High Precision Information Retrieval with Multi-Term Relation

Lixin Zhou
School of Software and Microelectronics,
Peking University
Beijing, China
zhoulx@vip.sina.com

Ruiguang Song
School of Software and Microelectronics,
Peking University
Beijing, China
ruiguangsong@gmail.com

ABSTRACT: *Enterprise search differs from Internet search in many ways. The notion of a "good" answer to a query is quite different. On an intranet, the notion of a "good" answer is often defined as the "right" answer. Finding the right answer is often more difficult than finding the best answer. On the other hand, nowadays when searching on Internet, a large amount of information returned for a query, the most important is how to improve the precision of a query. In order to improve the search relevance and precision, a search approach integrated multi-term relations into search process is presented in this paper. In this approach, two level mixed indices are established including term index and term relations index. The Maximum Match algorithm in Chinese segmentation is used to find multi-term relations from text data. With this approach, we can obtain the "right" answer quickly and efficiently.*

Keywords: high precision, information retrieval, term relation ,multi-term, enterprise

Received: 19 March 2009, Revised 29 April 2009, Accepted 20 May 2009

1. Introduction

The Web revolution has exposed hundreds of millions of people to the experiences of searching and taxonomy browsing and has reshaped their expectations of the knowledge retrieval process, not only while browsing the Web, but more importantly, while at work, performing their jobs. Unfortunately, study after study shows that at the enterprise level, these expectations are not being met [1]. Knowledge management in the enterprise setting and even simple document search functions are often perceived as disappointing[1].

The overwhelming majority of information in an enterprise is unstructured—that is, it is not resident in relational databases that tabulate the data and transactions occurring throughout the enterprise. This unstructured information exists in the form of HTML pages, documents in proprietary formats, and forms (e.g., paper and media objects). Together with information in relational and proprietary databases, these documents constitute the enterprise information ecosystem[1].

Enterprise search differs from Internet search in many ways [2, 3, 4]. First, the notion of a "good" answer to a query is quite different. On the Internet, it is vaguely defined. Because a large number of documents are typically relevant to a query, a user is often looking for the "best" or most relevant document. On an intranet, the notion of a "good" answer is often defined as the "right" answer. Users might know or have previously seen the specific document(s) that they are looking for. A large fraction of queries tend to have a small set of correct answers (often unique, as in "I forgot my Unix password"), and the answers may not have special characteristics. The correct answer is not necessarily the most "popular" document, which largely determines the "best" answer on the Internet. Finding the right answer is often more difficult than finding the best answer.

On the other hand, nowadays when searching on Internet, a large amount of information return for a query, the most important is how to improve the precision of a query.

The existing search engine are almost adopting word-based full text index such as Google. In order to improve the search relevance and precision, we present a multi-term relation based search approach in this paper. In this approach, we design a two level mixed multi-term relation based semantic full text indices including term index and multi-term relation index. The Maximum Matching method [5] is used to analyze multi-term relation in text data.

In this paper, related work is described in section 2, in section 3, we describe process of information retrieval with Multi-Term Relation, in section 4, we describe the architecture of an enterprise information retrieval system; in section 5, we presented the experiments design; in section 6, the conclusion and future work have been presented.

2. Related Work

A majority of traditional models of information retrieval (IR) mainly make use of surface linguistic information such as words/ terms. A common limitation of many retrieval models, including the statistical language model approach and the traditional TF-IDF (Term Frequency - Inverse Document Frequency) based VSM(Vector Space Model) approach, is that retrieval scores are solely based on exact matching of terms in the queries and documents, without allowing distinct but semantically related terms to match each other and contribute to the retrieval score [6] .

It is reasonable to expect better retrieval results if we can exploit deep linguistic information further. Previous studies of this sort have been carried out at both syntactic and semantic levels. Most of them focused on the former, because the recognition of syntactic structures is easier than that of semantic structures. Syntactic information possibly exploited in IR can be a simple syntactic relation between a pair of words, and can also be a complex structure tree.

It is natural to assume that semantic information is more useful in IR since it can capture the meaning of a sentence more precisely than syntax. Semantic information, both intra-sentential and inter-sentential, is usually represented by the so-called semantic relations between various entities involved.

