INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

ProQuest Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600

AUTOMATIC TERM EXTRACTION AND DOCUMENT SIMILARITY IN SPECIAL TEXT CORPORA

by

Li Dong

Submitted in partial fulfillment of the requirements for the degree of
Master of Computer Science

at

Dalhousie University
Halifax, Nova Scotia
March 2002

DALHOUSIE UNIVERSITY
DEPARTMENT OF COMPUTER SCIENCE

The undersigned hereby certify that they have read and recommend to the Faculty of Graduate Studies for acceptance a thesis entitled "AUTOMATIC TERM EXTRACTION AND DOCUMENT SIMILARITY IN SPECIAL TEXT CORPORA" by LI DONG in partial fulfillment of the requirements for the degree of Master of Computer Science.

Dated: April 29, 2002
Supervisor: E. Milios
Readers:

DALHOUSIE UNIVERSITY

DATE: April 29, 2002

AUTHOR: Li Dong

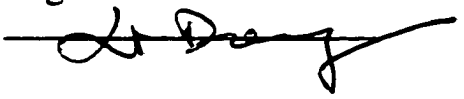
TITLE: Automatic Term Extraction and Document Similarity in Special Text Corpora

DEPARTMENT OF SCHOOL: Faculty of Computer Science

DEGREE: Master CONVOCATION: May,2002 YEAR: 2002

Permission is herewith granted to Dalhousie University to circulate and to have copied for non-commercial purposes, at its discretion, the above title upon the request of individuals or institutions.

Signature of Author



Automatic term extraction and document similarity in special text corpora

Li Dong

April 30, 2002

Abstract

The first objective of this thesis is to evaluate the performance of the C-value/NC-value methods, which are state-of-the-art methods for automatic term extraction in special text corpora, on a corpus composed of computer science articles and compare it with its published performance on a medical corpus. The C-value/NC-value method can automatically extract multi-word terms from special text corpora and can handle nested terms. It has been experimentally confirmed to outperform previously published automatic term extraction methods on a medical corpus. The second objective of the thesis is to use the extracted terms as features to estimate the similarity of papers in the computer science corpus using the standard Vector Space Model based on TF-IDF. Precision of the term-based method is evaluated and compared with the standard bag-of-words approach, as well as with a link-based method, which estimates the similarity of papers based on the overlap of their local neighborhoods in the citation graph.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Automatic term extraction in special text corpora is an interesting problem, that is becoming relevant as literature in specific scientific fields such as medicine, biology and computer science explodes making it difficult to track the evolving terminology in the field [8]. Early approaches to automatic term extraction were focused on information-theoretic approaches based on mutual information in detecting collocations (ch. 5 of [12]. Collocations are expressions that are composed of two or more words, the meaning of which is not easy to guess from the meanings of the component words. There are nuances in the detection of collocation that require linguistic criteria to resolve [7]. Shallow linguistic criteria are based on acceptable sequences of part-of-speech tags. Part-of-speech tagging can be performed automatically [3]. A key problem is that of nesting, where subsets of consecutive words of terms consisting of multiple words would satisfy the statistical criteria for "termhood", but they would not be called terms.

In the first part of this thesis, we describe experiments with a state-of-the-art method for automatic term extraction, that combines statistical and linguistic information [6], applied to a special text corpus of computer science articles, which is of a different nature from the medical corpus on which the method was originally tested. We confirmed that the performance of the method is equally good on our corpus, and we identified some adjustments that the method required.

In the second part of this thesis, we use the terms extracted to estimate the

similarity between two documents. We evaluate the quality of the similarity estimation based on terms in an information retrieval context. It is broadly believed that it is difficult to improve upon the bag-of-words representation as far as retrieval performance is concerned by using more sophisticated features or shallow linguistic techniques. Although retrieval based on terms did not show significant improvement over a bag-of-words representation, our long-term objective is to cluster special text corpora into subareas, and automatically generate lexical ontologies from the clusters [2]. Terms in this context are of interest in themselves, and not purely as a vehicle to information retrieval. We are, furthermore, interested in similarity criteria taking into account proximity of terms [9], for which again it is essential to work with terms, not words. The use of terms instead of words may also be preferable in information dissemination, where given a database of profiles (of users) and a document, the profiles that match the document must be identified efficiently [10].

The contribution of this thesis has been to confirm that a state-of-the-art method for automatic term extraction performs well in different special text corpora, and that similarity estimation based on terms performs at least as well as the standard bag-of-words representation in a document retrieval context. We further compared the performance of term-based retrieval with that of a method based only on links in the citation graph constructed based on the references (and citations) of the computer science articles considered [18]. We observed overlap in the results from the term-based and the link-based methods, but also relevant articles returned by one but not the other method. So it appears that the methods need to be combined in order to get the best retrieval performance.

Chapter 2 of the thesis describes the C-value/NC-value methods for automatic term extraction method [6], including the preprocessing performed on the papers, which had to be converted to plain text from postscript format, and cleaned up. It also reports the experimental results on the computer science corpus, and compares them with the results on the medical corpus reported in the original publication describing the method [6].

Chapter 3 of the thesis describes the similarity estimation based on terms, includ-

ing the vocabulary and parameter selection, and the experimental results from the term-based and word-based methods, and compares them with those of the link-based method applied to the same corpus.

Chapter 4 of the thesis discusses the approach and directions for future research.

Chapter 2

Automatic term extraction

2.1 Introduction

Technical terms, called terms in this paper, are linguistic representation of concepts [16]. For example, DISTRIBUTED SYSTEM, OPERATING SYSTEM, SOFTWARE ARCHITECTURE are all technical terms. Generally, automatic term extraction combines linguistic and statistical knowledge together. With linguistic knowledge, noun strings are extracted and with statistical knowledge, strings are ranked with their likelihood to be valid technical terms. In this research, C-value/NC-value method, a domain-specific method used to automatically extract terms, was implemented and tested on a computer science corpus. The precision and recall were compared to confirm the performance in the medical corpus.

2.2 The C-value method

C-value method[6] is a domain-specific method we use to automatically extract multi-word terms. It aims to get more accurate terms, especially those nested terms, such as MUTUAL INFORMATION is nested in MUTUAL INFORMATION PLOT and TYPICAL MUTUAL INFORMATION. The C-value method also has enhancement in non-nested terms because it put a term's length into consideration. With the C-value method, a computer science corpus is input and candidate terms are extracted

Before the tagging
Boolean Algebra forms a cornerstone of computer science and digital system design. Many problems in digital logic design and testing, artificial intelligence, and combinatorics can be expressed as a sequence of operations on Boolean functions.
After the tagging
Boolean/NNP Algebra/NNP forms/VB a/DT cornerstone/NN of/IN computer/NN science/NN and/CC digital/JJ system/NN design/NN ./. Many/JJ problems/NNS in/IN digital/JJ logic/NN design/NN and/CC testing/NN ,/, artificial/JJ intelligence/NN ,/, and/CC combinatorics/NNS can/MD be/VB expressed/VBN as/IN a/DT sequence/NN of/IN operations/NNS on/IN Boolean/NNP functions/NNS ./.

Table 2.1: Example of part-of-speech tagging. The meaning of the tags is available in appendix A

and ordered by their C-value, the larger value means the more probability for a candidate term to be a real term. C-value method combines linguistic knowledge (which consists of part-of-speech tagging, linguistic filters, stop list) and statistical knowledge (an improvement method based on other simple statistical methods), and put much focus on statistical information.

1. Linguistic part.

   The linguistic part consists of part of speech tagging, linguistic filter and stop list.

   POS tagging is the method to give each word in the corpus a grammatical tag as noun, verb, adjective, adverb or preposition, which is a prerequisite step to apply the corpus on linguistic filters [6] and get corresponding noun phrases. Table 2.1 shows a sample of tagging result using a rule-based part of speech tagging (see Appendix A for explanation of tags).

   Linguistic filter is used to extract all type of strings, such as noun phrases, adjective phrases. The choice of linguistic filters is decided by how many strings and how accurate they are we want to get.

   A stop list is used to store the words which are not expected to occur as term words[6].

2. Statistical part

Frequency of occurrence is a simple and popular statistical method. But in terms extraction it may not be a better method because a string with higher frequency of occurrence may not be a valid term. For example, N-DIMENSIONAL WEIGHT VECTOR is a term extracted from a computer science paper and its frequency of occurrence is 10, N-DIMENSIONAL WEIGHT as a noun substring is extracted too with frequency 10, but it is not a term.

C-value has been experimentally confirmed to be a better statistical method in medical corpus with higher precision on also-nested terms (terms have also appeared as nested), only-nested terms (terms have only appeared as nested) and non-nested terms (terms has no longer terms contain them). [6]

The C-value is given as following:

$$C\_value(a) = \begin{cases} \log_2 |a| . f(a) & \texttt{a is not nested,} \\ \log_2 |a| (f(a) - \frac{1}{P(T_a)} \sum_{b \in T_a} f(b)) & \texttt{otherwise} \end{cases} \quad (2.1)$$

where a is the candidate string,

$f(a)$ is the frequency of occurrence of a in the corpus,

T_a is the set of extracted candidate terms that contain a,

$P(T_a)$ is the number of these longer candidate terms.

From above, we get four components which determine the C-value, they are:

- the frequency of a candidate term, $f(a)$
- the length of a candidate term , $l(a)$
- if the candidate string is a nested string, the number of its longer strings, $P(T_a)$
- the total frequency of its longer strings, $F(b)$.

  If a candidate term is not a nested string, the C-value is determined by its frequency and its length. Otherwise, the number and frequency of its

frequency	string
136	SCHEDULING ALGORITHM
60	LOOP SCHEDULING ALGORITHM
6	DYNAMIC LOOP SCHEDULING ALGORITHM
5	ON-LINE SCHEDULING ALGORITHM
5	VARIOUS LOOP SCHEDULING ALGORITHM
5	AFFINITY SCHEDULING ALGORITHM
3	STATIC SCHEDULING ALGORITHM

Table 2.2: Strings that contain SCHEDULING ALGORITHM, illustrating nesting.

longer strings will be considered as the negative effect.

Table 2.2 is a set of extracted strings from a corpus. "SCHEDULING ALGORITHM" is a nested string because there is another 6 different strings include it. In this case, $f(a)$=136, $l(a)$=2, $P(T_a)$= 6, $F(b)$=60+6+5+5+5+3=84. So, C-value(SCHEDULING ALGORITHM)= $\log_2 2(136 - \frac{84}{6}) = 122$

The string LOOP SCHEDULING ALGORITHM is also a nested string because it is a part of DYNAMIC LOOP SCHEDULING ALGORITHM or VARIOUS LOOP SCHEDULING ALGORITHM. The calculation of C-value is in the same way with SCHEDULING ALGORITHM.

C-value(LOOP SCHEDULING ALGORITHM)= $\log_2 3(60 - \frac{11}{2}) = 86.38$

The other five strings do not appear in other longer strings and they are therefore assigned their C-value using the formula 1 (a is not a nested).

C-value(DYNAMIC LOOP SCHEDULING ALGORITHM)= $\log_2 4 * 6 = 12$

C-value(ON-LINE SCHEDULING ALGORITHM)= $\log_2 3 * 5 = 7.92$

C-value(VARIOUS LOOP SCHEDULING ALGORITHM)= $\log_2 4 * 5 = 10$

C-value(STATIC SCHEDULING ALGORITHM)= $\log_2 3 * 3 = 4.75$

In the C-value algorithm, the C-value formula takes into account the length (number of words) of candidate strings, because it assumes that a longer string has more possibility to be a term when its frequency is same as a shorter string. For a nested string, the reduction in C-value is the sum of the frequency of the longer strings containing it.

2.3 The application on a computer science corpus

Figure 2.1 gives an overview of the application of the C-value/NC-value method on a computer science corpus. Firstly, the corpus papers were downloaded from a technical web site; Secondly, the papers format were converted to fit for the POS tagging program; Thirdly, the preprocessing program was used to get rid of equations, references, adjust the punctuation to get higher quality candidate terms; Then rule-based part-of-speech tagging was applied to the papers; Finally the tagged papers were input into the term extraction program to get candidate terms list with the order of their likelihood to be true terms. The details for each step are presented in subsequent sections.

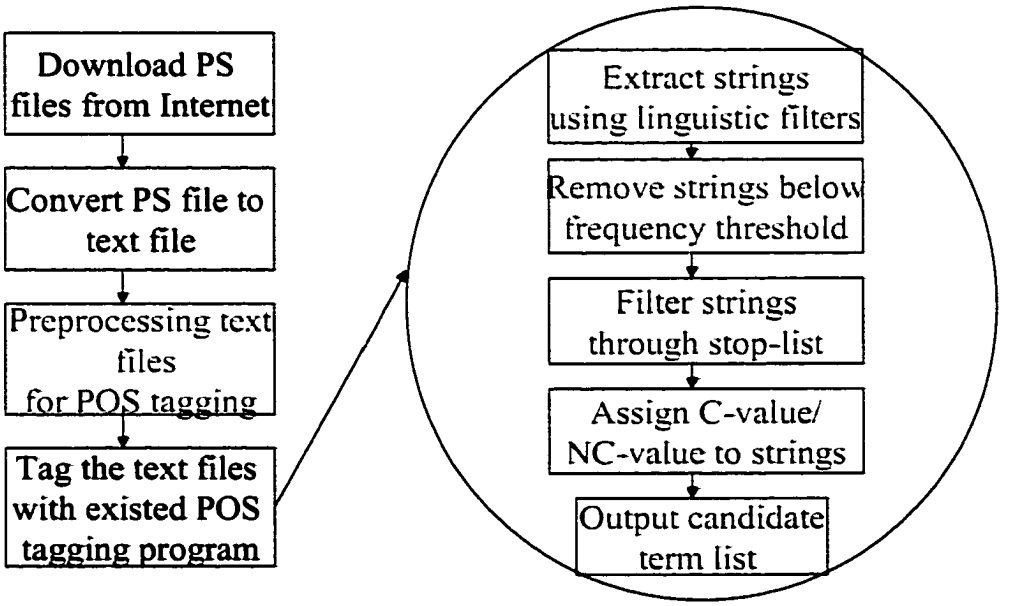


Figure 2.1: Overview of the term extraction process starting with postscript files of the articles of the corpus. The four steps on the left represent the preprocessing required.

