Master Thesis
Computer Science
Thesis no: MCS-2009:28
Jan 2009

Query Expansion Research and Application in Search Engine Based on Concepts Lattice

Jun Cui

School of Computing
Blekinge Institute of Technology
Soft Center
SE-37225 RONNEBY
SWEDEN

This thesis is submitted to the Department of Interaction and System Design, School of Engineering at Blekinge Institute of Technology in partial fulfillment of the requirements for the degree of Master of Science in Computer Science. The thesis is equivalent to 20 weeks of full time studies.

Contact Information:

Author(s):

Jun Cui 811120-5236

Address: Minervävagan 22A rum 5, Karlskrona, 37141 Sweden

E-mail: Jun.cui.xjtu@gmail.com

University advisor(s):

Guohua Bai

School of Computing

Yang Liu

School of Computing

School of Computing
Blekinge Institute of Technology
Soft Center
SE-37225 RONNEBY
SWEDEN

Internet : www.bth.se/tek
Phone : +46 457 38 50 00
Fax : + 46 457 102 45

Abstract

Formal concept analysis is increasingly applied to query expansion and data mining problems. In this paper I analyze and compare the current concept lattice construction algorithm, and choose iPred and Border algorithms to adapt for query expansion. After I adapt two concept lattice construction algorithms, I apply these four algorithms on one query expansion prototype system. The calculation time for four algorithms are recorded and analyzed.

The result of adapted algorithms is good. Moreover I find the efficiency of concept lattice construction is not consistent with complex analysis result. In stead, it is high depend on the structure of data set, which is data source of concept lattice.

Table of contents

List of figure

List of table

Introduction

With the development of the IT industry, especially the Internet technology, all kinds of information can be obtained through the Internet. However, the information on Internet is disorder and mass, and people can not find the useful and required information directly. Difficulty of obtaining and location required information has become restrict of Internet application. Search engine, as an easy and friendly tool, solved this problem.

A search engine is a tool designed to search for information on the World Wild Web. Search engine interacts with use through one kind of interface, gets the user's searching requirement, analyzes user's searching requirement, matches the searching requirement in its database, and finds the optional information. Finally search engine will return a result set to the user. The result set is ranked by the relevance. The best match web page will be placed on the top of the result set. Nowadays, the search engine divides the user's requirement into several key words, and uses these key words to match the documents and web pages. With the improvement of search engine technology, search engine has become the essential tool for Internet information retrieval.

Although search engine technology has achieved great success, there are still some issues should be solved. For example: the low accuracy, the result set contents many irrelevant documents; massive result set, it is very difficult for user to find concerned information; unintuitive result set, user must read original document, otherwise user can not know the content.

About these disadvantages in search engine, several methods have been proposed, and the query expansion is the one of important methods. The main reason of low accuracy is the low accuracy of initiation query information. This can be showed as following:

1. The users might not know which words can express their query intention. The searching key word does not match user's query intention.

2. The query intention can not express professionally. Normally, the users just use one or two searching key words. This can not indicate real query intention, and query intention should be further expressed by more professional worlds.

3. The users do not know how to use the symbolic logic to indicate their query intention. This is a very normal phenomenon for the general users. When do the multi-keywords queries, most of the users just use the keywords without any symbolic logic.

Considering the problem upon, query expansion uses the result set returned by search engine to help normal users to accurately state their real query intention.

This thesis attempts to employ **Formal Concept Analysis** and **Concept lattice** to implement the query expansion. Formal Concept Analysis (FCA) is proposed by German professor Rudolf Wille at 1982 [1]. It is a method which can be used for data analysis, knowledge express and data management, and it has many successful applications in these days. Formal concept analysis reflects the philosophy means of the concept, and is used for the concept discovery.

Concept lattice model is the kernel data structure in Formal Concept Analysis. It is the concept hierarchy structure construed by the binary relation. Concept lattice represents the connection between objects and attributions. It also represents generalization and instantiation relation among concepts. Employing the concept hierarchy structure the dependence and causality relation is easily constructed. Nowadays, application fields of formal concept analysis are extended to knowledge exploration, software engineering, information retrieval, semantics and economic. Using the hierarchy structure model which is constructed by concept lattice is conveniently for the query expansion research and implement.

In chapter 1, I will introduce query expansion methods, and give a brief review of how query expansion and data mining use formal concept analysis. In chapter 2, the main aim of this thesis and the research questions are given. I describe how I answer these research questions and achieve the main aim. I present the research methodology I used in thesis in chapter 3, and illustrate how I select concept lattice construction algorithm. In chapter 4, I give the background knowledge of formal concept analysis and explained how I adapted the selected algorithm. In chapter 5, I describe the design of the query expansion system. Chapter 6 depicts the experiment, and I analysis the result of the experiment.