An approach which merges case relations into the Vector Space Model (VSM) named Case relation-based VSM (C-VSM) was presented in [7], whose experimental results on the test set showed that C-VSM outperforms Word-based VSM by 3.4% on the average 11-point precision. But in this approach those 23 case relations were constructed by manual annotation.

The syntactic parsing of natural language can not achieve enough performance to be used in practice. Maximum Matching Method can achieve over 95% in recall and over 90% in precision in the process of Chinese word segmentation [8, 9]. Eric Brill's POS (Part of Speech) Tagging algorithm can achieve over 96% in recall and precision of POS (Part of Speech) Tagging [10] .

Most of the existing semantic search approaches didn't build full text index [11] . In this paper we have established two level mixed indices including term index, multi-term relation index.

3. Process of Information Retrieval with Multi-Term Relation

The process of information retrieval with multi-term relation is illustrated in Fig.1. In Figure 1, the text stands for pure text data, which can be extracted from unstructured documents such as PDF file, HTML file and WORD file etc. The most important part is the components of building multi-term relation index. A query is also analyzed to be parsed into terms and multi-term relations (multi-term phrases). Then we can do information retrieval by multi-term relation index and term index. Finally the query results will be presented.

A. Multi-Term Relation Lexicon

In this paper, we try to extract term relations between two terms or among 3 or more terms. The term relations include named entities such as "hard drive", "computer power", dependant relations such as "format hard drive", "make bootable floppy". We crawled 75 FAQ web pages from internet and obtained 1279 multi-term relation phrases which are stored in a Multi-Term Relation Lexicon. A multi-term relation is defined as follow,

Multi-term relation ::= [(word, POS)]2.. +length* ……..*(1)*

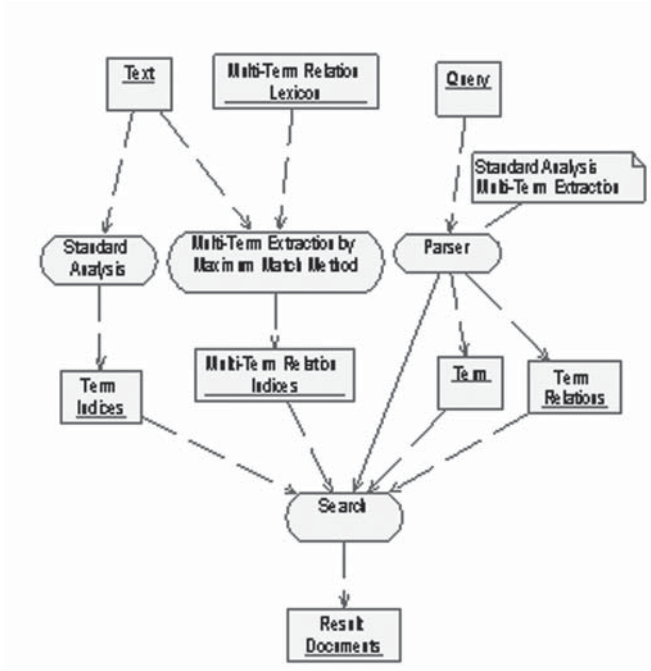

Figure1. The Process of Information Retrieval with Multi-Term Relation

In formula (1), (word, POS) stands for a word in a multi-term relation and the Part Of Speech of the word, "2..*" means there are two or more pairs of (word, POS) in a multi-term relation. The "length" stands for the maximal distance of characters of the multi-term relation which arises in the text. That means matching of a multi-term relation isn't required to regard the multi-term relation as an integrated term. The terms in a multi-term relation could appear in the text discontinuously, but should keep the relative sequence.

For example, from a sentence "Please tell me how to make a bootable floppy with the correct files to Fdisk and format this hard drive?", we can extract two multi-term relation patterns as follows,

"Make bootable floppy" ::= (make, V) +(bootable, A) +(floppy, N) + 20

"Format hard drive" ::= (Format, V) + (hard, A) + (drive, N) + 17

In above, V stands for verb, A stands for adjective, N stands for noun. Notice that the length of "Make bootable floppy" is 20, not including the word "a" and the length of "Format hard drive" is 17, not containing the word "this" in the sentence. So the multi-term relation we extracted is discrete in the text, and the sequence is important. We will discuss how to match the multi-term relation in the text in next part.