2.3.1 Corpus selection

The computer science corpus was selected from a database collected by Yuan An [1] by querying ResearchIndex [11] and crawling the citation graph. The 81 Software

Engineering papers, total 819,171 words, with the highest number of citations were chosen to form a mini-corpus for the purpose of this study. The mini-corpus consists of "authority" papers in software engineering. The process of constructing the software engineering corpus of approximately 20,000 papers, from which the 81 papers were selected, is described in [1]. The total number of words in the mini-corpus of 81 papers is consistent with that of the medical corpus, which consists of 810,719 words of a few thousand short eye-pathology medical records in the corresponding application. Because the selection of articles under the software engineering topic was automated, and terminology usage in computer science is probably less precise than in eye-pathology diagnoses, we expect the computer science corpus to be less coherent than the medical corpus, with more terms and fewer instances of each term present in the corpus.

2.3.2 File format conversion

Papers are typically available for downloading either in Postscript or in PDF format, and the Postscript format was used because of its ease to convert to text format, which is a recognized input file format to the existed part of speech tagging program.

2.3.3 Text preprocessing for POS tagging

The corpus was tagged with a simple rule-based part of speech program [4].

The rule-based POS program can automatically acquire its rules and tag the corpus with accuracy compared with other stochastic tagging methods. It has many advantages such as a vast reduction in stored information required, the perspicuity of a small set of meaningful rules, ease of finding and implementing improvements to the tagging, and better portability from one tag set, corpus genre or language to another[3].

The following errors often occur when an original corpus is tagged, (see Appendix A for explanation of tags).

- The last word of a sentence ended by symbol '.' is tagged as '/CD' or '/JJ' by

mistake.

1./CD Introduction/NN Statistical/NNP modeling/NN addresses/NNS the/DT problem/NN of/IN constructing/VBG a/DT stochastic/JJ model/NN to/TO predict/VBP the/DT behavior/NN of/IN a/DT random/JJ process./CD

Baseball/NN managers/NNS employ/VB batting/VBG averages,/NN compiled/VBN from/IN a/DT history/NN of/IN at-bats,/JJ to/TO gauge/NN the/DT likelihood/NN that/IN a/DT player/NN will/MD succeed/VB in/IN his/PRP$ next/JJ appearance/NN at/IN the/DT plate./JJ

- A word and a symbol are tagged together as one word. From the following, '*stochastic language processing:* ', '*bilingual sense disambiguation,*' and '*word reordering,*' finally were extracted as candidate terms.

  In/IN Section/NN 5/NN we/PRP describe/VB the/DT application/NN of/IN maximum/JJ entropy/NN ideas/NNS to/TO several/JJ tasks/NNS in/IN stochastic/JJ language/NN processing:/NN bilingual/JJ sense/NN disambiguation,/NN word/NN reordering./NN and/CC sentence/NN segmentation./JJ

- Equations are tagged as normal sentences.

  p(dans)/NN +/SYM p(en)/NN +/SYM p('a)/NN +/SYM p(au/NN cours/NNS de)/NN +/SYM p(pendant)/NN =/SYM 1/CD

  As *p(dans), p(en), p('a)* and *p(pendant) are tagged as nouns and their frequency of occurrence is high, they can easily gain higher likelihood in the candidate terms list and considered as real terms by mistake.*

Ways were found out to solve the above problems for accurate tagging.

- If words and symbols are separated using blank spaces, the first two problems were resolved. The following are correct results based on the first three examples.

  1/CD ./. Introduction/NN Statistical/NNP modeling/NN addresses/NNS the/DT problem/NN of/IN constructing/VBG a/DT stochastic/JJ model/NN to/TO predict/VBP the/DT behavior/NN of/IN a/DT random/JJ process/NN ./.

Baseball/NN managers/NNS employ/VB batting/VBG averages/NN ,/, compiled/VBN from/IN a/DT history/NN of/IN at-bats/NN ,/, to/TO gauge/NN the/DT likelihood/NN that/IN a/DT player/NN will/MD succeed/VB in/IN his/PRP$ next/JJ appearance/NN at/IN the/DT plate/NN ./.

In/IN Section/NN 5/NN we/PRP describe/VB the/DT application/NN of/IN maximum/JJ entropy/NN ideas/NNS to/TO several/JJ tasks/NNS in/IN stochastic/JJ language/NN processing/NN :/: bilingual/JJ sense/NN disambiguation/NN ./, word/NN reordering/NN ,/, and/CC sentence/NN segmentation/NN ./.

- Since equations do not contain semantic meaning, they are not relevant to our term-based method and therefore the best way to gain higher precision is to filter out all equations. Because no software tool exists for this task, we observed that the equations have the following properties, which allow such filtering:

  - the number of words and symbols with length of 1 is more than half of the total words number, OR
  - the average word length is smaller than 2, OR
  - the number of words is smaller than the number of symbols.

  Based on the above properties, we were able to filter out more than 95% equations and no non-equation text was filtered out.

The process of conversion from Postscript file to text file using the ps2ascii command may also introduce errors. For example, the converted file has some constant '*' or some letters were replaced by constant '*'.

For example, *A Logic of Authentication Michael Burrows Mart**nAbadi Roger Needham that was not previously possible*It has drawn attention to features account of the problem*goes on*

*Digital Equipment Corporation ***** ****

*Authentication would be straightforward in a su*ciently benign environment**

The simple rule-based part of speech tags any constant '*' as '/NNP'. Unless the problem is corrected, candidate terms like *** *** al* *** *** object*oriented database systems* will be obtained.

Sometimes the constant '*' is also used to separate paragraphs, for correct tagging, they were replaced by blank or blank lines. For example, the problem would be corrected by changing the following text:

In this paper, we study the feasibility of providing real-time services on a packet-switched store-and-forward wide-area network with general topology.
****

This research has been supported in part by the Defense Advanced Research Projects Agency

into

In this paper. we study the feasibility of providing real-time services on a packet-switched store-and-forward wide-area network with general topology.
This research has been supported in part by the Defense Advanced Research Projects Agency

References were cut off from papers as well because they could not give direct information about that paper. This research's intention is term extraction based on the paper's contents.

2.3.4 Application of POS tagging

After the preprocessing, we tagged the corpus with an existing rule-based POS tagging program [4]. The output tagged corpus is the input corpus for the automatic term extraction program.

2.4 Term extraction Program

Our term extraction program is to input a tagged corpus and output a terms list with an order of likelihood of being valid terms.

1. Linguistic filters The linguistic filter[6] is used to extract different grammatical types of strings. In this research, all terms are assumed as noun phrases. For extracting noun phrases of any length, the three linguistic filters were applied and compared. They are :

   (a) Noun+ Noun

   (b) (Adj|Noun)+Noun

   (c) ((Adj|Noun) + ((Adj|Noun)* (NounPrep)?) (Adj|Noun)*) Noun

   From the first one, only nouns can show up in a string, such as TYPE CONSTRUCTOR. From the second one, a string can consist of both adjectives and nouns, such as RECURRENT RADIAL BASIS FUNCTION. And the third one is a "open" filter, it can have prepositions in a string, such as STRUCTURE COMBINING WITH INDIVIDUAL NAVIGATION.

   Obviously, different filters have different results on precision and recall. The first one is a "closed filter" [6], the candidate terms extracted using it will have higher precision and lower recall. The last two are "open filters"[6] which have the tendency to get higher recall but lower precision. To choose the best one, we should balance the precision and recall first. In this research, each of these filters was used to measure the results.

2. Frequency threshold

   After the application of the linguistic filters on the corpus, finally we got three lists of candidate terms.

   We draw the diagram based on the frequency of occurrence of candidate terms and their ranks extracted using linguistic filter 2, see figure 2.2. It demonstrates Zipf's law distribution[12] which states that the product of the frequency of occurrence and the rank order is approximately constant.

   With linguistic filter 1, we get a total of 18,275 candidate terms. Table 2.3 shows the number of candidate terms with frequency of 1, 2, 3, 4, 5 and their

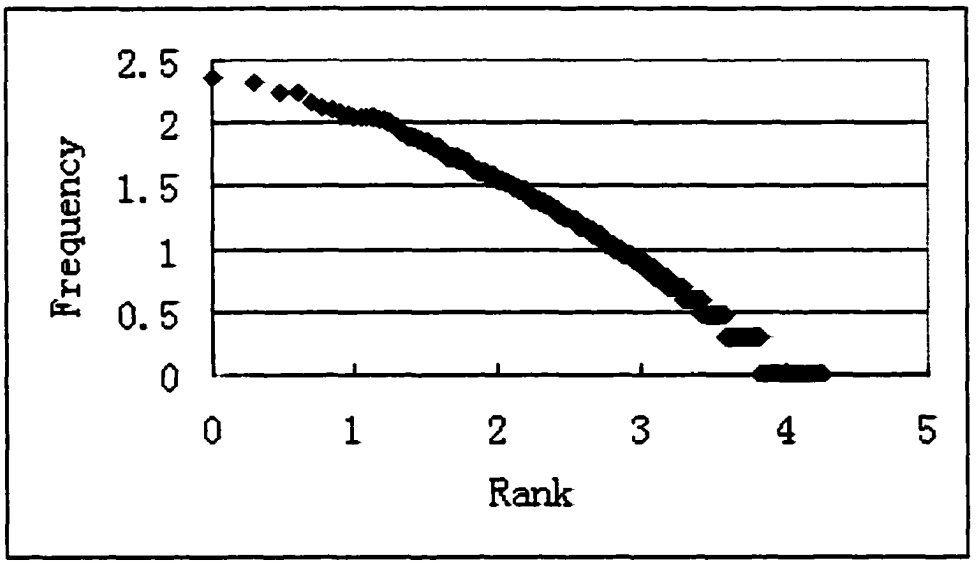


Figure 2.2: Frequency-rank diagram in log-log scale (base 10) for corpus terms. The approximate straight line confirms Zipf's law, that the frequency of a term is inversely proportional to its rank in the list, sorted by frequency of occurrence.

corresponding precision. From the table we know the number of terms with lower frequency is very large, for example, there are 11,536 candidate terms which occur only once with precision 14%, 2927 candidate terms occur only twice with precision 20%. In medical corpus, the lower frequent candidate terms, with frequency between 1 and 4, extracted with linguistic filter 3 have 30% precision.

Except the precision, the number of candidate terms from CS corpus is much larger than that of medical corpus. Table 2.4 gives us a comparison of term numbers extracted using linguistic filter 2 from CS corpus and medical corpus.

When C-value was applied on medical corpus, it did not use a frequency threshold. But in our application, we have much larger amount of candidate terms and lower precision on lower frequent strings. The value in Table 2.3 only shows the result of linguistic filter 1, we will get a much larger number of candidate terms and lower precision if linguistic filter 2 and filter 3 are used. To decrease the workload of manual evaluation, get better precision, not lose many terms,

frequency	extracted candidate terms number	precision
1	11,536	14%
2	2927	20%
3	1198	34%
4	687	49%
5	422	51%

Table 2.3: Precision for several frequency thresholds with linguistic filter 1 in page 13.

	Frequency larger than 2	Frequency larger than 0
Medical corpus	2,956 candidate terms	16,688 candidate terms
CS corpus	6,578 candidate terms	27,268 candidate terms

Table 2.4: Comparison of the number of extracted candidate terms with linguistic filter 2 in page 13 between the medical and the computer science corpus

a frequency threshold 3, that means all the candidate terms with frequency at least 3 were used.

3. Stop list

   A stop list contains words that are not likely to be helpful in a retrieval task. Common stop words are prepositions, pronouns, auxiliary verbs [12].

   See our stop list from Appendix B.

4. Calculation of the C-value

   To a string, if it doesn't appear in other longer strings, C-value is calculated C-value is calculated with the first equation in Equation 2.1; otherwise, with the second one in Equation 2.1.

   Tables 2.5, 2.6 and 2.7 give us samples of how the string nesting relationships look like and how unpredictably they can affect the computation.

   For example, when the string *HIDDEN MARKOV MODELS* was extracted from paper 9, it was a non-nested string and non-longer string. It changed to be a longer string after *MARKOV MODELS* was extracted from paper 170. Later on, after *COUPLED HIDDEN MARKOV MODELS*, *HIERARCHICAL HIDDEN MARKOV MODELS* and *AUTO-REGRESSIVE HIDDEN*

String ID	Paper ID	String
705	1	MUTUAL INFORMATION
533678	644	MUTUAL INFORMATION FEATURE
536404	650	JOINT MUTUAL INFORMATION
536589	651	RAW MUTUAL INFORMATION
536836	651	RAW MUTUAL INFORMATION ESTIMATE
536919	651	ADJUSTED MUTUAL INFORMATION ESTIMATE
539391	651	ADJUSTED MUTUAL INFORMATION
1399500	1867	TYPICAL MUTUAL INFORMATION
4319512	6029	MUTUAL INFORMATION PLOT

Table 2.5: Strings containing MUTUAL INFORMATION

String ID	Paper ID	String
9535	9	HIDDEN MARKOV MODELS
135754	170	MARKOV MODELS
1040474	1323	COUPLED MARKOV MODELS
2264691	3036	HIERARCHICAL MARKOV MODELS
4339920	6069	AUTO-REGRESSIVE MARKOV MODELS

Table 2.6: Strings containing MARKOV MODELS

MARKOV MODELS were extracted from paper 1323, 3036 and 6069, *HIDDEN MARKOV MODELS* became a nested string too.

