	6.06
P10	351-400	0.34	0.32	-5.33	0.36	7.69
P15	351-400	0.31	0.29	-7.63	0.32	1.68
P20	351-400	0.28	0.27	-1.79	0.30	6.45
P30	351-400	0.25	0.23	-4.61	0.26	7.34
P100	351-400	0.14	0.13	-4.97	0.15	4.12
P200	351-400	0.10	0.09	-5.61	0.10	0.41
P500	351-400	0.05	0.05	-6.77	0.05	-0.56
P1000	351-400	0.03	0.03	-11.25	0.03	-1.56

Table 4.6 shows retrieval performances from using *idf*-based and *f(idf,dd)* weighting against the baseline result compared on TREC 8.

Standard Measure	T 8	Baseline	IDF Probability		IDF-DD Interpolation	
	Query	Scores	Scores	%Chg	Scores	%Chg
num_q	401-450	50	50	0.00	50.00	0.00
num_ret	401-450	50,000	49127	-1.75	50,000.00	0.00
num_rel	401-450	4,522	4522	0.00	4,522.00	0.00
num_rel_ret	401-450	1,795	1701	-5.24	1,959.00	9.14
map	401-450	0.17	0.1732	2.12	0.20	15.74
R-prec	401-450	0.23	0.2201	-2.22	0.24	8.40
ircl_prn.0.00	401-450	0.69	0.5892	-15.05	0.71	1.87
ircl_prn.0.10	401-450	0.44	0.4041	-7.34	0.46	4.61
ircl_prn.0.20	401-450	0.30	0.3086	3.25	0.35	17.80
ircl_prn.0.30	401-450	0.23	0.2451	4.83	0.29	22.03
ircl_prn.0.40	401-450	0.17	0.1772	1.37	0.20	14.53
ircl_prn.0.50	401-450	0.13	0.1433	7.34	0.16	19.70
ircl_prn.0.60	401-450	0.08	0.1091	38.45	0.12	50.25
ircl_prn.0.70	401-450	0.05	0.0742	49.00	0.08	54.62
ircl_prn.0.80	401-450	0.04	0.0496	25.57	0.05	32.41
ircl_prn.0.90	401-450	0.02	0.0182	-4.71	0.03	33.51
ircl_prn.1.00	401-450	0.01	0.0159	10.42	0.02	20.83
P5	401-450	0.42	0.376	-9.62	0.47	12.50
P10	401-450	0.37	0.338	-7.65	0.40	9.29
P15	401-450	0.33	0.2947	-11.23	0.35	4.43
P20	401-450	0.30	0.28	-6.98	0.31	4.32
P30	401-450	0.25	0.2347	-7.60	0.27	4.72
P100	401-450	0.16	0.1398	-10.15	0.16	2.19
P200	401-450	0.11	0.0997	-7.69	0.11	3.52
P500	401-450	0.06	0.0566	-4.39	0.07	10.30
P1000	401-450	0.04	0.034	-5.29	0.04	9.19

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

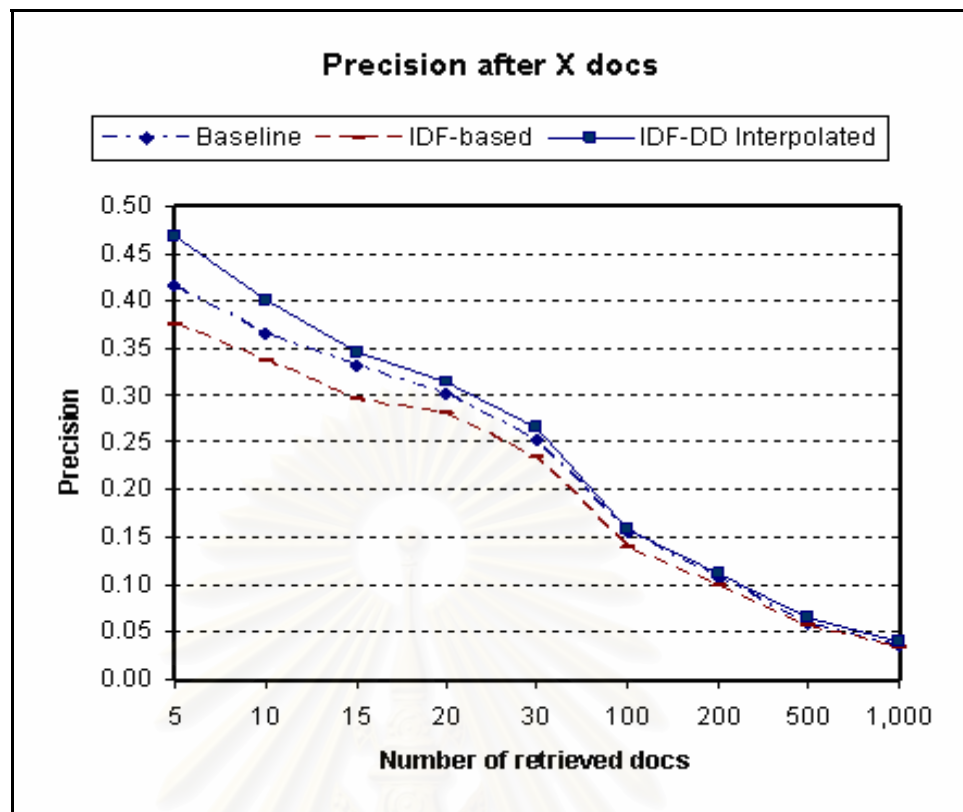
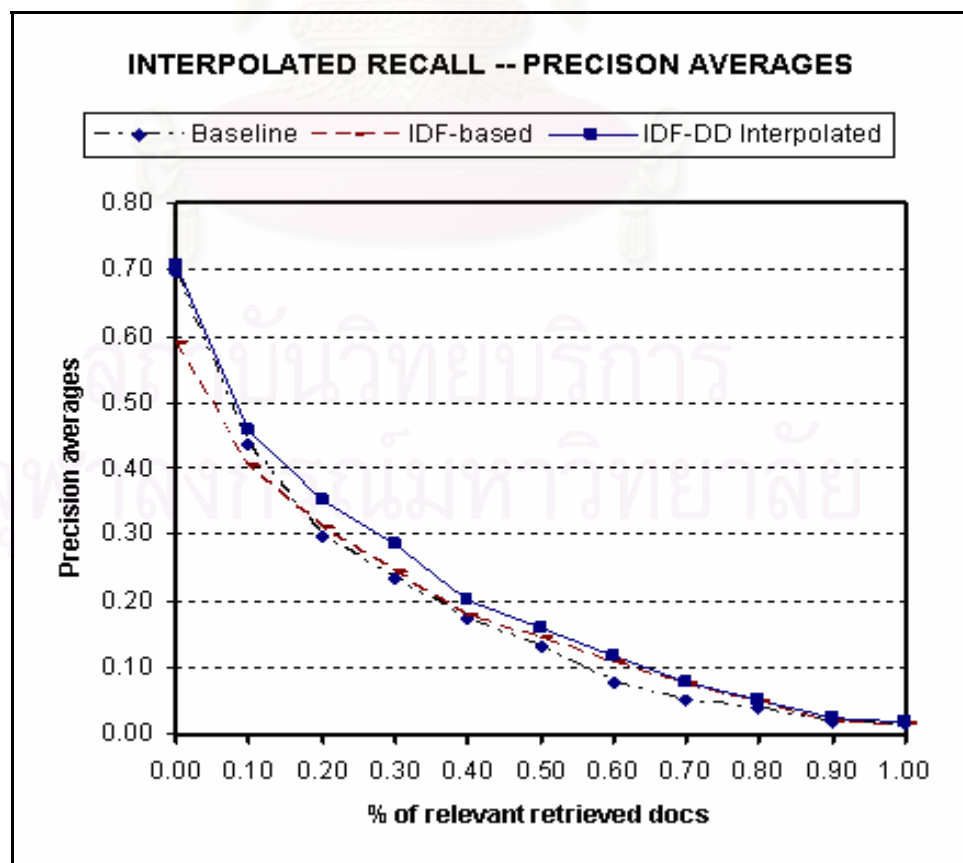


Figure 4.2 (a), above, shows a graph comparing R-precisions of baseline, *idf*, and *f(idf,dd)* results, and (b), below, shows comparisons by Recall-Precision Averages. Queries used from TREC 8.



4.3. Result Analysis

The experimental results as evaluated by IR standards show the following points:

1. The $f(idf,dd)$ approach performs exceptionally well in the area of average precisions. MAP jumps by 16.12% and 15.74% over the baseline for TREC 7 and TREC 8 respectively. R-precisions also improved significantly, 12.52% and 8.40% for TREC 7 and TREC 8 respectively. In contrast, the number of relevant document returns failed to improve consistently in the TREC 7 and TREC 8 runs. The improvement percentage is greater for TREC 8, at 9.14%, and is negative at -1.62% for TREC 7. The number of relevant document returns is indicative of the recall performance of a retrieval engine. In modern IR, precisions are more important than recall as it is difficult for users to sense recall performance than to evaluate precisions.
2. Given the chart in figure 4.2(a), the Precision After X for TREC 8 strongly improves in the initial stages of receiving documents: 12.50% after the first five documents returned, followed by 9.29% after 10 returned documents, 4.43 % after 15 returns, 4.32% after 20 and 4.72% after 30. Similar situation can be seen from TREC 7, although less obvious.

The results show that the $f(idf,dd)$ weighting does improve the precisions for the first groups of documents returned, which is a desirable situation for ad hoc information retrieval.

Table 4.7 shows the R-precision after X number of returned documents for TREC 8 query set.

After X docs	Query topics	Precision
P5	401-450	12.50
P10	401-450	9.29
P15	401-450	4.43
P20	401-450	4.32
P30	401-450	4.72
P100	401-450	2.19
P200	401-450	3.52
P500	401-450	10.30
P1000	401-450	9.19

Evaluation by the interpolated recall-precision averages cannot be used to analyze the particular point as its computation will always lead to higher percentage increase at its following stage. (Refer to *Evaluation Measures in Chapter 2* for explanation).

3. Given figure 4.1 illustrating a pie chart of gains and losses counted by TREC 7-TREC8 queries, 57 out of 100 weighted queries result in better

performance, 27 in negative changes, and 16 mixed. The factors that may affect the efficiency of the $f(idf, dd)$ efficiency and caused the negative results are as follows:

- a. The scheme counts on the *idf* weight as a global property input in its function. Hence, its efficiency is determined to a certain extent by the *idf* weighting. As it is generally known, the *idf* is just an indicator of term importance. By its definition, the *idf* weight is simply a figure showing how often it appears in a document collection. No more, no less. As mentioned in the early chapter, the *idf* weight is adopted because it is the *de facto* standard for term weighting but this can be replaced by another term weighting in our model.
- b. Stemming may affect the term weights. Stemming is primarily aimed at reducing the index size of a document database but one of its side effects is to distort the weight of some terms. For instances, “us” is the stemmed word from both “useful” and “US” and “am” is the stemmed word for “(I) am” and “(PAN) AM”. In the experiments for this thesis, porter stemming is used as is. There are no other mechanisms to distinguished capitalized letters or words from lower case terms. Abbreviations are also treated as simple terms.
- c. The weighting of terms is done on the query part in the experiments. Although this is sufficient in proving the hypothesis based on the assumption that weighting of document terms are transparent, the full capacity of the $f(idf, dd)$ term weighting would likely be demonstrated if and when the approach is also applied to the computation of document representation which is specific to different IR models. Applying the term weighting scheme on the document representation side, is beyond the objective and scope of the research.

CHAPTER 5

CONCLUSION AND COMMENTS

5.1. Conclusion

The master thesis proposes a novel approach that the importance of a term in a sentence is determined by the level of its contribution to the text concept, also represented by a term. The level can also be seen as a kind of relations between depending terms and the concept term.

We compute the term importance by first compute a $w = \alpha * \Delta pos + (1 - \alpha) * \Delta idf$ function which measures the term negligibility. The input function Δidf is the unsigned difference between the idf weight of the term and its reference concept term and the Δpos function computes a dependency distance between them. Like idf , the former is a global term weighting property and the latter a local property measured from an adjusted positional distance between the term and the respective reference concept. For the sake of convenience, the inverse function of $w = \alpha * \Delta pos + (1 - \alpha) * \Delta idf$ will be referred to as $f(idf, dd)$.

Based on a graphical illustration of dependency links, the thesis proposes an interpolated function to compute the localized term weights, $w = \alpha * \Delta pos + (1 - \alpha) * \Delta idf$ where Δpos is the adjusted term position distance and Δidf is the difference between the term weight and the weight of its reference concept term. The term weight input is a global property and is interpreted to be idf term weight in this research. The computed value is then transformed into a probability format using the notion $p = 1 - q$ where p is the probability of the term contribution to the key concept and q is the opposite probability that the term can be neglected for its great distance from the concept term. The q is computed from w with normalization. In the last step, conditional probability is employed to handle cases where there are multiple concept terms, the most important of which is classified as global concept term (global in the context of the particular text).

To our knowledge, the proposed $f(idf, dd)$ function and its underlining approach is novel in that term dependency is re-interpreted, term importance weights are extracted as local property, and that the computation requires only marginal costs with $O(mn)$ complexity, where m is the number of iterations required, and n is the length of the text.

The new approach is implemented in an experimentation design to test if query terms once weighted under the $f(idf, dd)$ scheme will yield better ad hoc information retrieval results than normal query terms. In the experimentation, three cases are tested:

1. Query terms are passed as they are to retrieval engine. In practice, query terms carry the equal weight of 1. This is the baseline design.
2. Query terms are first weighted by the idf method before they are passed to the retrieval engine. In practice, we are passing the weight as a global

property (the *idf* is statistically derived from a document collection) to the retrieval engine.

3. Query terms are first weighted by the $f(idf,dd)$ which localizes the *idf* weights, before they are passed to the retrieval engine.

For all three scenarios, the *tf* is implicitly retained by the number of terms sent. (i.e., some terms appear once each while the others appear more frequently in the query.). How the retrieval engine computes document representation from document and collection terms is assumed to be transparent in this experimentation.

The experiments are conducted within the language modeling framework using query likelihood scoring method and *Dirichlet* prior smoothing technique. They produce convincing gains for the new approach compared to the baseline and the *idf*-based results. Improvements are all-round significant given the results of all essential evaluation standards and are particularly outstanding in the precision area. Using TREC 7 and TREC 8 query sets, the experiments report a 16.12% and 15.74% increases in mean average precision (MAP) respectively. The $f(idf,dd)$ function also outperforms the *idf*-based scheme by 9.10%, and 13.34% for TREC 7 and TREC 8 query sets respectively.