Chapter 1: Background

1.1 Introduction of query expansion technology

When the man-computer interactive interface was implemented, the query expansion technology starts to be used in information retrieval system. It has relation with command language, menu selection, diagram operation, natural language communication and all other information retrieval methods. [2] [3]. In web search engine application, query expansion implements in many ways together. It includes the setting of the heuristic expansion interface, various interactive methods and some dynamic optimize method. Usually query expansions employ the database as the data source, and use several methods to do the data mining from multi-aspect. There are three kinds of query expansion technology. They are: static query expansion, query expansion based on data set and dynamic cluster query expansion [4] [5] [6].

1.1.1 Static query expansion

Static query expansion is using the setting in the query system interface to support query. Most search engines offer this kind of method. Besides basic query interface, the search engine always has the advanced query interface to offer the complex query. User can set which keyword should be included and which keyword should be excluded in the search result. Moreover, user can choose which language and which type of the document will be display in the result set. In this way user can express their query intention in detail. Take the Google as an example. User can use the Boolean expression to describe their search keyword. Also in the advance search page user can set the restriction of the language, zone, time, document type etc… the advantage of static query expansion is that it is easy for user to use, and the interface is simple. It help user to write a complex query keywords. User just needs to type the keyword and set restriction following the instruction. There are disadvantages of this method. Because all the setting is defined by the system developer, it can not satisfy all the users' special requirements. Especially it works not very well for the potential content express.

1.1.2 Query expansion based on data set

This kind method will use a predefined data set to help user doing the query expansion. The process of this method always uses the interactive way to communicate with the user. Common forms of this method include:

1. Automatic spelling correction for the query.
   This is the most common method to do the query expansion. Query system use the predefined spelling correction database to notice the user the normal spelling problem which occurs in the user input keyword. Meanwhile, the query system gives the user the suggestion about how to correct the wrong keywords. A lot of

query systems use the similar method—natural language query expansion technology, which is just the improvement of automatic spelling correction method. Natural language query expansion actually just employs the excluded word database to exclude useless words in the input keyword, and use the rest words to do the query.

2. Query expansion with the global user habit data set.
   This kind of method employs a global user habit data set which will record the query history of the user. After analysis the user habit data, system can find some rule to help the query expansion. This kind of method takes accounts the current user input and the all (global) user habit data to generate the expansion word. This method is easy to be applied and has a very high efficiency. It is the basic method for the traditional text information retrieval system. Some search engine, such as Alta Vista, Exite, Baidu and Google using this kind of method. Take Google as an example. When we search "classification", Google will give us the suggestion like "classification of animals", "classification of living things", etc.

   The advantage of this method is that it is easy to be applied and have a high practicability. However, this method just shows the query request which is popular on the Internet recently. It has the bias and can not express the current user's query intention. Moreover this method focuses on the text match, but can not express the query on concept level.

3. query expansion based on hierarchical thesaurus set
   This method uses the hierarchical thesaurus set to do the query expansion. Normally the thesaurus set is built according to the requirement and the system resource. It is an assistant system which is create based on the global requirement of the query system. The form of this method like the classical described wordlist and the hierarchical wordlist, but it is not as strict as the control wordlist. When do the query, system first match the user's query input in the hierarchical thesaurus set, and then with the help of the thesauruses to do the query expansion. The advantage of this method is that it can provide the concept level query expansion. The system create the hierarchical structure can offer the specific word or relative word based on the hierarchical structure. In this way the system get the better query result. The disadvantage of this method is that the hierarchical thesaurus set still can not generate automatically. That limits the size of the hierarchical thesaurus set, and it is very hard to apply this method into a general search engine. Furthermore, the hierarchical thesaurus set is created before using it. It can not satisfy the user in the special situation. This method is suitable for the size limited resource, especially for the professional text query system. General query system is very hard to get the benefit from this method.

1.1.3 Dynamic cluster query expansion

Dynamic cluster query expansion cluster the query result dynamically, and than do the query expansion based on the cluster result. Normally it works on the result of the user's query. After analysis and cluster the query result, system provides the expansion word to the user, which is relative with the original query keyword. The

common forms include: using the cluster result as the recourse to do the specificity query. The typical examples include Vivisimo, Allthe Web. Another using the cluster result to do both the expand query and specificity query. The typical examples include Teoma, GuideBeam. The advantage of this method is that the cluster result is created in real time. Therefore it can dynamically demonstrate the information in the resource, and can provide the query expansion for any field, object or level. It does not restricted by the predefined hierarchical thesaurus set. The disadvantage of this method is that this method use real time processing. The information and the situation are different from case to case, so it is very hard to implement query system which can handle all situations. Meanwhile the cluster result might be not useful in some situations. This will affect the efficiency of the query system. To deal the above problem, in practical application dynamic cluster query expansion always cooperate with other method in one query system. The dynamic cluster method as a great challenge in query expansion has been concerned by a lot of researchers.