B. Maximum Matching Algorithm

Maximum Matching algorithm is a powerful and most popular algorithm used in Chinese word segmentation. We use maximum matching method to analyze and extract multi-term relation phrases from text data.

The algorithm used in building multi-term relation index is described as follows,

BEGIN
READ the Multi-Term Relation Lexicon;
READ a sentence from a text file;
Tagging the text file obtaining the POS tag;
READ a word from the sentence;
BOOLEAN FIND = false;
BOOLEAN FINDLOOP = false;

```
    IF ( the word can be found in a multi-term relation in the lexicon )
        GET the multi-term relation in the lexicon;
        IF (the multi-term relation can be found in the sentence)
            THEN  { FIND = true; FINDLOOP = true ; }
           ELSE FIND = false;
       ENDIF
        WHILE (FINDLOOP = true) {GET the next longer multi-term relation
                in the lexicon;
          IF (the multi-term relation can be found in the sentence)
            THEN FIND = true;
              ELSE FINDLOOP = false;
          ENDIF
              }
       ENDIF
    IF FIND = true
      WRITE the found multi-term relation INTO Multi-Term Relation Index.
      ENDIF
END
```

In this algorithm, we need to find the multi-term relation in the sentence. As above mentioned, the multi-term relation could be discrete but sequential in the text, so it will be reasonable that we find out the multi-term relation from a certain range of words in the sentence discretely and by sequence. In the certain range we accept the term sequence of the multi-term relation as the found multi-term relation in the sentence, but out of the range we take the multi-term relation as not found in the sentence. The range can be set as needed.

C. Search Algorithm

When we get a query, the process of parsing a query and searching is as follows,

```
BEGIN
  Parsing the query by standard analysis algorithm;
 Tagging the query obtaining the POS tag;
 Parsing the query by maximum matching method
with multi-term relation lexicon;
  IF ( find multi-term relations in the query ) THEN {
    Search multi-term relation index;
    PUT the found documents INTO SET A;
    FOR ( each document[i] in SET A )  DO
      Multi-Term-Score[i] =
Σ (Frequent of a multi-term relation)*
(normalized length of the document);
}
  ENDIF
  Search term inverse index;
  PUT the found documents INTO SET B;
  FOR ( each document[j] in SET B) DO
    Calculate Term-Score[j];
  FOR ( each document [k] in both SET A and SET B ) DO
    Score[k] = α* Multi-Term-Score[k]
+ (1-α) * Term-Score[k];
  RETURN each document in SET A or SET B.
END
```

At first, a query is analyzed through POS tagging, tokenizing and stemming as needed. Then a query is parsed by maximum matching method with multi-term relation lexicon.

As for the documents found both in multi-term relation index and in term inverse index, we can obtain two scores, one is from multi-term relation phrases named Multi-Term-Score, another is from term vector named Term-Score, so the score is defined as,

Score = α* Multi-Term-Score + (1-α) * Term-Score,

$\alpha \in [0,1]$ …… (2)

The parameter α can be set as needed.

Then we could rank all returned documents by Score to form the search result. Above we don't give any specific calculating method because you could take your own method, but we recommend you to calculate multi-term-score and term-score using one unified method, for instance, all use a normalization method or not, on account of expecting the same order of magnitude which both two scores have. As a result, the definition above of two scores will make sense.

The process of search begins with parsing the query. If the result of the parsing by maximum matching method doesn't return a multi-term relation, the Multi-Term-Score will be 0, and the Score will be only (1-α) * Term-Score left. So if the parameter α is too big, the Score will be compressed to a much smaller range of the old value range when there is no multi-term relation returned from the query. This condition should be noticed.

4. The Architecture of System

According the information processing procedure and the algorithm described in the former sections, we designed and developed an enterprise information retrieval system. The system architecture is shown in Fig. 2. The system server has two parts: (1) index server, (2) search server.