5.2. Comments

This research work is not about refining query terms to improve ad hoc information results. Refining term weights employed in the experimentation is just a mean to prove the new proposition. Rather, this work is about a novel approach to compute term importance from the perspective that firstly, individual terms are basic language units and a tool to constitute to concepts. Secondly, terms are related to serve the purpose of clarifying the points to an audience. Thirdly, term importance, and hence its term weight, can be viewed in the said context. Fourthly, term importance should be localized as a term can carry different importance depending on the role it plays in a text, and fifthly, this local property should come into play in IR. The outcome of the work, however, can be extended to use in other IR applications.

5.3. Recommendations

Extending IR models to support the $f(idf,dd)$ term weighting. The assumption is that retrieval performance should be further enhanced if the scheme is integrated into the models from the level of preprocessing all the way to the computation of document representation and query/document ranking.

REFERENCES

1. Salton, G. and McGill, M. J. Introduction to Modern Information Retrieval, 1st edition. McGraw-Hill, New York, 1983
2. Barwise, J. and Seligman, J. Information Flow: The Logic of Distributed Systems, Cambridge Tracts in Theoretical Computer Science Series. Cambridge University Press, UK, 1997
3. Nanas, N., Uren, V., and Roeck, A.D. Building and Applying a Concept Hierarchy Representation of a User Profile. Annual ACM Conference on Research and Development in Information Retrieval, Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 2003: 198-204.
4. S. Robertson, S. Walker, S. Jones, M.Hancock Beaulieu, and M. Gatford. Okapi at TREC 3. NIST Special Publication 500-226: Overview of the Third Text Retrieval Conference (TREC-3), 3rd Annual Text Retrieval Conference, NIST - Gaithersburg, Maryland, 1994: 109-126.
5. Possas, B., Ziviani, N., Meira Jr., W., Ziviani, N., Ribeiro-Neto, B. Set-Based Model: A New Approach for Information Retrieval. Annual ACM Conference on Research and Development in Information Retrieval, Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 2002: pp 230-237.
6. Kim, Hee-soo, Choi, I., Kim, M. Refining Term Weights of Documents Using Term Dependencies. Annual ACM Conference on Research and Development in Information Retrieval, Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 2004: 552-553.
7. Pickens, J., MacFarlane, A. Term Context Models for Information Retrieval. Conference on Information and Knowledge Management, Proceedings of the 15th ACM international conference on Information and knowledge management, 2006: 559-566.
8. Song, F. and Croft, B. A General Language Model for Information Retrieval. Conference on Information and Knowledge Management, Proceedings of the Eighth International Conference on Information and Knowledge Management, 1999: 316-321.
9. Srikanth, M. and Srikanth, R. Bitern Language Models for Document Retrieval. Annual ACM Conference on Research and Development in Information Retrieval, Proceedings of the 25th Annual

- International ACM SIGIR Conference on Research and Development in Information Retrieval, 2002: 425-426.
10. Gao, J., Nie, J., Wu, G., Cao, G. Dependence Language Model for Information Retrieval. Annual ACM Conference on Research and Development in Information Retrieval, Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information, 2004: 170-177.
 11. Schutze, H. and Pedersen, J.O. A Co-occurrence-based Thesaurus and Two Applications to Information Retrieval. Information Processing and Management, volume 33, issue 3, 1997: pp 307-318.
 12. Voorhees, E. Query Expansion Using Lexical Semantic Relations. Annual ACM Conference on Research and Development in Information Retrieval, Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 1994: 61-69.
 13. Liu, S., Liu, F., Yu, C., and Meng, W. An Effective Approach to Document Retrieval via Utilizing WordNet and Recognizing Phrases. Annual ACM Conference on Research and Development in Information Retrieval, Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 2004: 266-272
 14. Mandala, R., Tokunaga, T., and Tanaka, H. Ad Hoc Retrieval Experiments Using WordNet and Automatically Constructed Theasuri. NIST Special Publication 500-242, The Seventh Text REtrieval Conference (TREC 7), 1998: 475-481.
 15. Cao, G., Nie, J.Y., Bai, J. Integrating Word Relationships into Language Models. Annual ACM Conference on Research and Development in Information Retrieval, Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 2005:298-305.
 16. Possas, B., Ribeiro-Neto, B., Ziviani, N., Meira Jr, M. Maximal Termsets as a Query Structuring Mechanism. Conference on Information and Knowledge Management, Proceedings of the 14th ACM International Conference on Information and Knowledge Management, 2005: 287-288.
 17. Metzler, D., Croft, W. B. A Markov Random Field Model for Term Dependencies. Annual ACM Conference on Research and Development in Information Retrieval, Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in information Retrieval, 2005: 472-479
 18. Burgess, C., Livesay, K. and Lund K. Explorations in Context Space: Words, Sentences, Discourse, Discourse Processes, 25(2&3)(1998): 211-

257.

19. Bruza P., Song, D. Inferring Query Models by Computing Information Flow. Conference on Information and Knowledge Management, Proceedings of the Eleventh International Conference on Information and Knowledge Management, 2002: 260-269.
20. Bai, J., Song, D., Bruza, P., Nie, J.Y., Cao, G. Query Expansion Using Term Relationships in Language Models for Information Retrieval. Conference on Information and Knowledge Management, Proceedings of the 14th ACM International Conference on Information and Knowledge Management, 2005: 688-695.
21. Lund, K. and Burgess C. Producing High-dimensional Semantic Spaces from Lexical Co-occurrence. Behavior Research Methods, Instrument, & Computers, 28(2)(1996): 203-208.
22. Zaki, M. J. Generating Non-Redundant Association Rules. Conference on Knowledge Discovery in Data, Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2000: 34-43.
23. Salton, G. and Lesk, M. E. Computer Evaluation of Indexing and Text Processing. Journal of the ACM (JACM), 15(1)(1968): 8-36.
24. Dominich, S. A Measure Theoretic Approach to Information Retrieval. IRFest, Information Retrieval Festival, University of Glasgow, Scotland, July 24, 2005.
25. Robertson, S. E., and Sparck Jones, K. Relevance Weighting of Search Terms. Journal of the American Society for Information Science 27, 1976: 129-146.
26. Ponte, J. M. and Croft, W.B. A Language Modeling Approach to Information Retrieval. Annual ACM Conference on Research and Development in Information Retrieval, Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information, 1998: 275-281.
27. Zhai, C. and Lafferty, J. Model-based Feedback in the KL-divergence Retrieval Model. Conference on Information and Knowledge Management, Proceedings of the Tenth International Conference on Information and Knowledge Management, 2001: 403-410.
28. Yuret, Deniz. Discovery of Linguistic Relations Using Lexical Attraction, Ph. D. thesis, Department of Electrical Engineering and Computer Science, MIT, 1998.
29. Lavrenko V. A. Generative Theory of Relevance, Ph. D. thesis, Computer Science, Graduate School, University of Massachusetts Amherst, 2004.



APPENDICES

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

APPENDIX A

INFORMATION RETRIEVAL MODELS

This appendix briefly reviews the more prominent information retrieval models that have been developed and put into practices: the Boolean Model, the Vector Space Model, the Probabilistic Model and the language modeling approach to Information Retrieval.

8.1. A.1. Boolean Model

A simple retrieval model based on the set theory and Boolean algebra, the Boolean model was adopted by many of the early commercial systems for its intuitive concept. In this system, a user issues a query comprised of terms joined together with logical operators (\wedge, \vee, \neg). The query, for an example, $[q = k_a \wedge (k_b \vee \neg k_c)]$ can be written in a disjunctive normal form (DNF) as $(1,1,1) \vee (1,1,0) \vee (1,0,0)$. Each of the three components is a binary weighted vector of the tuple (k_a, k_b, k_c) .

Its retrieval strategy is based on binary decision criterion, i.e. the document is predicted to be either relevant or not relevant. The model has been criticized for returning too few or too many documents in response to a query.

8.2. A.2. Vector Space Model

The vector model was proposed by Salton [2, 25]. The model is based on a similarity function whereby documents are ranked by their relative degrees of similarity to a user query. Documents and queries are seen as vector of distinct terms in the collection and the scoring function is the scalar product of the document and the query vectors.

$$sim(d_i, q) = \frac{\vec{d}_i \cdot \vec{q}}{|\vec{d}_i| \times |\vec{q}|} = \frac{\sum_{i=1}^t w_{i,j} * w_{i,q}}{\sqrt{\sum_{i=1}^t w_{i,j}^2} * \sqrt{\sum_{i=1}^t w_{i,q}^2}}$$

The weighting of each term is calculated from its frequency within the document tf , multiplied by the inverse document frequency idf , which indicates its importance within the collection.

Let $freq_{i,j}$ be the frequency of the term k_i in the document d_j , then the normalized frequency of the term is given by $f_{i,j} = \frac{freq_{i,j}}{\max_l freq_{l,j}}$, where $\max_l freq_{l,j}$ is the frequency of any term l in document j , which has the maximum occurrence frequency in the document.

Further, let idf be the inverse document frequency for term k_i . The idf is given by

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

The two leads to what has become the following best known term-weighting scheme. $w_{i,j} = f_{i,j} \times \log \frac{N}{n_i}$

The model, still one of the most popular today for its good performance, rests on the application of such mathematical terms as linear space, vector and inner product, to denote its underlying concepts. However, the mathematical meanings of these concepts have not been preserved. Rather they are used as mere computational constructs or metaphors [24].

Term independence is strictly assumed in the standard vector space model.

8.3. A.3. Probabilistic Model

The theoretical drawback of the vector space model has led to the introduction of the classical probabilistic model, which is based on intuition and strong probability foundation.

The model, introduced by [25] in 1976, sees ad hoc information retrieval as a task to retrieve documents “relevant” to a user query, which reflects information needs. In this model, terms remain the basic components of a document and a query.

The assumption is for the system to pick a document, examine it and estimate the probability of its relevance to a specific query. Using the Bayes’ Law, this probability is given by

$$P(R=1|D=\vec{d}_j) = \frac{P(R=1)P(D=\vec{d}_j|R=1)}{P(D=\vec{d}_j)}, \text{ where “1” denotes positive or}$$

relevant value of the R attribute and “0” means non-relevance.

The scoring function is the odds between the probability of relevance and the probability of not relevance. Hence, the formula is given by

$$sim(d_j, q) = \frac{P(R=1|D=\vec{d}_j)}{P(R=0|D=\vec{d}_j)} = \frac{P(R=1)P(D=\vec{d}_j|R=1)}{P(R=0)P(D=\vec{d}_j|R=0)} \approx \frac{P(D=\vec{d}_j|R=1)}{P(D=\vec{d}_j|R=0)}$$

The above estimation is based on the fact that $\frac{P(R=1)}{P(R=0)}$ is a constant for any \vec{d} in the collection.

The above equation is further divided by a $\frac{P(D=\vec{0}|R=1)}{P(D=\vec{0}|R=0)}$ which is construed to be the similarity value of an empty document, denoted by $\vec{0}$. As a result of the division, the equation has become the following:

$$sim(d_j, q) \approx \frac{P(D=\vec{d}_j|R=1)}{P(D=\vec{d}_j|R=0)} / \frac{P(D=\vec{0}|R=1)}{P(D=\vec{0}|R=0)}$$

Assuming that the indexing terms are independent from one another, the numerator of the right hand side of the equation can be further transformed into

$\prod_{k_i \in D} \frac{P(k_i=1|R=1)}{P(k_i=1|R=0)} \prod_{k_i \notin D} \frac{P(k_i=0|R=1)}{P(k_i=0|R=0)}$, where D is the set of all terms in the document

The denominator is transformed into $\prod_{k \in V} \frac{P(k_i=0|R=1)}{P(k_i=0|R=0)}$, where V is the set of all terms in the collection.

The division of the two expanded expressions leads to the following equation:

$$sim(d_i, q) \approx \prod_{k \in D} \frac{p_k(1-q_k)}{(1-p_k)q_k}, \text{ where } p_k \text{ stands for } P(k_i=1/R=1), q_k \text{ stands for } P(k_i=1/R=0), 1-p_k \text{ stands for } P(k_i=0/R=1), 1-q_k \text{ stands for } P(k_i=0/R=0)$$

Taking log, the equation turns into

$$sim(d_j, q) \approx \sum_{k \in D} \left(\log \frac{p_k}{1-p_k} + \log \frac{1-q_k}{q_k} \right)$$

A variant of the formula is to factor in the weights of terms in both the document and the query. It is shown here as

$$sim(d_j, q) \approx \sum_{i=1}^t w_{i,q} \times w_{i,j} \left(\log \frac{p_i}{1-p_i} + \log \frac{1-q_i}{q_i} \right)$$

According to the model, the key parameters that have to be specified are p_i and q_i . Since R is not known at start, p_i and q_i have to be estimated. The best guess for p_i is 0.5 if the term is in the query and $p_i=q_i$ otherwise. The estimate for q_i is interesting as it is approximated by $\frac{n_i}{N}$ where n_i is the number of documents which contain the term k_i and N the number of all documents in the collection. The similarity of q_i to document frequency is obvious.

The problem with the classical probability model is with the initial estimate for p_i , which is a result of the fact that R is not known. This information is updated by term statistics in the documents retrieved by the query.

Efforts to introduce term dependence into the classical model have failed [29]. In fact, the key problem with the model is with the estimates related to the unknown relevant variables, rather than the independence problem.

8.4. A.4. Language Model

The Language modeling approach has been successfully employed in areas related to natural languages such as automatic speech recognition, natural language processing, optical character recognition, handwriting recognition and machine translation for about two decades before it was first introduced to IR by [26] in 1998.

The model is about estimating the likelihood or probability of a word string. Formally, the probability of a word string is denoted by $P(W)$ where W is a string of words. What is wanted is the most likely W^* , which will be acquired on the basis of maximum likelihood estimation (MLE). In automatic speech recognition where the language model was first adopted in the 80's, the problem is to determine the W^* that best corresponds to the input acoustic signal by solving the following equation:

$$W^* = \arg \max_{W \in Gen(A)} P(W | A) = \arg \max_{W \in Gen(A)} P(W)P(A | W), \text{ where } A \text{ denotes acoustic signals and } Gen(A) \text{ is a set of all possible strings } W \text{ which may correspond to } A.$$

$P(W)P(A/W)$ is derived from $P(W/A)$ given the Bayes' law which stipulates that $P(W,A)=P(A)P(W/A)=P(W)P(A/W)$ and the MLE principle which preserves orders despite the removal of $P(A)$ from the equation.

$P(W)$ on the right-hand side of the equation is the source model and is called language model. In this framework, some source generates W^* with probability $P(W)$ and transmitted the word string through a noisy channel that transform the intended W to the observation A with probability $P(A/W)$.

The framework can be adopted in many other natural language processing applications. The basic components are the language model (the source probability distribution over a word string or $P(W)$), the transformation (A in the case of ASR), and the noisy channel through which the transformation has to undergo.

In IR, the task is to retrieve a ranked list of relevant documents D given a query Q . The retrieved documents are ranked by the posterior probability $P(D/Q)$, i.e., the probability that D is generated from the observed Q . By applying Bayes' rule and dropping the constant denominator, we get $P(D/Q) \propto P(D)P(Q/D)$. We now have the source-channel models for IR.