Because the nested relationship has unpredictable effects when the method applied to a large corpus, the design of the data structure is an important part in the implementation. A database was designed to store the candidate terms and compute the C-value. In our database, we set up four tables named as follows:

String ID	Paper ID	String
2405	4	RADIAL BASIS FUNCTION
3148	4	RADIAL BASIS FUNCTION MODEL
76647	89	RADIAL BASIS FUNCTION NETWORK
97497	119	GAUSSIAN RADIAL BASIS FUNCTION
852989	1068	RADIAL BASIS FUNCTION PARAMETER
1054695	1336	RADIAL BASIS FUNCTION PARAMETER ESTIMATION
1626759	2185	RECURRENT RADIAL BASIS FUNCTION NETWORK
1627114	2185	RECURRENT RADIAL BASIS FUNCTION

Table 2.7: Strings containing RADIAL BASIS FUNCTION

(a) *temp*, which is a temporary table to record all the candidate terms extracted from the corpus through the frequency threshold, their length, frequency of occurrence and some other features,

(b) *term*, which records all the distinct candidate terms come from table *temp*, total frequency C-value and NC-value (See section 2.6 for detailed introduction to NC-value) .

(c) *relationship* which records the relationship between the nested strings and their longer strings,

(d) *wordweight*, which is used for weighting the words in the corpus by recording every unique word that appeared in the corpus. This four tables are enough to support our further computation of C-value.

For each paper,

- Use frequency threshold to extract candidate strings.
- Insert strings into the table *temp*.
- For each string,
  - Insert every word appearing in this string into table *wordweight*.
  - Search table *term* to find out whether this string exists (i.e. extracted from the papers processed so far). If it already exists, modify the frequency in both table *temp* and table *relationship*.
  - If not, add it to table *term* and search all the strings in table *term* to get all the longer strings, which contain this string, or the shorter strings nested in this one. Record all their relationships into table *relationship*.

Using existing data to calculate words weight, C-value.

Figure 2.3 gives an example of the data processing after the insertion of the candidate term JOINT MUTUAL INFORMATION. First, insert the candidate term into table *temp* to record which paper it appeared in and other features,

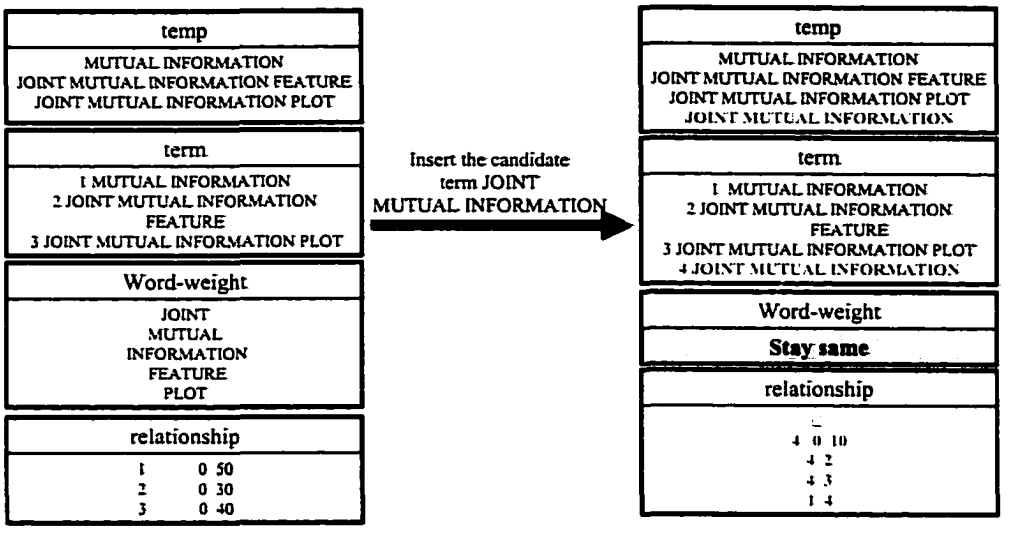


Figure 2.3: Data processing

such as frequency and number of words in the term and their weighting. Second, find out whether the candidate term exists in table *term*. If not, insert it and assign an ID to it (the ID is 4 in the example of Figure 2.3). At the same time, put all its context words into table *wordweight*. Finally, modify the *relationship* table to record the nesting relationships of the candidate terms. For example, records "4 2" and "4 3" in the table represent the nesting of candidate term with ID 4 within the candidate terms with ID numbers 2 and 3. This data will be used for the accurate C-value/NC-value calculation.

To all strings, calculate NC-value using their C-value and their term weights.

5. Term list output

   The term list is output sorted by the confidence in being true terms.

2.5 Evaluation for C-value

Since the automatic term extraction process is empirical, [8], we evaluated the C-value from its precision and recall in computer science corpus and compare the result with that of medical corpus.

	Relevant	Not relevant
Selected	t_p	f_p
Not selected	f_n	t_n

Table 2.8: The definitions of Precision and Recall

2.5.1 Introduction to Precision and Recall

The definitions of precision and recall are based on a 2*2 matrix showed in Table 2.8 [12].

t_p: number of relevant items retrieved by the system

f_p: number of not relevant items retrieved by the system

f_n: number of relevant items not retrieved (missed) by the system

t_n: number of not relevant items not retrieved by the system

Precision is the fraction of selected items that are relevant, equals to $\frac{t_p}{t_p+f_p}$, while recall is the fraction of relevant items that are selected, equals to $\frac{t_p}{t_p+f_n}$.

2.5.2 Precision evaluation

We compared the precision of C-value/frequency based on the following idea which was used in the medical corpus evaluation. For each linguistic filter, we measure the precision of C-value/frequency on

- the also-nested terms
- the only-nested terms
- all terms

Table 2.9 is the statistical result from a computer science corpus and Table 2.10 is from the medical corpus based on the same statistical purpose.

The precision from both corpora give very similar conclusion:

- For only-nested candidate terms, C-value shows almost 10% higher precision than frequency occurrence. Among three filters, filter two not only extracts more candidate terms but also achieves higher precision than the other two.

		1st filter	2nd filter	3rd filter
also-nested	C-v	64.76	73.39	58.45
	freq	58.36	70.26	33.8
only-nested	C-v	56.81	65.625	32.08
	freq	47.19	56.87	22.19
all	C-v	61.63	61.5	47.06
	freq	60.15	61.12	31.95

Table 2.9: Comparison of precision of term extraction on the CS corpus of C-value vs Frequency methods, and for each linguistic filter, described in page 13

		1st filter	2nd filter	3rd filter
also-nested	C-v	40.76	44.18	39.58
	freq	34.4	37.59	31.96
only-nested	C-v	50	60	54.54
	freq	18.57	22	12.91
all	C-v	38	36	31
	freq	36	35	30

Table 2.10: Comparison of precision of term extraction on the medical corpus of C-value vs Frequency methods, and for each linguistic filter, described in page 13. Data was obtained from [6].

- For also-nested candidate terms, C-value also shows higher precision than frequency occurrence but not quite apparent than only-nested results. Filter two gets higher precision than the other two filters.

- For all the candidate terms, C-value shows better results than frequency although the difference is very small. It also shows that filter one gains 0.1% higher precision than filter two but filter two can get twice as many candidate terms as filter one.

- Comparing the three filters, filter one is a closed linguistic filter which gets higher precision; filter three is an open linguistic filter with lower precision which extracts three times more candidate terms than filter one and two times more candidate terms than filter two. In our application, filter two is a satisfactory filter which will not lose much precision and get more real terms.

- Computer science corpus shows better precision than medical corpus with the use of a frequency threshold.

	1st filter	2nd filter	3rd filter
Strings number after using frequency threshold 3	3,812	6,578	12,029
Term number	2,293	4,020	3,864
Terms number after using C-value threshold 0	2,293	3,944	3,677

Table 2.11: Terms number after using two thresholds

1st filter	2nd filter	3rd filter
99.96%	98.1%	95.16%

Table 2.12: Recall from CS corpus using C-value

2.5.3 Recall evaluation

In the processing, first we use frequency threshold equal to 3 to get the candidate terms. This means all the string frequencies are greater than 3 or equal to 3, and then we use additional C-value threshold 0 to filter the remaining strings, that means we cut off all the only-nested terms that appeared only in one longer string.

Table 2.11 gives the term number after using the two thresholds. From it, we can know how many terms we lost after we applied the C-value threshold.

For both corpora:

1. Table 2.12 is the recall from computer science corpus after using C-value threshold. With C-value filter, the recall falls 0.1% with the first linguistic filter, 2% with the second filter and 5% with the third one.

2. Table 2.13 is the recall from medical corpus. Using C-value filter, recall falls less than 2% with the first linguistic filter, and around 2.5% with the second and the third filter.

3. Recall almost do not lose at all using C-value with first two linguistic filters for both corpora.

4. This shows exactly that C-value 'attracts' real terms more than pure frequency of occurrence, placing them closer to the top of the extracted list [6].

1st filter	2nd filter	3rd filter
98.22%	97.41%	97.47%

Table 2.13: Recall from medical corpus using C-value

2.5.4 Analysis on the C-value method

From precision and recall, C-value method shows better result than Frequency of occurrence when performing automatic term extraction. We can also see their difference in assigning term likelihood to candidate terms from some simple cases.

1. For non-nested candidate terms.

   C-value method only has a little difference compared with frequency of occurrence when a candidate term is not a nested string. In this case, C-value take the candidate term's length into consideration.

   In Table 2.14, when *PARALLELIZABLE INTERFERENCE GRAPH* and *TEMPORAL LOGIC* occur same times in a corpus, C-value method gives the higher term likelihood to the longer candidate term because it trusts that with same frequency of occurrence a longer candidate term has more possibility to be a real term .

2. For nested terms.

   C-value method not only considers a candidate term's length but also takes into account the number of times of its independent occurrences, which leads to a significant improvement over using the frequency of occurrence. Table 2.15 shows how the C-value method adjusts frequency of occurrence on nested terms. For example, the term *CONSTRAINED EXPRESSION* appears 111 times in the corpus, excluding the times appearing within longer strings, its C-value tunes down the frequency value and gives more precise way to judge its term likelihood.

Candidate term	C-value	Frequency
PARALLELIZABLE INTERFERENCE GRAPH	85	54
TEMPORAL LOGIC	54	54
TYPE CONSTRUCTORS	49	49
CONTROL MODES	49	49

Table 2.14: Likelihood of termhood given by C-value vs Frequency for non-nested terms

Candidate term	C-value	Frequency	Term
MARY SHAW ABSTRACTIONS	52	41	YES
SHAW ABSTRACTION	0	41	NO
CONSTRAINED EXPRESSION	109	111	YES
CONSTRAINED EXPRESSION FORMALISM	17	11	YES
CONSTRAINED EXPRESSION TOOLSET	33	21	YES

Table 2.15: Likelihood of termhood given by C-value vs Frequency for nested terms

2.6 Introduction to the NC-value method[6]

Context words are the words appearing in the terms. For example, both TEMPORAL and LOGIC are context words in the term TEMPORAL LOGIC, PARALLELIZABLE, INTERFERENCE and GRAPH are context words in the term PARALLELIZABLE INTERFERENCE GRAPH. NC-value incorporates context information to re-rank the top list of candidate terms. As we can imagine, when OPERATING SYSTEM and LINGUISTIC FILTER have the same C-value from a computer science corpus, we should give higher rank to OPERATING SYSTEM because the context words are more popular in the computer science domain. NC-value method aims to re-rank candidate terms by assigning them a context weight.

Our assumption is that the more frequent a word is in terms extracted from a specific domain/corpus, the higher probability it is related to this domain/corpus. In computer science domain, *computer*, *system*, *software*, *network*, *package* will gain higher weight when compared to *extraction*, *context*, *stage*. So in a corpus, we measure a word's weight from:

$$weight(w) = \frac{t(w)}{n} \tag{2.2}$$

where w is the context word, $weight(w)$ is the weight of w in this specific domain/corpus, $t(w)$ the number of terms w appears in, n the total number of different terms considered in this domain/corpus. The NC-value is computed by the following equation:

$$NC\_value(a) = 0.8\ C\_value(a) + 0.2 \sum_{b \in C_a} f_a(b) weight(b) \tag{2.3}$$

where a is the candidate term, C_a is the set of distinct context words of a, b is a word from C_a, $f_a(b)$ is the frequency of b as a term context word of a, $weight(b)$ is the weight of b as a term context word. C-value and context information have been assigned the weights 0.8 and 0.2 respectively as in [6]. To apply the NC-value method, three stages of computation should be performed.

1. We use C-value method to get a list of candidate terms and order them by C-value.

2. We extract context words from each term and record their appearance times. This stage will contribute to the third stage and improve the term distribution in the list. Every word is assigned a weight according to equation 2.2 to show its importance rating in this corpus. For example, we get some most popular words from the corpus we used in this paper, SYSTEM is assigned weight 0.02598 for 171 times appearance and STATE is assigned weight 0.02097 for 138 times appearance.

