BUILDING AN INTELLIGENT FILTERING SYSTEM USING IDEA INDEXING

Li Yang

Thesis Prepared for the Degree of

MASTER OF SCIENCE

UNIVERSITY OF NORTH TEXAS

August 2003

APPROVED:

Rada Mihalcea, Major Professor
Kathy Swigger, Committee Member
Robert Brazile, Committee Member
Krishna Kavi, Chair of the Department of Computer Sciences
C. Neal Tate, Dean of the Robert B. Toulouse School of Graduate Studies

The widely used vector model maintains its popularity because of its simplicity, fast speed, and the appeal of using spatial proximity for semantic proximity. However, this model faces a disadvantage that is associated with the vagueness from keywords overlapping. Efforts have been made to improve the vector model. The research on improving document representation has been focused on four areas, namely, statistical co-occurrence of related items, forming term phrases, grouping of related words, and representing the content of documents.

In this thesis, we propose the idea-indexing model to improve document representation for the filtering task in IR. The idea-indexing model matches document terms with the ideas they express and indexes the document with these ideas. This indexing scheme represents the document with its semantics instead of sets of independent terms. We show in this thesis that indexing with ideas leads to better performance.
ACKNOWLEDGMENTS

I feel that I owe a great deal to the computer science department at the University of North Texas. I have always dreamed to be an engineer. I will be graduating soon as a software engineer from UNT. If it were not for the computer science department, I would never have had a chance to be a computer science major. So, I would say that the computer science department made my dream come true. I am grateful to the department not only for the foregoing reason, but also for the great professors she gave me. Dr. Brazile, Dr. Tate, Dr. Fisher, Dr. Tarau, and Dr. Parberry have built in me a solid programming and theoretical background. I am thankful to Dr. Swigger for leading me into the field of natural language processing and for her encouraging words that helped me build up my confidence in this field. I have special thanks to Dr. Mihalcea not only because she helped me with my projects and theses but also because she pointed out the directions for my future career. Special thanks to Mr. Duncan for his valuable suggestions on my system programming knowledge.
4.2 Similarity Measured by Common Idea Terms ........................................... 26
4.3 Ranking ................................................................. 26
4.4 Seeking Idea Categories ......................................................... 27
  4.4.1 Roget’s International Thesaurus .............................................. 27
  4.4.2 Domain Labels ............................................................ 28
4.5 Hypothesis ................................................................. 29
4.6 Potential Problem with Idea Indexing .............................................. 29
4.7 Idea Indexing Example .......................................................... 30

5 IMPLEMENTATION ......................................................... 32
  5.1 Components of the Overall System .............................................. 32
  5.2 Indexing with Vectorial Terms .................................................. 34
  5.3 Indexing with Idea Categories .................................................. 34
  5.4 Combining Idea Indexing and Vector Models ............................... 36
  5.5 Evaluation Measures .......................................................... 36

6 EXPERIMENT AND RESULTS ............................................. 39
  6.1 Test Data ................................................................. 39
    6.1.1 TREC Topics ......................................................... 39
    6.1.2 Building User Profiles ............................................... 40
    6.1.3 Test Collection ...................................................... 40
    6.1.4 Relevance File ....................................................... 41
  6.2 Experiments ........................................................................... 41
  6.3 Results ............................................................................... 42
CHAPTER 1
INTRODUCTION

An Information Retrieval (IR) system performs two main tasks, *ad hoc* retrieval and the *filtering*. The former is a conventional IR task, where new user queries over a document collection keep arriving at the system while the document collection remains static in the system. The latter is a relatively newer task where user queries or profiles remain static at the system and new documents arrive at the system and are compared to the user profiles.

For the above definitions, the two tasks are distinct. However, they are similar tasks for four reasons. Firstly, both tasks build logical representations for document content. Second, query content in both tasks is also represented logically. Third, both tasks compute the similarities between documents and a user query. Finally, documents are ranked with regard to their similarity to the user query in both tasks.

Since the late 1960’s, research on the *ad hoc* task has yielded quite a few IR models, including the classic boolean, vector, and probabilistic models. Latest models on the Text REtrieval Conference (TREC) and the Topic Detection and Tracking (TDT) program display a trend in combining various statistical models to achieve increased system performance. Due to the four similarities between the *ad hoc* and the *filtering* tasks, the models built to handle the former task can be applied to the latter task as well, as shown in recent TREC and TDT participating systems.

Although a large number of models have been proposed, the number of the models that have been implemented and are still in use in commercial systems is very small. For more
than three decades, the vector model has been the most popular model, regardless of its disadvantages of assuming term independence and the use of term overlap. To improve the vector model, many efforts have been made to improve document/query representation, weighting methods, and similarity computing.

Research on improving document/query representation has been focusing on four areas since the 1980’s

- generating sets of related items based on the statistical co-occurrence of the words in certain contexts within the document collection
- forming term phrases consisting of one or more governing terms together with corresponding dependent terms
- using word grouping methods of the kind provided by thesauri, where classes of related words are grouped under common headings. These class headings can then be assigned for content identification instead of the individual terms contained in the classes.
- constructing knowledge bases to represent the content of the subject area under consideration. Entries from the knowledge bases are then used to represent the content of documents and queries.

Following research in the third and fourth areas above, we propose to build an idea indexing model for the filtering task. This model builds document/query representations by indexing the ideas expressed by the terms in documents/queries, contrary to vector model’s indexing all terms in documents. To match terms in documents with the idea(s) they belong to, the idea indexing model relies on either Roget’s idea categories or the Dewey Decimal
Category domain labels.

To test the idea indexing model, we ran experiments with the Reuter’s corpus used in the TREC 2002 filtering task. We also implemented the vector model, as a baseline system, and ran it with the same data set. Results from the experiments show that idea indexing gains higher recall and F-measure, with a tradeoff of lower precision.

In the chapters that follow, we describe the idea indexing model for the filtering task. Chapter 2 defines the filtering task and a general IR model. Chapter 3 describes the vector model and efforts made to improve it. Chapter 4 defines the idea indexing model. The implementation of idea indexing system is discussed in chapter 5. Experiments and results are discussed in chapter 6. Chapter 7 gives the conclusions and directions for future work.
CHAPTER 2

FILTERING TASK IN IR

With the expansion of the World Wide Web, modern Information Retrieval (IR) has seen its uses in two scenarios on the Internet. In the first scenario, a user searches information of interest online via some search engine. The search engine brings the interested information to the user. This is the "pulling" action of the World Wide Web. In this scenario, IR happens at the search engine side, where the search engine finds, computes, and returns the information that matches the user’s query. What is key to the search engine is its search system. In the second scenario, the Internet sends information towards a user, regardless of the user’s interests. This is the "pushing" action of the World Wide Web. We may compare the information pushed to the user with the junk mail in the user’s mailbox. IR comes into picture when it filters through the incoming information and only presents to the user those that he/she is interested in reading about [3, 4].

As the above two scenarios indicate to us, one type of IR system works at the remote site of some search engine. Another type of IR system is installed locally, performing the filtering task. The former system used to be more popular and hence more researched than the latter type. Different models have been developed in previous decades to handle the search system. However, in the following sections, after we give more formal description to both systems, we will show how technologies developed for the search system may also be applied to the filtering system without much difficulty, as filtering systems in recent Text REtrieval Conferences (TREC) and the Topic Detection and Tracking (TDT) program have
shown to us [5, 7]. Therefore, at the end of the chapter we give an overview of the major IR models for the search system.

2.1 Search System and Filtering System

An IR system is always built upon some IR model. Before we describe the search and filtering systems in more detail, we define a general IR model [3] in figure 2.1.

Definition

An information retrieval model is a quadruple \([D, Q, F, R]\) where

1. \(D\) is a set that consists of logical representations for the documents in a collection.
2. \(Q\) is a set that contains logical representations for the user information needs or queries.
3. \(F\) is a framework for operations on document representations queries, and for computing their relationships, i.e. similarities.
4. \(R(q_i, d_j)\) is a ranking function that defines an ordering among the documents with regard to a user query.

Figure 2.1: A general IR Model

Document and query representations are necessary to this model for the following reason. A document conveys ideas of its author to its reader. Sentences in a document follow specific logic flows. To process a document for IR purposes means to extract the ideas and logic flows in the document and represent them in a way that a IR system can process. Same applies for a user query. Hence, we have logical representations of documents and queries.

The third element of the definition expresses the central problem in IR systems. That is, determining which documents are relevant or which are not to a user query. The ranking
function figures out how much more relevant some documents are relevant than others to the query.

With the above definition in mind, we are ready to discuss more about the search and filtering systems. We give a high level description of a search system below referring to the above general IR model.

A search system is a conventional IR system. It uses the operational model as in figure 2.2.

1. The search system computes the logical representations of various documents and store the representations into a collection. This collection of document representations remain relatively static during the procedures of each retrieval task.
2. A query is submitted to the system.
3. The system computes the logical representation of the query.
4. After the system computes the similarities between a query and the collection, it ranks the documents that are computed to be similar to the query and returns the top ranking documents.

Figure 2.2: Search System Operational Model

While in a search system the collection remains relatively static and the query is dynamic, in a filtering system it is the query that remains relatively static and the collection are dynamic. We see this difference more in the following operational model of a filtering system in figure 2.3.

Despite documents and queries being static or dynamic in both systems, figures 2.2 and 2.3 show that both systems are very similar. First of all, both systems compute the logical representations of the documents and queries. Secondly, both compute similarities between the query and documents. Finally, both rank the documents and return the top ranking
1. The filtering system constructs a user profile describing the user’s interests.
2. The system computes the logical representation of the user profile. This representation remains relatively static during the filtering process.
3. The system computes the logical representations of various incoming documents.
4. The system compares the logical representation of the user profile to the document representations and computes their similarity.
5. The system ranks the incoming documents and presents to the user the top ranking ones.