In practice, a uniform distribution of the prior probability $P(D)$ is assumed, so the ranking function only takes $P(Q/D)$ into account. Since it is very difficult to estimate $P(Q/D)$ directly, $P(Q/D)$ is usually approximated by $P(Q/M_D)$, where M_D is the language model trained on D . In experiments, a language model is estimated for each document. Since the document is sometimes too small to train a reliable model, smoothing techniques are also required.

When [26] introduced language modeling approaches to IR in 1988, their original work considers a document or a query a vector of $|V|$ binary values of V where V is the set of distinct vocabulary or terms in the document collection. In the work, the language model M_d is a vector of $|V|$ probabilities, one for each term v in V , and two steps are involved: firstly, M_d is derived statistically from the document, and secondly, the query is observed for the probability that it is generated by M_d .

The probability of the observed query being generated by the model is given by

$$P(Q = \bar{q} | M_d = \bar{p}_d) = \prod_{v \in Q} p_{d,v} \times \prod_{v \notin Q} (1 - p_{d,v}) , \text{ where } p_{d,v} \text{ is the probability}$$

of term v being present in the query.

The model is formally called a multiple *Bernoulli* language model, and generally known as a “bag of words” model.

The multiple Bernoulli model soon gave way to the traditional language modeling approaches widely used in the natural language processing field. Assuming a different event space for variable Q (and document variable D), the model is named multinomial LM in IR because Q is seen as a sequence of n random variables q_i , where n is the length of the query, and each q_i variable can be assigned any word in V as its value.

Owing to the sparse-data problem, most state-of-the-art language modeling approaches to IR use unigram models and do not consider any dependency between words.

Unigram is a special case of n-gram models. A unigram is the possibility of a word given zero words preceding it. A bigram (2-gram model) is the possibility of a word given one preceding word. A trigram is the possibility of a word given two preceding words. An n-gram model is the possibility of a word given n-1 preceding words. In the n-gram models, the preceding words must be adjacent to the present word under consideration.

A typical unigram model used in IR is given here.

$P(Q = \bar{q} | M_d = \bar{p}_d) = \prod_{i=1}^n p_{d, q_i}$, where p_{d, q_i} is the probability of the specific language model generating the word q_i at i in the query.

In a variant of the multinomial language model, the probability p_{d, q_i} is given by

$$p_{d, v} = \lambda \frac{tf_{d, v}}{|d|} + (1 - \lambda) \frac{\sum_{d' \in c} tf_{d', v}}{\sum_{d' \in c} |d'|}$$

Here λ is an empirical parameter used to control the variance in the estimator and to allow for interpolation from the second part of the right hand side of the equation. The part $(1 - \lambda) \frac{\sum_{d' \in c} tf_{d', v}}{\sum_{d' \in c} |d'|}$ can be seen as a simple way to smooth $p_{d, v}$.

The symbol $tf_{d, v}$ is the frequency of term v in document d . $|d|$ is the number of term in the document; and c is the set of all documents in the collection.

Language modeling approaches to IR may also be categorized by scoring methods. The basic scoring method is to rank documents by estimating the probability of the query being generated by the language model of a document and is therefore called “query likelihood”.

Another popular method is to derive a language model from the query and compare it with the estimated language model of a document. Known as KL-divergence (*Kullback-Leibler divergence*), the model computes the relative entropy of the query and the document language models. Given two probability mass functions $p(x)$ and $q(x)$, the KL-divergence between p and q is defined as

$$D(p \parallel q) = \sum_x p(x) \log \frac{p(x)}{q(x)}$$

Since the cross entropy $-\sum_x p(x) \log q(x)$ is always greater than the entropy $-\sum_x p(x) \log p(x)$, it can be shown that $D(p \parallel q)$ will always be positive. It is zero if $p(x)$ is equal to $q(x)$, i.e. when the two models are the same.

The similarity (or difference) between the estimated query language model θ_q and the estimated document language model θ_d , then the relevance value of d with respect to q can be measured by the negative KL-divergence function [27]

$$-D(p \parallel q) = \sum_w p(w | \hat{\theta}_b) \log p(w | \hat{\theta}_b) + (-\sum_w p(w | \hat{\theta}_q) \log p(w | \hat{\theta}_q))$$

The second part of the right-handed side of the equation is a query-dependent constant and can be dropped for the purpose of document ranking.



สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

APPENDIX B

EXPERIMENTAL RESULTS BY QUERIES (TOPIC 351-360)

Table B.1 compares retrieval results of baseline, IDF-based and two IDF-DD weighting methods.

Standard Measures	Query topic	Baseline	IDF-based Probability		IDF-DD Pythagoras		IDF-DD Interpolation	
		Score	Score	%Chg	Score	%Chg	Score	%Chg
num_ret	351	1,000.00	1,000.00	0.00	1,000.00	0.00	1,000.00	0.00
num_rel	351	48.00	48.00	0.00	48.00	0.00	48.00	0.00
num_rel_ret	351	23.00	25.00	8.70	26.00	13.04	26.00	13.04
map	351	0.20	0.19	-9.45	0.26	27.41	0.27	30.05
R-prec	351	0.29	0.33	14.26	0.33	14.26	0.33	14.26
bpref	351	0.25	0.26	5.37	0.30	19.11	0.30	20.51
recip_rank	351	1.00	0.50	-50.00	1.00	0.00	1.00	0.00
ircl_prn.0.00	351	1.00	0.67	-33.33	1.00	0.00	1.00	0.00
ircl_prn.0.10	351	0.67	0.63	-6.25	0.90	34.99	0.91	36.36
ircl_prn.0.20	351	0.67	0.56	-16.66	0.77	15.37	0.91	36.36
ircl_prn.0.30	351	0.23	0.38	62.48	0.50	116.64	0.54	132.11
ircl_prn.0.40	351	0.03	0.05	41.33	0.07	97.69	0.07	111.85
ircl_prn.0.50	351	0.00	0.03	N.A.	0.04	N.A.	0.04	N.A.
ircl_prn.0.60	351	0.00	0.00	N.A.	0.00	N.A.	0.00	N.A.
ircl_prn.0.70	351	0.00	0.00	N.A.	0.00	N.A.	0.00	N.A.
ircl_prn.0.80	351	0.00	0.00	N.A.	0.00	N.A.	0.00	N.A.
ircl_prn.0.90	351	0.00	0.00	N.A.	0.00	N.A.	0.00	N.A.
ircl_prn.1.00	351	0.00	0.00	N.A.	0.00	N.A.	0.00	N.A.
P5	351	0.60	0.60	0.00	0.80	33.33	0.80	33.33
P10	351	0.60	0.60	0.00	0.90	50.00	0.90	50.00
P15	351	0.67	0.53	-20.01	0.67	0.00	0.67	0.00
P20	351	0.55	0.50	-9.09	0.60	9.09	0.60	9.09
P30	351	0.40	0.43	8.33	0.50	25.00	0.50	25.00
P100	351	0.16	0.17	6.25	0.17	6.25	0.18	12.50
P200	351	0.08	0.09	12.50	0.09	12.50	0.09	12.50
P500	351	0.04	0.04	11.11	0.04	22.22	0.05	27.78
P1000	351	0.02	0.03	8.70	0.03	13.04	0.03	13.04
num_ret	352	1,000.00	1,000.00	0.00	1,000.00	0.00	1,000.00	0.00
num_rel	352	246.00	246.00	0.00	246.00	0.00	246.00	0.00
num_rel_ret	352	7.00	6.00	-14.29	7.00	0.00	7.00	0.00
map	352	0.01	0.01	34.29	0.01	34.29	0.01	32.86
R-prec	352	0.02	0.02	20.20	0.02	20.20	0.02	0.00
bpref	352	0.02	0.02	13.54	0.02	8.33	0.02	10.42
recip_rank	352	1.00	1.00	0.00	1.00	0.00	1.00	0.00
ircl_prn.0.00	352	1.00	1.00	0.00	1.00	0.00	1.00	0.00
ircl_prn.0.10	352	0.00	0.00	N.A.	0.00	N.A.	0.00	N.A.
ircl_prn.0.20	352	0.00	0.00	N.A.	0.00	N.A.	0.00	N.A.
ircl_prn.0.30	352	0.00	0.00	N.A.	0.00	N.A.	0.00	N.A.
ircl_prn.0.40	352	0.00	0.00	N.A.	0.00	N.A.	0.00	N.A.
ircl_prn.0.50	352	0.00	0.00	N.A.	0.00	N.A.	0.00	N.A.
ircl_prn.0.60	352	0.00	0.00	N.A.	0.00	N.A.	0.00	N.A.

Standard Measures	Query topic	Baseline	IDF-based Probability		IDF-DD Pythagoras		IDF-DD Interpolation	
		Score	Score	%Chg	Score	%Chg	Score	%Chg
ircl_prn.0.70	352	0.00	0.00	N.A.	0.00	N.A.	0.00	N.A.
ircl_prn.0.80	352	0.00	0.00	N.A.	0.00	N.A.	0.00	N.A.
ircl_prn.0.90	352	0.00	0.00	N.A.	0.00	N.A.	0.00	N.A.
ircl_prn.1.00	352	0.00	0.00	N.A.	0.00	N.A.	0.00	N.A.
P5	352	0.40	0.40	0.00	0.40	0.00	0.40	0.00
P10	352	0.20	0.30	50.00	0.30	50.00	0.30	50.00
P15	352	0.13	0.20	50.04	0.20	50.04	0.20	50.04
P20	352	0.10	0.20	100.00	0.20	100.00	0.20	100.00
P30	352	0.10	0.17	66.70	0.13	33.30	0.13	33.30
P100	352	0.04	0.05	25.00	0.05	25.00	0.05	25.00
P200	352	0.02	0.03	50.00	0.03	25.00	0.03	25.00
P500	352	0.01	0.01	0.00	0.01	0.00	0.01	0.00
P1000	352	0.01	0.01	-14.29	0.01	0.00	0.01	0.00
num_ret	353	1,000.00	1,000.00	0.00	1,000.00	0.00	1,000.00	0.00
num_rel	353	114.00	114.00	0.00	114.00	0.00	114.00	0.00
num_rel_ret	353	44.00	51.00	15.91	51.00	15.91	51.00	15.91
map	353	0.12	0.22	87.26	0.24	98.42	0.24	99.92
R-prec	353	0.26	0.39	49.96	0.40	53.31	0.41	56.65
bpref	353	0.24	0.28	18.84	0.34	41.04	0.34	42.00
recip_rank	353	1.00	0.50	-50.00	1.00	0.00	1.00	0.00
ircl_prn.0.00	353	1.00	0.75	-25.00	1.00	0.00	1.00	0.00
ircl_prn.0.10	353	0.41	0.67	64.45	0.59	45.76	0.58	43.88
ircl_prn.0.20	353	0.35	0.50	43.47	0.52	48.26	0.53	52.74
ircl_prn.0.30	353	0.20	0.47	135.55	0.48	139.43	0.49	141.92
ircl_prn.0.40	353	0.00	0.40	N.A.	0.42	N.A.	0.42	N.A.
ircl_prn.0.50	353	0.00	0.00	N.A.	0.00	N.A.	0.00	N.A.
ircl_prn.0.60	353	0.00	0.00	N.A.	0.00	N.A.	0.00	N.A.
ircl_prn.0.70	353	0.00	0.00	N.A.	0.00	N.A.	0.00	N.A.
ircl_prn.0.80	353	0.00	0.00	N.A.	0.00	N.A.	0.00	N.A.
ircl_prn.0.90	353	0.00	0.00	N.A.	0.00	N.A.	0.00	N.A.
ircl_prn.1.00	353	0.00	0.00	N.A.	0.00	N.A.	0.00	N.A.
P5	353	0.60	0.60	0.00	0.80	33.33	0.80	33.33
P10	353	0.40	0.60	50.00	0.60	50.00	0.60	50.00
P15	353	0.40	0.67	66.68	0.47	16.68	0.47	16.68
P20	353	0.35	0.60	71.43	0.55	57.14	0.55	57.14
P30	353	0.33	0.57	70.03	0.53	60.01	0.53	60.01
P100	353	0.26	0.42	61.54	0.44	69.23	0.44	69.23
P200	353	0.18	0.26	45.71	0.26	45.71	0.26	45.71
P500	353	0.08	0.10	24.39	0.10	24.39	0.10	24.39
P1000	353	0.04	0.05	15.91	0.05	15.91	0.05	15.91
num_ret	354	1,000.00	1,000.00	0.00	1,000.00	0.00	1,000.00	0.00
num_rel	354	361.00	361.00	0.00	361.00	0.00	361.00	0.00
num_rel_ret	354	65.00	50.00	-23.08	61.00	-6.15	67.00	3.08
map	354	0.02	0.04	55.95	0.03	11.45	0.02	0.44
R-prec	354	0.11	0.12	7.69	0.09	-12.78	0.09	-12.78
bpref	354	0.10	0.09	-3.98	0.08	-15.02	0.09	-13.07
recip_rank	354	0.20	1.00	400.00	0.25	25.00	0.20	0.00
ircl_prn.0.00	354	0.20	1.00	392.37	0.35	73.76	0.33	64.11
ircl_prn.0.10	354	0.12	0.16	26.89	0.09	-25.20	0.09	-25.76