In addition, some search engine provides the “Similar to” option in the query result. This method uses the query result as the resource to do query expansion. Also, the link of each query result will be considered in this method. The implementation of this kind of system is quite simple. System just needs to expand the query when the item is selected.

1.2 Current research of query expansion based on formal concept analysis

Jie Wei, Stephane Bressan and Beng Chin Ooi propose a technique for mining term association rules in automatic global query expansion in [7]. Through expanding the original query terms using the relevant terms from the thesaurus or from the mined rules, the precision and recall can be improved. This method is association rules mining. Yufeng Hai, Yajun Du and Haiming Li use formal concept analysis on it in [8]. Without scanning every node of lattice, search engine can provide additional relevant web pages and reduce useless ones to users. W.C.Cho and D.Richards also mentioned a formal concept analysis method in [9]. This method reduces query ambiguity effectively and return the accurate information that users need. They have improved precision and recall for information retrieval. Bing Zhang, YaJun Du et al. proposed a query expansion method based on topics of interest. The authors adopted the TREC as the contexts and built concept lattices as the expansion source [10]. Besides, a distinguishing point of this paper is that the expansion source terms are corresponding to the nodes of ODP directory tree in order to extract the interest topics. Nicolas Spyratos and Carlo Meghini et al. [11], Ferre, S. and Ridoux, O. [12]both came up with a new query approach which combines navigation and querying into a single process. They extend formal concept analysis-based query by considering user preferences. The former authors provide detailed examples to explain all their assumptions clearly. The latter authors proposed Logical Concept Analysis where logical formulas took place of attributes as formal descriptions. F. Grootjen and T. van der Weide generated a local thesaurus by projecting global collection information onto the top ranked document [13]. T.I. Wang, T.C. Hsieh et al. proposed a query-based partial ontology knowledge acquisition system [14]. Authors gave a brief introduction for the formal concept analysis and applied formal

concept analysis approach and users' query to construct a query-based partial ontology. N. Stojanovic proposed a query refinement approach, which interacted with user and provided appropriate query [15]. The author used formal concept analysis to calculate the Quantifying content-related ambiguity. B. Safar and H. Kefi harnessed domain ontology and formal concept analysis for implementing an interactive querying system on a topical resources repository [16]. Jon D. et al. developed D-SIFT to provide untrained users with practical and intuitive access to the core functionality of formal concept analysis to explore relational database schema [17]. Sergei O. et al compared several concept lattice construction algorithms for generating concept lattices [18]. They gave the conclusion about how we should choose algorithm in different situations. Baixeries J et al. proposed a new and fast algorithm to build Hasse diagram for concept lattice [19]. They improved the algorithm proposed in article [20] and compared these two algorithm in complexity analysis and experimental. In article [21] authors explained the benefit of using the Hasse diagram when apply formal concept analysis. Authors in article [22] [23] [24] talk about the application of formal concept analysis on data mining. Meghini C. et al. applied formal concept analysis on digital library [25].

Chapter 2: Problem definition

In recent years, it is popular for people to retrieve information by using search engine. The most important task of search engine is presenting more additional relevant web pages and reducing those web pages which are useless. Query expansion is an efficient method for search engine optimization. Through query expansion, we can increase the quality of the search results particularly in increasing recall, precision and relevance [26]. There are many ways to implement query expansion. We choose formal concept analysis as our research since formal concept analysis is a method for data analysis, knowledge representation and information management [27], which are the main characteristics of formal concept analysis.

The **main aim** of my thesis is to adapt a formal concept analysis algorithm which is suitable for the query expansion system. To achieve this main aim I separated this aim in to follow research question.

RQ1: What is the current application situation of query expansion based on formal concept analysis?

SQ1: What are the main characteristics of formal concept analysis?

SQ2: How is formal concept analysis theory applied on query expansion?

RQ2: Which lattice construction algorithm is the best to be adapted?

SQ3: What are the main features of existing concept lattice construction algorithms?

SQ4: Which lattice construction algorithm can significantly reduce calculation cost and fulfill the query expansion requirements after being adapted?

RQ3: How to reduce the calculation cost for query expansion based on formal concept analysis?

SQ5: How can we adapt the selected lattice construction algorithm?