3. In the third stage, we combine word weight with C-value to get NC-value.

2.7 Evaluation of the NC-value method

2.7.1 Precision evaluation

The statistical data for NC-value method comes from the same corpus C-value method used in Section 2.3.

	[top-10]	(10-6]	(6-4]	(4-3]
NC-value	54%	51.4%	45.4%	43.6%
C-value	53.7%	48%	44.6%	44.1%
Frequency	25.4%	33.5%	37.5%	32.8%

Table 2.16: Distribution of precision on CS corpus according to the rank of the candidate terms, when candidate terms are listed according to likelihood of termhood. NC-value achieves higher precision on the terms at the top of the list.

Figure 2.4 and Figure 2.5 show the precision obtained from the computer science corpus and the medical corpus. Both of them used linguistic filter 3, the open filter. We had to use filter 3 on the computer science corpus, because the results of the NC-value method reported on the medical corpus were based on filter 3. The break-down of the horizontal axis into intervals is corpus-dependent and it is based on the requirement that each interval contain about the same number of candidate terms [6]. The frequency, C-value and NC-value intervals respectively for the medical corpus used in [6] are $[\geq 40]$, $(40, 10]$, $(10, 4)$, $[4, 1]$, while the intervals for the computer science corpus are $[\geq 10]$, $(10, 6]$, $(6, 4]$, $(4, 3]$. The vertical axis shows the precision on the terms belonging to the corresponding interval.

From the CS corpus, we notice the following:

- The NC-value method is more precise when applied on the candidate terms which frequency are larger than 4 .
- When the frequency is decreased to 3, the precision of the NC-value dropped somewhat.

The precision result from the CS corpus confirms that of the medical corpus: NC-value achieves higher precision on the terms at the top of the list than the other two methods. The differences in precision values between the computer science and the medical corpus may be explained by the medical corpus being likely to make more focused use of fewer terms.

Table 2.16 contains the data showed in Figure 2.4.

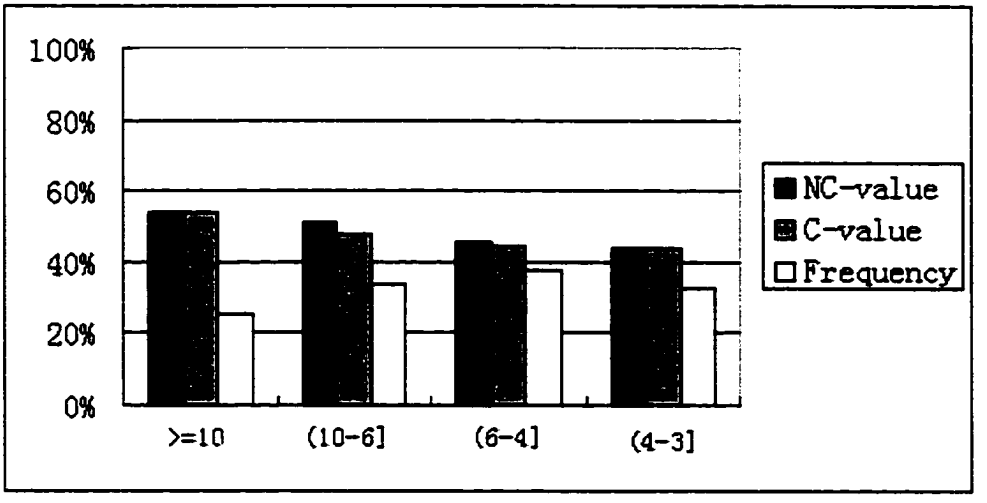


Figure 2.4: Distribution of precision on CS corpus: NC-value,C-value vs Frequency with linguistic filter 3 (see page 13). Horizontal axis corresponds to ranges of frequency/C-value/NC-value respectively.

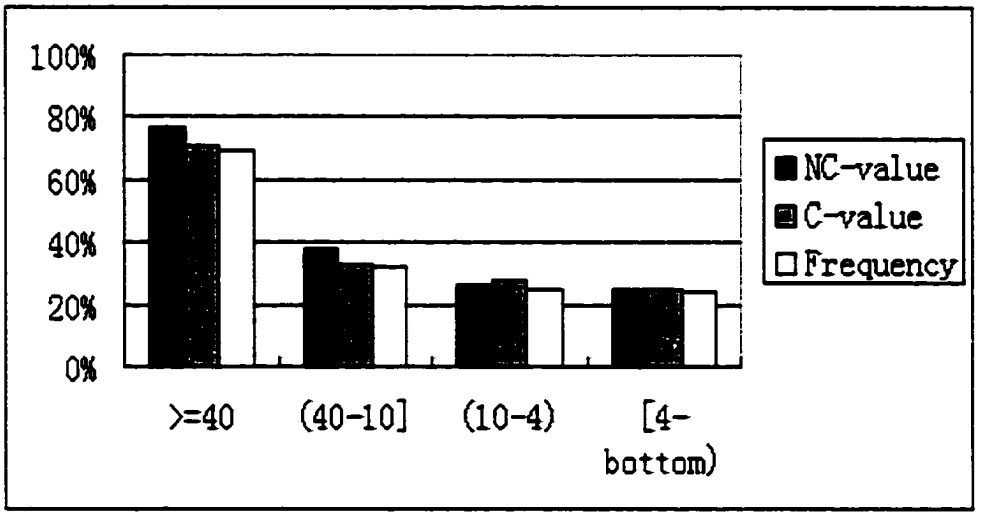


Figure 2.5: Distribution of precision on medical corpus: NC-value,C-value vs Frequency with linguistic filter 3

2.7.2 Recall evaluation

Due to the fact that the NC-value method is an improvement to the C-value method which assigns 0.8 weight on C-value and another 0.2 weight on term context words[6], the candidate term list is same as that of C-value method. The recall of the NC-value method is the same as that of C-value method.

2.7.3 Analysis on NC-value method

1. Context word weights.

   Table 2.17 lists 10 context words from top/middle/bottom of the context words list. From that we know 'SYSTEM','STATE','I-O','PROCESS','DATA' gain more weight in Software Engineering domain. If we do the same experiment on a Neural Network domain, the context word weights list will be changed.

2. Examples of how NC-value re-ranks terms list.

   Table 2.18 gives the examples of top 30 candidate terms with their Frequency, C-value and NC-value.

   RECOVERABLE SYSTEM and CONTROL FLOW have the same C-value in the top term list, in another words, they have the same rank in the list. After the NC-value method puts some focus on context words, the ranks are different. In our computer science corpus, SYSTEM gains first weight 0.02598, RECOVERABLE gains weight 0.00274, CONTROL gains weight 0.0117, FLOW gains weight 0.00714. After context words consideration , NC-value gives 79 to RECOVERABLE SYSTEM and 75 to CONTROL FLOW. Actually, the advantage of NC-value is not only an improvement of C-value, but the new idea of using context information. NC-value method can serve as a postprocessing step in any statistical term extraction method to act as a tuner to attract real terms to the top of the list.

word	weight	num. of terms
SYSTEM	0.02598	171
STATE	0.02097	138
I-O	0.01915	126
PROCESS	0.01687	111
DATA	0.01656	109
OBJECT	0.01489	98
TIME	0.01322	87
NEW	0.01307	86
TYPE	0.01292	85
INTERFACE	0.00851	56
ADAPTIVE	0.00091	6
ADDRESS	0.00091	6
AUXILIARY	0.00091	6
AVERAGE	0.00091	6
BALANCING	0.00091	6
BENCHMARK	0.00091	6
BOUND	0.00091	6
BOUNDED	0.00091	6
BUILDING	0.00091	6
COUNTER	0.00091	6
ABSOLUTE	0.00030	2
ACCEPTING	0.00030	2
ACCESS	0.00030	2
ACK	0.00030	2
ACKNOWLEDGEMENT	0.00030	2
ACL	0.00030	2
ACTIVITIES	0.00030	2
ACTUAL-IN	0.00030	2
ACTUAL-OUT	0.00030	2
AESOP	0.00030	2

Table 2.17: 10 context words from the top/middle/bottom of the list of context words sorted by weight.

NC-value	C-value	frequency	candidate term	term?
239	305	333	OPERATING SYSTEM	1
196	246	250	STATE INTERVAL	1
162	202	213	DEPENDENCE GRAPH	1
147	188	205	DISTRIBUTED SYSTEMS	1
145	182	188	SYSTEM STATE	1
136	171	173	COMMON KNOWLEDGE	1
133	169	174	SOFTWARE ARCHITECTURES	1
116	148	167	PROGRAMMING LANGUAGE	1
114	142	145	STATE FUNCTION	1
112	142	147	MESSAGE LOGGING	1
103	130	136	SCHEDULING ALGORITHM	1
101	128	131	CONCURRENCY CONTROL	1
96	121	125	STABLE STORAGE	1
94	118	126	RECOVERY STATE	1
89	112	114	CONTEXT GRAPH	1
87	109	111	CONSTRAINED EXPRESSION	1
87	109	110	LOOP SCHEDULING	1
85	106	71	CURRENT RECOVERY STATE	1
85	110	118	VIRTUAL MEMORY	1
80	100	102	ACTIVE COPY	1
79	96	99	RECOVERABLE SYSTEM	1
78	98	101	PROCESS MODELS	0
75	96	104	CONTROL FLOW	1
75	93	97	CURRENT RECOVERY	0
75	97	115	DATA STRUCTURE	1
74	93	60	LOOP SCHEDULING ALGORITHMS	1
74	93	97	CONTEXT SWITCH	1
73	91	61	RECOVERABLE SYSTEM STATE	1
72	90	103	DATA TYPES	1
72	92	108	RESPONSE TIMES	1

Table 2.18: The top 30 candidate terms extracted by NC-value. In the term? column 1 is Yes and 2 is No.

Chapter 3

Document similarity based on technical terms

In chapter 2, we used three method frequency/C-value/NC-value to get corpus terms and testified C-value/NC-value is better than frequency on the term precision. In this chapter, the terms extracted by these three methods will be used to form document vector space for similarity assessment and the similarity precision will be analyzed among the different methods.

3.1 Introduction to the Vector Space Model

The Vector Space Model is widely used for the measurement of similarity between documents [12] because of its conceptual and computational simplicity. Documents and queries are represented as vectors in a vector space, where the dimensions correspond to "features" (words or terms). Similarity is measured by the angle between vectors [12].

Our objective is to evaluate the use of terms as features in a vector space model. In Figure 3.1, we show a two-dimensional vector space, corresponding to the terms DECISION TREE and NEURAL NETWORK. The query document is represented by the vector $q_1(10, 10)$ which means DECISION TREE and NEURAL NETWORK have the same occurrence times in this query paper. The vector $d_1(2, 13)$ represents

the first document in which NEURAL NETWORK is a main topic and DECISION TREE may be only a passing reference. Vectors $d_2(8, 11)$ and $d_3(1, 12)$ also show the frequency of occurrence of both terms. From the figure, it is easy to see that d_2 is the most similar document with q_1 because it has smallest angle with q_1.

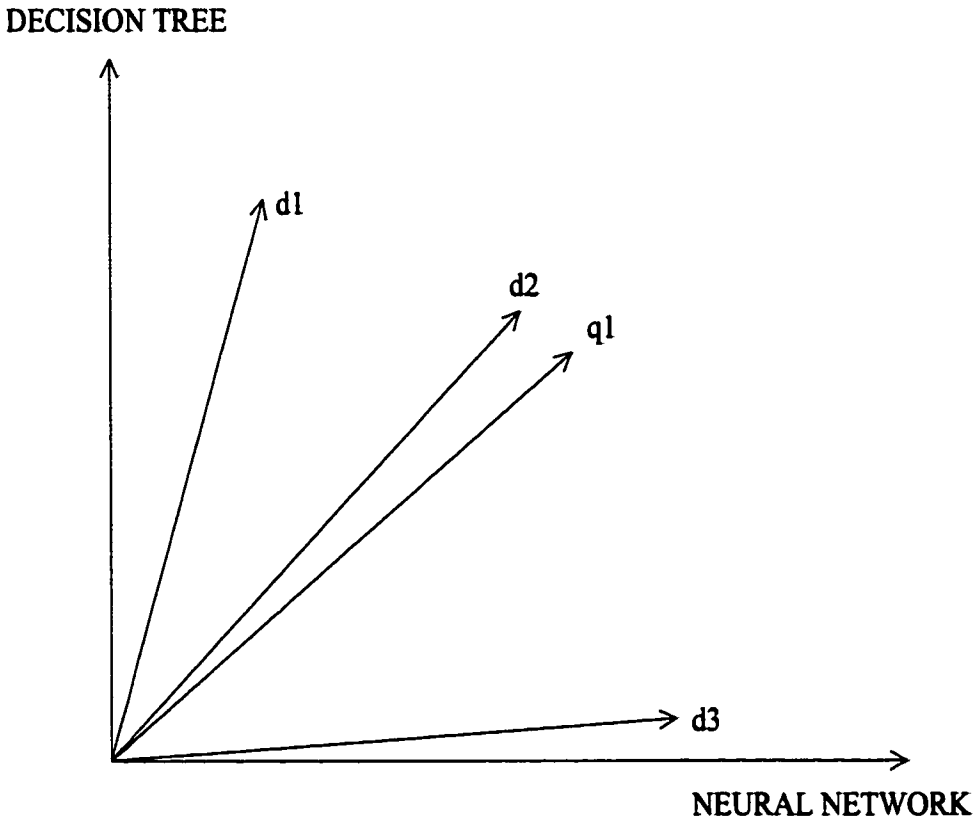


Figure 3.1: The Vector Space approach to document similarity. Dimensions correspond to terms, coordinate values correspond to the TF-IDF value for the particular term and document.