Standard Measures	Query topic	Baseline	IDF-based Probability		IDF-DD Pythagoras		IDF-DD Interpolation	
		Score	Score	%Chg	Score	%Chg	Score	%Chg
ircl_prn.0.20	354	0.00	0.00	N.A.	0.00	N.A.	0.00	N.A.
ircl_prn.0.30	354	0.00	0.00	N.A.	0.00	N.A.	0.00	N.A.
ircl_prn.0.40	354	0.00	0.00	N.A.	0.00	N.A.	0.00	N.A.
ircl_prn.0.50	354	0.00	0.00	N.A.	0.00	N.A.	0.00	N.A.
ircl_prn.0.60	354	0.00	0.00	N.A.	0.00	N.A.	0.00	N.A.
ircl_prn.0.70	354	0.00	0.00	N.A.	0.00	N.A.	0.00	N.A.
ircl_prn.0.80	354	0.00	0.00	N.A.	0.00	N.A.	0.00	N.A.
ircl_prn.0.90	354	0.00	0.00	N.A.	0.00	N.A.	0.00	N.A.
ircl_prn.1.00	354	0.00	0.00	N.A.	0.00	N.A.	0.00	N.A.
P5	354	0.20	0.80	300.00	0.20	0.00	0.20	0.00
P10	354	0.20	0.50	150.00	0.30	50.00	0.20	0.00
P15	354	0.13	0.33	150.04	0.27	100.08	0.20	50.04
P20	354	0.15	0.25	66.67	0.30	100.00	0.20	33.33
P30	354	0.20	0.27	33.35	0.30	50.00	0.23	16.65
P100	354	0.16	0.20	25.00	0.17	6.25	0.16	0.00
P200	354	0.14	0.16	18.52	0.13	-3.70	0.12	-11.11
P500	354	0.10	0.09	-4.17	0.09	-8.33	0.09	-10.42
P1000	354	0.07	0.05	-23.08	0.06	-6.15	0.07	3.08
num_ret	355	1,000.00	1,000.00	0.00	1,000.00	0.00	1,000.00	0.00
num_rel	355	43.00	43.00	0.00	43.00	0.00	43.00	0.00
num_rel_ret	355	19.00	15.00	-21.05	16.00	-15.79	18.00	-5.26
map	355	0.10	0.10	6.55	0.11	15.56	0.12	21.39
R-prec	355	0.21	0.23	11.13	0.23	11.13	0.21	0.00
bpref	355	0.14	0.16	13.61	0.17	21.14	0.18	22.68
recip_rank	355	1.00	1.00	0.00	1.00	0.00	1.00	0.00
ircl_prn.0.00	355	1.00	1.00	0.00	1.00	0.00	1.00	0.00
ircl_prn.0.10	355	0.26	0.36	35.68	0.38	42.48	0.40	51.98
ircl_prn.0.20	355	0.23	0.26	13.96	0.38	66.67	0.36	60.00
ircl_prn.0.30	355	0.08	0.03	-65.86	0.03	-62.09	0.03	-61.60
ircl_prn.0.40	355	0.03	0.00	-100.00	0.00	-100.00	0.02	-35.24
ircl_prn.0.50	355	0.00	0.00	N.A.	0.00	N.A.	0.00	N.A.
ircl_prn.0.60	355	0.00	0.00	N.A.	0.00	N.A.	0.00	N.A.
ircl_prn.0.70	355	0.00	0.00	N.A.	0.00	N.A.	0.00	N.A.
ircl_prn.0.80	355	0.00	0.00	N.A.	0.00	N.A.	0.00	N.A.
ircl_prn.0.90	355	0.00	0.00	N.A.	0.00	N.A.	0.00	N.A.
ircl_prn.1.00	355	0.00	0.00	N.A.	0.00	N.A.	0.00	N.A.
P5	355	0.40	0.40	0.00	0.40	0.00	0.40	0.00
P10	355	0.20	0.30	50.00	0.30	50.00	0.30	50.00
P15	355	0.20	0.33	66.65	0.33	66.65	0.40	100.00
P20	355	0.25	0.25	0.00	0.30	20.00	0.40	60.00
P30	355	0.20	0.27	33.35	0.30	50.00	0.30	50.00
P100	355	0.09	0.10	11.11	0.10	11.11	0.10	11.11
P200	355	0.07	0.05	-23.08	0.06	-15.38	0.06	-7.69
P500	355	0.03	0.03	-23.53	0.03	-23.53	0.03	-23.53
P1000	355	0.02	0.02	-21.05	0.02	-15.79	0.02	-5.26
num_ret	356	1,000.00	1,000.00	0.00	1,000.00	0.00	1,000.00	0.00
num_rel	356	16.00	16.00	0.00	16.00	0.00	16.00	0.00
num_rel_ret	356	2.00	0.00	-100.00	2.00	0.00	2.00	0.00
map	356	0.01	0.00	-100.00	0.01	3.45	0.01	8.62

Standard Measures	Query topic	Baseline	IDF-based Probability		IDF-DD Pythagoras		IDF-DD Interpolation	
		Score	Score	%Chg	Score	%Chg	Score	%Chg
R-prec	356	0.00	0.00	N.A.	0.00	N.A.	0.00	N.A.
bpref	356	0.00	0.00	N.A.	0.00	N.A.	0.00	N.A.
recip_rank	356	0.05	0.00	-100.00	0.05	-9.51	0.05	-9.51
ircl_prn.0.00	356	0.05	0.00	-100.00	0.05	-7.22	0.05	0.00
ircl_prn.0.10	356	0.04	0.00	-100.00	0.05	22.00	0.05	31.50
ircl_prn.0.20	356	0.00	0.00	N.A.	0.00	N.A.	0.00	N.A.
ircl_prn.0.30	356	0.00	0.00	N.A.	0.00	N.A.	0.00	N.A.
ircl_prn.0.40	356	0.00	0.00	N.A.	0.00	N.A.	0.00	N.A.
ircl_prn.0.50	356	0.00	0.00	N.A.	0.00	N.A.	0.00	N.A.
ircl_prn.0.60	356	0.00	0.00	N.A.	0.00	N.A.	0.00	N.A.
ircl_prn.0.70	356	0.00	0.00	N.A.	0.00	N.A.	0.00	N.A.
ircl_prn.0.80	356	0.00	0.00	N.A.	0.00	N.A.	0.00	N.A.
ircl_prn.0.90	356	0.00	0.00	N.A.	0.00	N.A.	0.00	N.A.
ircl_prn.1.00	356	0.00	0.00	N.A.	0.00	N.A.	0.00	N.A.
P5	356	0.00	0.00	N.A.	0.00	N.A.	0.00	N.A.
P10	356	0.00	0.00	N.A.	0.00	N.A.	0.00	N.A.
P15	356	0.00	0.00	N.A.	0.00	N.A.	0.00	N.A.
P20	356	0.05	0.00	-100.00	0.00	-100.00	0.00	-100.00
P30	356	0.03	0.00	-100.00	0.03	0.00	0.03	0.00
P100	356	0.02	0.00	-100.00	0.02	0.00	0.02	0.00
P200	356	0.01	0.00	-100.00	0.01	0.00	0.01	0.00
P500	356	0.00	0.00	-100.00	0.00	0.00	0.00	0.00
P1000	356	0.00	0.00	-100.00	0.00	0.00	0.00	0.00
num_ret	357	1,000.00	1,000.00	0.00	1,000.00	0.00	1,000.00	0.00
num_rel	357	270.00	270.00	0.00	270.00	0.00	270.00	0.00
num_rel_ret	357	78.00	45.00	-42.31	56.00	-28.21	57.00	-26.92
map	357	0.11	0.02	-79.84	0.05	-54.14	0.06	-51.06
R-prec	357	0.20	0.09	-54.54	0.14	-29.11	0.16	-23.61
bpref	357	0.19	0.08	-60.59	0.13	-32.43	0.14	-30.22
recip_rank	357	1.00	0.25	-75.00	1.00	0.00	1.00	0.00
ircl_prn.0.00	357	1.00	0.45	-54.55	1.00	0.00	1.00	0.00
ircl_prn.0.10	357	0.53	0.10	-81.81	0.23	-57.05	0.23	-56.20
ircl_prn.0.20	357	0.24	0.00	-100.00	0.06	-74.01	0.07	-70.99
ircl_prn.0.30	357	0.00	0.00	N.A.	0.00	N.A.	0.00	N.A.
ircl_prn.0.40	357	0.00	0.00	N.A.	0.00	N.A.	0.00	N.A.
ircl_prn.0.50	357	0.00	0.00	N.A.	0.00	N.A.	0.00	N.A.
ircl_prn.0.60	357	0.00	0.00	N.A.	0.00	N.A.	0.00	N.A.
ircl_prn.0.70	357	0.00	0.00	N.A.	0.00	N.A.	0.00	N.A.
ircl_prn.0.80	357	0.00	0.00	N.A.	0.00	N.A.	0.00	N.A.
ircl_prn.0.90	357	0.00	0.00	N.A.	0.00	N.A.	0.00	N.A.
ircl_prn.1.00	357	0.00	0.00	N.A.	0.00	N.A.	0.00	N.A.
P5	357	0.60	0.20	-66.67	0.60	0.00	0.60	0.00
P10	357	0.60	0.40	-33.33	0.40	-33.33	0.50	-16.67
P15	357	0.60	0.33	-44.45	0.47	-22.22	0.47	-22.22
P20	357	0.55	0.30	-45.45	0.45	-18.18	0.40	-27.27
P30	357	0.57	0.23	-58.83	0.37	-35.29	0.37	-35.29
P100	357	0.43	0.13	-69.77	0.21	-51.16	0.26	-39.53
P200	357	0.26	0.10	-61.54	0.16	-40.38	0.18	-30.77
P500	357	0.13	0.06	-55.22	0.09	-29.85	0.09	-29.85

Standard Measures	Query topic	Baseline	IDF-based Probability		IDF-DD Pythagoras		IDF-DD Interpolation	
		Score	Score	%Chg	Score	%Chg	Score	%Chg
P1000	357	0.08	0.05	-42.31	0.06	-28.21	0.06	-26.92
num_ret	358	1,000.00	1,000.00	0.00	1,000.00	0.00	1,000.00	0.00
num_rel	358	51.00	51.00	0.00	51.00	0.00	51.00	0.00
num_rel_ret	358	48.00	14.00	-70.83	30.00	-37.50	30.00	-37.50
map	358	0.30	0.01	-96.68	0.09	-71.53	0.09	-70.40
R-prec	358	0.31	0.06	-81.26	0.20	-37.49	0.20	-37.49
bpref	358	0.29	0.02	-91.96	0.15	-47.11	0.15	-46.07
recip_rank	358	0.33	0.03	-90.91	0.33	0.00	0.33	0.00
ircl_prn.0.00	358	0.67	0.07	-90.22	0.40	-40.00	0.43	-35.71
ircl_prn.0.10	358	0.63	0.04	-93.47	0.40	-36.00	0.43	-31.42
ircl_prn.0.20	358	0.45	0.03	-93.64	0.19	-58.72	0.20	-54.89
ircl_prn.0.30	358	0.36	0.00	-100.00	0.06	-82.64	0.06	-83.34
ircl_prn.0.40	358	0.35	0.00	-100.00	0.04	-89.02	0.04	-87.84
ircl_prn.0.50	358	0.29	0.00	-100.00	0.03	-88.60	0.04	-86.20
ircl_prn.0.60	358	0.26	0.00	-100.00	0.00	-100.00	0.00	-100.00
ircl_prn.0.70	358	0.20	0.00	-100.00	0.00	-100.00	0.00	-100.00
ircl_prn.0.80	358	0.14	0.00	-100.00	0.00	-100.00	0.00	-100.00
ircl_prn.0.90	358	0.07	0.00	-100.00	0.00	-100.00	0.00	-100.00
ircl_prn.1.00	358	0.00	0.00	N.A.	0.00	N.A.	0.00	N.A.
P5	358	0.60	0.00	-100.00	0.20	-66.67	0.20	-66.67
P10	358	0.50	0.00	-100.00	0.20	-60.00	0.20	-60.00
P15	358	0.60	0.00	-100.00	0.40	-33.33	0.40	-33.33
P20	358	0.50	0.00	-100.00	0.35	-30.00	0.35	-30.00
P30	358	0.43	0.00	-100.00	0.27	-38.45	0.27	-38.45
P100	358	0.28	0.03	-89.29	0.12	-57.14	0.12	-57.14
P200	358	0.19	0.04	-81.08	0.07	-62.16	0.07	-62.16
P500	358	0.09	0.03	-71.11	0.04	-55.56	0.04	-53.33
P1000	358	0.05	0.01	-70.83	0.03	-37.50	0.03	-37.50
num_ret	359	1,000.00	1,000.00	0.00	1,000.00	0.00	1,000.00	0.00
num_rel	359	28.00	28.00	0.00	28.00	0.00	28.00	0.00
num_rel_ret	359	10.00	6.00	-40.00	7.00	-30.00	8.00	-20.00
map	359	0.06	0.00	-95.12	0.00	-92.04	0.01	-86.62
R-prec	359	0.07	0.00	-100.00	0.00	-100.00	0.00	-100.00
bpref	359	0.05	0.00	-100.00	0.00	-100.00	0.00	-100.00
recip_rank	359	1.00	0.02	-97.87	0.02	-97.87	0.02	-97.73
ircl_prn.0.00	359	1.00	0.02	-97.87	0.02	-97.63	0.04	-96.21
ircl_prn.0.10	359	0.09	0.02	-82.84	0.02	-73.93	0.04	-58.31
ircl_prn.0.20	359	0.06	0.01	-87.24	0.02	-70.60	0.03	-47.33
ircl_prn.0.30	359	0.03	0.00	-100.00	0.00	-100.00	0.00	-100.00
ircl_prn.0.40	359	0.00	0.00	N.A.	0.00	N.A.	0.00	N.A.
ircl_prn.0.50	359	0.00	0.00	N.A.	0.00	N.A.	0.00	N.A.
ircl_prn.0.60	359	0.00	0.00	N.A.	0.00	N.A.	0.00	N.A.
ircl_prn.0.70	359	0.00	0.00	N.A.	0.00	N.A.	0.00	N.A.
ircl_prn.0.80	359	0.00	0.00	N.A.	0.00	N.A.	0.00	N.A.
ircl_prn.0.90	359	0.00	0.00	N.A.	0.00	N.A.	0.00	N.A.
ircl_prn.1.00	359	0.00	0.00	N.A.	0.00	N.A.	0.00	N.A.
P5	359	0.20	0.00	-100.00	0.00	-100.00	0.00	-100.00
P10	359	0.10	0.00	-100.00	0.00	-100.00	0.00	-100.00
P15	359	0.07	0.00	-100.00	0.00	-100.00	0.00	-100.00

Standard Measures	Query topic	Baseline	IDF-based Probability		IDF-DD Pythagoras		IDF-DD Interpolation	
		Score	Score	%Chg	Score	%Chg	Score	%Chg
P20	359	0.10	0.00	-100.00	0.00	-100.00	0.00	-100.00
P30	359	0.07	0.00	-100.00	0.00	-100.00	0.00	-100.00
P100	359	0.06	0.01	-83.33	0.01	-83.33	0.01	-83.33
P200	359	0.04	0.01	-87.50	0.02	-50.00	0.03	-25.00
P500	359	0.02	0.01	-44.44	0.01	-33.33	0.01	-22.22
P1000	359	0.01	0.01	-40.00	0.01	-30.00	0.01	-20.00
num_ret	360	1,000.00	1,000.00	0.00	1,000.00	0.00	1,000.00	0.00
num_rel	360	151.00	151.00	0.00	151.00	0.00	151.00	0.00
num_rel_ret	360	87.00	103.00	18.39	102.00	17.24	101.00	16.09
map	360	0.13	0.31	143.59	0.33	161.35	0.32	152.83
R-prec	360	0.23	0.38	62.86	0.32	39.99	0.33	42.84
bpref	360	0.23	0.36	55.01	0.34	43.28	0.34	43.19
recip_rank	360	1.00	1.00	0.00	1.00	0.00	1.00	0.00
ircl_prn.0.00	360	1.00	1.00	0.00	1.00	0.00	1.00	0.00
ircl_prn.0.10	360	0.32	0.68	110.24	0.95	194.77	0.84	162.01
ircl_prn.0.20	360	0.24	0.61	157.00	0.71	200.68	0.67	184.95
ircl_prn.0.30	360	0.20	0.42	107.47	0.39	90.07	0.37	79.50
ircl_prn.0.40	360	0.12	0.37	201.14	0.32	159.85	0.32	159.69
ircl_prn.0.50	360	0.10	0.27	161.84	0.24	132.58	0.25	140.70
ircl_prn.0.60	360	0.00	0.18	N.A.	0.16	N.A.	0.16	N.A.
ircl_prn.0.70	360	0.00	0.00	N.A.	0.00	N.A.	0.00	N.A.
ircl_prn.0.80	360	0.00	0.00	N.A.	0.00	N.A.	0.00	N.A.
ircl_prn.0.90	360	0.00	0.00	N.A.	0.00	N.A.	0.00	N.A.
ircl_prn.1.00	360	0.00	0.00	N.A.	0.00	N.A.	0.00	N.A.
P5	360	0.60	0.80	33.33	1.00	66.67	1.00	66.67
P10	360	0.40	0.80	100.00	1.00	150.00	1.00	150.00
P15	360	0.33	0.73	120.01	0.93	180.02	0.93	180.02
P20	360	0.25	0.70	180.00	0.90	260.00	0.80	220.00
P30	360	0.30	0.63	111.10	0.80	166.67	0.73	144.43
P100	360	0.22	0.45	104.55	0.43	95.45	0.44	100.00
P200	360	0.21	0.34	63.41	0.32	53.66	0.32	53.66
P500	360	0.12	0.18	48.33	0.18	48.33	0.18	50.00
P1000	360	0.09	0.10	18.39	0.10	17.24	0.10	16.09

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

APPENDIX C

IDF AND IDF-DD WEIGHTS COMPUTED FOR TOPIC 351-360

The following is a table generated by a main application used in the experimentation. Brief explanations are:

Pass: If the IDF-DD probability exceeds 0, Pass = 1 else Pass = 0. In effect, the query terms with 0 pass value is dropped before the query is passed to a retrieval engine. Pass indicates the importance of a term in Boolean.