SQ6: How can we design and implement query expansion prototype to compare adapted algorithm and original algorithm?

RQ stands for research question and SQ stands for Sub question. In the following section I give the explanation for these research questions and how can I answer these questions.

2.1 Background study

The first research question is: What is the current application situation of query expansion based on formal concept analysis? In my research, this question is divided into two sub questions:

1. What are the main characteristics of formal concept analysis?
2. How is formal concept analysis theory applied on query expansion?

For doing the research for query expansion based on formal concept analysis, I need to understand the definitions, concept and theories in formal concept analysis.

Because “formal concept analysis has been originally developed as a subfield of applied mathematics based on the mathematization of concept and concept hierarchy” [28], there are lots of complex theories in this research field. For understanding how concept lattice constructions works and adapted them in the future, I need to learn about this research field.

Because I want to apply formal concept analysis on query expansion, besides learning basic theories of formal concept analysis I must find out how it works with query expansion. I need to find out how to implement a query expansion system with formal concept analysis and what is the benefit and drawback of query expansion system based on formal concept analysis.

To answer the sub question 1, I need read some books and articles which talk about the formal concept analysis theory. Like book [29]. To answer the sub question 2, I read several articles which have list in chapter 1.

2.2 Algorithm selection

The seconded research question is: Which lattice construction algorithm is the best to be adapted? To answer this question I separated it into two sub questions.

1. What are the main features of existing concept lattice construction algorithms?
2. Which lattice construction algorithm can significantly reduce calculation cost and fulfill the query expansion requirements after being adapted?

As I have talked the main aim of this research is to adapt a concept lattice construction algorithm for the query expansion system. Therefore, I need to select few algorithms to adapt. To select algorithms, I require finding out the main features of these concept lattice construction algorithm. These algorithms should be sorted into groups by the way they construct concept lattice. I should learn the characters for different algorithm groups.

After deep understanding the concept lattice construction algorithms, I start select the suitable algorithms which will be adapted later. Because I will apply the selected algorithms on query expansion system, the selected algorithms must create the concept lattice diagram which will be required by query expansion system. Furthermore, the selected algorithms should have the potential to be adapted to improve the efficiency in query expansion system.

To full fill above requirement, I read several articles which are about the compare of concept lattice construction algorithms and about proposing new algorithms. After compare the complexity of these algorithms and the result of experiments for these algorithms, I choose few algorithms to adapt based on the requirement of the query expansion system.

2.3 Adaptation and implement algorithms

The third research question is: How to reduce the calculation cost for query

expansion based on formal concept analysis? In my research I divide it in to two sub question.

1. How can we adapt the selected lattice construction algorithm?
2. How can we design and implement query expansion prototype to compare adapted algorithm and original algorithm?

After select the suitable algorithms, I adapt them in the way which can improve the efficiency of query expansion system and also satisfy the requirement for the precision and recall in query expansion system.

When I adapt the selected algorithm, I implement a query expansion prototype system to test the efficiency of these algorithms. The query expansion prototype system works with the concept lattice diagram. It does the query expansion for the Google that is using Google as the information resource. It returns the query expansion words for the user, and records the calculation time of construct concept lattice diagram for different algorithms. With these calculation times, we can compare the efficiency of each algorithm.

After solve the above questions, I achieved the main aim of this thesis. Adapted concept lattice construction algorithms, and test it on a query expansion prototype system.

Chapter 3: Methodology

In my thesis, I use both the qualitative methods and quantitative methods to do the research. At the beginning of thesis work, I do the literature review to understand the definition, the theories and the application of the formal concept analysis. After literature review, I do the case study to select the concept lattice construct algorithm which will be adapted and applied on my query expansion prototype system. Also I implement a prototype system to do the experiment as the quantitative methods.

3.1 Literature review

At the beginning of my research, I conduct literature survey: literature search and literature review. I adopt this method because it is a feasible approach to interpret and evaluate all available research relevant to a particular research question or topic area [30].I search a large number of relevant books, journals and articles from online databases such as Compendex/Inspec, Science Direct, Google scholar and so on. To learn more about the background, I start with a board search on "what has already been done on query expansion?" Then I focus my search on query expansion based on formal concept analysis to get a deep insight into the working principle of it. Besides, I put forward lots of appraisal criteria and conduct a critical evaluation of final set of articles while doing a systematic review. In the process, I can discover some areas that need further research. Finally, I analyze the pros and cons of query expansion based on formal concept analysis to know the application of formal concept analysis and identify what I can do to fill some gaps.

The result of my literature review can be found in Chapter 1.