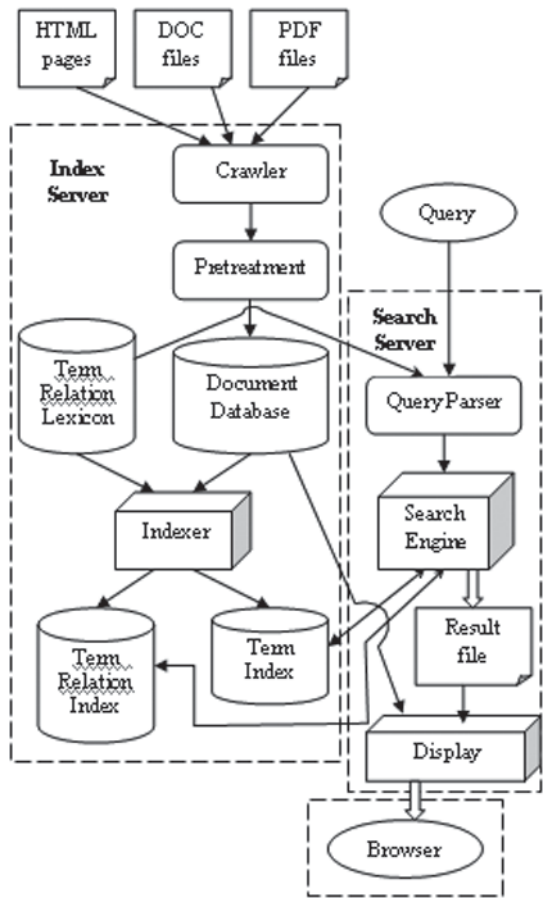

Figure 2. The Structure of System

A. Index Server

The index server has three computing modules: (1) Crawler, (2) Pretreatment, (3) Indexer; and four databases: (4) document database, (5) multi-term relation lexicon, (6) multi-term relation index, and (7) term index. We will describe these parts in detail in the following.

(1) Crawler

Firstly, the raw information from the enterprise website and other enterprise information database are crawled by the web crawler component and gathered into temp files.

(2) Pretreatment

Secondly, the pre-treatment component process the raw information from the temp files, which are saved into document database after pre-converted.

(3) Indexer

Index Server is designed to gather documents residing on the document database and can satisfy the search solution needs of the enterprise information retrieval. The tasks of the Indexer are building two kinds of index: one is general term index, another is the multi-term relation index. The Indexer is the most important part of the system. The algorithm of building multi-term relation index is described in Section 3.

Indexing happens in the background only when there are sufficient system resources available to index without adversely affecting system performance. We design some parameters controlling indexing performance which can be set by the administrator to tune performance to the site's needs.

As for incremental indexing, since files are constantly changing in a file system, so Index Server has been designed to seamlessly assimilate constantly changing documents. Index Server strives to make the index available at all times.

(4) Document Database

The document database collectively refers to all the documents you want to include in your index.
Index Server gathers documents that are individual files in the file systems in an enterprise, and crawls the Internet or the intranet to gather documents of interest.

(5) multi-term relation lexicon

It is described in Section 3.

(6) multi-term relation index

The multi-term relation inverted index is an index data structure storing a mapping from content, multi-term relation, to its locations in a database file, in this case allowing full text search with multi-term relations. It is the most popular data structure used in document retrieval systems.

The multi-term relation inverted index contains the positions of each multi-term relation found within a document. The form of multi-term relation index offers more functionality (like phrase searches), but needs more time and space to be created.

Multi-term index appears in a format like so:

A multi-term relation : (first character position of the multi-term relation appearing in a document, last character position of the multi-term relation appearing in a document, document ID)

(7) term index

The term inverted index is an index data structure storing a mapping from content, term or word, to its locations in a database file, in this case allowing full text search.

The term inverted index contains the positions of each term within a document.

Term index appears in a format like so:
A term :
(first character position of the term appearing in a document, document ID)

Documents are much more than a stream of characters. They contain concepts threaded into a stream of thought using syntactic and semantic constructs of varying complexity provided by the language. Text processors implemented in software are generally not capable of "understanding" natural language documents as thoroughly as we humans can.

The term relation level of abstraction is to be able to extract noun phrases (for example, “Software Architecture”). The concept explicitly expressed by “Software Architecture” is very distinct from the concepts expressed by “Software” and “Architecture”. This phrase-recognition capability enables our users to lock on to documents containing “Software Architecture” while ignoring documents that merely contain the component words.