IDF Weight: Computed from $idf = \log \frac{N}{n}$ function where N = number of collection documents and n = number of document which contains the term.

IDF-DD Probability: The weight computed by $f(idf, dd)$. Please refer to *Chapter 2* for details.

Direction: 1 = UP, 2 = DOWN, 3 = DOWNUP, 4 = UPDOWN, 6 = PEAK. Please refer to *Chapter 2* for details.

Table C.1 shows the outputs of IDF-DD computation

Query	Pass	IDF Weight	IDF-DD Prob	Direction
<QUERY 351>				
explor	1	3.971	0.111	4
what	1	1.383	0.080	3
inform	1	2.178	0.107	4
is	0	0.314	0.000	3
avail	1	2.821	0.132	4
on	0	0.198	0.000	3
petroleum	1	4.585	0.175	6
explor	1	3.971	0.165	2
in	0	0.062	0.000	2
the	0	0.025	0.000	3
south	0	2.313	0.000	1
atlant	0	4.346	0.000	4
near	0	2.562	0.000	2
the	0	0.025	0.000	3
falkland	1	7.306	1.000	6
island	1	3.665	0.645	3
</QUERY>				
<QUERY 352>				
impact	1	3.232	0.381	4
what	1	1.383	0.317	3
impact	1	3.232	0.479	4
ha	0	0.494	0.000	2

Query	Pass	IDF Weight	IDF-DD Prob	Direction
the	0	0.025	0.000	3
chunnel	1	10.970	1.000	6
had	0	0.878	0.000	2
on	0	0.198	0.000	2
the	0	0.025	0.000	3
british	1	2.624	0.344	4
economi	1	2.396	0.330	2
and	0	0.072	0.000	3
or	0	0.967	0.000	4
the	0	0.025	0.000	3
life	1	2.262	0.175	1
style	1	3.210	0.233	4
of	0	0.047	0.000	2
the	0	0.025	0.000	3
british	1	2.624	0.099	4
</QUERY>				
<QUERY 353>				
identifi	1	3.337	0.143	3
systemat	1	5.233	0.197	6
explor	1	3.971	0.170	2
and	0	0.072	0.000	3
scientif	1	3.972	0.157	4
investig	1	2.798	0.131	2
of	0	0.047	0.000	3
antarctica	1	7.712	1.000	6
current	1	2.005	0.373	2
or	0	0.967	0.000	3
plan	1	1.570	0.259	4
</QUERY>				
<QUERY 354>				
identifi	1	3.337	0.206	3
instanc	1	3.659	0.246	6
where	1	1.639	0.184	3
journalist	1	3.753	0.459	6
ha	0	0.494	0.000	3
been	0	0.748	0.000	1
put	1	1.832	0.238	4
at	0	0.370	0.000	3
risk	1	3.038	0.449	4
kill	1	2.801	0.505	3
arrest	1	3.226	0.611	6
or	0	0.967	0.000	3
taken	1	2.220	0.515	1
hostag	1	5.130	1.000	6
in	0	0.062	0.000	2
the	0	0.025	0.000	3
perform	1	2.428	0.420	4

Query	Pass	IDF Weight	IDF-DD Prob	Direction
of	0	0.047	0.000	3
hi	0	0.945	0.000	1
work	1	1.359	0.045	4
</QUERY>				
<QUERY 355>				
identifi	1	3.337	0.254	4
document	1	3.058	0.278	2
discuss	1	2.217	0.268	2
the	0	0.025	0.000	3
develop	1	1.628	0.274	4
and	0	0.072	0.000	3
applic	1	3.577	0.427	4
of	0	0.047	0.000	3
spaceborn	1	12.761	1.000	6
ocean	1	4.370	0.512	3
remot	1	4.619	0.527	4
sens	1	3.060	0.436	3
</QUERY>				
<QUERY 356>				
identifi	1	3.337	0.097	4
document	1	3.058	0.098	2
discuss	1	2.217	0.086	2
the	0	0.025	0.000	3
us	0	0.838	0.000	4
of	0	0.047	0.000	3
estrogen	1	8.870	0.238	6
by	0	0.296	0.000	3
postmenopaus	1	10.970	1.000	6
women	1	3.062	0.273	2
in	0	0.062	0.000	3
britain	1	3.065	0.245	4
</QUERY>				
<QUERY 357>				
identifi	1	3.337	0.812	6
document	1	3.058	0.776	2
discuss	1	2.217	0.666	2
intern	1	1.558	0.580	3
boundari	1	4.956	1.000	6
disput	1	3.346	0.742	3
relev	1	4.186	0.876	4
to	0	0.066	0.000	2
the	0	0.025	0.000	3
200	1	2.880	0.661	1
mile	1	3.262	0.810	6
special	1	2.280	0.682	2
econom	1	1.795	0.619	3
zone	1	3.500	0.683	6

Query	Pass	IDF Weight	IDF-DD Prob	Direction
or	0	0.967	0.000	3
12	1	1.983	0.109	1
mile	1	3.262	0.293	4
territori	1	3.031	0.341	2
water	1	2.861	0.394	3
subsequ	1	3.976	0.563	6
to	0	0.066	0.000	2
the	0	0.025	0.000	3
pass	0	2.569	0.000	4
of	0	0.047	0.000	2
the	0	0.025	0.000	3
intern	1	1.558	0.224	1
convent	1	3.698	0.419	6
on	0	0.198	0.000	2
the	0	0.025	0.000	3
law	0	2.205	0.000	4
of	0	0.047	0.000	2
the	0	0.025	0.000	3
sea	1	3.443	0.276	6
</QUERY>				
<QUERY 358>				
what	1	1.383	0.094	3
role	1	2.539	0.162	4
doe	1	2.012	0.189	3
blood	1	4.004	0.279	1
alcohol	1	4.549	0.332	6
level	1	2.021	0.270	3
plai	1	2.092	0.271	4
in	0	0.062	0.000	3
automobil	1	4.643	0.666	6
accid	1	3.930	0.631	3
fatal	1	4.722	1.000	6
</QUERY>				
<QUERY 359>				
ar	0	0.604	0.000	3
there	0	0.987	0.000	1
reliabl	1	4.418	0.479	4
and	0	0.072	0.000	3
consist	1	3.300	0.435	1
predictor	1	8.933	1.000	6
of	0	0.047	0.000	3
mutual	1	3.691	0.470	4
fund	1	2.194	0.335	3
perform	1	2.428	0.356	4
</QUERY>				
<QUERY 360>				
what	1	1.383	0.271	4

Query	Pass	IDF Weight	IDF-DD Prob	Direction
ar	0	0.604	0.000	2
the	0	0.025	0.000	3
benefit	1	2.714	0.476	6
if	1	1.125	0.357	3
ani	1	1.438	0.380	4
of	0	0.047	0.000	3
drug	1	3.162	1.000	6
legal	1	2.750	0.935	3
</QUERY>				
<QUERY 361>				
identifi	1	3.337	0.243	4
document	1	3.058	0.270	2
that	0	0.271	0.000	3
discuss	1	2.217	0.238	1
cloth	1	3.916	0.470	1
sweatshop	1	8.471	1.000	6
</QUERY>				
<QUERY 362>				
identifi	1	3.337	0.273	3
incid	1	3.460	0.312	6
of	0	0.047	0.000	3
human	1	2.967	0.492	1
smuggl	1	5.013	1.000	6
</QUERY>				
<QUERY 363>				
what	1	1.383	0.143	3
disast	1	4.237	0.395	6
have	0	0.543	0.000	3
occur	1	3.275	0.286	4
in	0	0.062	0.000	3
tunnel	1	5.271	1.000	6
us	0	0.838	0.000	2
for	0	0.194	0.000	3
transport	1	3.057	0.395	4
</QUERY>				
<QUERY 364>				
identifi	1	3.337	0.440	4
document	1	3.058	0.492	2
discuss	1	2.217	0.518	2
case	1	2.072	0.577	2
where	1	1.639	0.623	3
rabi	1	8.208	1.000	6
have	0	0.543	0.000	3
been	0	0.748	0.000	1
confirm	1	2.982	0.621	4
and	0	0.072	0.000	3
what	1	1.383	0.479	4

Query	Pass	IDF Weight	IDF-DD Prob	Direction
if	1	1.125	0.467	3
anyth	1	2.888	0.550	4
is	0	0.314	0.000	3
be	0	0.393	0.000	1
done	1	2.587	0.404	4
about	0	0.927	0.000	2
it	0	0.308	0.000	3
</QUERY>				
<QUERY 365>				
what	1	1.383	0.042	3
effect	1	2.102	0.183	4
have	0	0.543	0.000	3
been	0	0.748	0.000	1
attribut	1	4.049	0.458	4
to	0	0.066	0.000	3
el	1	3.674	0.480	1
nino	1	7.879	1.000	6
</QUERY>				
<QUERY 366>				
what	1	1.383	0.105	4
ar	0	0.604	0.000	2
the	0	0.025	0.000	3
industri	1	1.804	0.215	4
or	0	0.967	0.000	3
commerci	1	2.848	0.392	4
us	0	0.838	0.000	2
of	0	0.047	0.000	3
cyanid	1	7.849	1.000	6
or	0	0.967	0.000	2
it	0	0.308	0.000	3
deriv	1	4.623	0.522	4
</QUERY>				
<QUERY 367>				
what	1	1.383	0.037	3
modern	1	3.375	0.312	1
instanc	1	3.659	0.419	4
have	0	0.543	0.000	3
there	0	0.987	0.000	4
been	0	0.748	0.000	2
of	0	0.047	0.000	3
old	1	1.990	0.335	1
fashion	1	3.885	0.599	1
piraci	1	7.170	1.000	6
the	0	0.025	0.000	3
board	1	2.519	0.101	4
or	0	0.967	0.000	3
take	1	1.294	0.094	1

Query	Pass	IDF Weight	IDF-DD Prob	Direction
control	1	2.057	0.121	4
of	0	0.047	0.000	3
boat	1	4.471	0.177	6
</QUERY>				
<QUERY 368>				
identifi	1	3.337	0.305	4
document	1	3.058	0.334	2
that	0	0.271	0.000	3
discuss	1	2.217	0.304	4
in	0	0.062	0.000	3
vitro	1	8.251	1.000	6
fertil	1	5.172	0.674	3
</QUERY>				
<QUERY 369>				
what	1	1.383	0.034	4
ar	0	0.604	0.000	2
the	0	0.025	0.000	3
caus	1	2.315	0.154	4
and	0	0.072	0.000	3
treatment	1	3.632	0.309	4
of	0	0.047	0.000	3
anorexia	1	9.717	0.887	1
nervosa	1	10.564	1.000	6
and	0	0.072	0.000	3
bulimia	1	9.989	0.462	6
</QUERY>				
<QUERY 370>				
what	1	1.383	0.123	4
ar	0	0.604	0.000	2
the	0	0.025	0.000	3
law	1	2.205	0.274	4
deal	1	2.180	0.370	2
with	0	0.327	0.000	2
the	0	0.025	0.000	3
qualiti	1	3.036	0.524	4
and	0	0.072	0.000	3
process	1	2.304	0.572	4
of	0	0.047	0.000	3
food	1	2.938	0.712	1
beverag	1	5.808	1.000	6
or	0	0.967	0.000	3
drug	1	3.162	0.727	4
</QUERY>				

APPENDIX D

THE TREC RETRIEVAL CONFERENCE

The Text REtrieval Conference (TREC), co-sponsored by the National Institute of Standards and Technology (NIST) and U.S. Department of Defense, was started in 1992 as part of the TIPSTER Text program. Its purpose was to support research within the information retrieval community by providing the infrastructure necessary for large-scale evaluation of text retrieval methodologies. In particular, the TREC workshop series has the following goals:

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

TREC is overseen by a program committee consisting of representatives from government, industry, and academia. For each TREC, NIST provides a test set of documents and questions. Participants run their own retrieval systems on the data, and return to NIST a list of the retrieved top-ranked documents. NIST pools the individual results, judges the retrieved documents for correctness, and evaluates the results. The TREC cycle ends with a workshop that is a forum for participants to share their experiences.

This evaluation effort has grown in both the number of participating systems and the number of tasks each year. Ninety-three groups representing 22 countries participated in TREC 2003.

The TREC test collections and evaluation software are available to the retrieval research community at large, so organizations can evaluate their own retrieval systems at any time. TREC has successfully met its dual goals of improving the state-of-the-art in information retrieval and of facilitating technology transfer. Retrieval system effectiveness approximately doubled in the first six years of TREC.

TREC has also sponsored the first large-scale evaluations of the retrieval of non-English (Spanish and Chinese) documents, retrieval of recordings of speech, and retrieval across multiple languages. TREC has also introduced evaluations for open-domain question answering and content-based retrieval of digital video. The TREC test collections are large enough so that they realistically model operational settings. Most of today's commercial search engines include technology first developed in TREC.

APPENDIX E

LEMUR TOOLKIT

To facilitate research on information retrieval, particularly language modeling, the Computer Science Department at the University of Massachusetts and the School of Computer Science at Carnegie Mellon University have jointly launched the Lemur Project to provide tools for IR researchers.

The Lemur Project is sponsored by the Advanced Research and Development Activity in Information Technology (ARDA) under its Statistical Language Modeling for Information Retrieval Research Program and by the National Science Foundation.