3.2 Case study for lattice construction algorithm selection

Concept lattice construction algorithm is the foundation for applying the concept lattice. The process of concept lattice construction is the process of concept clustering. Concept lattice has completeness relation, which means the order of sort is not influent by the data or attribute's order and different construction algorithm should generate a unique lattice. After concept lattice theory is proposed for twenty years, a lot of researches have proposed several concept lattice construction algorithms. From the way to construction, the construction algorithm can be categorized into 3 groups: batch algorithm, incremental algorithm and parallel algorithm.

3.2.1 Introduction for different kind of algorithm

1. Batch algorithm
   The basic principle of batch algorithm is to generate all concepts from formal

concept context. After that the algorithm generates the relation between the concepts. Batch algorithm has two steps. (1) Generates all concept lattice nodes set. (2) Generates the immediate predecessor and immediate successor relation of concept lattice nodes. There are two ways to implement algorithm. One is to generate all the concept lattice nodes set, and then create the graph structure of concept lattice. Another is to generate part of concept lattice nodes, after that add those nodes into concept graph structure.

According to the construction order, the batch algorithm can be divided into 3 groups: (1) Top to bottom algorithm. First generate the nodes on the top, and then spread downward. Like Bordat [31] algorithm and OSHAM [32] algorithm. (2) Bottom to top algorithm. Oppositely, this kinds of algorithm create the nodes in the bottom first, and the spread upward, such as Chein algorithm [33], iPred Algorithm [19] and Border Algorithm [20]. (3) Enumeration algorithm. Enumeration algorithm will enumerate all the nodes in some order, after that create the Hasse diagram, i.e. the relation between nodes. This kind of algorithm includes Ganter [33]algorithm, Noruine[34] algorithm.

2. incremental algorithm
   The basic principal of incremental concept lattice algorithm is that, it supports N nodes have been generated in the concept lattice diagram. When the (n+1)th concept will be added into concept lattice diagram, update the previous concept lattice diagram. Repeat this step until generate the whole concept lattice structure.

   Incremental algorithm should consider those issues, when adds new node into concept lattice structure. (1) Generate the new node. (2) Avoid generate the reduplicate node. (3) Update the Hasse diagram.

   The Godin[35] algorithm and Carpinet[34] algorithm is the typical incremental concept lattice construction algorithm.

3. parallel algorithm
   Parallel algorithm is to process concept lattice by the distribute computing. Parallel algorithm divides formal context into several sub-formal context, which is distribute stored.[36] Then it will generate the sub-concept lattices and combine these sub-concept lattices into concept lattice. This kind of algorithm divides it into a number of sub-assignment; each sub-assignment will be processed by different processor or computer in the same time. With the widely using of network technology, the parallel concept lattice construction algorithm has the foundation for implement and practical use.

3.2.2 Algorithm selection

It is know that how to generate the concept lattices is a #P-complete problem and the number of the concept lattices can be exponential in the size of the formal context. Therefore to find a suitable concept lattice construction algorithm is very important

for query expansion.

Not all the algorithms can create the concept lattice diagram in the same time. Therefore we just consider the algorithms generate diagram. In article [37] the authors compare several algorithms in 2002, for the algorithms which can generate diagrams; they focus on the Bordat algorithm and Godin algorithm. They find although in article [35] authors said their Godin algorithm can get better result, in some situation the quite old Bordat algorithm [34] has better efficiency. They give the conclusion that: when calculate small and sparse formal context the Godin algorithm is better, but for the large and dense formal context it is better to use Bordat algorithm. In 2009, J. Baixeries et al [19] propose a new algorithm for building concept lattice and the concept lattice diagram. They compared their iPred algorithm with Border [20] algorithm. From complexity and experiment, iPred algorithm is better than Border algorithm.

It is not always necessary to do whole calculation in the concept lattice construction algorithm when we apply formal concept analysis to query expansion. In the concept lattice construction algorithm, there are lots of works for maintaining the mathematic properties of the lattice, like completeness relation etc. However, some of these mathematic properties are useless for applying formal concept analysis to query expansion. That means an 'incomplete' lattice can also work well for query expansion. Just like F. Grootjen et al. [13], they did partial calculation while constructing lattice. Therefore, I will select an algorithm which can easily be adapted to generate the suitable lattice for query expansion.

For the requirement of the query expansion I choose the iPred algorithm and the Border algorithm to improve. Here I give reasons to choose them below.

1. I will choose the batch algorithm to apply on query expansion system. Just like article [13] said, to do the query expansion, system just needs to do the partial calculation for concept lattice construction algorithm. We need to find which algorithm can be possible to do the partial calculation.