Figure 2.3: Filtering System Operational Model

ones. Therefore, we can view a filtering system as a conventional search system in which the documents are the ones which keep arriving at the system.

2.2 Classic, Extended, and Latest IR Models

Due to the similarities between search and filtering systems discussed in section 2.1, traditional IR models developed for search systems can be applied to filtering systems. This application was first seen in systems participated in the TREC-1 filtering task in 1992, e.g. the CPGCN system [9]. More IR models have been used filtering systems in recent years. In this section, we give an informal overview of the three classic IR models, extensions to them, and latest IR models applied to filtering systems.

However, since document/query representation is a process common to different IR models, we first introduce a generic document representation method, as presented in [3].

2.2.1 Document/Query Representation Methodology

The components of a document/query representation are a set of index terms and their weights. Index terms may be either a set of representative keywords of a document or every
one of its words. The latter is referred to as full text indexing, whereas the former is known as the key term indexing. The weight of each index term is an assigned numerical value, which quantifies the importance of the index term for describing the document semantic contents.

Document Representation Definition Let $t$ be the number of index terms in the system and $k_i$ be a generic index term, $K = \{k_1, ..., k_t\}$ is the set of all index terms. A weight $w_{i,j} > 0$ is associated with each index term $k_i$ of a document $d_j$. For an index term which does not appear in the document text, $w_{i,j} = 0$. With the document $d_j$ is associated an index term vector $\vec{d}_j = (w_{1,j}, w_{2,j}, ..., w_{t,j})$. Further, let $g_i$ be a function that returns the weight associated with the index term $k_i$ in any $t$-dimensional vector (i.e., $g_i(\vec{d}_j) = w_{i,j}$).

While IR models share the above generic method for document/query representation, their approaches to representing documents and computing term weights and query-document similarity scores are different as in the models described in the next chapter.

2.2.2 IR Model Overview

The three classic models in information retrieval are called Boolean, vector, and probabilistic. Over the years, alternative modeling paradigms for each classic model have been proposed. New models have also been developed. In this section, we briefly review the main IR models.

Set Theoretic Models

The first of the three classic models is the Boolean model. Two extensions of the Boolean model are the Fuzzy model and the Extended Boolean model. Since the Boolean model
and its alternatives are based on classic set theory, we say that the Boolean model and its extensions are set theoretic models.

Probabilistic Models

Next on the classic-model list is the probabilistic model. Two extended models of the probabilistic model are the Inference Network and Belief Network. Since the probabilistic model and its extensions make a probabilistic interpretation of document relevance to a given user query, the probabilistic model and its alternative versions are probabilistic.

Algebraic Models

The Vector model is another traditional IR model. Three extended models of the vector model are the Generalized Vector model, Latent Semantic Indexing model, and the Neural Networks. The Vector model is based on representing documents and queries as vectors of index terms in a t-dimensional space. This approach wins for the vector model and its alternatives the name of algebraic model.

Latest Models in TREC and TDT

In recent years, both TREC and TDT invited research groups worldwide to participate in annual test runs on various TREC and TDT tasks. The TREC filtering task [7] and the TDT tracking task [5] are among these tasks. In fact, both tasks are so similar that we do not differentiate them here [1, 2].

In TREC and TDT filtering systems developed in recent years, we have seen a hybrid of statistical models. For example, CMU researchers use k-nearest-neighbors classification,
language models, decision trees, and a variant of the Rocchio approach for topic tracking. The Dragon Systems uses both Beta-Binomial and Unigram systems.

The latest models in TREC and TDT display the state-of-art technologies in text filtering. Some of the models are so successful that Allen states in [1] that they are ready to be developed as commercial products. What is the current status of the classic models and their alternatives? Among all the alternatives proposed for the classic models, only LSI has been made a product for automated essay assessment [8]. All other alternatives, being too complicated, still stay at the experimental stage. For the classic models, the Boolean model made itself to some early bibliographical products. The vector model still maintains its popularity among researchers, practitioners and the Web community.

In the next chapter, we discuss efforts made to improve filtering systems in terms of the four elements of the general IR model shown in figure 2.1.
CHAPTER 3

IMPROVING THE VECTOR MODEL

Built in 1970 with the SMART system by Salton [17], the vector model is one of the most widely used models in commercial IR applications due to its conceptual simplicity, fast speed, and the appeal of the underlying metaphor of using spatial proximity for semantic proximity [14, 3]. Later, more research has been done to improve the vector model in terms of weighting methods and document/query representation. Salton and Buckley in [18] studied various weighting methods for optimal retrieval results. LSI researchers proposed the singular value decomposition approach to representing the semantics of documents [8]. With conceptual indexing, Woods represented documents with the concepts in them [19]. These efforts to improve the vector model lend us ideas for our current work. Before we report our work, we review the work listed above.

3.1 The Vector Model

The SMART system is one of the first, and still best IR systems available. SMART was developed by Gerard Salton of Cornell University. It represents documents as a t-dimensional vector space. SMART performs automatic indexing by removing stop words from a predetermined list, stemming via suffix deletion, and weighting. Given a new query, it converts it to a vector, and then uses a similarity measure to compare it to the documents in the vector space. SMART ranks the documents, and returns the top $n$, where $n$ is a number determined by the user. SMART can perform relevance feedback based on the result of the
retrieval [17]. As we will see in the following sections, the back end of the SMART system is the vector model. Next, we formally define the vector model in terms of document/query representation, similarity computation, term weighting, and ranking.

3.1.1 Document/Query Representation

In the following definition of document/query representation, we see that both a document and a user query are represented as t-dimensional vectors [3, 14].

Definition Let $t$ be the number of index terms in the system and $k_i$ be a generic index term, $K = \{k_1, ..., k_t\}$ is the set of all index terms. A weight $w_{i,j} > 0$ is associated with each index term $k_i$ of a document $d_j$. The weight $w_{i,j}$ is positive and non-binary. The index terms in the query are also weighted. Let $w_{i,q}$ be the weight associated with the pair $[k_i, q]$, where $w_{i,q} \geq 0$. Then, the query vector $\vec{q}$ is defined as $\vec{q} = (w_{1,q}, w_{2,q}, ..., w_{t,q})$.

As before, the vector for a document $d_j$ is represented by $\vec{d}_j = (w_{1,j}, w_{2,j}, ..., w_{t,j})$.

In the above definition, the word term is used for both words and phrases. Also, term weights rather than word weights are used because dimensions in the vector space can correspond to phrases as well as words.

3.1.2 Similarity as Correlation

The vector model evaluates the degree of similarity between document $d_j$ and query $q$ as the correlation between the vectors $\vec{d}_j$ and $\vec{q}$. This correlation can be usually quantified by the cosine of the angle between these two vectors [3, 14]. The formula below gives this cosine measure.
sim(d_j, q) = \frac{d_j \cdot q}{|d_j| \times |q|} = \frac{\sum_{i=1}^{T} w_{i,j} \times w_{i,q}}{\sqrt{\sum_{i=1}^{T} w_{i,j}^2} \times \sqrt{\sum_{j=1}^{Q} w_{i,j}^2}}.

We illustrate this similarity correlation with the example in figure 3.1.2.

![Figure 3.1: A vector space with two dimensions. The two dimensions correspond to the terms vector and model. One query and three documents are represented in the space](image)

In figure 3.1.2, we show a vector space with two dimensions, corresponding to the words vector and model. The entities represented in the space are the query q represented by the vector (0.60, 0.64), and three documents d1, d2, and d3 with the following coordinates: (0.12, 0.86), (0.92, 0.56), and (0.90, 0.26). The elements of the vectors or the coordinates are term weights calculated with the method discussed in the next section. If we plug these
numbers into the formula shown earlier in this section, the cosine of the angle between \( q \) and \( d_2 \) is larger than the cosines of the angles between \( q \) and \( d_1 \) and between \( q \) and \( d_3 \). This is because the angle between \( q \) and \( d_2 \) in figure 3.1.2 is the smallest. The cosine formula quantifies the similarity correlation between the query and documents.

We can see from this example that term weights are used to quantify the semantic importance of document/query terms.

3.1.3 Term Weights as Semantic Indicators

From the document/query definition and the vector model example, we see that terms are weighted. We look at term weighting in more detail here. The basic information used in term weighting is term frequency (tf), document frequency (df), and sometimes collection frequency.

The information that is captured by term frequency is how salient a word is within a given document. In other words, term frequency provides one measure of how well that term describes the document contents. The higher the term frequency the more likely it is that the word is a good description of the content of the document.

Document frequency can be interpreted as an indicator of informativeness. A semantically focussed word will often occur several times in a document if it occurs at all. Semantically unfocused words are spread out homogeneously over all documents. Terms that appear in many documents are not very useful for distinguishing a relevant document from a non-relevant one. This observation indicates that a semantically focussed word might have lower document frequency than an unfocused term.

With the above discussion, we are now ready to present the definition of TF.IDF weighting
method from [3], as follows.

**TF.IDF Definition** Let $N$ be the total number of documents in the system and $n_i$ be the number of documents in which the index term $k_i$ appears. Let $freq_{i,j}$ be the raw frequency of term $k_i$ in the document $d_j$ (i.e., the number of times the term $k_i$ is mentioned in the text of the document $d_j$). Then, the normalized frequency $f_{i,j}$ of term $k_i$ in document $d_j$ is given by

$$tf_{i,j} = \frac{freq_{i,j}}{\max_l freq_{l,j}}$$

(3.1)

where the maximum is computed over all terms which are mentioned in the text of the document $d_j$. If the term $k_i$ does not appear in the document $d_j$ then $f_{i,j} = 0$.