Language modeling has recently emerged as an attractive new framework for text information retrieval, leveraging work on language modeling from other areas such as speech recognition and statistical natural language processing. Research carried out at a number of sites has confirmed that the language modeling approach is an effective and theoretically attractive probabilistic framework for building information retrieval (IR) systems.

The Lemur Toolkit is designed to facilitate research in language modeling and information retrieval, where IR is broadly interpreted to include such technologies as ad hoc and distributed retrieval, cross-language IR, summarization, filtering, and classification.

The toolkit supports indexing of large-scale text databases, the construction of simple language models for documents, queries, or sub collections, and the implementation of retrieval systems based on language models as well as a variety of other retrieval models. The system is written in the C and C++ languages, and is designed as a research system to run under Unix operating systems, although it can also run under Windows.

The toolkit can be downloaded from <http://www.lemurproject.org> .

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

APPENDIX F

IDF-DD CODE IMPLEMENTATION

The $f(idf,dd)$ function is implemented in the *TermweightQueryRep* class developed as a library for this research within the development framework of the Lemur Toolkit. The class declarations and definitions are coded in *TermweightQueryRep.hpp* and *TermweightQueryRep.cpp*. In addition to the class, we write *TestTermWeightApp.exe* application to invoke the *TermWeightQueryRep* class and modify *RetEval.exe* of the *Lemur* project to accept the output from *TestTermWeightApp.exe* by using a new *WeightedDocStream* class, which is subclassed from *DocStream* like the *BasicDocStream* class.

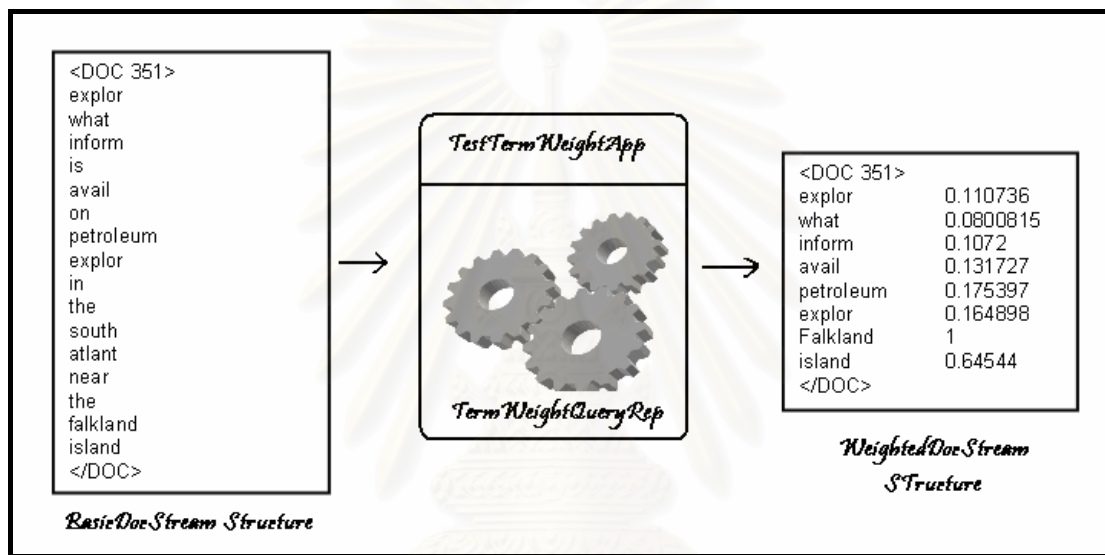


Figure F.1: A diagram showing an input query file in *BasicDocStream* class format going through an application *TestTermWeightApp.exe*, which invokes *TermweightQueryRep* class library and outputs a transformed query file structured to conform to the *WeightedDocStream* class.

In our research, we take the following steps before using *TermWiegghtQueryRep* class:

1. Develop an application *TestTermWeightApp.exe* to invoke the class.

The application does three four things: open a database connection, open an input file of a query set and iterate through each query, invoke *TermweightQueryRep* library, and output a file of transformed queries. A *TermweightQueryRep* object is created for each query topic.

The *TermweightQueryRep* class is initialized with three parameters: database index pointer, a query topic, and an option. Three options are available: SIMPLEIDF, PITHAGORAS, and INTERPOLATE.

2. Prepare the input file of a query set. The structure of the query sets are required by the *BasicDocStream* class of the *Lemur* Project.

3. Modify *RetEval.exe*, an application bundled with the *Lemur* Project so it can accept input file of weighted query terms in addition to a query file structure conforming to the *BasicDocStream* class requirement. The modification requires the subclassing of *DocStream* to create the *WeightedDocStream* class. By default, a query file of the *BasicDocStream* structure is used as an input to *RetEval.exe* which will initialize a retrieval engine. In our implementation, a weighted query file is used instead. Consult the Lemur project development guide on the use of *RetEval.exe*.

The source codes of *TermweightedQueryRep* and *WeightedDocStream* classes, as well as the modified *RetEval.exe* application are published in the following pages.



สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

c:\Lem\src\lemur-4.2\retrieval\include\TermweightQueryRep.hpp

1

```

/*=====
                APPENDIX F:
            IDF-DD Term Weighting Library

/*
/*=====
*
* This library is authored by Sompong Kittinaradorn as part of a master
* thesis research at Chulalongkorn University. 2006-2007.
*
* It is written to work with the Lemur Toolkit for Language Modeling and
* Information Retrieval.
*
* Use of the Lemur Toolkit is subject to the terms of the software license
* set forth in the LICENSE file included with this software, and also
* available at http://www.lemurproject.org/license.html
* Copyright (c) 2001 Carnegie Mellon University. All Rights Reserved.
*
*=====
*/

#ifndef _TERMWEIGHTQUERYREP_HPP
#define _TERMWEIGHTQUERYREP_HPP

#include "TextQueryRep.hpp"
namespace lemur
{
    namespace retrieval
    {
        {
            enum tDirection {STARTER = 0, UP=1, DOWN = 2, DOWNUP = 3, UPDOWN = 4, MIDLOW = 5, PEAK = 6};
            enum twOption {SIMPLEIDF = 0, PITHAGORAS = 1, INTERPOLATE = 2};
            class TermweightQueryRep {
            public:
                TermweightQueryRep(int size, string qid, twOption calOption, const lemur::api::TermQuery &
qry, const lemur::api::Index &dbx);
                virtual ~TermweightQueryRep();

            protected:
                virtual void calIDF(vector<int> refList, const lemur::api::Index &dbIndex);
                virtual void calIDFDD();
                virtual void transformQuery();
                lemur::utility::ArrayCounter<double> * counter;
                twOption calweightOpt;

            private:
                string getName(int ti){
                    return orderednames[ti];
                }
                string getNamefromid(int id){
                    int i = 0;
                    for (i=0; i<orderedtermids.size(); i++){
                        if (id==orderedtermids[i]){
                            break;
                        }
                    }
                    return orderednames[i];
                }
                string curqueryid;
                string indexpath;
                map<int, double> refWeight;
                double maxval, minval;

                vector<int> orderedtermids;
                vector<double>orderedtermweights;
                vector<tDirection>termdirections;
                vector<bool> orderedresults;
                vector<double> orderreddists;
        }
    }
}

```


c:\Lem\src\lemur-4.2\retrieval\include\TermweightQueryRep.hpp

2

```

    vector<string> orderednames;

    ofstream logf;
};
}
#endif /* _TERMWEIGHTQUERYREP_HPP */

/*=====
 *
 * This library is authored by Sompong Kittinaradorn as part of a master
 * thesis research at Chulalongkorn University. 2006-2007.
 *
 * It is written to work with the Lemur Toolkit for Language Modeling and
 * Information Retrieval.
 *
 * Use of the Lemur Toolkit is subject to the terms of the software license
 * set forth in the LICENSE file included with this software, and also
 * available at http://www.lemurproject.org/license.html
 * Copyright (c) 2001 Carnegie Mellon University. All Rights Reserved.
 *
 *=====
 */

#include "TermweightQueryRep.hpp"
#include "InvFPIndex.hpp"
using namespace lemur::api;
using namespace lemur::index;

lemur::retrieval::TermweightQueryRep::TermweightQueryRep(int size, string qid, twOption calOption,
const TermQuery &qry, const Index &dbx):
curqueryid(qid), counter(new lemur::utility::ArrayCounter<double>(size)), calweightOpt(calOption)
{
    vector <int> vRefId;
    vector <int> vPos;
    map<int, int, less<int>> mPos;

    indexpath = "c:/TrecDatabase/bin/workspace/";
    string fname = indexpath + "termsetquery8rank.log";

    logf.open(fname.c_str(), ios_base::out|ios_base::ate|ios_base::app);
    if (!logf)
        cerr<<"error creating log file "<<endl;
    logf.seekp(0, ios_base::end);
    logf<<endl<<"***** LOGGING ["<<curqueryid<<"] *****"<<endl<<endl;
    const InvFPIndex * indx = dynamic_cast<const InvFPIndex*> (&dbx);
    int a = 0, beforeid = 0, afterid = 0;

    qry.startTermIteration();
    while (qry.hasMore()){
        const Term *t = qry.nextTerm();
        char * qryterm = const_cast<char*>(t->spelling());
        TERMID_T ti = dbx.term(qryterm);
        if (strlen(indx->term(ti).c_str())==1)
            continue;
        cerr<<"TermID-> "<<ti<<", Spelling-> "<<t->spelling()<<endl;
        logf<<"TermID-> "<<ti<<", Spelling-> "<<t->spelling()<<endl;

        orderedtermids.push_back(ti);
        string sp = t->spelling();
        orderednames.push_back(sp);

        if (ti>0){
            lemur::index::InvFPDocList* dlist = dynamic_cast<lemur::index::InvFPDocList*>(indx
->docInfoList(ti));
            double N = static_cast<double>(indx->docCount());
            double n = static_cast<double>(dlist->docFreq());
            double termweight = log (N/n);

```

```

        orderedtermweights.push_back(termweight);
        delete dlist;
        counter->incCount(ti,1);
        if (static_cast<int>(counter->count((int)ti)>1)) continue;
        vRefId.push_back(ti);
    } else
        orderedtermweights.push_back(0.0);
    }
    sort(vRefId.begin(),vRefId.end());

    calIDF(vRefId,dbx);
    calIDFDD();
    transformQuery();
    logf.close();
}

void lemur::retrieval::TermweightQueryRep::calIDFDD(){
    termdirections.clear();
    double result = orderedtermweights[1]-orderedtermweights[0];
    result>0 ? termdirections.push_back(DOWNUP):termdirections.push_back(UPDOWN);

    for (int i=1; i<orderedtermids.size()-1; i++){
        result = orderedtermweights[i+1] - orderedtermweights[i];
        if (result>0){
            if (termdirections[i-1]==DOWN || termdirections[i-1]==UPDOWN)
                termdirections.push_back(DOWNUP);
            else
                termdirections.push_back(UP);
        } else{
            if (termdirections[i-1]==UP || termdirections[i-1]==DOWNUP)
                termdirections.push_back(UPDOWN);
            else
                termdirections.push_back(DOWN);
        }
    }
    result = orderedtermweights[orderedtermweights.size()-1]-orderedtermweights
[orderedtermweights.size()-2];
    result>0 ? termdirections.push_back(UPDOWN):termdirections.push_back(DOWNUP);

    multimap<double, int, greater<double>> rankedterms;

    for (int i=0; i<orderedtermids.size();i++)
    {
        orderedresults.push_back(false);
        ordereddists.push_back(0.0);
        if (termdirections[i]==4){
            pair<double, int> paer(orderedtermweights[i],i);
            rankedterms.insert(paer);
        }
    }

    vector<int> orderedpeaks;
    multimap<double,int,greater<double>>::const_iterator iter = rankedterms.begin();
    int maxpeaks = orderedtermids.size()/5, cntpeak = 0;
    while (iter!=rankedterms.end() && cntpeak<=maxpeaks) {
        int index = (*iter).second;
        double w = (*iter).first;
        if (w>0.6*(maxval) && termdirections[index]==UPDOWN){
            bool fakeone1 = (index-2>=0 && termdirections[index-1]==DOWNUP && orderedtermweights
[index]-orderedtermweights[index-1]<1 && index+1<orderedtermids.size() && orderedtermweights[index]
<orderedtermweights[index-2] && termdirections[index+1]!=DOWNUP);
            bool fakeone2 = (index+2<orderedtermids.size() && termdirections[index+1]==DOWNUP &&
orderedtermweights[index]-orderedtermweights[index+1]<1 && index-1>0 && orderedtermweights[index]
<orderedtermweights[index+2] && termdirections[index-1]!=DOWNUP);
            bool fakeone3 = (index-1>0 && index+1<orderedtermids.size() && termdirections[index-1]
==DOWNUP && termdirections[index+1]==DOWNUP && orderedtermweights[index]-orderedtermweights[index-
1]<1 && orderedtermweights[index]-orderedtermweights[index+1]<1);
            if (fakeone1 || fakeone2 || fakeone3) {

```

```

        iter++;
        continue;
    }
    cntpeak++;
    termdirections[index] = PEAK;
    orderedpeaks.push_back(index);
    cerr<<getName(index)<<"...inserted["<<index<<"]"<<endl;
}
else {
    iter++;
    continue;
}
iter++;
}
cerr<<"# of peaks"<<orderedpeaks.size()<<endl;

int post = -1;
int bpost = -1;

double horzlen = orderedtermids.size()+1;
double bhorzlen = orderedtermids.size()+1;
int beginmarker = 0, endmarker = -1;
vector<int> peaklist;
bool lastround = false, firstround = false;
vector<int> skippeak;
int round = 0;

sort(orderedpeaks.begin(), orderedpeaks.end());

double minpeakweight = 100.0;
int winlen = 0, maxpeak = 0;
for (int i=0; i<orderedpeaks.size();i++){
    if (orderedtermweights[orderedpeaks[i]]==maxval){
        maxpeak = i;
        break;
    }
}

double pausethreshold = 1.0;

for (int i=0; i<orderedpeaks.size();i++){
    int priorbound = 0, postbound = 0;
    double priorboundweight = 100.0, postweight = 100.0;
    int firsthalf = 0, lasthalf = 0;
    int j = orderedpeaks[i];

    if (minpeakweight>orderedtermweights[orderedpeaks[i]]){
        minpeakweight = orderedtermweights[orderedpeaks[i]];
    }

    if (i==0){
        priorbound = -1;
        firsthalf = j;

        for (int k=0; k<j;k++)
            if (orderedtermweights[k]<pausethreshold)
                firsthalf--;
        if (orderedpeaks.size()==1){
            postbound = orderedtermids.size();
            lasthalf = postbound-1-j;
            for (int k=j+1; k<postbound;k++)
                if (orderedtermweights[k]<pausethreshold)
                    lasthalf--;
        } else{
            for (int k=j; k<orderedpeaks[i+1]; k++){
                if (orderedtermweights[k]<postweight){
                    postweight = orderedtermweights[k];
                    postbound = k;
                }
            }
        }
    }
}