   As I have written in the section 4.3, when apply formal concept analysis on query expansion, we need to create the concept lattice and concept lattice diagram at first. After that, the mining frequent item sets or association rules, mining frequent closed item sets or other condensed representations of frequent patterns will be applied the created concept lattice [22]. To use these data mining technologies, the extension of each concept lattice node must be available for the query expansion system. The query expansion system will find the concept lattice nodes which have high quantity of extension. Later than the system try to discover the relation between the high quantity extension concept lattice nodes and the query keyword.

   Therefore the query system just woks on the high quantity extension concept lattice nodes. Some low quantity extension concept lattice nodes can be skipped during the query expansion process.

   For the Batch algorithm, it first generates all concept lattice nodes, and then creates the relation between them. Before the process of creating the relation, system can eliminate the low quantity extension concept lattice nodes, and just

create the relation for the concept lattice nodes with high quantity extension. In this way query system can save the computing time.

However for the incremental algorithm, it creates the concept lattice node one by one and inserts it into the concept lattice diagram. During the inserts process, system will change the relation between the concept lattice nodes and the extension of these nodes. Hence, before the whole concept lattice diagram created, the extension of all concept lattice nodes can be changed, and we can not decide the high quantity extension concept lattice nodes. Therefore, the incremental algorithm is very hard to be adapted for the query expansion system.

Now we talk about the specific algorithm. The Godin algorithm is the incremental algorithm, so it is not suitable for our query expansion system. Although the Bordat is a kind of batch algorithm, the computing of concept lattice note and the relation are interleaved and difficult to separate [19]. Therefore the Bordat algorithm also not selected as our candidate construct algorithm.

2. In article [37], authors have compared the Bordat algorithm and the Godin algorithm. They find when the cardinality of attribute set in formal context g is 25, the Godin algorithm works very good. But when the cardinality of attribute set in formal context is 50, the efficiency of the Godin algorithm decreases dramatically.

   In the query expansion system, the cardinality of attribute set in formal context is normally larger than 50, therefore this is the second reason we eliminate the Godin algorithm from our candidate list.

3. In 2008 B. Martin et al. proposed the Border algorithm [20], the method is "to the best of our knowledge, the first attempt to address the precedence computation problem with data mining concerns in mind. In fact, the method only considers the set of all (frequent) intents and organizes them into a graph representing the Hasse diagram of the (iceberg) lattice." [19]

   In 2009 B.Jaume et al. improved the Border algorithm [19]. They call their new method iPred algorithm. The authors have significant improve the efficiency the new algorithm. The complexity of iPred algorithm is

   $$| C | \times \omega(L) \times |M|$$

   $| C |$ is the size of the input set. $|M|$ is the size attribution. $\omega(L)$ is the width of the lattice in the worst case. Compare to the complexity of Border algorithm

   $$| C | \times \omega(L) \times |M|^2,$$

   the iPred algorithm change the $|M|^2$ to $|M|$, that means that they has improve the algorithm by a factor linear on the size of the attribute set.

Because these two algorithms are really new and have great efficiency, I decide to apply these two algorithms on my prototype system and adapt them for the requirement of query expansion.

Chapter 4: Theoretical work

4.1 Basic theorem on formal concept analysis

To apply mathematical methods on concepts and concept hierarchy relations, there must be a mathematical model which can mathematically express the *object, attributes* and relationships which indicate an attribute belong to an object. This model was proposed in article [1] and it is called "formal context". The formal context is the basic for the applied mathematics: *Formal Concept Analysis*.

4.1.1 Formal context

Definition 1: A ***formal context*** is defined as a triple *K:= (G, M, I)* where *G* is the set of objects (in German: Gegenstände) , *M* is the set of attributes (in German: Merkmale) . *I* is the binary relation between objects set *G* and attributes set *M*. For $\forall x \in G$, $y \in M$, if x has the attributes y, x is relative to y, write as xIy or $(x, y) \in I$. It is read: the object *x* has the attribute *y*.

To define the formal concept form formal context *(G, M, I)*, we need to define following derivation operators for arbitrary $A \subseteq G$ and $B \subseteq M$ at first.

$$g : A \Rightarrow A' := \{m \in G \mid (\forall n \in A) nIm\}$$
$$f : B \Rightarrow B' := \{n \in M \mid (\forall m \in B) nIm\}$$

These two derivation operators satisfy following condition:
(1) $Z_1 \subseteq Z_2 \Rightarrow Z_1' \supseteq Z_2'$, similar as $Z_1 \subseteq Z_2 \Rightarrow f(Z_1) \supseteq f(Z_2)$
(2) $Z \subseteq Z''$, similar as $Z \subseteq g(f(Z))$
(3) $Z''' = Z'$, similar as $f(g(f(Z))) = f(Z)$

4.1.2 Formal concept

Definition 2: The ***formal concept*** of a formal context K:= *(G, M, I)* is a pair (A, B), where $A \in P(G)$, $B \in P(M)$, $P(G)$ and $P(M)$ represent the power set of objects and attributes sets, and $A' = B$, $B' = A$. A is called an **Extension** of concept (A, B), B is called as **Intension** of concept (A, B). They can be write as $Exten(C)$ and $Inten(C)$.