Further, let $idf_i$, inverse document frequency for $k_i$, be given by

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

(3.2)

The TF.IDF weighting scheme is given by

$$w_{i,j} = f_{i,j} \times \log \frac{N}{n_i}$$

(3.3)

3.1.4 Ranking

The vector model is a ranking model for the following reason. In the cosine similarity formula, since $w_{i,j} > 0$, $sim(q, d_j)$, a cosine value, ranges from 0 to 1. Thus, instead of attempting to predict whether a document is relevant or not, the vector model ranks the documents
according to their degree of similarity to the query. A document might be retrieved even if it matches the query only partially.

For this reason, one can establish a threshold on $sim(d_j, q)$ and retrieve the documents with a degree of similarity above the threshold.

3.1.5 Advantages and Disadvantages of Vector Model

Allowing partial matching is in fact one of the main advantages of the vector model. Other advantages of the vector are:

- The cosine measure indicates degrees of similarity between a query and documents. Hence the cosine allows the vector model to rank documents.
- Term weights and cosine measures are easy to compute. Hence the vector model is fast.

As discussed earlier, the vector model summarizes the contents of documents and queries through a set of index terms. However, this may lead to two disadvantages. First, many unrelated documents might be included in the answer set. Second, relevant documents which are not indexed by any of the query keywords are not retrieved. The main reason for these two disadvantages is the inherent vagueness associated with a retrieval process which is based on keyword sets.

3.2 Improving Document Representation

In the vector model, as well as in most of the early models, such as the Boolean model, single terms alone are used for document semantic representation. In many experiments,
single-term representation of documents yields quite effective retrieval results. However, the answer to the question “can sets of single terms provide complete semantic representation of documents?” still remains negative. For this reason, many enhancements in content analysis and text indexing procedures have been proposed over the years in an effort to generate complex document representations. Salton summarized the following possibilities in this connection in the 1970’s and the 1980’s [18]:

- The generation of sets of related items based on the statistical co-occurrence characteristics of the words in certain contexts within the document collection. The assumption normally made is that words that co-occur with sufficient frequency in the documents of a collection are in fact related to each other.

- The formation of term phrases consisting of one or more governing terms (the phrases heads) together with corresponding dependent terms (the phrase components). Phrases are often chosen by using word frequency counts and other statistical methods, possibly supplemented by syntactic procedures designed to detect syntactic relationships between governing and dependent phrase components.

- The use of word grouping methods of the kind provided by thesauri, where classes of related words are grouped under common headings; these class headings can then be assigned for content identification instead of the individual terms contained in the classes. Alternatively, term relationships useful for content identification may also be obtainable by using existing machine-readable dictionaries and lexicons.

- The construction of knowledge bases and related artificial intelligence structures designed to represent the content of the subject area under consideration; entries from
the knowledge bases are then used to represent the content of documents and queries.

The above four possibilities are still widely used in IR research with new tools and knowledge bases. For example, the conceptual-indexing at Sun Microsystems relies on the WordNet knowledge base to extract the concepts of documents and queries. Other possibilities to represent documents and queries have also been explored. The latent semantic indexing model converts each document and query vector into a lower dimensional space using singular value decomposition so that the transformed vectors are associated with concepts.

In the rest of this section, we review the latent semantic indexing model and Woods’ conceptual indexing at the Sun Microsystems Labs.

### 3.2.1 Latent Semantic Indexing Model

Latent semantic indexing (LSI) is a technique that projects queries and documents into a space with "latent" semantic dimensions. Co-occurring terms are projected onto the same dimensionals, non-co-occurring terms are projected onto different dimensions. LSI extracts and represents the contextual-usage meaning of words, or co-occurrence of words, by statistical computations applied to a large corpus of text. The underlying idea is that the aggregate of all the word contexts in which a given word does and does not appear provides a set of mutual constraints that largely determines the similarity of meaning of words and sets of words to each other. In the latent semantic space, a query and a document can have high cosine similarity even if they do not share any terms as long as their terms are semantically similar according to the cooccurrence analysis [14, 11].

We describe how LSI works in terms of the document/query representation, similarity computation, and ranking.
Term-Document Association Matrix

Association Matrix Definition Let \( t \) be the number of index terms in the collection and \( N \) be the total number of documents. Define \( \tilde{M} = (M_{ij}) \) as a term-document association matrix with \( t \) rows and \( N \) columns. To each element \( M_{ij} \) of this matrix is assigned a weight \( w_{i,j} \) associated with the term-document pair \([k_i, d_j]\). This \( w_{i,j} \) weight could be generated using the \( tf - idf \) weighting technique from the classic vector model.

Computing Semantic Matrix

The key to LSI is to use Singular Value Decomposition (SVD), as a method of word co-occurrence analysis, to convert the \( \tilde{M} \) association matrix to a lower dimensional semantic matrix. The SVD on \( \tilde{M} \) proceeds as follows [14, 3]:

1. Decompose \( \tilde{M} \) in three components so that

\[
\tilde{M} = \tilde{K} \tilde{S} \tilde{D}^t
\]

The matrix \( \tilde{K} \) is the matrix of eigenvectors derived from the term-to-term correlation matrix given by \( \tilde{M} \tilde{M}^t \). \( \tilde{D}^t \) is the matrix of eigenvectors derived from the transpose of the document-to-document matrix given by \( \tilde{M}^t \tilde{M} \). The matrix \( \tilde{S} \) is an \( r \times r \) diagonal matrix of singular values where \( r = min(t, N) \) is the rank of \( \tilde{M} \).

2. Keep the \( s \) largest singular values of \( \tilde{S} \) along with their corresponding columns in \( \tilde{K} \) and \( \tilde{D}^t \). Delete the remaining singular values of \( \tilde{S} \). Then, the resultant \( \tilde{M}_s \) matrix is the matrix of \( s \) which is closest to the original matrix \( \tilde{M} \) in the least square sense. \( \tilde{M}_s \)
is given by

\[ \tilde{M}_s = \tilde{K}_s \tilde{S}_s \tilde{D}_s^t \]

where \( s, s < r \), is the dimensionality of a reduced concept space. The selection of a value for \( s \) attempts to balance two opposing effects. First, \( s \) should be large enough to allow fitting all the structure in the real data. Second, \( s \) should be small enough to allow filtering out all the non-relevant representational details (which are present in the conventional index-term based representation).

Computing Similarity

The similarity between any two documents in the reduced space of dimensionality \( s \) can be obtained from the \( \tilde{M}^t \tilde{M} \) matrix given by

\[
\tilde{M}_s^t \tilde{M}_s = (\tilde{K}_s \tilde{S}_s \tilde{D}_s^t)^t \tilde{K}_s \tilde{S}_s \tilde{D}_s^t \\
= \tilde{D}_s \tilde{S}_s \tilde{K}_s^t \tilde{K}_s \tilde{S}_s \tilde{D}_s^t \\
= \tilde{D}_s \tilde{S}_s \tilde{S}_s \tilde{D}_s^t \\
= (\tilde{D}_s \tilde{S}_s)(\tilde{D}_s \tilde{S}_s)^t.
\]

The similarity between documents \( d_i \) and \( d_j \) is quantified by the \((i, j)\) element in the matrix \( \tilde{M}_s^t \tilde{M}_s \).
Document Ranking

To rank documents with regard to a given user query, LSI models the query as a *pseudo-document* in the original $\tilde{M}$ term-document matrix. Assume the query is modeled as the document with number 0. Then the first row in the matrix $\tilde{M}_s'\tilde{M}_s$ provides the ranks of all documents with respect to this query.

Challenge to LSI

Although the LSI model displays some advantages over the vector model, it faces the challenge of high time complexity, when it performs SVD, which keeps LSI from being a widely used model. According to [14, 6], the actual computation of SVD is quadratic in the rank of the document by term matrix and cubic in the number of singular values that are computed. The rank is bounded by the smaller of the number of documents and the number of terms.

3.2.2 Conceptual Indexing

The researchers at Sun Microsystems Labs, led by Woods, built the Sun Precision Content Retrieval system [19, 15]. The system consists of two parts:

- *Conceptual Indexing* builds a structured conceptual taxonomy of words and phrase extracted from the indexed material.

- *Specific Passage Retrieval* finds specific passages and ranks them according to relevance to the query.

Conceptual indexing is the document representation part of the Sun Precision Content Retrieval system. The conceptual indexing process proceeds as follows [19]:

1. Organize all of the words and phrases of the indexed material into a "conceptual taxonomy" that explicitly links each concept to its "most specific subsumers" and to other semantically-related concepts.

2. Analyze the conceptual structure of phrases extracted from the material.

3. Use semantic relationships between words and concepts to establish connections between the terminology used in the user query and other related terminology that may provide the information the user needs.

The conceptual taxonomy, organized by the most-specific-sum-sumer (MSS) relationship, provides a topological structure for the space of conceptual descriptions that can be navigated to find concepts related to each other, and can also support efficient conceptual search and retrieval.

3.3 Improving Weighting Methods

Salton and Buckley in [18] identified eight principal weighting components and some well-known term-weighting systems based on different combinations of the weighting components. We list the eight weighting components in table 3.1 and five well-known term-weighting formulas in table 3.2.