In the above figure, we use frequency as term weights which means the value of coordinates are the numbers of occurrence of the terms in the documents. Actually, there is a large family of so-called TF.IDF weighting schemes.

In this research, we applied the following equations [12] to measure the term weighting.

$$weight(i,j) = \begin{cases} (1 + \log tf_{i,j}) . \log \frac{N}{df_i} & if \quad tf_{i,j} \geq 1 \\ 0 & if \quad tf_{i,j} = 0 \end{cases} \tag{3.1}$$

in which

- $tf_{t,d}$ is the frequency of term t in document d
- df_t is the number of documents term t occurs in
- N is the total number of documents in the corpus

For example, when we give the word weighting to FORMULA and NETWORK from a corpus of a computer science magazine, FORMULA can occur in every article because almost all of them contain mathematical models, in contrast, NETWORK only occurs in the papers focus on computer network topic. That is the reason we can not give them the same weight when they have the same frequency occurrence in the corpus. When a term occurs in every document in the whole corpus, its weight is zero, since $\log(N) - \log(df_i) = \log(N) - log(N) = 0$, while when a term only occurs in one document, the IDF term in equation 3.1 is maximized, since $\log(N) - \log(1) = \log(N)$.

Documents are ranked in the vector space model by measuring their similarities with the query vector using the cosine distance.

$$cos(\vec{q}, \vec{d}) = \frac{\sum_{i=1}^{n} q_i d_i}{\sqrt{\sum_{i=1}^{n} q_i^2} \sqrt{\sum_{i=1}^{n} d_i^2}} \tag{3.2}$$

where $\vec{q}$ and $\vec{d}$ are n-dimensional vectors.

3.2 Introduction to paper similarity assessment

Figure 3.2 gives an overview of our objective and methods used in the similarity assessment. Firstly, randomly choose query papers, (in this research, we used 9 query papers which were used in [18] for document similarity assessment with link-based method,) tracked similar papers for each of the given 9 query papers using technical

terms from a computer science corpus, which is a collection of 10,426 papers. And then each top 10 similar papers were evaluated as related (score 1), somewhat related (score 0.5) or not related (score 0) papers.

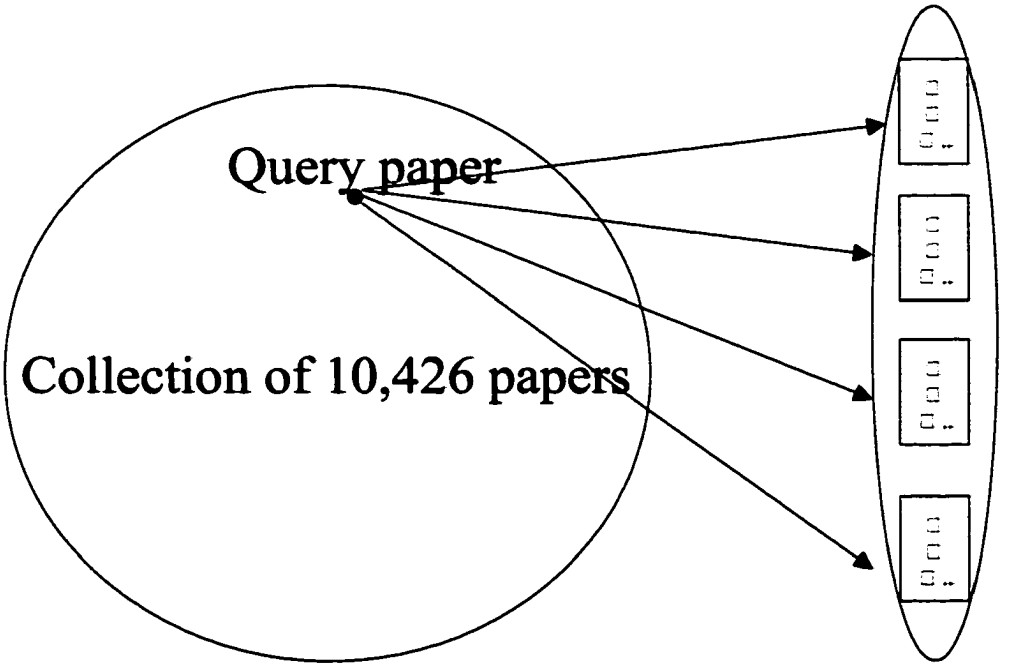


Figure 3.2: Overview of the process to evaluate document similarity estimation. On the right, a list of returned papers is shown, ranked by similarity to the query paper.

The different methods used are listed in the Table 3.1. In the text-based methods, we used both term-based method and word-based method to perform similarity assessment. We had two ways to get the corpus terms. One way was getting them from full papers with frequency of occurrence, C-value and NC-value, correspondingly the papers similarity is based on the full papers. Another way was getting the terms from papers abstracts with frequency of occurrence, C-value and NC-value method, correspondingly the papers similarity is based on the abstracts. In the word-based method, we extracted all the nouns from the corpus as features. The text- and word-based methods are compared with a link-based method using the same 9 query papers [18].

From text-based methods, we chose the best one to compare with the link-based method based on the precision and computation time.

	Text-based method: term-based	
Term Source	Term extraction method	Doc similarity source
Full papers	Frequency/C-value/NC-value	Full papers
Abstracts	Frequency/C-value/NC-value	Abstracts
	Text-based method: word-based	
Word Source	Word extraction method	Doc similarity source
Full papers	Frequency of occurrence	Full papers
	Link-based method	

Table 3.1: Similarity assessment methods

3.3 The application on a Neural Network corpus

First, we extract candidate terms from the corpus with linguistic filter 2, which was evaluated as having higher precision and recall, and frequency threshold 1 to get more candidate terms. 189,043 distinct candidate terms were extracted from the corpus.

Second, given a query document, we computed its distance to each document in the corpus, according to the following conceptual steps:

1. Select a subset of candidate terms as features.
2. For each document in the corpus, set up a vector in this feature space. The coordinates of this vector are equal to the term weight in this document, as described previously.
3. Set up query vector.
4. After all the documents are represented as vectors, compute distance between the query vector and all other vectors.
5. Record the distance, rank them order by distance.

3.3.1 Choice of terms to be used as features

There are 189,043 candidate terms extracted from the whole corpus with linguistic filter 2. But not all the terms are suitable for information retrieval [17]. We specified

two cut-offs to exclude the most frequent and the least frequent terms and we extracted the subset of terms with frequency between them [17]. We experimented with both frequency/C-value/NC-value and document frequency to choose the cut-offs, and with different cut-off points.

3.4 Evaluation on the similarity assessment

The top 10 similar papers for each of 9 query papers were judged by the domain experts (Dr. N. Japkowicz and Dr. E. Milios), and were assigned a score 1 (related), 0.5 (somewhat related) or 0 (not related).

3.4.1 Choosing cut-offs with frequency/C-value/NC-value

In the term-based method based on full papers, we experimented with two choices of cut-off points. In the first experiment, we chose the upper cut-off point as 2500 and the lower cut-off as 50, i.e. we only selected the candidate terms whose frequency/C-value/NC-value is between 50 and 2500. There are 10719 candidate terms within this area. In the second experiment, we reduced the spread by using 6100 candidate terms with the upper cut-off as 2000 and the lower cut-off as 100.

Table 3.2 is the term-based precision based on the full papers using three different cut-offs, while Figures 3.3 and 3.4 are the corresponding diagrams. From the figures, although in some ranks frequency shows higher precision than C/NC-value while some other ranks C/NC-value is better than frequency, generally, the three curves have almost the similar precision when using the full paper terms. That means for the papers similarity assessment, term extraction with frequency of occurrence is a good way and we do not need to apply the extra computation to extract terms with C-value/NC-value.

The tighter frequency cut-off interval $(100, 2000)$ gave better precision than the looser cut-off interval $(50, 2500)$, see figure 3.5. This is to be expected, because with fewer terms precision increases, but recall decreases.

In the term-based method based on paper abstracts, we used two cut-offs to

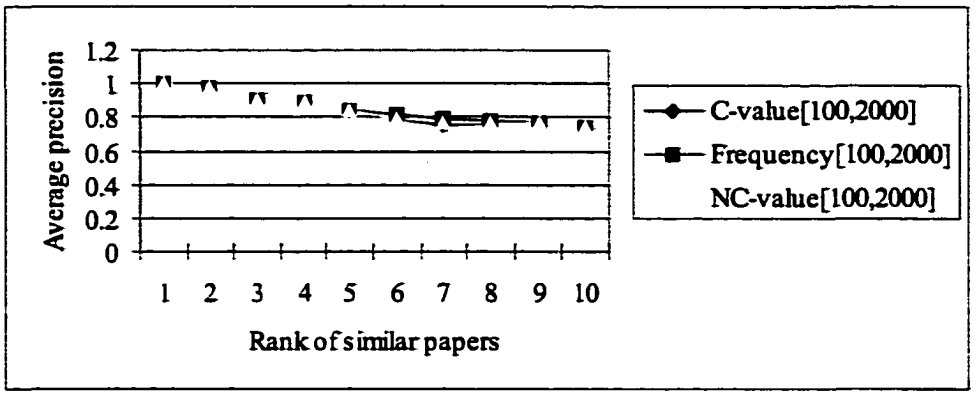


Figure 3.3: Precision diagram of term-based method on full papers using frequency/C-value/NC-value [50,2500] as cut-offs. The precision for rank N is the fraction of papers deemed related to the query paper by the expert among the top N most similar papers returned by the algorithm, averaged over the 9 query papers.

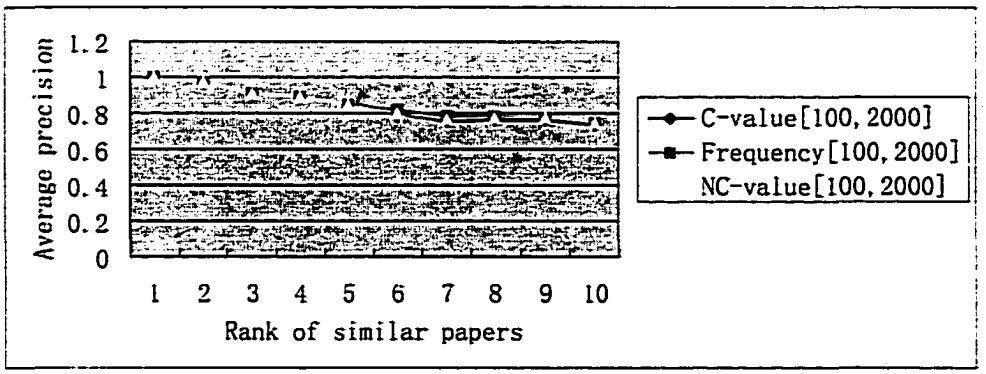


Figure 3.4: Precision diagram of term-based method on full papers using frequency/C-value/NC-value [100,2000] as cut-offs.

Rank No.	Fre[50,2500]	Cv	NCv	Fre[100,2000]	Cv	NCv
1	1	1	1	1	1	1
2	0.944	0.944	0.944	0.944	0.944	0.944
3	0.833	0.778	0.778	0.778	0.778	0.778
4	0.722	0.722	0.722	0.833	0.833	0.833
5	0.5	0.5	0.611	0.667	0.667	0.667
6	0.944	0.944	0.944	0.667	0.556	0.556
7	0.5	0.556	0.556	0.611	0.5	0.556
8	0.778	0.778	0.778	0.778	0.833	0.833
9	0.556	0.556	0.556	0.667	0.778	0.778
10	0.5	0.5	0.389	0.5	0.5	0.5

Table 3.2: Precision of Term-based method on full papers

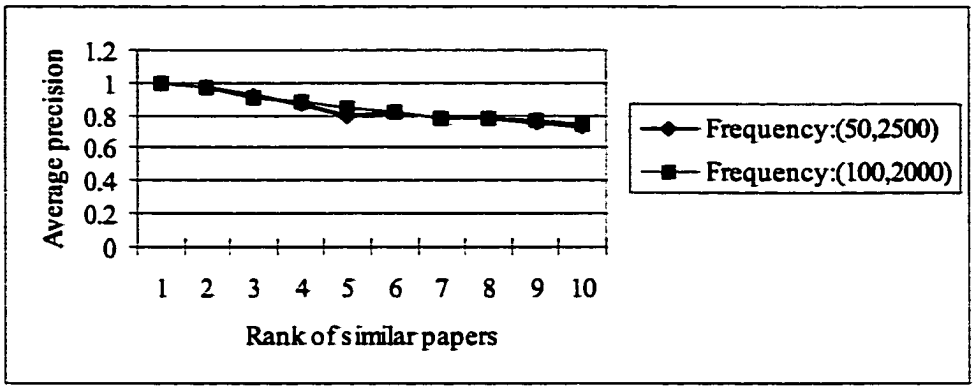


Figure 3.5: Precision comparison between frequency cut-offs

do the similarity assessment too. One is from 2 to 140 of frequency/C-value/NC-value containing 10300 candidate terms and another is from 5 to 100 of frequency/C-value/NC-value containing 6000 candidate terms. In figure 3.6 all the terms were extracted using frequency/C-value/NC-value between 2 and 140, while in figure 3.7 all the terms were extracted using frequency/C-value/NC-value between 5 and 100. Although the term extraction from papers abstract is faster because fewer terms were involved, it does not show better precision than full papers because abstracts are too short to perform statistical term extraction.