Definition 3: Let ***C*** as the set of all concepts that generate from the set of object *G*, the set of attribute *M* and their relation *I*.

The concepts in *C* have the precedence relation within them. They are order in following way:

Definition 4: The concept $C_1 = (A_1, B_1)$ is more specific than concept $C_2 = (A_2, B_2)$ if it has a restricted extension. It is mathematized by

$$C_1 \le C_2 \Leftrightarrow Exten(C_1) \subseteq Exten(C_2)$$
$$\text{Or } (A_1, B_1) \le (A_2, B_2) \Leftrightarrow A_1 \subseteq A_2 \Leftrightarrow B_1 \supseteq B_2$$

Having a restricted extension is equivalent with having an augmented intension. The precedence relation $\le$ within $c,\ c \in C$ is a partial order.

Definition 5: for two concepts C_1 and C_2, and $C_1 \le C_2$. If there is no concept C_3 with $C_3 \ne C_2,\ C_3 \ne C_1, C_1 \le C_3 \le C_2$ the concept C_1 is called the immediate predecessor of C_2, and denoted as $C_1 < C_2$.

Definition 6: the formal concept C with the precedence relation $\le$ is denoted by $L(K) =< C, \le_L >$, where $\le_L \subset C \times C$. *w(L)* is the width of *L*, and *d(L)* is the degree of all the elements in *L*.

Definition 7: for formal concepts $C_1, C_2 \in C$, the intersection of these two elements is the intersection of their Extension and the union of their Intension.

$$C_1 \cap C_2 = ((Exten(C_1) \cup Exten(C_2)), (Inten(C_1) \cap Inten(C_2)))$$

The union of these two elements is the intersection of their Extension and the union of their Intension.

$$C_1 \cup C_2 = ((Exten(C_1) \cap Exten(C_2)), (Inten(C_1) \cup Inten(C_2)))$$

Definition 8: *K:= (G, M, I)* is a formal context. Then *L(K)* is a complete lattice, called the concept lattice diagram of *(G, M, I)*. The infimum and supremum of the concept lattice diagram are described as follows:

$$\underset{t \in T}{\wedge}(A_t, B_t) = (\underset{t \in T}{\cap} A_t, (\underset{t \in T}{\cup} B_t)'')$$
$$\underset{t \in T}{\vee}(A_t, B_t) = ((\underset{t \in T}{\cup} A_t)'', \underset{t \in T}{\cap} B_t)$$

4.1.3 An example of formal concept

A formal context can be easily understood if it is depicted by a cross table as for example the formal context about words in document in table 1.

Table 1. Formal concept example

Word document	A	B	C	D	E
1	**X**	**X**	**X**		
2		**X**	**X**	**X**	
3	**X**			**X**	**X**
4			**X**	**X**	**X**
5	**X**				

6	**X**

According to the above formal context, we can generate the formal concept in table 2

Table 2. Formal concepts

concept	Extension	Intension	concepts	Extension	Intensio
1	123456	Φ	7	34	DE
2	135	A	8	1	ABC
3	1246	C	9	2	BCD
4	234	D	10	3	ADE
5	12	BC	11	4	CDE
6	24	CD	12	Φ	ABCDE

In the figure 1 it shows the (Hasse) Formal concept diagram for the above formal concepts.