Salton and Buckley suggested in [18] that the following single-term weighting systems for the vector model be used as a standard for comparison with enhanced text analysis systems using thesauruses and other knowledge tools to produce complex multi-term semantic identifications:

- Best document weighting \( tfc, nfc \) (or \( tpc, npc \))
**Term Frequency Component**

- **b** 1.0
  - binary weight equal to 1 for terms present in a vector (term frequency is ignored)

- **t** $tf$
  - raw term frequency (number of times a term occurs in a document or query text)

- **n** $0.5 + 0.5 \frac{tf}{maxtf}$
  - augmented normalized term frequency (tf factor normalized by maximum tf in the vector, and further normalized to lie between 0.5 and 1.0)

**Collection Frequency Component**

- **x** 1.0
  - no change in weight; use original term frequency component ($b$, $t$, or $n$)

- **f** $log \frac{N}{n}$
  - multiply original tf factor by an inverse collection frequency factor
  - ($N$ is total number of documents in collection, and $n$ is number of documents to which a term is assigned)

**Normalization Component**

- **x** 1.0
  - no change; use factors derived from term frequency and collection frequency only (no normalization)

- **c** $\frac{1}{\sqrt{\sum_{i=1}^{w} w_i^2}}$
  - use cosine normalization where each term weight $w$ is divided by a factor representing Euclidean vector length

---

**Table 3.1: Term-weighting Components**

<table>
<thead>
<tr>
<th>Weighting System</th>
<th>Document Term Weight</th>
<th>Query Term Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>Best fully weighted system $tfc \cdot nfx$</td>
<td>$\frac{tf \cdot log \frac{N}{n}}{\sqrt{\sum_{i=1}^{w} (tf_i \cdot log \frac{N}{n})^2}}$</td>
<td>$(0.5 + \frac{0.5tf}{maxtf}) \cdot log \frac{N}{n}$</td>
</tr>
<tr>
<td>Best weighted probabilistic weight $nxx \cdot bpx$</td>
<td>$0.5 + \frac{0.5tf}{maxtf}$</td>
<td>$log \frac{N}{n}$</td>
</tr>
<tr>
<td>Classical idf weight $bf \cdot bfx$</td>
<td>$log \frac{N}{n}$</td>
<td>$log \frac{N}{n}$</td>
</tr>
<tr>
<td>Binary term independence $bxx \cdot bpx$</td>
<td>1</td>
<td>$log \frac{N}{n}$</td>
</tr>
<tr>
<td>Standard tf weight: $txc.txx$</td>
<td>$\frac{tf}{\sqrt{\sum_{i=1}^{w} (tf_i)^2}}$</td>
<td>$tf$</td>
</tr>
<tr>
<td>Coordination level $bxx.gxx$</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

---

**Table 3.2: Term-weighting Formulas**
• Best query weighting $nfx$, $tfx$, $bfx$ (or $npx$, $tpx$, $bpx$)

3.4 Chapter Summary

As summarized in section 3.2, researchers started seeking alternative approaches to representing document content/semantics other than key-term indexing in the early 1970’s. In this trend are LSI’s semantic dimension compacting, Sun’s conceptual taxonomy, content classification of the thesauri approach, as well as the knowledge-base entry representation of the documents. Although the approaches are very different, they are based on the following rationale:

• Documents communicate ideas to the reader from the author. The ideas in a document are more related to the concepts expressed by the index terms rather than the index terms themselves. Thus, the process of matching documents to a given query could be based on concept matching instead of index term matching. This would allow the retrieval of documents even when they are not indexed by query index terms.

Based on this rationale, an IR system is potentially able to handle the classic synonymy/polysemy or paraphrase problem.

Following the same rationale, we propose the idea indexing model as the core of our intelligent filtering system. We discuss the idea indexing model in the next chapter.
CHAPTER 4
FILTERING WITH IDEA INDEXING

In chapter 3, we reviewed some major models proposed to represent documents with document semantics instead of index terms per se. In this chapter, we propose an intelligent filtering system using the idea indexing model which indexes documents with the idea category that each word or phrase belongs to and utilizes the speed, simplicity and ranking advantages of the vector model. The filtering system is intelligent because the index items are the ideas expressed in each word and phrase, and such indexing is supposed to catch the ideas documents convey. We first define the idea indexing model in terms of the four elements of our general IR model.

4.1 Representing Documents with Ideas

In the following definition, we represent a document or a user query as a vector of ideas as expressed by the words and phrases in the document or query.

**Idea Indexing Definition** Let an idea index term be an idea expressed by a word or phrase in a document \( d \) or a query \( q \). Let \( t \) be the number of ideas in the system and \( k_i \) be a generic idea index term. \( K = \{k_1, ..., k_t\} \) is the set of all idea index terms. A weight \( w_{i,j} > 0 \) is associated with each idea index term \( k_i \) of a document \( d_j \). The weight \( w_{i,j} \) is positive and non-binary. The idea index terms in the query are also weighted. Let \( w_{i,q} \) be the weight associated with the pair \([k_i, q]\), where \( w_{i,q} \geq 0 \). Then, the query idea vector \( \vec{q} \) is defined as \( \vec{q} = (w_{1,q}, w_{2,q}, ..., w_{t,q}) \). The idea vector the vector for a a document \( d_j \) is represented by \( \vec{d}_j = (w_{1,j}, w_{2,j}, ..., w_{t,j}) \).
4.2 Similarity Measured by Common Idea Terms

For the vector model, the degree of similarity between document $d_j$ and query $q$ is measured by the term correlation between the vectors $\vec{d}_j$ and $\vec{q}$. For the idea indexing model, since document $d$ and query $q$ are both converted to vectors of ideas expressed in both $q$ and $d$, the degree of similarity between the two is measured by how many common ideas are shared in both idea vectors $\vec{d}_j$ and $\vec{q}$. However, due to the fact that both idea vectors are built the same as term vectors of the vector model and idea weights are similarly computed as term weights, we use the cosine measure to quantify the similarity between two idea vectors. For that reason, we repeat the cosine similarity measure from section 3.1.2 below:

$$sim(d_j, q) = \frac{\sum_{i=1}^{t} w_{i,j} \times w_{i,q}}{\sqrt{\sum_{i=1}^{t} w_{i,j}^2} \times \sqrt{\sum_{j=1}^{t} w_{i,q}^2}}$$

4.3 Ranking

The vector model is a ranking model because the cosine similarity measure varies between 0 and 1, with term weight $w > 0$. The idea indexing model uses similar weighting method and similarity measure as the vector model. This gives the idea indexing model the ability to rank the documents according to how similar they are to the query. For this reason, one can also establish a threshold on $sim(d_j, q)$ and retrieve the documents with a degree of similarity above the threshold.
4.4 Seeking Idea Categories

As discussed in the above sections, the idea indexing model matches a word or phrase of a document with the idea it expresses and index this idea. Then two questions arise regarding (1) how the model finds out about the match, and (2) which ideas to select in the case where a word or phrase expresses multiple ideas. In this section, we look at two knowledge bases using which the idea indexing model matches a word or phrase with its idea(s). One of the two knowledge bases we are interested in is the Roget’s International Thesaurus [10]. The other knowledge base we are interested in is an extension to WordNet developed by Magnini et. al. in [12, 13].

4.4.1 Roget’s International Thesaurus

The revolutionary achievement of Dr. Peter Mark Roget’s first edition in 1852 was the development of a brand-new principle: the arrangement of words and phrases according to their meanings. Dr. Roget’s system brings together in one place all the terms associated with a single thought or concept. The meanings, thoughts, or concepts are in turn grouped into different categories, which are called idea categories. The revised and updated sixth edition in 2001 features thousands of new words and phrases, including those from slang, every language and scientific and technological terminology. The sixth edition includes 330,000 words and phrases that are organized into 1,075 categories. The words and phrases that are grouped under the same category are not only from synonyms, antonyms, and semantically related words and phrases but also from different parts-of-speech, including nouns, verbs, adjectives, adverbs, interjections and prepositions. It is common that some word or phrase
belongs to multiple categories of ideas.

From the above short description of *Roget’s Thesaurus*, we see that the thesaurus makes a good tool for the idea indexing model for several reasons:

- Since a word or phrase is categorized under a specific category of idea, it is handy to find what idea category or categories the word or phrase belongs to. Hence, indexing with idea(s) is possible.

- The sixth edition includes as many as 330,000 words and phrases. This size suffices our vocabulary needs.

- The sixth edition includes 1,075 categories of ideas. The idea indexing model builds the collection index file using only these 1,075 categories of ideas. This approach achieves low space complexity comparing to indexing with all document terms of the vector model.

- Within any one of the idea categories, related words of different of parts-of-speech are included. This inclusion allows the idea indexing model to cover most of the words or phrases of a document, hence most of the content of the document.

4.4.2 Domain Labels

Using two hundred of the Dewey Decimal Classification (DDC) labels, Magnini et al. mapped the nouns, verbs, adjectives and adverbs from the WordNet databases into these 200 labels. Magnini’s original mapping was done between the DDC labels and WordNet 1.6. We have extended the mapping to WordNet 1.7.1.
The DDC domain labels also make a good tool for the idea indexing model for the same four reasons we listed above for the thesaurus. However, the thesaurus has 1,075 idea categories. Magnini only developed 200 DDC domain labels. This difference in categories makes the latter a coarser-grained classification and the former a finer-grained classification.

4.5 Hypothesis

Following the many efforts made to effectively represent document semantics discussed in section 3.4, we propose the idea indexing model to represent documents with the ideas expressed in document terms so that similarity computing will not be based on term overlaps but on the number of common ideas expressed in documents. Since the idea indexing model is designed to overcome the problem that the vector model cannot solve properly, the idea indexing model is supposed to lead to positive performance gain in filtering tasks.

4.6 Potential Problem with Idea Indexing

Both Roget’s Thesaurus and the DDC domain labels enable the idea indexing model to eliminate the classic synonymy problem because synonyms and related words from the same document will be collected/indexed under the same idea category which avoids the disadvantage brought by sense overlaps in both Boolean and vector models. However, the second question we brought up in section 4.4 still does not have a formal solution yet. This is the polysemy problem, where one word or phrase belongs to multiple idea categories.

Although we still do not have a formal solution to the classic polysemy problem, we hope the idea term weighting process assigns lower weights to the less-important idea categories and filter out the un-important categories. However, the fact that words or phrases may
belong to multiple idea categories presents a potential problem for representing the semantics of a document appropriately.