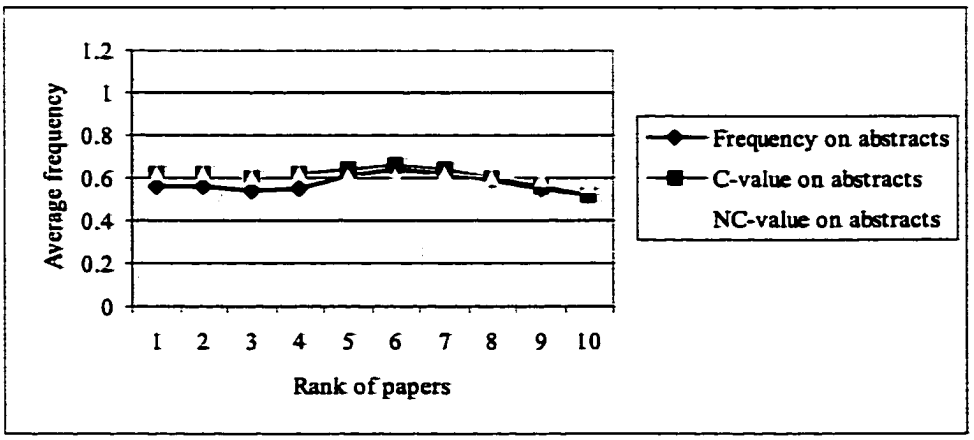


Figure 3.6: Precision of term-based method on papers abstract using cut-off points [2,140]

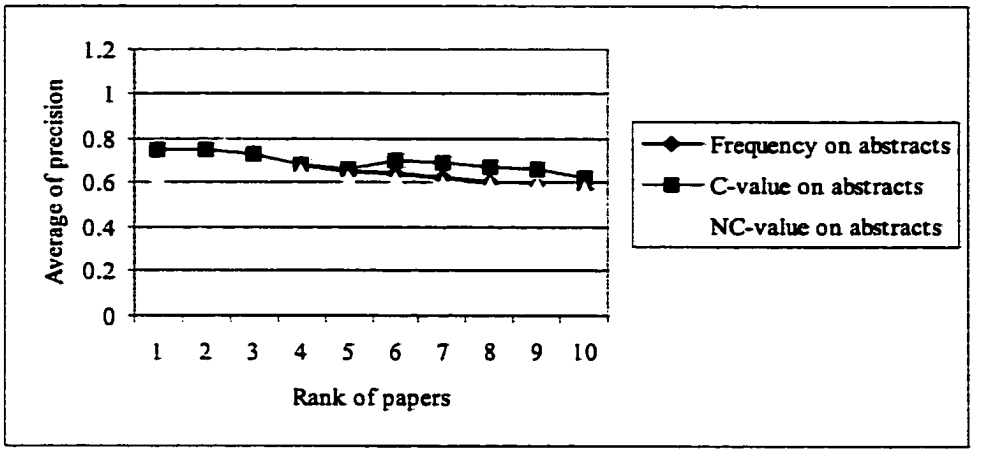


Figure 3.7: Precision of term-based method on papers abstract using cut-off points [5,100]

Ranked No.	Fre[2,140]	Cv[2,140]	NCv[2,140]	Fre[5,100]	Cv[5,100]	NCv[5,100]
1	0.5625	0.625	0.625	0.75	0.75	0.625
2	0.5625	0.625	0.625	0.75	0.75	0.75
3	0.5	0.5625	0.5625	0.687	0.688	0.625
4	0.5625	0.6875	0.6875	0.563	0.563	0.625
5	0.875	0.6875	0.5625	0.5	0.563	0.625
6	0.75	0.75	0.625	0.625	0.875	0.5
7	0.5	0.5625	0.625	0.5	0.688	0.625
8	0.375	0.3125	0.5	0.5	0.5	0.5
9	0.25	0.25	0.556	0.5	0.562	0.5
10	0.3125	0.1875	0.25	0.313	0.5	0.437

Table 3.3: Precision of Term-based method on papers abstract

3.4.2 Choosing cut-offs with document frequency[12]

In another experiment, a term document frequency is the number of documents of the collection in which the term occurs [12]. After we list the document frequency of the corpus terms with descending order, we experimented with three cut-off intervals: (7, 200), (4, 200) and (4, 250).

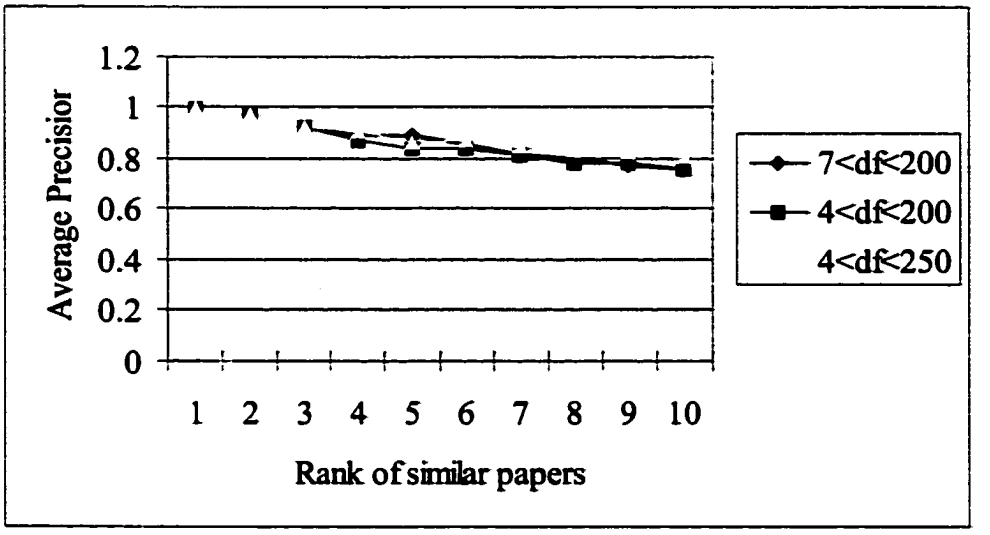


Figure 3.8: Terms chosen by document frequency to form the document vectors for similarity

The cut-off with document frequency (4, 250) shows better precision than the other two document frequency cut-offs (see figure 3.8), and frequency cut-offs (see figure 3.9).

3.4.3 Paper similarity assessment using word-based method

In a final experiment, we would like to compare the effectiveness of using terms as features to the standard approach of using words (nouns) as features. In the word-based method, we extracted all the nouns from the corpus and then generate a corpus words list ordered by their frequency. Figure 3.10 is a frequency-rank diagram for word list, which appears to follow a Zipf's law distribution by demonstrating the product of the frequency of occurrence and the rank order is approximately constant.

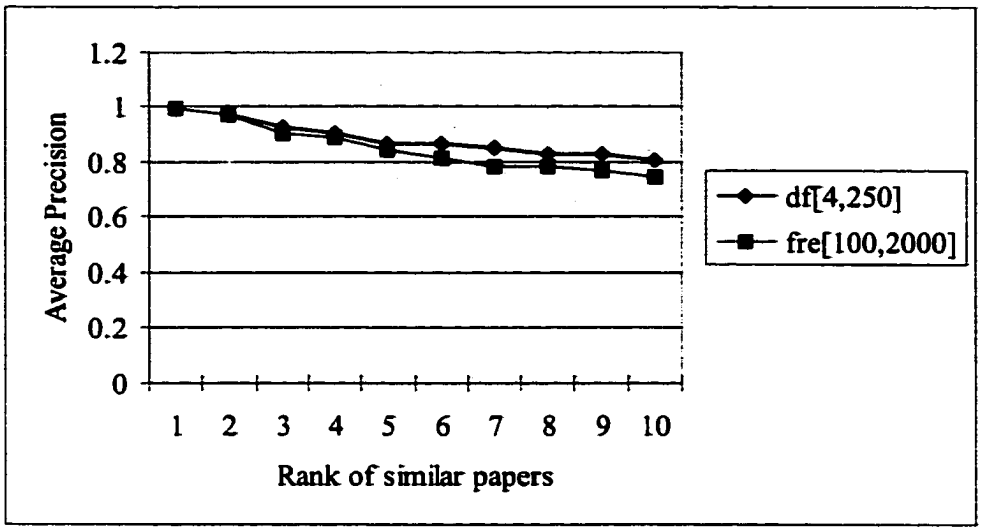


Figure 3.9: Precision comparison using cut-off with document frequency vs. frequency in the selection of terms to be used to form the document vectors

There were 2,475,125 words extracted and we used 11,060 words whose document frequency are from 12 to 8700 as features in the vector space representation.

Table 3.4 is the similarity precision result using word-based method and document frequency cut-off.

Ranked No.	DF[12,8700]
1	1
2	0.833
3	0.667
4	0.833
5	0.722
6	0.778
7	0.611
8	0.556
9	0.611
10	0.667

Table 3.4: Precision of word-based method

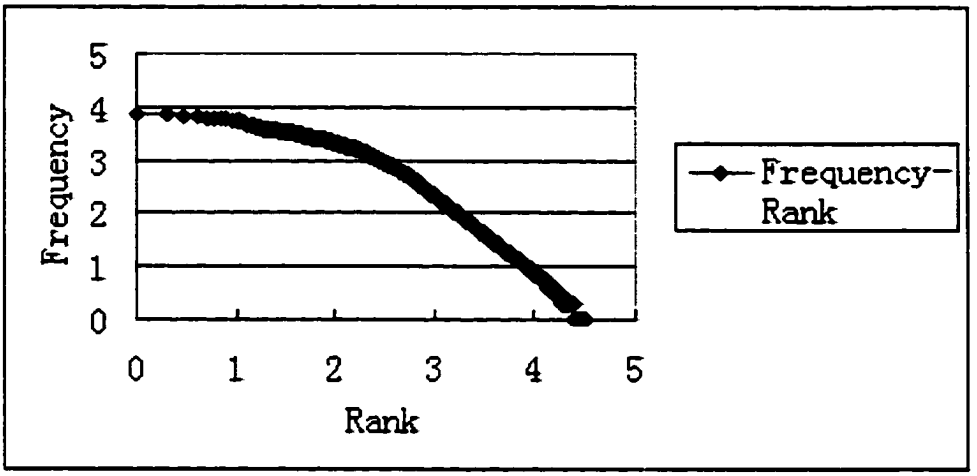


Figure 3.10: The Frequency-Rank Diagram in log-log scale (base 10) for corpus words.

3.4.4 Precision comparison between term-based and word-based method

We observe that the term-based method gave slightly better precision than word-based method, see figure 3.11. Whether this difference is statistically significant is deferred to future research.

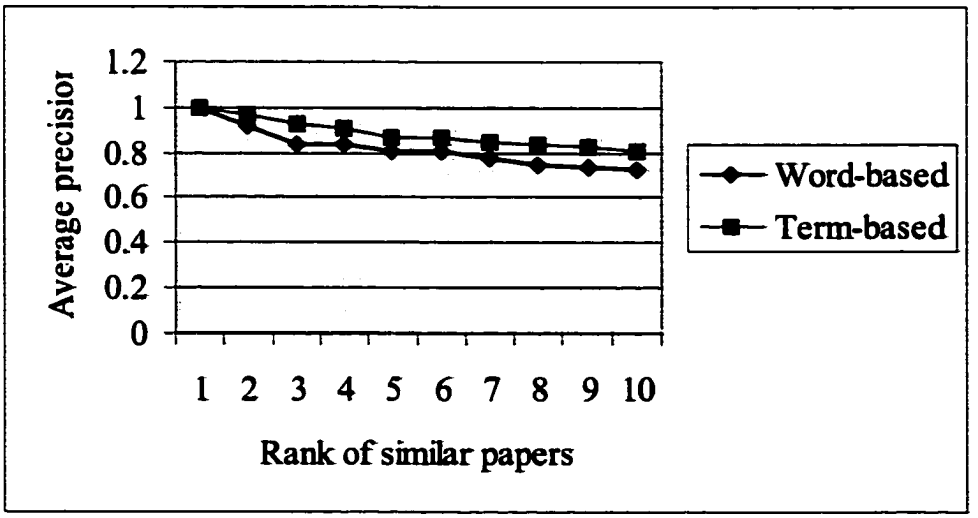


Figure 3.11: Precision comparison between term-based and word-based method

3.4.5 Complementarity of the term-based and word-based method

The word-based and term-based methods complement each other by producing different sets of related papers, see Figure 3.12. Average over 9 query papers, they have 4.4 relevant papers in common against top 10 similar papers and for remaining non-common papers, 3.8 papers are judged as relevant with term-based method and 3 papers are judged as relevant with word-based method.

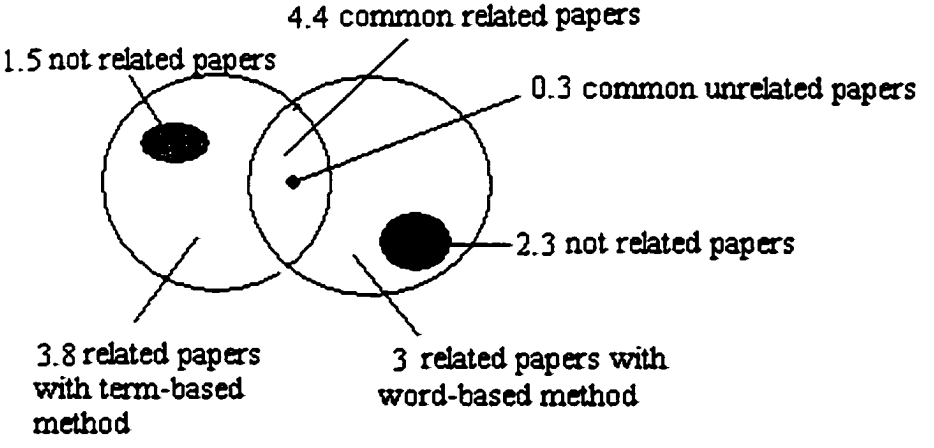


Figure 3.12: Venn diagram for the complementarity of the results from the term-based and word-based methods

3.4.6 Precision comparison between text-based and link-based method

Figure 3.13 is a precision comparison diagram between text-based method and link-based method. Text-based method showed higher precision than link-based method.