```

```

        }
    }
    lasthalf = postbound - j;
    for (int k=j+1; k<=postbound;k++)
        if (orderedtermweights[k]<pausethreshold)
            lasthalf--;
    }
}
else if (i==orderedpeaks.size()-1){
    for (int k=j; k>orderedpeaks[i-1]; k--){
        if (orderedtermweights[k]<priorboundweight){
            priorboundweight = orderedtermweights[k];
            priorbound = k;
        }
    }
    firsthalf = j - priorbound;
    for (int k=priorbound; k<j;k++)
        if (orderedtermweights[k]<pausethreshold)
            firsthalf--;

    postbound = orderedtermids.size();
    lasthalf = postbound - j;
    for (int k=j+1; k<postbound;k++)
        if (orderedtermweights[k]<pausethreshold)
            lasthalf--;
}
else {
    for (int k=j; k>orderedpeaks[i-1]; k--){
        if (orderedtermweights[k]<priorboundweight){
            priorboundweight = orderedtermweights[k];
            priorbound = k;
        }
    }
    firsthalf = j - priorbound;
    for (int k=priorbound; k<j;k++)
        if (orderedtermweights[k]<pausethreshold)
            firsthalf--;

    for (int k=j; k<orderedpeaks[i+1]; k++){
        if (orderedtermweights[k]<postweight){
            postweight = orderedtermweights[k];
            postbound = k;
        }
    }
    lasthalf = postbound - j-1;
    for (int k=j+1; k<=postbound; k++)
        if (orderedtermweights[k]<pausethreshold)
            lasthalf--;
}
int range = firsthalf+lasthalf;
if (winlen<range)
    winlen = range;
}

double normalizefactor;
double framernormfactor;
double peakdist = 0.0;
double alpha = 0, falpha = 0;
switch (calweightOpt)
{
    case SIMPLEIDF:
        if (maxval==minpeakweight)
            minpeakweight = minval;
        normalizefactor = maxval-minpeakweight;
        framernormfactor = maxval-minval;
        break;
    case PITHAGORAS:

```

c:\Lem\src\lemur-4.2\retrieval\include\TermweightQueryRep.hpp

6

```

        normalizefactor = sqrt(orderedpeaks.size()*orderedpeaks.size()+
minpeakweight)*(maxval-minpeakweight));
        //framenormfactor = sqrt(winlen*winlen + (maxval-minval)*(maxval-minval));
        framenormfactor = sqrt(winlen*winlen + (maxval-pausedthreshold)*(maxval-
pausedthreshold));

        break;
    case INTERPOLATE:
        alpha = orderedpeaks.size()/(orderedpeaks.size()+
(maxval-minpeakweight));
        normalizefactor = alpha*orderedpeaks.size()+
(1-alpha)*(maxval-minpeakweight);

        /*falpha = winlen/(maxval-minval+winlen);
        framenormfactor = falpha*winlen + (1-falpha)*(maxval-minval);
        */
        falpha = winlen/(maxval-pausedthreshold+winlen);
        framenormfactor = falpha*winlen + (1-falpha)*(maxval-pausedthreshold);
        break;
    default:
        alpha = orderedpeaks.size()/(orderedpeaks.size()+
(maxval-minpeakweight));
        normalizefactor = alpha*orderedpeaks.size()+
(1-alpha)*(maxval-minpeakweight);
        //framenormfactor = alpha*winlen + (1-alpha)*(maxval-minval);
        //falpha = winlen/(maxval+winlen);
        falpha = winlen/(maxval-pausedthreshold+winlen);
        framenormfactor = falpha*winlen + (1-falpha)*(maxval-pausedthreshold);
        calweightOpt = INTERPOLATE;
        break;
}

// Compute PEAK term prob
vector<double> peakprob;

for (int j=0; j<orderedpeaks.size();j++){
    int i = orderedpeaks[j];
    int frommax = 0;
    double weightdif = maxval-orderedtermweights[i];
    if (orderedtermweights[i]== maxval){
        maxpeak = j;
        frommax = 0;
    } else
        frommax = abs(maxpeak-j);
    if (calweightOpt==INTERPOLATE||calweightOpt==SIMPLEIDF)
        peakdist = alpha*frommax+(1-alpha)*weightdif;
    else
        peakdist = sqrt(frommax*frommax+weightdif*weightdif);
    double prob = 1-(peakdist/normalizefactor);
    peakprob.push_back(prob);
}

int inelements = 0, outelements = 0;
for (int j=0; j<orderedpeaks.size(); j++){
    int i = orderedpeaks[j];
    inelements++;
    int p = i;

    bool truebound = false, bound = false;
    int priorbound = 0, postbound = 0;
    double priorboundweight = 100.0, postweight = 100.0;
    int firsthalf = 0, lasthalf = 0;

    if (j==0){
        priorbound = -1;
        firsthalf = i;
        firsthalf = i - priorbound;

        if (orderedpeaks.size()==1){
            postbound = orderedtermids.size();
            lasthalf = postbound - p;

```



```

    }
    else {
        for (int k=p; k<orderedpeaks[j+1]; k++){
            if (orderedtermweights[k]<postweight){
                postweight = orderedtermweights[k];
                postbound = k;
            }
        }
        lasthalf = postbound - p;
    }
}
else if (j==orderedpeaks.size()-1){
    for (int k=p; k>orderedpeaks[j-1]; k--){
        if (orderedtermweights[k]<priorboundweight){
            priorboundweight = orderedtermweights[k];
            priorbound = k;
        }
    }
    firsthalf = i - priorbound;

    postbound = orderedtermids.size();
    lasthalf = postbound - p;
}
else {
    for (int k=p; k>orderedpeaks[j-1]; k--){
        if (orderedtermweights[k]<priorboundweight){
            priorboundweight = orderedtermweights[k];
            priorbound = k;
        }
    }
    firsthalf = i - priorbound;
    for (int k=p; k<orderedpeaks[j+1]; k++){
        if (orderedtermweights[k]<postweight){
            postweight = orderedtermweights[k];
            postbound = k;
        }
    }
    lasthalf = postbound - p;
}

//Compute satellite term prob
int range = firsthalf+lasthalf;
double normalize = framernormfactor;
int gaps = 0;
while (!bound&&gap>=0){

    double childprob = 0.0;
    if (!truebound){
        double wdif = orderedtermweights[i] - orderedtermweights[p];
        double pdif = abs(i-p);
        pdif -= gaps;
        double ddif = 0.0;
        if (calweightOpt==INTERPOLATE||calweightOpt==SIMPLEIDF)
            ddif = falpha*pdif+(1-falphi)*wdif;
        else
            ddif = sqrt(wdif*wdif+pdif*pdif);
        childprob = (1-(ddif/normalize))*peakprob[j];
        ordereddists[p] = childprob;
        inelements++;
    } else
        gaps++;

    if (p==0)
        break;
    p--;
    truebound = orderedtermweights[p]<pausethreshold;
    bound = p == priorbound;
}

```

```

    }
    p = i;
    bound = false;

    gaps = 0;
    while (!bound && p < orderedtermids.size() - 1) {

        double childprob = 0.0;
        double pdif = 0;
        double ddif = 0;
        p++;
        truebound = orderedtermweights[p] < pausethreshold;
        if (!truebound) gaps++;
        bound = p == postbound;
        if (!truebound) {
            double wdif = orderedtermweights[i] - orderedtermweights[p];
            pdif = abs(i - p);
            pdif -= gaps;
            if (calweightOpt == INTERPOLATE || calweightOpt == SIMPLEIDF)
                ddif = falpha * pdif + (1 - falpha) * wdif;
            else
                ddif = sqrt(wdif * wdif + pdif * pdif);
            childprob = (1 - (ddif / normalize)) * peakprob[j];
            orderedreddists[p] = childprob;
            inelements++;
        }
        bound = p == postbound;
    }
    endmarker = p;
}

for (int i = 0; i < orderedreddists.size(); i++)
    orderedresults[i] = orderedreddists[i] > 0;

cerr << endl << endl;
if (inelements == 0) {
    if (orderedresults.size() > 0) {
        for (int k = 0; k < orderedresults.size(); k++) {
            orderedresults[k] = true;
            orderedreddists[k] = 1000.0;
        }
    } else {
        for (int k = 0; k < orderedtermids.size(); k++) {
            orderedresults.push_back(true);
            orderedreddists.push_back(1000.0);
        }
    }
}
}

lemur::retrieval::TermweightQueryRep::~TermweightQueryRep()
{
    if (counter != NULL)
        delete counter;
}

void lemur::retrieval::TermweightQueryRep::transformQuery()
{
    string fname = indxpath + "testrun8nostops.qry";
    string fnameW = indxpath + "termweight_testrun8nostops.qry";

    ofstream ofile(fname.c_str(), ios_base::out | ios_base::ate | ios_base::app);
    ofstream ofileW(fnameW.c_str(), ios_base::out | ios_base::ate | ios_base::app);

    if (!ofile)
        cerr << "error creating summary file " << endl;

    ofile.seekp(0, ios_base::end);

```

```

ofile<<"<DOC "<<curqueryid<<">"<<endl;

ofileW.seekp(0,ios_base::end);
ofileW<<"<QUERY "<<curqueryid<<">"<<endl;

logf<<endl<<"***** SUMMARY ["<<curqueryid<<"] *****"<<endl<<endl;

for (int k=0; k<orderedresults.size(); k++){
    map<int,double>::const_iterator iter = refWeight.find(orderedtermids[k]);
    if (iter!=refWeight.end()){

        double wght = (*iter).second;
        if (orderedresults[k]){
            ofile<<getName(k)<<"\t"<<ordereddist[k]<<endl;
            ofileW<<getName(k)<<"\t"<<true<<"\t"<<wght<<"\t"<<ordereddist[k]<<"\t"<
<termdirections[k]<<endl;
            cerr<<getName(k)<<"\t"<<true<<"\t"<<ordereddist[k]<<endl;
            logf<<getName(k)<<"\t"<<true<<endl;
        } else {
            ofileW<<getName(k)<<"\t"<<false<<"\t"<<wght<<"\t"<<ordereddist[k]<<"\t"<
<termdirections[k]<<endl;
            cerr<<getName(k)<<"\t"<<false<<"\t"<<ordereddist[k]<<endl;
            logf<<getName(k)<<"\t"<<false<<endl;
        }
    }
}

ofile<<"</DOC>"<<endl;
ofile.close();
ofileW<<"</QUERY>"<<endl;
ofileW.close();
}

void lemur::retrieval::TermweightQueryRep::calIDF(std::vector<int> refList, const lemur::api::Index &
&dbIndex){
    const lemur::index::InvFPIndex * indx = dynamic_cast<const lemur::index::InvFPIndex*> (&
    dbIndex);
    vector<double>lstweight;
    maxval = 0.0;
    minval = 1000.0;

    double freq = 0.0;
    int id;

    for (int i=0; i<refList.size(); i++){
        id = refList[i];
        freq = counter->count(id);
        lemur::index::InvFPDocList* dlist = dynamic_cast<lemur::index::InvFPDocList*>(indx->
        docInfoList(id));

        double N = static_cast<double>(indx->docCount());
        double n = static_cast<double>(dlist->docFreq());
        delete dlist;

        double weight = log (N/n);

        if (weight>maxval)
            maxval = weight;

        if (weight<minval)
            minval = weight;

        lstweight.push_back(weight);
        pair<int,double> paer(refList[i],weight);
        refWeight.insert(paer);

        cerr<<getNamefromid(id)<<": "<<weight<<endl;
        logf<<getNamefromid(id)<<": "<<weight<<endl;
    }
}

```

```
}  
}
```



สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

c:\Lem\src\lemur-4.2\utility\include\BasicDocStream.hpp

1

```

/*=====
 * WeightedDocStream and WeightedTokenDoc authored by Sompong Kittinaradorn
 * for a master thesis research at Chulalongkorn University's Faculty of
 * Engineering
 *
 * Copyright (c) 2001 Carnegie Mellon University. All Rights Reserved.
 *
 * Use of the Lemur Toolkit for Language Modeling and Information Retrieval
 * is subject to the terms of the software license set forth in the LICENSE
 * file included with this software, and also available at
 * http://www.lemurproject.org/license.html
 *
 *=====
 */

#ifndef _BASICFILESTREAM_HPP
#define _BASICFILESTREAM_HPP
#include "common_headers.hpp"
#include <cassert>
#include <cstdio>
#include <cstring>
#include "DocStream.hpp"
#include "Exception.hpp"

namespace lemur
{
    namespace parse
    {
        /// A basic DocStream implementation
        /*!
        BasicDocStream is an implementation of DocStream that recognizes
        the following format:

        <PRE>
        &lt;DOC unique_document_identifier&gt;
        this
        is
        an
        example
        &lt;/DOC&gt;
        </PRE>

        The following is a typical example of using BasicDocStream(or DocStream)
        :

        <PRE>
        DocStream *docStr = new BasicDocStream("source");
        docStr->startDocIteration();

        while (docStr->hasMore()) {

        Document *doc = docStr->nextDoc();
        cout << "doc id: " << doc->getID() << endl;
        doc->startTermIteration();
        while (doc->hasMore()) {
        TokenTerm *term = thisDoc->nextTerm();
        cout << "term: " << term->spelling() << endl;
        }
        }
        </PRE>
        */
    }
}

```



```

#define MAXLINE 65536

/// doc representation for BasicDocStream

class BasicTokenDoc : public lemur::api::Document {
public:
    BasicTokenDoc() {
    }
    BasicTokenDoc(ifstream *stream): docStr(stream) {
    }
    void startTermIteration() const;

    const char *getID() const { return id;}

    bool hasMore() const{ return (strcmp(curWord, "</DOC>") != 0);}

    const lemur::api::Term * nextTerm() const;

    void skipToEnd() const;
    friend class BasicDocStream;
private:
    void readID();
    mutable char *curWord;
    mutable char buf1[20000];
    mutable char buf2[20000];
    char id[2000];
    ifstream *docStr;
    streampos startPos; // starting position of the terms in the file
    //replace static BasicTokenTerm t; with attribute
    mutable lemur::api::Term t;
};

/// A DocStream handler of a stream with the basic lemur format
class BasicDocStream : public lemur::api::DocStream
{
public:
    BasicDocStream() {}
    BasicDocStream (const string &inputFile);

    virtual ~BasicDocStream() { delete ifs;}

public:

    bool hasMore();

    void startDocIteration();

    lemur::api::Document *nextDoc();

private:
    char file[1024];
    ifstream *ifs;
    char buf[2000];
    bool nextTokenRead;
    // replace static BasicTokenDoc doc; with attribute
    BasicTokenDoc doc;
};

class WeightedTokenDoc : public lemur::api::Document {
public:
    WeightedTokenDoc() {
    }
    WeightedTokenDoc(ifstream *stream): docStr(stream) {
    }
    void startTermIteration() const;