B. Search Server

The search server has three computing modules: (1) query parser, (2) search engine, (3) display; and one data file: (4) result file. We will describe these parts in detail in the following.

(1) Query parser

The Query parser parses a query into terms and multi-term relations (multi-term phrases). In the process of parsing, the multi-term relation lexicon should be used.

(2) Search engine

The search engine search information by multi-term relation index and term index together. The search algorithm is described in Section 3.

(3) Display

The display module takes charge of deciding how to present the results and present them.

(4) Result file

The document list searched from the document database is stored into the result file.

5. Experiments Design

We crawled 75 FAQ web pages from internet and obtained 1279 multi-term relation phrases which are stored in a Multi-Term Relation Lexicon. There are some examples in Table 1.

Multi-term relation	Description	Length
Make bootable floppy	(make, V) (bootable, A) (floppy, N)	20
Format hard drive	(format, V)(hard, V)(drive, N)	17
Computer power	(computer, N)(power, N)	14
System boot	(system, N)(boot, V)	11
Computer network	(computer, N)(network, N)	16
Custom folder	(Custom, N)(folder, N)	13
Password protection	(Password, N)(protection, N)	19
Display setting	(Display, N) (setting, N)	15
System restore CD	(System, N) (restore, N) (CD,N)	17

Table 1. Examples from Lexicon

We tested 10 multi-term queries by the approach presented in this paper and by Google in order to compare. In the process of searching, the parameter α is set as 0.3. As for each multi-term query, the following processes have been done :

At first we search with the query by Google, the relevance of the top ten return pages were evaluated;

Then we download the top fifty return web pages from the results by Google. After the fifty pages were processed by the approach presented in this paper, the top ten pages to the query were obtained, and the relevance of return pages were evaluated.

The result were showed in Table 2.

Multi-term query	**precision@10 by Google**	**Precision@10 by our approach**
Display setting	0.8	1.0
Make bootable floppy	0.9	1.0
System boot	0.9	1.0
System restore CD	0.3	1.0
Password protection	1.0	1.0
Format hard drive	1.0	1.0
Computer power	0.8	1.0
Custom folder	0.8	1.0
Computer secure	0.8	1.0
modem broken	0.7	1.0
Average	0.80	1.0

Table 2. Results

More than 10% of precision at the first ten return pages can be obtained by the approach presented in this paper.

In Table 2, as to query “password protection”, the value of precision@10 by Google is the same as the value of precision@10 by our approach, which is 1.0. but as to the query “System restore CD”, the value of precision@10 by our approach is 1.0, the value of precision@10 by Google is 0.3. This means when searching by Google, the first return HTML page is not the good answer for query “System restore CD”. So, with our approach presented in this paper, the relevance and precision of a query have been improved.

For another experiment we crawled 1,384 FAQ and help web pages from some commercial websites. We could obtain more than 3000 multi-term relation phrases and we manually extracted more than 30 multi-term relation phrases to verify another experiment. Examples of phrases are given in Table 3.

By the algorithm presented in Section III.B, our system builds a multi-term relation index containing 1,618 records.

We also test two groups of natural language queries. Group 1 consists of queries which only contain one multi-term relation and queries of Group 2 contain more multi-term relation phrases which are involved with each other. We executed two groups search by the approach presented in this paper with the parameter α set as 0.3 and by the usual approach without the multi-term relation approach, in order to compare two results.

Multi-term relation	**Description**	**Length**
add a collation	(add, VB) (a, DT) (collation, NN)	15
grant tables	(grant, NN) (tables, NNS)	12
map array elements	(map, NN) (array, NN) (elements, NNS)	18
Access denied	(Access, NN) (denied, VBD) (error, NN)	13
system variable	(system, NN) (variable, JJ)	15
Can’t connect	(Ca, MD) (n’t, RB) (connect, VB)	13
Create a table	(Create, VB) (a, DT) (table, NN)	14
character set	(Unicode, NNP) (character, NN) (sets, NNS)	13
host value	(host, NN) (value, NN)	10

Table 3. Examples of Phrases

In Table 4 we give the result of Group 1 containing one multi-term relation. The result of Group 2 containing two multi-term relation phrases is presented in Table 5.