4.7 Idea Indexing Example

In this section, we show how the idea indexing model builds document representations for the following quotation from British philosopher Bertrend Russell’s prologue to his book *A history of western philosophy* [16], using both Roget’s idea categories and the DDC domain labels.

*Three passions, simple but overwhelmingly strong, have governed my life: the longing for love, the search for knowledge, and unbearable pity for the suffering of mankind. These passions, like great winds, have blown me hither and thither, in a wayward course, over a great ocean of anguish, reaching to the very verge of despair ... I have sought love ... With equal passion I have sought knowledge. I have wished to understand the hearts of men. I have wished to know why the stars shine ...*

Using the framework defined in 4.1 and Roget’s idea categories, the indexing model builds a document representation for the above passage shown in table A.1 in Appendix A. In table A.1, the first column lists the ideas that occur in the passage. The third column shows the number of times each idea occur in this passage. The second column lists the term(s) that express(es) the corresponding ideas. In the actual representations in the implemented system, the second column does not exist. It is shown here only for illustration purposes. However, table A.1 only shows the ideas with document frequency greater than 2 to save space.

Table B.1 shows a DDC domain label based idea representation of the same passage. The
information in its three columns corresponds with that in table A.1.

We make two observations from both representations as follows:

- Table A.1 gives a richer presentation of the content of the passage than table B.1. However, we will not further explore this observation in current work.

- In both tables, some terms belong to multiple idea categories. This illustrates the potential problem we brought up in section 4.6. We will come back to this point in section 6.5.

In the next chapter, we report on how we implemented the systems for our experiments.
Our intelligent filtering system is designed to work in the following scenario: given the pushing action of the World Wide Web, some news agency or information provider pushes large amount of information towards a customer. If the customer is only interested in reading a certain topic from all incoming information, he/she specifies the topic to the system. Then the system filters out the unwanted information and only reports the information of interest.

To simulate the above scenario, we embedded our filtering system in a two-interface system. One interface is for the information provider, and the other is for the customer. The filtering system is installed locally on the customer’s machine. At certain time of the day, the news agency sends documents to the customer. Then at a later time of the day, the customer picks a topic and reads relevant information.

We have implemented the two-interface system, a vector model based filtering system, an idea indexing model based system, and a system based on both idea indexing and vectorial term indexing. We describe the main system and the sub-systems in that order.

5.1 Components of the Overall System

The two-interface system has the following components.

- News Agency System - News Agency System consists of two parts, the sender interface and the underlying sender program.

The sender interface
A Web implementation. The command generated at the sender interface is transmitted to the underlying sender program.

The sender program

A CGI program written in Perl. At the interface, a news agent may select some specific types of news stories or a bundle of them and specify a customer’s address for the stories to be sent to. After selecting the news stories and specifying the customer’s address, the agent sends the stories to the user.

The news agency interface is available online at: http://students.csci.unt.edu/~lyangePostmanep.htm.

- Customer System - The customer system has three components.

  Receiving System

  A receiver program. It is a program running in the back of the customer system all the time. It receives news stories from the news agency and notifies the customer whenever new items have arrived.

  Customer Interface

  A customer selects the topic that he/she is interested in and reads the ones that are computed to be of his/her interest.

  The Filtering System

  The topic that the customer is interested in is submitted to the underlying filtering system which computes and recommends to the customer the ones that are relevant to the selected topic.

The customer interface is available online at: http://mira.csci.unt.edul/yangePostmanreader2.htm.
5.2 Indexing with Vectorial Terms

We implemented the filtering system at the customer side using the vector model. Figure 5.1 shows our implementation of the vector model. This implementation is in linear order. It is so because we implemented the vectors as hash of hashes so that any operations, including similarity computation, are in constant time.

1. Read in all the incoming news stories.
2. Build a document vector for all stories by following the steps below:
   2.1 Preprocess each story so that they are tokenized
   2.2 Create an inverted index file for the tokenized stories
      2.2.1. Record term frequencies.
      2.2.2. Record document frequencies
3. Process the topic story defined by the user following the same steps in 2 so that a query vector is built.
4. Assign weights to the document and query vector, using tfc and nfx weighting methods respectively.
5. Compute document–query similarities using the cosine measure
6. Return stories with cosine measure greater than or equal to a threshold value.

Figure 5.1: Vector Model Implementation

5.3 Indexing with Idea Categories

In this section, we report on some of the implementation details of our systems using idea categories. Figure 5.2 lists the system procedures to filter through the incoming stories for user interested stories.

We followed the steps in figure 5.3 to build the inverted document index file. Two important sub-procedures in figure 5.3 are

- getting base forms of words.
1. Build idea document index for a topic document, using tf.idf
2. Build idea document index for all files in the collection, using tf.idf
3. Assign weights to the idea terms for the collection index, using tfc
4. Assign weights to the idea terms for the topic index, using tfx
5. Compute similarity between the topic and a document from the collection, using cosine similarity function
6. If the similarity is greater than a threshold value, then the document is selected as a similar document
7. Otherwise, it is not similar and filtered out

Figure 5.2: Filtering System

- recognizing phrases of length 5 that exist in the knowledge base

1. read in the stories in the collection one at a time
2. preprocess each story so that a sentence is stored as a separate element in a document array,
3. for each word/phrase in the sentence,
4. if it exists in the thesaurus
5. then index the idea/concept mapped with the word/phrase
6. also, record the number of times this idea/concept occurs in this document
7. also, record the number of documents the idea/concept occurs in
8. if a word does not have a mapping idea/concept in the thesaurus and starts with an upper case letter, then index it like traditional vector model. This word is most likely a proper noun

Figure 5.3: Building Idea Vector

To get the base forms of the words in a document, we had to use the *morph* API in WordNet.

To recognize phrases in the document, the steps in figure 5.4 were followed.

In figure 5.4, step 4.2 was designed to handle proper nouns that are not in the thesaurus.
Figure 5.4: Recognizing Phrases

5.4 Combining Idea Indexing and Vector Models

We combined both idea indexing and term indexing into one system as shown in figure 5.5.

As shown in figure 5.5, we combined both models by adding the similarity scores for the same story that exists in both similarity vectors.

5.5 Evaluation Measures

We use precision, recall, and F-measure to evaluate the system performance. Before these measures are defined, we define a contingency table in table 5.6 from [14].

The numbers in each box show the frequency or count of the number of items in each region of the space. The cases accounted for by \( tp \) (true positives) and \( tn \) (true negatives) are the cases the system got right. The wrongly selected cases in \( fp \) are called false positives. The cases in \( fn \) that failed to be selected are called false negatives.
1. Follow steps 1 to 5 of the vector model shown in figure 4.1 to compute a similarity vector V1 for all news stories and a topic story.
2. Follow steps 1 to 5 of the idea indexing model shown in figure 4.2 to compute a similarity vector V2 for all news stories and a topic story.
3. Combine V1 and V2 into similarity vector V by adding the similarity scores associated with the same story from both V1 and V2.
4. If a similarity score in V is greater than or equal to a threshold value, then the corresponding document is selected as a relevant document to a topic story.
5. Otherwise, the story is off topic and filtered out.

Figure 5.5: Idea and Vector Models Combined

<table>
<thead>
<tr>
<th>System</th>
<th>target</th>
<th>−target</th>
</tr>
</thead>
<tbody>
<tr>
<td>selected</td>
<td>tp</td>
<td>fp</td>
</tr>
<tr>
<td>−selected</td>
<td>fn</td>
<td>tn</td>
</tr>
</tbody>
</table>

Figure 5.6: Contingency Table
Precision is defined as a measure of the proportion of selected items that the system got right:

\[
\text{precision} = \frac{tp}{tp + fp}
\]

Recall is defined as the proportion of the target items that the system selected:

\[
\text{recall} = \frac{tp}{tp + fn}
\]

F-measure combines precision and recall into a single measure of overall performance:

\[
F - \text{measure} = \frac{2PR}{P + R}
\]

We implemented three filtering systems with the vector model, the idea indexing model, and a combination of the vector and idea indexing. In the next chapter, we discuss the experiments we have done with these systems and the results obtained from the experiments.
CHAPTER 6

EXPERIMENT AND RESULTS

We carried out three sets of experiments with the three systems we implemented. The first set included experiments with the three systems using idea categories from *Roget’s International Thesaurus*. The second set was based on the DDC domain labels. The third set contained only the experiment using vectorial term indexing, which was designed to be the baseline system. In all experiments, used the Reuter’s Corpus for TREC 2002 as the data set. In the following, we introduce the data set. We follow the introduction with the experiments and the results.

6.1 Test Data

For our experiments with the filtering task, we used the documents, relevance judgments and 50 topics from TREC 2002.

6.1.1 TREC Topics

One hundred topics have been specially constructed for evaluating the filtering systems, and are of two types. A set of 50 are of the usual TREC type, developed by the NIST assessors (with assessor relevance judgments, see below). A second set of 50 have been constructed artificially from intersections of pairs of Reuters categories.

In current work, we only considered the 50 topics of the first type.

The topics are defined in standard TREC format. Figure 6.1 gives an example of the TREC definition for topic #104.
6.1.2 Building User Profiles

In our experiments, we built the user profiles or topics by combining all files that are annotated to be relevant in the training data to a topic into one corresponding topic file, or filtering profile, for the same topic. This gave us a topic profile well representing the corresponding topic definition. We created a profile for each of the 50 topics.

6.1.3 Test Collection

We used the document collection in Reuters Corpus Volume 1. The documents are divided into training and test sets. The training set consists of all documents with dates up to and including 30 September 1996 (83,650 documents). The test set consists of all remaining 723,141 documents up to 1997. In our experiments, we ran the systems on a subset of the test set. The size of the subset is about 61,654.
6.1.4 Relevance File

The results from the systems were compared against the relevance judgments. All TREC files mentioned in this section are available at [7].