```

```

    const char *getID() const { return id;}

    bool hasMore() const{ return (strcmp(curWord, "</DOC>") != 0);}

    const lemur::api::Term * nextTerm() const;

    void skipToEnd() const;
    friend class WeightedDocStream;
private:
    void readID();
    mutable char *curWord;
    mutable double curWeight;
    mutable char buf1[20000];
    mutable char buf2[20000];
    char id[2000];
    ifstream *docStr;
    streampos startPos; // starting position of the terms in the file
    //replace static BasicTokenTerm t; with attribute
    mutable lemur::api::Term t;
};

// A DocStream handler of a stream with the basic lemur format
class WeightedDocStream : public lemur::api::DocStream
{
public:
    WeightedDocStream() {}
    WeightedDocStream (const string &inputFile);

    virtual ~WeightedDocStream() { delete ifs;}

public:

    bool hasMore();

    void startDocIteration();

    lemur::api::Document *nextDoc();

private:
    char file[1024];
    ifstream *ifs;
    char buf[2000];
    bool nextTokenRead;
    // replace static BasicTokenDoc doc; with attribute
    WeightedTokenDoc doc;
};
}
#endif

/*=====
 * Copyright (c) 2001 Carnegie Mellon University. All Rights Reserved.
 *
 * Use of the Lemur Toolkit for Language Modeling and Information Retrieval
 * is subject to the terms of the software license set forth in the LICENSE
 * file included with this software, and also available at
 * http://www.lemurproject.org/license.html
 *
 *=====
 */

#include <cstring>
#include <cctype>
#include <cassert>
#include "BasicDocStream.hpp"

```

```

void lemur::parse::BasicTokenDoc::startTermIteration() const
{
    // ensure the start position of the terms
    docStr->seekg(startPos);
    curWord = buf1;
    // peek one term
    *docStr >> curWord;
}

void lemur::parse::BasicTokenDoc::skipToEnd() const
{
    startTermIteration();
    while (hasMore()) {
        nextTerm();
    }
}

const lemur::api::Term * lemur::parse::BasicTokenDoc::nextTerm() const
{
    // static BasicTokenTerm t;
    t.spelling(curWord);
    if (curWord == buf1) {
        curWord = buf2;
    } else {
        curWord = buf1;
    }
    *docStr >> curWord;
    return &t;
}

void lemur::parse::BasicTokenDoc::readID()
{
    // get id
    *docStr >> id;
    int len= strlen(id);
    if (id[len-1]!='>') {
        throw lemur::api::Exception("BasicTokenDoc", "ill-formatted doc id, > expected");
    }
    id[len-1]='\0';
    startPos = docStr->tellg(); // record the start position of terms
}

lemur::parse::BasicDocStream::BasicDocStream (const string &inputFile)
{
    strcpy(file, inputFile.c_str());
    ifs = new ifstream(file, ios::in);
    if (ifs->fail() ) {
        throw lemur::api::Exception("BasicDocStream", "can't open BasicDocStream source file");
    }
}

bool lemur::parse::BasicDocStream::hasMore()
{
    bool moreDoc = false;
    if (!nextTokenRead) {
        moreDoc = *ifs >> buf;
        nextTokenRead = true;
        if (moreDoc && strcmp(buf, "<DOC")) {
            cerr << " actual token seen: "<< buf << endl;
            throw lemur::api::Exception("BasicDocStream", "begin doc marker expected");
        }
    }
}

return moreDoc;

```

```

}

void lemur::parse::BasicDocStream::startDocIteration()
{
    ifs->close();
    ifs->open(file);
    ifs->seekg(0);
    ifs->clear();
    nextTokenRead = false;
}

lemur::api::Document *lemur::parse::BasicDocStream::nextDoc()
{
    // fails to initialize properly, preventing reuse of a
    // BasicDocStream (or opening more than one).
    // static BasicTokenDoc doc(ifs);
    // make it an instance attribute
    // static BasicTokenDoc doc;
    doc.docStr = ifs;
    doc.readID();
    nextTokenRead = false;
    return &doc;
}
//*****
void lemur::parse::WeightedTokenDoc::startTermIteration() const
{
    // ensure the start position of the terms
    docStr->seekg(startPos);
    curWord = buf1;
    // peek one term
    *docStr >> curWord;
    // curWeight = 0;
    *docStr >> curWeight;
    // cerr<<"curWord from startterminteration is "<<curWord<<" and curweight is "<<curWeight<
    <endl;
}

void lemur::parse::WeightedTokenDoc::skipToEnd() const
{
    startTermIteration();
    while (hasMore()) {
        nextTerm();
    }
}

const lemur::api::Term * lemur::parse::WeightedTokenDoc::nextTerm() const
{
    // static WeightedTokenTerm t;
    t.spelling(curWord);
    t.weighting(curWeight);
    if (curWord == buf1) {
        curWord = buf2;
    } else {
        curWord = buf1;
    }
    *docStr >> curWord;
    //curWeight = 0;
    if (strcmp(curWord, "</DOC>"))
        *docStr >> curWeight;
    //cerr<<"curWord from nextterm is "<<curWord<<" and curweight is "<<curWeight<<endl;
    return &t;
}

void lemur::parse::WeightedTokenDoc::readID()
{
    // get id
    *docStr >> id;
}

```

```

int len= strlen(id);
if (id[len-1]!='>') {
    throw lemur::api::Exception("WeightedTokenDoc","ill-formatted doc id, > expected");
}
id[len-1]='\0';
startPos = docStr->tellg(); // record the start position of terms
}

lemur::parse::WeightedDocStream::WeightedDocStream (const string &inputFile)
{
    strcpy(file, inputFile.c_str());
    ifs = new ifstream(file, ios::in);
    if (ifs->fail() ) {
        throw lemur::api::Exception("WeightedDocStream", "can't open WeightedDocStream source file");
    }
}

bool lemur::parse::WeightedDocStream::hasMore()
{
    bool moreDoc = false;
    if (!nextTokenRead) {
        moreDoc = (*ifs >> buf);
        nextTokenRead = true;
        cerr << " actual token seen: " << buf << endl;
        if (moreDoc && strcmp(buf, "<DOC")) {
            cerr << " actual token seen: " << buf << endl;
            throw lemur::api::Exception("WeightedDocStream", "begin doc marker expected");
        }
    }

    return moreDoc;
}

void lemur::parse::WeightedDocStream::startDocIteration()
{
    ifs->close();
    ifs->open(file);
    ifs->seekg(0);
    ifs->clear();
    nextTokenRead =false;
}

lemur::api::Document *lemur::parse::WeightedDocStream::nextDoc()
{
    // fails to initialize properly, preventing reuse of a
    // BasicDocStream (or opening more than one).
    // static BasicTokenDoc doc(ifs);
    // make it an instance attribute
    // static BasicTokenDoc doc;
    doc.docStr = ifs;
    doc.readID();
    nextTokenRead = false;
    return &doc;
}

```


c:\Lem\src\lemur-4.2\app\src\RetEval.cpp

1

```

/*=====
 * Modified by Sompong Kittinaradorn
 * Copyright (c) 2001 Carnegie Mellon University. All Rights Reserved.
 *=====
 */
#include "common_headers.hpp"
#include "IndexManager.hpp"
#include "BasicDocStream.hpp"
#include "RetMethodManager.hpp"
#include "ResultFile.hpp"
#include "TmpInvFPIndex.hpp"

using namespace lemur::api;

void GetAppParam()
{
    RetrievalParameter::get();
    // for rel model test.
    SimpleKLParameter::get();
}

int AppMain(int argc, char *argv[]) {

    Index *baseind, *ind;
    bool isMyType = atoi(argv[2]);
    bool isWeightedDoc = atoi(argv[3]);
    string mode = argv[4], indexpath;

    if (mode=="-w")
        indexpath = "D:\\TrecDatabase\\workspace\\";
    else if (mode=="-s")
        indexpath = "D:\\TrecD4\\FT\\data_sentencemode\\";
    else indexpath = mode;
    cerr<<"Indexpath:"<<indexpath<<endl;
    if (isMyType)
        cerr<<"Evaluating termset ad hoc retrieval"<<endl;
    else cerr<<"Evaluating single term retrieval"<<endl;

    try {
        baseind = IndexManager::openIndex(RetrievalParameter::databaseIndex);
    }
    catch (Exception &ex) {
        ex.writeMessage();
        throw Exception("RelEval", "Can't open index, check parameter index");
    }
    DocStream *qryStream;
    try {
        if (isWeightedDoc)
            qryStream = new lemur::parse::WeightedDocStream(RetrievalParameter::textQuerySet);
        else
            qryStream = new lemur::parse::BasicDocStream(RetrievalParameter::textQuerySet);
    }
    catch (Exception &ex) {
        ex.writeMessage(cerr);
        throw Exception("RetEval",
            "Can't open query file, check parameter textQuery");
    }
}

ofstream result(RetrievalParameter::resultFile.c_str());

ResultFile resFile(RetrievalParameter::TRECresultFileFormat);

ifstream *workSetStr;
ResultFile *docPool;
qryStream->startDocIteration();
TextQuery *q;

```

c:\Lem\src\lemur-4.2\app\src\RetEval.cpp

2

```

IndexedRealVector workSetRes;
bool ignoreWeights = true;
bool doingRelModel = (RetrievalParameter::fbDocCount > 0 &&
    RetrievalParameter::retModel == "k1" &&
    (SimpleKLPParameter::qryPrm.fbMethod ==
        SimpleKLPParameter::RM1 ||
        SimpleKLPParameter::qryPrm.fbMethod ==
        SimpleKLPParameter::RM2));

while (qryStream->hasMore()) {
    Document *d = qryStream->nextDoc();
    //cerr<<"to go "<<qryStream->hasMore()<<endl;;

    q = new TextQuery(*d);
    cout << "query : "<< q->id() << endl;
    if (!isMyType)
        ind = baseind;
    else
    {
        // need input for indexname
        cerr<<"latest version"<<endl;
        //string indexpath =
        string tmpindexname = indexpath + q->id() + ".tmp";
        cout<<RetrievalParameter::workSetFile<<endl;

        try {
            ind = IndexManager::openIndex(tmpindexname);
        }
        catch (Exception &ex) {
            ex.writeMessage();
            throw Exception("RetEval", "Can't open index, check parameter index");
        }
        lemur::index::TmpInvFPIndex* tmpind = dynamic_cast<lemur::index::TmpInvFPIndex*>
(ind);
        //lemur::index::TmpInvFPIndex* tmpind = new lemur::index::TmpInvFPIndex
(tmpindexname);
        tmpind->setBaseIndex((lemur::index::InvFPIndex*)baseind);
        ind = tmpind;
    }

    resFile.openForWrite(result, *ind);

    if (RetrievalParameter::useWorkingSet) {
        workSetStr = new ifstream(RetrievalParameter::workSetFile.c_str(), ios::in);
        if (workSetStr->fail()) {
            throw Exception("RetEval", "can't open working set file");
        }
        docPool = new ResultFile(false); // working set is always simple format
        docPool->openForRead(*workSetStr, *ind);
    }
    lemur::retrieval::ArrayAccumulator accumulator(ind->docCount());
    IndexedRealVector results(ind->docCount());
    RetrievalMethod *model = NULL;
    model = RetMethodManager::createModel(ind, &accumulator,
        RetrievalParameter::retModel);
    QueryRep *qr = model->computeQueryRep(*q);
    PseudoFBDocs *workSet;

    if (RetrievalParameter::useWorkingSet) {
        docPool->getResult(q->id(), workSetRes);
        workSet = new PseudoFBDocs(workSetRes, -1); // -1 means using all docs
        model->scoreDocSet(*qr, *workSet, results);
    } else {
        model->scoreCollection(*qr, results);
    }
    results.Sort();
}

```

c:\Lem\src\lemur-4.2\app\src\RetEval.cpp

3

```

// prune to number of feedback docs.
if (RetrievalParameter::fbDocCount > 0 &&
    results.size() > RetrievalParameter::fbDocCount)
    results.erase(results.begin() + RetrievalParameter::fbDocCount,
                  results.end());
if (doingRelModel) {
    if (SimpleKLParameter::qryPrm.adjScoreMethod !=
        SimpleKLParameter::QUERYLIKELIHOOD) {
        throw Exception("RetEval:FB",
                        "Relevance models require query likelihood scores.");
    }
    ignoreWeights = false;
    results.LogToPosterior();
}
if (RetrievalParameter::retModel == "indri")
    ignoreWeights = false;
if (RetrievalParameter::fbDocCount > 0) {
    PseudoFBDocs *topDoc = new PseudoFBDocs(results,
                                             RetrievalParameter::fbDocCount,
                                             ignoreWeights);

    model->updateQuery(*qr, *topDoc);

    if (RetrievalParameter::useWorkingSet) {
        model->scoreDocSet(*qr, *workSet, results);
    } else {
        model->scoreCollection(*qr, results);
    }
    results.Sort();
    delete topDoc;
}
resFile.writeResults(q->id(), &results, RetrievalParameter::resultCount);

if (RetrievalParameter::useWorkingSet) {
    delete workSet;
}
delete qr;
delete q;
if (RetrievalParameter::useWorkingSet) {
    delete docPool;
    delete workSetStr;
}
delete model;
if (isMyType)
    delete ind;
}
result.close();
delete qryStream;

delete baseind;
return 0;
}

```

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

BIOGRAPHY

Personal Information:

Name (English): Sompong Kittinaradorn

Name (Thai): นาย สมพงษ์ กิตตินราดอร์

Birth Date: 20 April 1956

Address: 96/3, Moo 12, Manthana 2, Srinakharin Rd., Tambon Bangkaew, Bangplee District, Samut Prakarn Province, 10540, phone#: 08-9029-2968

Work Experience:

Motif Technology Public Company Limited (April, 2007 – present)

1. **Senior vice president for project management**, from April 2007

Nation Multimedia Group Plc (14, Oct, 1980 – 15, Jan, 2007):

2. **Senior vice president**, Central Office – Executive, from Jan 2002- Jan 2007
3. Senior editorial staff in charge of IT support for all editorials,
4. **Sub-editor**, *The Nation*
5. **General manager**, *Nation On-line*
6. **Editor**, Nation On-line, Set Daily
7. **Political news editor**, *The Nation's* editorial
8. **Reporter**, *The Nation's* editorial
9. **Rewriter**, *The Nation's* editorial

Sankei Shimbun, Bangkok bureau (Jan, 1980-Sept, 1980)

10. **Assistant Foreign Correspondent**, Bangkok Bureau of Japanese daily *Sankei Shimbun*.

Education & Training:

1. Bachelor degree (second honorary degree), Faculty of Communications Arts, Chulalongkorn University, majored in journalism, graduated in June, 1980
2. Fulbright professional scholarship, training at Center for Foreign Journalists, USA, three-month on-the-job internship at *Bermont Enterprise*, a community newspaper based in Texas, of *Hearst Corp*.
3. High school graduation in science from Triam Udomsuksa School (Phyathai).