Compared Table 4 and Table 5, we find that the precision of Group 1 drops more obviously than Group 2. So we presume that when the query contains more multi-term relation phrases, the precision of our approach may drop slowly and stably. But more experiments are required to work out the conclusion and the relevant condition.

1 Multi-term query	**Precision@10 by usual approach**	**Precision@10 by our approach**
add a collation	0.7	0.9
grant tables	0.7	0.8
map array elements	0.4	0.5
Access denied	0.9	1.0
system variable	1.0	1.0
Can’t connect	0.8	1.0
Create a table	0.9	1.0
character set	0.8	1.0
host value	0.3	0.5
increase the size	0.7	0.8
Average	0.72	0.85

Table 4. Result of Group 1

2+ Multi-term query	**Precision@10 by usual approach**	**Precision@10 by our approach**
add a collation, character set	0.6	0.8
grant tables, Access denied	0.5	0.7
character set , map array elements	0.4	0.7
Can’t connect, Access denied	0.6	0.9
Create a table, increase the size	0.5	0.8
host value, IP numbers	0.7	0.9
Average	0.55	0.80

Table 5. Relation phrases

6. Conclusions and Future Work

An search approach integrated with multi-term relations has been presented in this paper. In this approach, we design a two level mixed indices including term inverse index and multi-term relations index. The result proved the search precision are improved significantly. And when more multi-term relations in the query we presume the search precision may drop more slowly than the usual search approach.

The performance of the approach in this paper depends on the quality of the multi-term relation lexicon heavily. But in Enterprise search, the precision and relevance are very important. The approach presented in this paper can be used in Enterprise search to improve the precision and relevance of a query.

In this paper, the multi-term relations in the lexicon are obtained manually. In the future, we will try to find methods to form the multi-term relations in the lexicon and figure out whether the decline of precision is more slowly or some relevant conditions it happens.

References

[1] Rajat Mukherjee and Jianchang Mao. Enterprise Search: Tough Stuff. ACM Queue vol. 2, no. 2 - April 2004.

[2] Fagin, R., Kumar, R., McCurley, K., Novak, J., Sivakumar, D., Tomlin, J. A., and Williamson, D. P. Searching the workspace Web. Proceedings of the 12th International World Wide Web Conference, Budapest, Hungary (May 2003) 366-375.

[3] Raghavan, P. Structured and unstructured search in enterprises. Data Engineering 24, 4 (Dec. 2001) 15-18.

[4] Hawking, D. Challenges in enterprise search. Proceedings of the Australasian Database Conference, Dunedin, New Zealand (Jan. 2004) 15-24.

[5] Yuan Liu, Qiang Tan, and Kun Xu Shen. The Word Segmentation Rules and Automatic Word Segmentation Methods for Chinese Information Processing (in Chinese). Qing Hua University Press and Guang Xi Science and Technology Press, 1994, page 36.

[6] Hui Fang and ChengXiang Zhai. "Semantic Term Matching in Axiomatic Approaches to Information Retrieval". In Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pages 115-122, Seattle, Washington, USA, 2006.

[7] Wang Hongtao, Sun Maosong, Liu Shaoming. Merging Case Relations into VSM to Improve Information Retrieval Precision. Proceedings of Computational Linguistics and Intelligent Text Processing (CICLing 2005), 6th International Conference, Mexico City, Mexico, February 13-19, 2005, 584-592.

[8] Ka-Po Chow. Andy C. Chin. Wing Fu Tsoi. Maximal Match Chinese Segmentation Augmented by Resources. The 2nd International Joint Conference on Natural Language Processing (IJCNLP-05), October 11-13, 2005.

[9] Jianfeng Gao, Mu Li. Chinese Word Segmentation : A Pragmatic Approach. Microsoft Research, MSR-TR-2004-123, Nov. 2004.

[10] Eric Brill. Transformation-Based Error-Driven Learning and Natural Language Processing: A Case Study in Part of Speech Tagging. Computational Linguistics, Dec. ‘95.

[11] C. Tang, S. Dwarkadas and Z. Xu. On Scaling Latent Semantic Indexing for Large Peer-to-Peer Syatems. SIGIR04.