6.2 Experiments

Three sets of experiments have been done to evaluate the three systems, namely, the vector model, the idea indexing model, and the combined model.

- In the first set, we built a baseline system using the vector model and had the system filter through the test data.

- Using the Roget’s idea categories, we ran both the idea based system and the combined system on the test data.

- Using the DDC domain labels, we ran both the idea based system and the combined system on the test data.

All sets of experiments were performed following four steps:

1. Build the system.

2. For each of the 50 topic files or user profiles, filter through the text collection and evaluate each document as relevant or irrelevant.

3. Output the documents that the system computes to be relevant.

4. Use the relevance file to compute precision, recall, and F-measure scores from the output.
We report on the experiment results in the next section.

6.3 Results

We plot the results from the experiments in this section. First are the results from the baseline vectorial system.

6.3.1 Baseline System

Figure 6.2 gives the recall, precision, and F-measure of the vector model at threshold values 0-1 respectively at an interval of 0.1. A threshold value describes how much a document has to be similar to a topic file. A threshold of 0.4 says that the document has to be 40% similar to the topic file. In terms of similarity scores, for a threshold of 0.4 the similarity score has to be greater than or equal to 0.4 so that a document can be claimed as similar to the topic file. In our experiments, we set the threshold values from 0 to 1 at an interval of 0.1.

![Figure 6.2: Vector Model Recall, Precision, F](image)

Figure 6.2 shows following trends in each of the measures:
• Recall starts high at 0.691 at threshold 0.1 and drops quickly to 0.351 at threshold 0.2, and then below 0.2 at 0.3. After threshold 0.3, it stays mostly below 0.050.

• F-measure rises from 0.240 at threshold 0.1 to its highest 0.253 at threshold 0.3. Then it stays mostly below 0.050.

• Precision reaches the highest of 0.351 at threshold 0.5, and its lowest is at 0.06 at threshold 1.0.

6.3.2 Indexing with Roget’s Idea Categories

Figures 6.3, 6.5, and 6.4 give the recall, precision, and F-measure of the idea indexing model and the combined model using ideas, as compared to the baseline system, with threshold values 0-1 respectively at an interval of 0.1. Corresponding with figure 6.3, table 6.1 lists the positive and negative recall gain (if multiplied by 100, it becomes percentage improvement) of both idea and combined indexing over the baseline system. Corresponding with figure 6.4, table 6.2 lists the positive and negative F-measure gain (if multiplied by 100, it becomes the percentage improvement) of both idea and combined indexing over the baseline system. Corresponding with figure 6.5, table 6.3 lists the positive and negative precision gain (if multiplied by 100, it becomes the percentage improvement) of both idea and combined indexing over the baseline system.

<table>
<thead>
<tr>
<th>Threshold</th>
<th>0.0</th>
<th>0.1</th>
<th>0.2</th>
<th>0.3</th>
<th>0.4</th>
<th>0.5</th>
<th>0.6</th>
<th>0.7</th>
<th>0.8</th>
<th>0.9</th>
<th>1.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Idea Indexing</td>
<td>0.00</td>
<td>0.43</td>
<td>1.72</td>
<td>3.66</td>
<td>7.36</td>
<td>12.44</td>
<td>21.00</td>
<td>44.10</td>
<td>84.75</td>
<td>128.00</td>
<td>195.00</td>
</tr>
<tr>
<td>Combined Index</td>
<td>0.00</td>
<td>0.44</td>
<td>1.76</td>
<td>3.91</td>
<td>8.07</td>
<td>14.74</td>
<td>26.60</td>
<td>59.50</td>
<td>127.25</td>
<td>214.50</td>
<td>353.00</td>
</tr>
</tbody>
</table>

Table 6.1: Recall Gain of Idea indexing and Combined Indexing Over Vector Model

From figure 6.3 and table 6.1, we can make following observations:
The idea indexing model using Roget’s idea categories gains much higher recall values at all threshold values 0.1-1.0. For example, with threshold = 0.5, the idea model gains a recall of 0.672. With the same threshold, the vector model has a recall of 0.050. So the former model has a recall 1244% higher than the latter.

The combined model of idea and vectorial indexing using Roget’s idea categories obtains much higher recall values at all threshold values 0.1-1.0. For example, with threshold = 0.5, the combined model has a recall of 0.787. With the same threshold, the vector model has a recall of 0.050. The former has a recall 1474% higher than the latter.

For all three models, recall values drop with higher threshold values.

The vector model drops its recall values below 0.050 after threshold 0.5. While the idea indexing model maintains its recall values high above 0.2, with a minimum recall
of 0.2 at threshold 1.0. The combined model keeps its recall high above 0.35, with a minimum recall of 0.354 at threshold 1.0.

- The combined model gains higher recall than the idea indexing model for threshold larger than 0.2.

![Idea, Vector, and Combined Model F Measures](image)

**Figure 6.4:** F-measures for 3 Systems

<table>
<thead>
<tr>
<th>Threshold</th>
<th>0.0</th>
<th>0.1</th>
<th>0.2</th>
<th>0.3</th>
<th>0.4</th>
<th>0.5</th>
<th>0.6</th>
<th>0.7</th>
<th>0.8</th>
<th>0.9</th>
<th>1.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Idea Indexing</td>
<td>-0.39</td>
<td>-0.38</td>
<td>-0.14</td>
<td>0.50</td>
<td>1.52</td>
<td>3.58</td>
<td>9.58</td>
<td>21.75</td>
<td>40.25</td>
<td>72.50</td>
<td></td>
</tr>
<tr>
<td>Combined Index</td>
<td>-0.39</td>
<td>-0.39</td>
<td>-0.17</td>
<td>0.46</td>
<td>1.52</td>
<td>3.74</td>
<td>10.61</td>
<td>25.50</td>
<td>51.50</td>
<td>100.00</td>
<td></td>
</tr>
</tbody>
</table>

**Table 6.2:** F-measure Gain of Idea indexing and Combined Indexing Over V.M.

We can make following observations from figure 6.4 and table 6.2 about the F-measures of the three systems.

First, for threshold values between 0.1 and 0.3 inclusive, the vector model achieves higher F-measures than both combined and idea indexing models. Second, the F-measure of the vector model drops below 0.1 and hence below those of both combined and idea indexing models with threshold larger than 0.4.
Figure 6.5: Precision for 3 Systems

<table>
<thead>
<tr>
<th>Threshold</th>
<th>0.0</th>
<th>0.1</th>
<th>0.2</th>
<th>0.3</th>
<th>0.4</th>
<th>0.5</th>
<th>0.6</th>
<th>0.7</th>
<th>0.8</th>
<th>0.9</th>
<th>1.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Idea Indexing</td>
<td>0.00</td>
<td>-0.47</td>
<td>-0.62</td>
<td>-0.66</td>
<td>-0.65</td>
<td>-0.62</td>
<td>-0.51</td>
<td>-0.21</td>
<td>0.16</td>
<td>0.58</td>
<td>1.72</td>
</tr>
<tr>
<td>Combined Index</td>
<td>0.00</td>
<td>-0.48</td>
<td>-0.63</td>
<td>-0.67</td>
<td>-0.67</td>
<td>-0.64</td>
<td>-0.53</td>
<td>-0.23</td>
<td>0.18</td>
<td>0.65</td>
<td>1.90</td>
</tr>
</tbody>
</table>

Table 6.3: Precision Gain of Idea Indexing and Combined Indexing Over V.M.
Following observations can be made from figure 6.5 and table 6.3.

Firstly, for threshold values from 0.1 up to 0.7, the vector model gains higher precision than the idea model, from a minimum 18% gain at the threshold of 0.7 to a maximum 58% gain at 0.4. Second, for threshold value from 0.1 up to 0.7, the vector model obtains higher precision than the combined model, from a minimum 15% gain at 0.7 to a maximum 61% gain at 0.4. Third, for threshold values between 0.8 to 1.0, both combined and idea models win higher precision.

6.3.3 Indexing with Domain Labels

Figures 6.6, 6.8, and 6.7 give the recall, precision, and F-measure of the idea indexing model and the combined model using the DDC domain labels, as compared to the baseline system, with threshold values 0-1 respectively at an interval of 0.1. Corresponding with figure 6.6, table 6.4 lists the positive and negative recall gain (if multiplied by 100, it becomes the percentage improvement) of both idea and combined indexing over the baseline system. Corresponding with figure 6.7, table 6.5 lists the positive and negative F-measure gain (if multiplied by 100, it becomes the percentage improvement) of both idea and combined indexing over the baseline system. Corresponding with figure 6.8, table 6.6 lists the positive and negative precision gain (if multiplied by 100, it becomes the percentage improvement) of both idea and combined indexing over the baseline system.

<table>
<thead>
<tr>
<th>Threshold</th>
<th>0.0</th>
<th>0.1</th>
<th>0.2</th>
<th>0.3</th>
<th>0.4</th>
<th>0.5</th>
<th>0.6</th>
<th>0.7</th>
<th>0.8</th>
<th>0.9</th>
<th>1.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Idea Indexing</td>
<td>0.00</td>
<td>0.33</td>
<td>1.24</td>
<td>2.45</td>
<td>4.83</td>
<td>8.20</td>
<td>14.04</td>
<td>29.60</td>
<td>61.50</td>
<td>102.5</td>
<td>171.00</td>
</tr>
<tr>
<td>Combined Indexing</td>
<td>0.00</td>
<td>0.41</td>
<td>1.57</td>
<td>3.17</td>
<td>6.31</td>
<td>11.04</td>
<td>20.24</td>
<td>45.20</td>
<td>99.50</td>
<td>167.50</td>
<td>285.00</td>
</tr>
</tbody>
</table>

Table 6.4: Recall Gain of Idea indexing and Combined Indexing Over V.M.