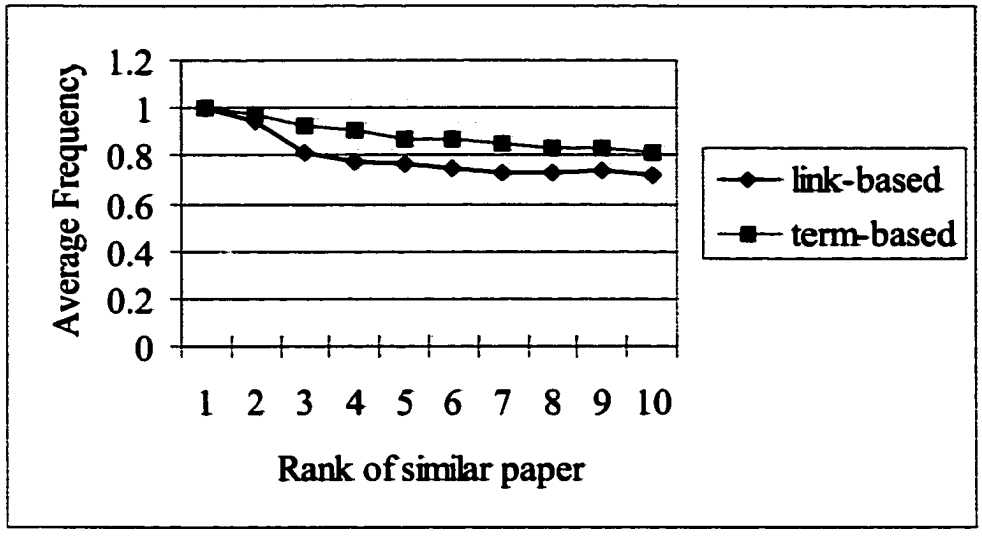


Figure 3.13: Precision comparison between term-based and link-based method

3.4.7 Complementarity of term-based and link-based methods

The link-based and text-based methods complement each other by producing different sets of related papers, see Figure 3.14. Average over 9 query papers, they have 2.4 relevant papers in common against top 10 similar papers and for remaining non-common papers, 5.3 papers are judged as relevant with text-based method and 4.8 papers are judged as relevant with link-based method.

Text-based method can get higher precision but needs time to preprocess the paper content and build an inverted index. They can work together to gain higher precision and attract more similar papers to the top similar papers list.

Figure 3.15 demonstrate the link-based and word-based methods complement each other too by producing different sets of related papers. Average over 9 query papers, they have 2.8 relevant papers in the 2.9 common papers against top 10 similar papers and for remaining non-common papers, 4.6 papers are judged as relevant with word-based method and 4.4 papers are judged as relevant with link-based method.

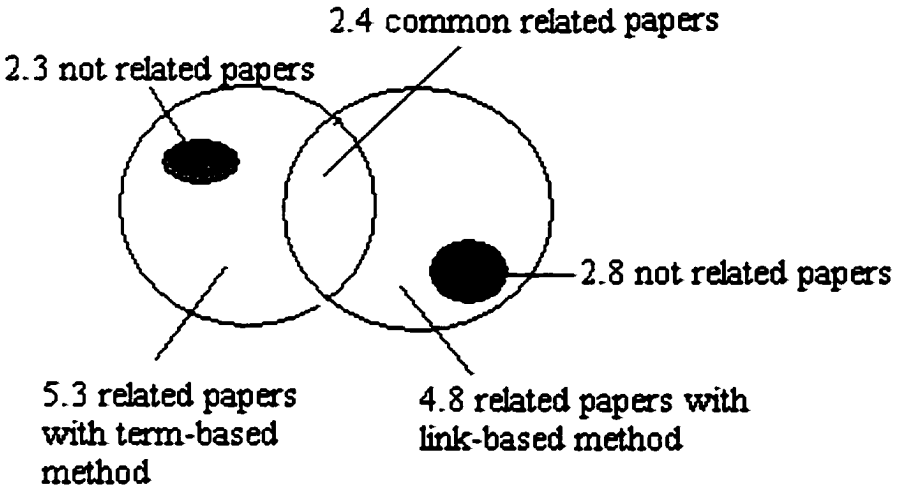


Figure 3.14: Venn diagram for the complementarity of the results from the term-based and link-based methods

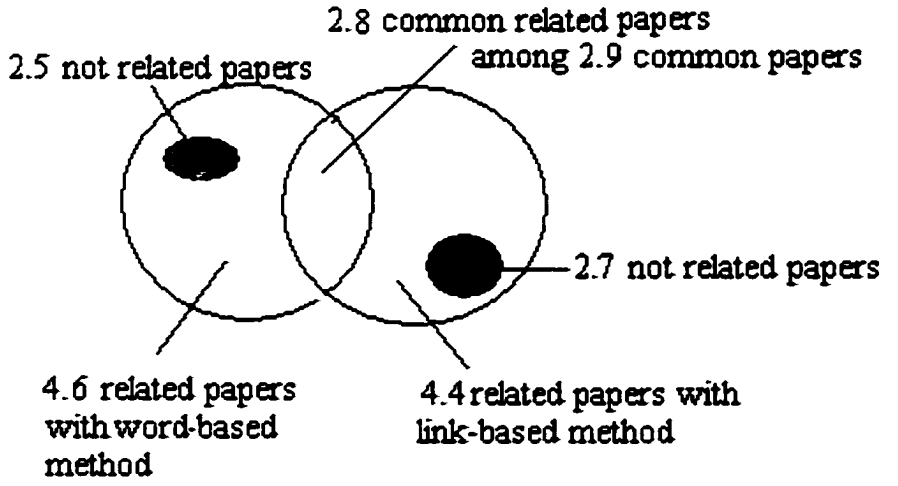


Figure 3.15: Venn diagram for the complementarity of the results from the word-based and link-based methods

3.4.8 Sub-conclusion

For the similarity assessment, terms extracted by frequency of occurrence has almost same precision effect compared with C-value/NC-value.

With text-based method, term-based method can get higher precision than word-based method.

Text-based method is better than link-based method from assessment precision, but the two methods are complementary in that they return sufficiently different sets of similar papers to the query.

Chapter 4

Discussion

In this thesis, we confirmed that the performance of the C-value/NC-value methods in automatic term extraction from a corpus consisting of computer science articles is as good as that on the medical corpus, on which the method was tested by its authors.

Furthermore, we used automatically extracted terms as features in an information retrieval task, in which we are given a database of papers and we are searching for the most similar papers in the database to a given query paper. Terms of intermediate frequency were selected, according to standard practice in information retrieval. We compared the precision obtained using terms, words and a link-based method that is based entirely on the information encoded in the citation graph. Precision of term-based retrieval appears to be slightly higher than link-based retrieval, and comparable to word-based retrieval. The papers returned by the methods have substantial overlap, but there are several papers returned by one method and not the others. So from an information retrieval perspective, the methods are complementary and they should be used together.

Future research involves several short-term directions.

- Develop a statistical model to allow the assessment of the statistical significance of the differences between the precision graphs obtained from the different methods.

- Efficient implementation using the proper data and file structures for handling large special text corpora and number of terms.
- Integration of term-based and link-based method for document retrieval.

A long-term objective is to apply term-based similarity to clustering of special text corpora, aiming to automatically discover hierarchical organizations of scientific disciplines and the associated induced lexical ontologies [2, 5, 13, 14], and knowledge mining through the Web [15].

Acknowledgements I would like to thank my supervisor, Dr. Milios, for giving me instruction to finish this thesis research, Dr. Japkowicz for helping me do the evaluation, Dr. Shepherd for his many critical comments that helped substantially improve the quality of the thesis, and Dr. Zincir-Heywood for reading my paper and being on my defense committee. To compare the text-based method with the link-based method, I got a lot of help from from Wangzhong Lu and Yuan An, who completed their theses on link-based similarity and the characterization of the citation graph, respectively.

Bibliography

[1] Y. An, J. Janssen, and E. Milios. Characterizing and mining the citation graph of the computer science literature. Technical Report CS-2001-02, Faculty of Computer Science, Dalhousie University, 2001.

[2] H. Ayad and M. Kamel. Topic discovery from text using aggregation of different clustering methods. In *AI'2002: The Fifteenth Canadian Conference on Artificial Intelligence*, 27-29 May, 2002 2002.

[3] E. Brill. A simple rule-based part of speech tagger. In *3rd Conference on Applied Natural Language Processing (ANLP-92)*, pages 152–155, 1992.

[4] Eric Brill. *Eric Brill's Home Page.*

[5] C. Clifton, R. Cooley, JM Zytkow, and J. Rauch. TopCat: data mining for topic identification in a text corpus. In *Principles of Data Mining and Knowledge Discovery. Third European Conference, PKDD'99*, pages 174–183, 1999.

[6] K. Frantzi, S. Ananiadou, and H. Mima. Automatic recognition of multi-word terms: the c-value/nc-value method. *International Journal on Digital Libraries*, 3(2), 2000.

[7] J. Justeson and S. Katz. Technical terminology: some linguistic properties and an algorithm for identification in text. *Natural Language Engineering*, 1:9–27, 1995.

[8] Kyo Kageura and Bin Umino. Methods of automatic term recognition -a review-. *Terminology*, 3(2):259–289, 1996.

[9] M. Koubarakis. Boolean queries with proximity operators for information dissemination. In *International Workshop on FOUNDATIONS OF MODELS FOR INFORMATION INTEGRATION (FMII-2001), as the 10th Workshop in the Series Foundations of Models and Languages for Data and Objects (FMLDO), immediately after VLDB-2001*, 16-18 September, 2001, Viterbo (near Rome), Italy.

[10] M. Koubarakis, T. Koutris, P. Raftopoulou, and C. Tryfonopoulos. Efficient dissemination of textual information using the boolean model. In *2nd Hellenic Conference on Artificial Intelligence*, April 11-12, 2002, Thessaloniki, Greece.

[11] Steve Lawrence, C. Lee Giles, and Kurt Bollacker. Digital libraries and Autonomous Citation Indexing. *IEEE Computer*, 32(6):67–71, 1999.

[12] C. Manning and H. Schuetze. *Foundations of Statistical Natural Language Processing*. MIT Press Cambridge, Massachusetts, 1999.

[13] D. Maynard and S. Ananiadou. Creating and using domain-specific ontologies for terminological applications. *Proceedings of Second International Conference on Language Resources and Evaluation, Athens*, 2000.

[14] D. Maynard and S. Ananiadou. Identifying terms by their family and friends. *The 18th International Conference on Computational Linguistics, COLING 2000*, 2000.

[15] H. Mima, S. Ananiadou, and J. Tsujii. A web-based integrated knowledge mining aid system using term-oriented nlp. *Proceedings of Natural Language Processing Pacific Rim Symposium 99, Beijing*, 1999.

[16] J. Sager, D. Dungworth, and P. McDonald. English special languages: principles and practice in science and technology. *Oscar Brandstetter Verlag KG, Wiesbaden*, 1980.

[17] C. J. van RIJSBERGEN. *Information Retrieval.* **http://www.dcs.gla.ac.uk/~iain/keith/index.htm** (accessed April 17, 2002), 1999 (2nd ed.).

[18] W., J. Janssen, E. Milios, and N. Japkowicz. Node similarity in networked information spaces. *Technical Report CS-2001-03 Dalhousie University*, Sep. 2001.

Appendix A

Tagger of simple rule-based POS

- CC conjunction, coordinating (and)
- CD number, cardinal (four)
- BEDR were
- BEDZ was
- BEG being
- BEM am
- BEN been
- CD number, cardinal (four)
- CS conjunction, subordinating (until)
- DO do
- DOD did
- DOG doing
- DON done
- DOZ does

- DT determiner, general (a, the, this, that)
- EX existential there
- FW foreign word (ante, de)
- HV have
- HVD had (past tense)
- HVG having
- HVN had (past participle)
- HVZ
- has
- IN preposition (on, of)
- JJ adjective, general (near)
- JJR adjective, comparative (nearer)
- JJS adjective, superlative (nearest)
- MD modal auxiliary (might, will)
- NN noun, common singular (action)
- NNS noun, common plural (actions)
- NNP noun, proper singular (Thailand, Thatcher)
- OD number, ordinal (fourth)
- PDT determiner, pre- (all, both, half)
- PN pronoun, indefinite (anyone, nothing)
- POS possessive particle (', 's)

- PP pronoun, personal (I, he)
- PP$ pronoun, possessive (my, his)
- PPX pronoun, reflexive (myself, himself)
- RB adverb, general (chronically, deep)
- RBR adverb, comparative (easier, sooner)
- RBS adverb, superlative (easiest, soonest)
- RP adverbial particle (back, up)
- SYM symbol or formula (US$500, R300)
- TO infinitive marker (to)
- UH interjection (aah, oh, yes, no)
- VB verb, base (believe)
- VBP verb, (are)
- VBG verb, -ing (believing)
- VBN verb, past participle (believed)
- VBZ verb, -s (believes)
- WDT det, wh- (what, which, whatever, whichever)
- WP pronoun, wh- (who, that)
- WP$ pronoun, possessive wh- (whose)
- WRB adv, wh- (how, when, where, why)
- XNOT negative marker (not, n't)

Appendix B

Stop list

This is a manually constructed stop list specifically for the computer science corpus. The usual stop words are excluded from consideration by the linguistic filters.