From figure 6.6 and table 6.4, we can make following observations:
The idea indexing model using DDC domain labels gains much higher recall values at all threshold values. For example, with threshold = 0.5, the idea model gains a recall of 0.460. With the same threshold, the vector model has a recall of 0.050. So the former model has a recall 153% higher than the latter.

The combined model of idea indexing and vectorial indexing using Roget’s idea categories obtains much higher recall values at all threshold values. For example, with threshold = 0.5, the combined model has a recall of 0.602. With the same threshold, the vector model has a recall of 0.050. The former has a recall 170% higher than the latter.

For all three models, recall values drop with higher threshold values.

The vector model drops its recall values below 0.050 after threshold 0.5. While the idea indexing model maintains its recall values high above 0.17, with a minimum recall
of 0.175 at threshold 1.0. The combined model keeps its recall high above 0.35, with a minimum recall of 0.286 at threshold 1.0.

- The combined model gains higher recall than the idea indexing model for threshold larger than 0.1.

![Figure 6.7: F-measure for 3 Systems](image)

<table>
<thead>
<tr>
<th>Threshold</th>
<th>0.0</th>
<th>0.1</th>
<th>0.2</th>
<th>0.3</th>
<th>0.4</th>
<th>0.5</th>
<th>0.6</th>
<th>0.7</th>
<th>0.8</th>
<th>0.9</th>
<th>1.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Idea Indexing</td>
<td>0.00</td>
<td>-0.34</td>
<td>-0.29</td>
<td>-0.04</td>
<td>0.60</td>
<td>1.53</td>
<td>3.30</td>
<td>8.61</td>
<td>19.00</td>
<td>36.00</td>
<td>65.50</td>
</tr>
<tr>
<td>Combined Indx</td>
<td>0.00</td>
<td>-0.37</td>
<td>-0.32</td>
<td>-0.05</td>
<td>0.63</td>
<td>1.70</td>
<td>3.88</td>
<td>10.44</td>
<td>24.12</td>
<td>46.50</td>
<td>88.50</td>
</tr>
</tbody>
</table>

Table 6.5: F-measure Gain of Idea indexing and Combined Indexing Over Vector Model

We can make following observations for from figure 6.7 and table 6.5 about the F-measures of the three systems. First, for threshold values 0.1 and 0.2 inclusive, the vector model achieves higher F-measures than both combined and idea indexing models. Second, the F-measure of the vector drops below 0.1 and hence below those of both combined and idea indexing models with threshold larger than 0.4.
Figure 6.8: Precision for 3 Systems

Table 6.6: Precision Gain of Idea indexing and Combined Indexing Over Vector Model

<table>
<thead>
<tr>
<th>Threshold</th>
<th>0.0</th>
<th>0.1</th>
<th>0.2</th>
<th>0.3</th>
<th>0.4</th>
<th>0.5</th>
<th>0.6</th>
<th>0.7</th>
<th>0.8</th>
<th>0.9</th>
<th>1.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Idea Indexing</td>
<td>0.00</td>
<td>-0.42</td>
<td>-0.55</td>
<td>-0.57</td>
<td>-0.58</td>
<td>-0.56</td>
<td>-0.46</td>
<td>-0.18</td>
<td>0.20</td>
<td>0.59</td>
<td>1.62</td>
</tr>
<tr>
<td>Combined Indx</td>
<td>0.00</td>
<td>-0.45</td>
<td>-0.58</td>
<td>-0.60</td>
<td>-0.61</td>
<td>-0.57</td>
<td>-0.46</td>
<td>-0.15</td>
<td>0.26</td>
<td>0.68</td>
<td>1.82</td>
</tr>
</tbody>
</table>
Following observations can be made from figure 6.8 and table 6.6.

Firstly, for threshold values from 0.1 up to 0.7, the idea model makes negative gains. Second, for threshold value from 0.1 up to 0.7, the combined model also makes negative gains. Third, for threshold values between 0.8 to 1.0, both combined and idea models makes positive gains.

6.3.4 Result Summary

The observations we made in sections 6.3.2 and 6.3.3 indicate the experiment results from Roget’s idea category based systems match with those from DDC domain label based systems in following respects.

First, the idea indexing model and combined indexing model yields much higher recall than the vector model in both types of idea categories.

Second, the idea indexing and combined indexing models obtain higher F-measures than the vector model when the threshold is greater than or equal to 0.4. Both models’ F-measures fall below vector model’s when the threshold is 0.1 and 0.2.

Third, precision of idea indexing and combined indexing models is below vector model’s when the threshold is between 0.1 and 0.7, inclusive.

With the above summary, is it safe to conclude that the idea indexing based systems are more reliable than vector based systems (we label the combined indexing as a idea indexing based system) because of the higher recall and F-measure? If it is more reliable, then why its precision is mostly not as high as a vectorial system?

We explain in the next section that the high precision achieved by the vector model does not necessarily make it a more reliable system than the idea indexing based systems.
6.4 Higher Precision, Better Performance?

The vector model gives higher average precision for most of the threshold values. But it does not necessarily perform better than the idea indexing based systems. First of all, its recall and F-measure are not as good. Secondly, its higher precision is not convincing. We try to prove this in two ways:

- the number of documents the vector model retrieves for each of the 50 topics

- Threshold and user satisfaction

6.4.1 Number of Retrieved Documents

Recall the definition of precision, which is the proportion of the selected documents that the system correctly retrieves.

A system can retrieve a small number of documents and still get high precision because the number of retrieved incorrect documents is small. This is exactly the case with the vector model in all the experiments. Figures 6.9, 6.10, 6.11, and 6.12 compare the number of documents retrieved by the vector system and the two idea-based systems for each of the 50 topics at the threshold values of 0.1, 0.3, 0.4, and 0.6 respectively. Figures 6.13, 6.14, 6.15, and 6.16 compare the number of documents retrieved by the vector system and the two domain-label-based systems for each of the 50 topics at the threshold values of 0.1, 0.3, 0.4, and 0.6 respectively.

Observing from figures 6.9 through 6.16, we can conclude two trends common to both Roget-based and domain-based idea indexing models and the vector model. If we plot the numbers for the whole range of threshold values, we will see the same trend.
- While the threshold increases, the number of documents the vector model retrieves decreases. When the threshold is greater than or equal to 0.3, the vector model returns few or even no documents for many topics. This trend helps to explain a couple of issues we mentioned earlier. First, since the vector model returns a small number of documents, it possibly gains higher precision than the idea based systems, as shown in figures 6.5 and 6.8. Second, the small number of documents retrieved by the vector model leads to low recall measures as shown in figures 6.3 and 6.3. This means, for most of the topics, the vector system can return very few documents that are of the user’s interest at a threshold value greater than 0.2. Hence, the vector system does not help much in the filtering tasks with a threshold greater than 0.2.

- With all threshold values shown in these figures, both versions of idea indexing models return a large number of documents, among which a good proportion are relevant to the user defined topics. This explains the high recall measures for most of the threshold values in figures 6.3 and 6.6. Although the large number implies low precision measures, as shown in figures 6.5 and 6.8, the idea-based indexing models are still more reliable even when the threshold rises to higher values because it still manages to return documents that the user is interested in.

6.4.2 Threshold and User Satisfaction

We have been comparing the vectorial system with the idea indexing systems at the full range of threshold values only for the purposes of experiments. In real life situations, the user probably would be more satisfied with the documents that are computed to be more similar to the user profile. This satisfaction is guaranteed only with a not too low threshold,
for example, 0.4 or higher. If a selection of high threshold value is required, then the idea-based systems performs more reliably than the vectorial system.

6.5 Causes for Low Precision

So far, the low precision of the idea-based systems has not been accounted for. We try to do so here. Recall the potential problem we brought up in section 4.6. We expected in that
section that a term’s belonging to multiple idea categories could cause some problem. We assume here that it causes the idea systems’ low precision scores. We attempt to explain why this is so below.

According to Roget’s Thesaurus, passion belongs to six idea categories, namely, FEELING, MUSIC, LOVE, ELOQUENCE, UNPLEASURE, EXCITEMENT, DESIRE, and EAGERNESS. From the document presentation of Russell’s passage in table A.1, we see that passion appear in all six categories. A careful reading of the passage would tell us that
passion means DESIRE in this context. The idea category DESIRE is supposed to contain more content that other categories that passion appears in. Hence, category DESIRE is more important that other categories that passion appears in in terms of contributing more the meaning of the passage. However, our current idea indexing model is unable to tell the more important categories from the less important ones and hence cannot remove the less important categories and give more weight to the important ones. Now if a different document containing the word passion which means ELOQUENCE comes into the filtering system, the idea indexing model might mark it as relevant to Ressull’s passage because the representation for the latter document also has categories for passion exactly the same as Ressull’s, although in the latter representation ELOQUENCE is more important that the other categories. Now suppose the latter document contains more than one identical word forms of Russell’s passage and each word belongs to multiple categories. Given our filtering system cannot figure out which category for each identical word form has more weight, when computing similarity it will also add up the less important categories for each word form. Thus, the system might declare similar documents that are not so relevant and returns more relevant documents than there actually are. This lowers the precision of the system.

We can make similar conclusion about the representation with the DDC domain labels in table B.1, because the issue of a word’s appearing in multiple categories also exists in this representation.
CHAPTER 7

CONCLUSIONS AND FUTURE WORK

Efforts have been made as early as the vector model in the 1970’s to represent the content and semantics of a document effectively in IR research. Research on representing the content and semantics of a document or a query has been focused on four areas, namely, improving document representation, improving query representation, optimizing similarity computing and ranking algorithms.

In our current work we contributed to the same efforts in representing document content by building an intelligent representation of document semantics using the idea indexing model. The idea indexing model resembles the advantages of the classic vector model in that it inherits the simplicity, fast speed, the effective weighting method and cosine similarity function, and ranking algorithm from the latter. However, the idea indexing model differs from the vector model in that the former indexes the document and query with the ideas that are expressed by the terms instead of each independent term from the document and query. Our hypothesis is that indexing with ideas catches the meaning of a document more than independent term indexing used in the vector model.

Our systems have shown improvement comparing to the vector model in terms of higher recall and F-measure. However, our systems are yet to achieve high precision. Therefore, obtaining higher precision with the idea indexing model has become our next goal.

Specifically, our immediate next step is to make the idea indexing model more intelligent so that it can compute the critical idea categories among many less important categories. In
other words, the idea indexing model should be able to compute the centroid from all the categories of a document.
APPENDIX A

DOCUMENT REPRESENTATION - ROGET’S IDEA CATEGORY-BASED
<table>
<thead>
<tr>
<th>IDEA CATEGORY</th>
<th>COORESPONDING TERMS</th>
<th>TERM FEQUENCY</th>
</tr>
</thead>
<tbody>
<tr>
<td>FEELING</td>
<td>over, passion, love</td>
<td>3</td>
</tr>
<tr>
<td>THE UNIVERSE ASTRONOMY</td>
<td>wind, ocean of, star, great</td>
<td>4</td>
</tr>
<tr>
<td>WIND</td>
<td>wind, strong, great, blow</td>
<td>4</td>
</tr>
<tr>
<td>MUSIC</td>
<td>wind, passion, blow</td>
<td>3</td>
</tr>
<tr>
<td>IMPAIRMENT</td>
<td>over, strong, blow</td>
<td>3</td>
</tr>
<tr>
<td>EVENT</td>
<td>course, wind, suffer, know, life</td>
<td>5</td>
</tr>
<tr>
<td>IMPORTANCE</td>
<td>star, great, life</td>
<td>3</td>
</tr>
<tr>
<td>LOVE</td>
<td>like, shine, passion, love, life</td>
<td>5</td>
</tr>
<tr>
<td>INQUIRY</td>
<td>over, search for, seek</td>
<td>3</td>
</tr>
<tr>
<td>ELOQUENCE</td>
<td>shine, strong, passion</td>
<td>3</td>
</tr>
<tr>
<td>DEATH</td>
<td>over, great, life</td>
<td>3</td>
</tr>
<tr>
<td>ADVERSITY</td>
<td>wind, in a, blow, life</td>
<td>4</td>
</tr>
<tr>
<td>LOCATION</td>
<td>over, thither, hither</td>
<td>3</td>
</tr>
<tr>
<td>GREATNESS</td>
<td>like, unbearable, strong, great</td>
<td>4</td>
</tr>
<tr>
<td>UNPLEASURE</td>
<td>suffer, despair, passion, anguish, blow</td>
<td>5</td>
</tr>
<tr>
<td>REPRODUCTION PROCREATION</td>
<td>over, great, love</td>
<td>3</td>
</tr>
<tr>
<td>INTELLIGIBILITY</td>
<td>know, simple, understand</td>
<td>3</td>
</tr>
<tr>
<td>CERTAINTY</td>
<td>shine, know, but</td>
<td>3</td>
</tr>
<tr>
<td>SEX</td>
<td>knowledge, love, life</td>
<td>3</td>
</tr>
<tr>
<td>GOODNESS</td>
<td>star, great, equal</td>
<td>3</td>
</tr>
<tr>
<td>EXCITEMENT</td>
<td>over, in a, passion</td>
<td>3</td>
</tr>
<tr>
<td>SUPERIORITY</td>
<td>over, star, great</td>
<td>3</td>
</tr>
<tr>
<td>PAIN</td>
<td>suffer, unbearable, anguish</td>
<td>3</td>
</tr>
<tr>
<td>POWER POTENCY</td>
<td>men, strong, life</td>
<td>3</td>
</tr>
<tr>
<td>DESIRE</td>
<td>like, wish to, knowledge, passion, love, long for</td>
<td>6</td>
</tr>
<tr>
<td>LOUDNESS</td>
<td>wind, like, blow</td>
<td>3</td>
</tr>
<tr>
<td>EAGERNESS</td>
<td>over, passion, life</td>
<td>3</td>
</tr>
<tr>
<td>SKILL</td>
<td>star, know, great</td>
<td>3</td>
</tr>
<tr>
<td>PLEASURE</td>
<td>like, over, love</td>
<td>3</td>
</tr>
<tr>
<td>TRUTH</td>
<td>the very, simple, life</td>
<td>3</td>
</tr>
<tr>
<td>REPUTE</td>
<td>shine, star, great</td>
<td>3</td>
</tr>
<tr>
<td>END</td>
<td>course, over, blow</td>
<td>3</td>
</tr>
<tr>
<td>PUNISHMENT</td>
<td>wind, suffer, blow</td>
<td>3</td>
</tr>
<tr>
<td>THE FUTURE</td>
<td>verge of, course, life</td>
<td>3</td>
</tr>
<tr>
<td>KNOWLEDGE</td>
<td>know, knowledge, understand</td>
<td>3</td>
</tr>
<tr>
<td>IMPULSE IMPACT</td>
<td>course, over, blow</td>
<td>3</td>
</tr>
<tr>
<td>INTERPRETATION</td>
<td>heart of, over, understand</td>
<td>3</td>
</tr>
<tr>
<td>FACILITY</td>
<td>course, like, simple</td>
<td>3</td>
</tr>
<tr>
<td>DISTANCE REMOTENESS</td>
<td>over, thither, reach to</td>
<td>3</td>
</tr>
</tbody>
</table>

Table A.1: Doc Rprsnt. of Russell’s Passage tf >= 3
APPENDIX B

DOCUMENT REPRESENTATION - DDC Domain Label Based
<table>
<thead>
<tr>
<th>IDEA CATEGORY</th>
<th>COORESPONDING TERMS</th>
<th>TERM FREQUENCY</th>
</tr>
</thead>
<tbody>
<tr>
<td>literature</td>
<td>long</td>
<td>1</td>
</tr>
<tr>
<td>economy</td>
<td>men, blow</td>
<td>2</td>
</tr>
<tr>
<td>physiology</td>
<td>wind, passion, love, blow</td>
<td>4</td>
</tr>
<tr>
<td>pedagogy</td>
<td>course</td>
<td>1</td>
</tr>
<tr>
<td>time, period</td>
<td>long, life</td>
<td>2</td>
</tr>
<tr>
<td>psychology</td>
<td>shine, know, wayward, understand, blow, pity</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>suffer, anguish</td>
<td>8</td>
</tr>
<tr>
<td>zoology</td>
<td>course, blow</td>
<td>2</td>
</tr>
<tr>
<td>music</td>
<td>wind, blow</td>
<td>2</td>
</tr>
<tr>
<td>cinema</td>
<td>star</td>
<td>1</td>
</tr>
<tr>
<td>history</td>
<td>equal</td>
<td>1</td>
</tr>
<tr>
<td>politics</td>
<td>govern</td>
<td>1</td>
</tr>
<tr>
<td>pedagogy</td>
<td>course</td>
<td>1</td>
</tr>
<tr>
<td>theatre</td>
<td>star</td>
<td>1</td>
</tr>
<tr>
<td>medicine</td>
<td>suffer, simple, blow</td>
<td>3</td>
</tr>
<tr>
<td>meteorology</td>
<td>wind, blow</td>
<td>2</td>
</tr>
<tr>
<td>number</td>
<td>three</td>
<td>1</td>
</tr>
<tr>
<td>linguistics</td>
<td>star, long, understand</td>
<td>3</td>
</tr>
<tr>
<td>body, care</td>
<td>shine</td>
<td>1</td>
</tr>
<tr>
<td>biology</td>
<td>blow, life</td>
<td>2</td>
</tr>
<tr>
<td>town, planning</td>
<td>verge</td>
<td>1</td>
</tr>
<tr>
<td>building, industry</td>
<td>course</td>
<td>1</td>
</tr>
<tr>
<td>person</td>
<td>star, simple, love, equal</td>
<td>4</td>
</tr>
<tr>
<td>anatomy</td>
<td>heart</td>
<td>1</td>
</tr>
<tr>
<td>quality</td>
<td>like, strong, simple, great, long, equal</td>
<td>6</td>
</tr>
<tr>
<td>sexuality</td>
<td>know, passion, love, blow</td>
<td>4</td>
</tr>
<tr>
<td>tourism</td>
<td>love</td>
<td>1</td>
</tr>
<tr>
<td>art</td>
<td>star</td>
<td>1</td>
</tr>
<tr>
<td>publishing</td>
<td>star</td>
<td>1</td>
</tr>
<tr>
<td>computer, science</td>
<td>search, seek</td>
<td>2</td>
</tr>
<tr>
<td>geography</td>
<td>ocean</td>
<td>1</td>
</tr>
<tr>
<td>hunting</td>
<td>course</td>
<td>1</td>
</tr>
<tr>
<td>play</td>
<td>love</td>
<td>1</td>
</tr>
<tr>
<td>cricket</td>
<td>over</td>
<td>1</td>
</tr>
<tr>
<td>alimentation</td>
<td>strong</td>
<td>1</td>
</tr>
<tr>
<td>telecommunication</td>
<td>star</td>
<td>1</td>
</tr>
<tr>
<td>grammar</td>
<td>strong</td>
<td>1</td>
</tr>
<tr>
<td>botany</td>
<td>simple</td>
<td>1</td>
</tr>
<tr>
<td>sport</td>
<td>course, love, equal</td>
<td>3</td>
</tr>
</tbody>
</table>

Table B.1: Doc Rprsnt. of Russell’s Passage tf >= 1
BIBLIOGRAPHY