ACCORDING DAY INC. KIND KG KM PERFORMANCE PERFORMED PERHAPS PLENTY REPORT RESULTS SAKE SIDEWAYS STUDIES TYPE FIGURE ILLUSTRATION RESEARCH YEAR MONTH DAY HOUR MINUTE P.O. SYMBOL TABLE FUNCTION STRUCTURE WAY KIND TYPE ACCORDING DAY INC. KG KM PERFORMANCE PERFORMED PERHAPS PLENTY REPORT RESULT SAKE SIDEWAY STUDY FIGURE ILLUSTRATION YEAR MONTH HOUR MINUTE SYMBOL TABLE FUNCTION STRUCTURE GOAL SECTION FORM INPUT OUTPUT DISCUSSION SUM VALUE COVER STABILITY EQUALITY PROPERTY COMPONENT INDEX INSTITUTE DIFFERENCE RESEARCH USE LOG GAMMA KNOWLEDGE WORLD ERROR NUMBER CODE METHOD SHOW CASE BASIS COMMON SIDE MUSIC DELTA PRODUCT RULE FIELD TIME COST LENGTH RESPECT REVIEW UNIVERSITY APPROACH RIGHT ROOM OFFICE FUTURE SCALE CONDITION WORK STEP CAUSE GUIDE NEED EXAMPLE FIRST SECOND THIRD

Appendix C

Paper similarity results

The following tables give out the top 10 similar papers (represented by unique code)to each query paper with term-based (with document frequency between 4 and 250), word-based and link-based method. The evaluation is set to 1 (related), 0.5 (somewhat related) and 0 (not related). The detailed article could be found from the URL listed in tables C.10, C.11, C.12, C.13.

term-based	evaluation	word-based	evaluation	link-based	evaluation
2746	1	2746	1	10406	1
2763	0.5	2763	0.5	30	1
2306	1	2177	1	2746	1
2764	1	2764	1	2744	1
5553	0.5	2749	1	2873	1
2330	1	4909	1	10407	1
3359	0.5	3301	1	10408	1
2749	1	30	1	1050	1
1026	1	1023	1	10409	1
3519	1	2744	1	40	0.5

Table C.1: Top 10 similar papers to paper 2762

term-based	evaluation	word-based	evaluation	link-based	evaluation
403	1	403	1	333	1
308	1	323	1	10411	1
323	1	1130	0.5	10413	0.5
366	1	327	1	466	0
8021	1	2309	1	2526	0
2309	1	8017	0	6384	0
1127	0	8021	1	138	0
4047	0	2331	1	209	1
2517	1	4768	1	384	0
1312	0	308	1	434	0

Table C.2: Top 10 similar papers to paper 10410

term-based	evaluation	word-based	evaluation	link-based	evaluation
10422	1	10422	1	2277	1
2265	1	2265	1	1411	1
1416	1	1411	1	2265	1
2277	1	6021	1	714	0.5
6825	1	1266	0.5	296	0.5
2324	1	2277	1	10422	1
1411	1	9316	0.5	4	0.5
7087	1	1416	1	23	0.5
1618	1	170	0	25	0.5
6021	1	6350	0	4802	0.5

Table C.3: Top 10 similar papers to paper 10419

term-based	evaluation	word-based	evaluation	link-based	evaluation
2148	1	2148	1	2148	1
9	1	442	0	10424	1
4700	1	3146	0.5	1498	0.5
6827	1	10291	0	2174	0
6116	1	7295	0	129	1
3079	1	9	1	9	1
2511	1	2066	1	130	1
1498	0.5	3588	0	1715	1
1847	1	3430	0.5	1868	1
734	0.5	8105	0	6254	0

Table C.4: Top 10 similar papers to paper 10423

term-based	evaluation	word-based	evaluation	link-based	evaluation
5493	1	5493	1	2728	1
2728	1	2728	1	1284	1
312	1	2309	0	2866	0
2177	0.5	2177	0.5	13	1
2411	0.5	1169	1	5493	1
2666	0	4983	0	10426	1
4112	1	2764	0	2318	1
27	0	29	1	2301	0
13	1	2748	0	4344	1
60	0.5	13	1	2306	1

Table C.5: Top 10 similar papers to paper 2292

term-based	evaluation	word-based	evaluation	link-based	evaluation
2530	1	2530	1	2564	1
2583	1	2583	1	2481	0
7307	0.5	2486	1	10416	0.5
2480	1	2483	1	2530	1
2742	0	2177	0	1139	1
1139	1	1139	1	3896	1
6029	0	2048	0	2275	0
2493	1	2485	0	2508	1
2309	0	2500	1	2286	1
397	1	2493	1	2507	1

Table C.6: Top 10 similar papers to paper 10415

term-based	evaluation	word-based	evaluation	link-based	evaluation
4047	1	4047	1	2627	1
2627	1	2627	1	2361	1
412	1	2485	1	2608	0.5
1229	1	2621	1	2621	1
2679	0	2361	1	407	1
2666	1	407	1	10398	1
2361	1	534	0.5	411	1
2048	1	3148	0	408	1
2485	1	5855	1	2537	1
407	1	2666	1	412	0.5

Table C.7: Top 10 similar papers to paper 2641

term-based	evaluation	word-based	evaluation	link-based	evaluation
2169	1	2169	1	2169	1
2307	1	2207	1	2207	1
2217	0	2217	0	2217	0
2324	0	1745	1	2220	1
1745	1	9936	1	2228	0
9936	1	3146	1	10418	0
542	1	3147	0.5	2185	0
8175	1	2392	1	2224	0
7525	0	2200	0.5	2318	1
2309	0	2177	0	2221	0

Table C.8: Top 10 similar papers to paper 2162

term-based	evaluation	word-based	evaluation	link-based	evaluation
462	1	462	1	10404	1
10399	1	10399	1	10405	1
1836	1	1836	1	10402	1
10402	1	10405	1	482	0.5
10405	1	10402	1	10399	1
4122	1	482	1	10401	0
2164	1	8108	1	2164	1
9184	1	1956	0	1836	1
2375	1	9419	0.5	10400	1
7713	1	8107	1	1956	1

Table C.9: Top 10 similar papers to paper 10403

Code	URL
2762	citeseer.nj.nec.com/jh91adaptive
2177	citeseer.nj.nec.com/ring94continual
4909	citeseer.nj.nec.com/schmidhuber91reinforcement
3301	citeseer.nj.nec.com/schmidhuber-artificial
30	citeseer.nj.nec.com/schmidhuber91neural
1023	citeseer.nj.nec.com/wyatt97exploration
2744	citeseer.nj.nec.com/schmidhuber91learning
10410	http://citeseer.nj.nec.com/41706.html
1130	citeseer.nj.nec.com/mani95design
327	citeseer.nj.nec.com/heemskerk95overview
8017	citeseer.nj.nec.com/dehon-robust
2331	citeseer.nj.nec.com/ienne95digital
4768	citeseer.nj.nec.com/ferrucci94acme
10419	citeseer.nj.nec.com/200102
1266	citeseer.nj.nec.com/freitas99nonlinear
9316	citeseer.nj.nec.com/stensmo95adaptive
170	citeseer.nj.nec.com/dietterich97machine
6350	citeseer.nj.nec.com/isard98visual
10423	citeseer.nj.nec.com/46254
442	citeseer.nj.nec.com/costantino96financial
3146	citeseer.nj.nec.com/mcmahon94statistical
10291	citeseer.nj.nec.com/task96applications
7295	citeseer.nj.nec.com/rosenfeld94adaptive
2066	citeseer.nj.nec.com/reutterer-combined
3588	citeseer.nj.nec.com/tam-datacube
3430	citeseer.nj.nec.com/webb95multidimensional
8105	citeseer.nj.nec.com/fu96discovery
2292	citeseer.nj.nec.com/williams129learning
2309	citeseer.nj.nec.com/lehmann94hardware
1169	citeseer.nj.nec.com/williams92some

Table C.10: Paper code with corresponding URL

Code	URL
4983	citeseer.nj.nec.com/schmidhuber90reinforcement
2764	citeseer.nj.nec.com/schmidhuber90learning
29	citeseer.nj.nec.com/williams90adaptive
2748	citeseer.nj.nec.com/schmidhuber-line
10415	citeseer.nj.nec.com/smieja91neural
2483	citeseer.nj.nec.com/mourlfileo98efficient
2177	citeseer.nj.nec.com/ring94continual
2048	citeseer.nj.nec.com/harvey-artificial
2485	citeseer.nj.nec.com/harvey95artificial
2500	citeseer.nj.nec.com/parekh-mupstart
2641	citeseer.nj.nec.com/gruau95automatic
2621	citeseer.nj.nec.com/kodjabachian96evolutionary
2361	citeseer.nj.nec.com/kodjabachian95evolution
3148	citeseer.nj.nec.com/bill96genetic
5855	citeseer.nj.nec.com/kodjabachian98evolution
482	citeseer.nj.nec.com/scott98parcel
1956	citeseer.nj.nec.com/kohavi96data
9419	citeseer.nj.nec.com/zheng96constructing
8107	citeseer.nj.nec.com/liu96feature
2162	citeseer.nj.nec.com/lawurlfilece95applicability
2207	citeseer.nj.nec.com/lawurlfilece00natural
3146	citeseer.nj.nec.com/mcmahon94statistical
3147	citeseer.nj.nec.com/resnik93selection
2392	citeseer.nj.nec.com/christiansen93toward
2200	citeseer.nj.nec.com/plate94distributed
5553	citeseer.nj.nec.com/mataric94interaction
3359	citeseer.nj.nec.com/wright97emotional
1026	citeseer.nj.nec.com/hexmoor95representing
2306	citeseer.nj.nec.com/thrun90adaptive
1925	citeseer.nj.nec.com/bala95hybrid

Table C.11: Paper code with corresponding URL

Code	URL
3519	citeseer.nj.nec.com/kirman94predicting
2749	citeseer.nj.nec.com/schmidhuber91possibility
2324	citeseer.nj.nec.com/burrows96speech
2958	citeseer.nj.nec.com/lawurlfilece912mixture
3079	citeseer.nj.nec.com/kohonen96very
74	citeseer.nj.nec.com/japkowicz99conceptlearning
2511	citeseer.nj.nec.com/merkl97en
482	citeseer.nj.nec.com/scott912parcel
1847	citeseer.nj.nec.com/merkl97cluster
1667	citeseer.nj.nec.com/besch94how
8985	citeseer.nj.nec.com/fang912computing
734	citeseer.nj.nec.com/lampinen95distortion
2411	citeseer.nj.nec.com/siegelmann93foundations
2666	citeseer.nj.nec.com/dellaert95toward
27	citeseer.nj.nec.com/plaut96understanding
7307	citeseer.nj.nec.com/brousse91generativity
2486	citeseer.nj.nec.com/bering-emergence
2309	citeseer.nj.nec.com/lehmann94hardware
397	citeseer.nj.nec.com/kwok95objective
2493	citeseer.nj.nec.com/yang97distal
6249	citeseer.nj.nec.com/kodjabachian-evolution
534	citeseer.nj.nec.com/jakobi912minimal
3389	citeseer.nj.nec.com/kuscu94design
213	citeseer.nj.nec.com/ling93answering
4848	citeseer.nj.nec.com/holst97use
8807	citeseer.nj.nec.com/doyle91two
2309	citeseer.nj.nec.com/lehmann94hardware
6270	citeseer.nj.nec.com/jacobs912class
230	citeseer.nj.nec.com/kohavi94useful
2535	citeseer.nj.nec.com/hallinan99simultaneous

Table C.12: Paper code with corresponding URL

Code	URL
1926	citeseer.nj.nec.com/bala96using
2564	citeseer.nj.nec.com/fahlman90cascade-correlation
10424	citeseer.nj.nec.com/michael-neural
10413	citeseer.nj.nec.com/abdi-neural
1498	citeseer.nj.nec.com/graepel-statistical
466	citeseer.nj.nec.com/medler98brief
714	citeseer.nj.nec.com/jordan94hierarchical
2174	citeseer.nj.nec.com/ripley96pattern
2526	citeseer.nj.nec.com/camargo-learning
296	citeseer.nj.nec.com/mackay92practical
129	citeseer.nj.nec.com/kaski96comparing
6384	citeseer.nj.nec.com/terry92acquisition
3896	citeseer.nj.nec.com/marco97optimal
138	citeseer.nj.nec.com/tumer-linear
4	citeseer.nj.nec.com/mackay92bayesian
130	citeseer.nj.nec.com/putten-utilizing
209	citeseer.nj.nec.com/burr91digital
23	citeseer.nj.nec.com/david97gated
1715	citeseer.nj.nec.com/honkela97self-organizing
2508	citeseer.nj.nec.com/alpaydin91gal
384	citeseer.nj.nec.com/orponen94computational
25	citeseer.nj.nec.com/david97stock
1868	citeseer.nj.nec.com/flexer97limitations
2286	citeseer.nj.nec.com/kwok97constructive
434	citeseer.nj.nec.com/warner96understanding
4802	citeseer.nj.nec.com/ueda99smem
6254	citeseer.nj.nec.com/flexer-data
2507	citeseer.nj.nec.com/fiesler94comparative

Table C.13: Paper code with corresponding URL