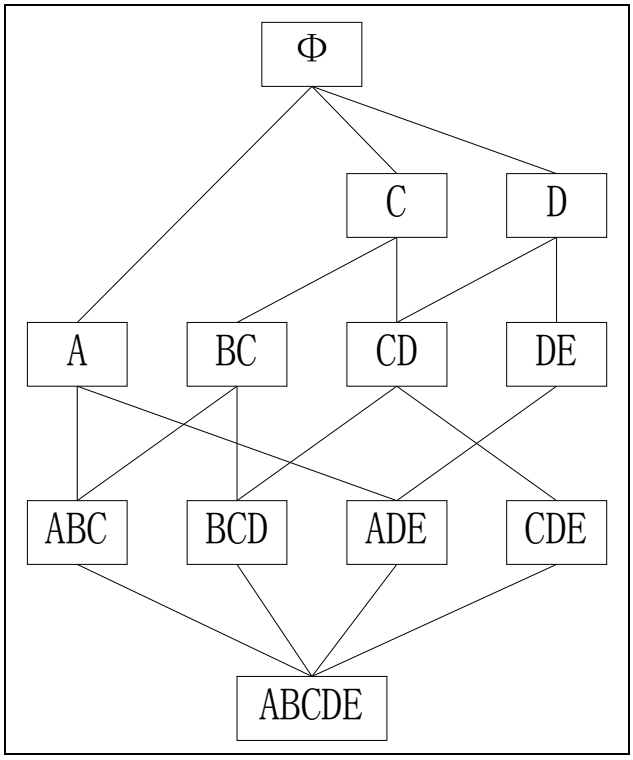

Figure 1. Formal concept diagram

In figure 1 just shows the intension of each concept lattice, you can find the extension of each concept lattice in table 2. The concept lattice (135, A) is the predecessor of concept lattice (1, ABC), (3, ADE) and (Φ, ABCDE). But (135, A) is just the immediate predecessor for (1, ABC) and (3, ADE), not for (Φ, ABCDE).

More detail of the formal concept analysis can be found in book [38].

4.2 Adapted algorithms

In section 3.2 I have discussed that I will choose Border and iPred algorithm as the original algorithms which will be adapted later in this chapter. To understand how I

adapt the original algorithms, I give the introduction for these two algorithms at first.

4.2.1 Introduction for original algorithms

First I give the pseudo code for these two algorithms.

The table 3 shows the pseudo code for Border algorithms and the table 4 shows the pseudo code for iPred algorithm. The more detail can be find in article [19], [20], [39].

1. Border algorithm

Table 3. The Border algorithm

Input: $C = \{c_1, c_2 \ldots c_l\}$
Output: $L = < C, \leq_L >$
1 Sort(C);
2 *Border* $\leftarrow \{c_1\}$;
3 **foreach** i $\in \{2, l\}$ do
4 *Candidate* $\leftarrow \{c_i \cap c' \mid c' \in Border\}$
5 *Cover* $\leftarrow$ *Maxima (Candidate);*
6 $\leq_L \leftarrow \leq_L \cup \{(c_i, c') \mid c' \in Cover\}$;
7 *Border* $\leftarrow$ (*Border – Cover*) $\cup$ c_i ;
8 **end**

I use the example in 4.1.3 to explain how this algorithm works. For convenience I just do the calculation for the intension here. The input is:

$\{\Phi, c, d, a, bc, cd, de, abc, bcd, ade, cde, abcde\}$

We assume the element Φ, *c, d, a, bc, cd, de* and *abc* have been insert into the L, and the next element to be processed is *bcd.* Figure 2 show the current situation.

Table 4. Maxima function

Input: *Candidate* = $\{c_1, c_2 \ldots c_l\}$
Output: *Cover*
1 *Cover* $\leftarrow \Phi$
2 **Foreach** *c'* $\in$ *reverse(Intents)*
3 *ismin* = 1
4 **foreach** $c \in$ *Cover*
5 **if** $c' \cap c = c'$ **then**
6 *Ismin* = 0
7 **end**
8 **end**
9 **if** *ismin* = 1 **then**
10 *Cover* $\leftarrow$ *Cover* $\cup c'$
11 **end**
12 **end**

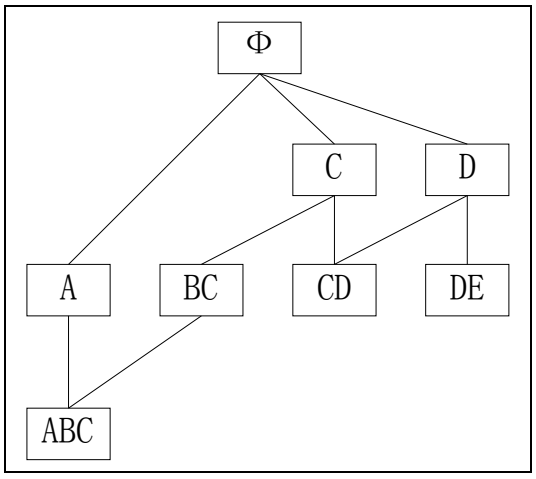

Figure 2. Iceberg of the already processed elements

The *Border* set now is the elements {*cd, de, abc}*, which are generated when insert *abc* into the *L*. The *Candidate* set computed in line 4 will get the following set: {*cd, bc, d, Φ*}, which is the intersection of *bcd* with border set. In line 5 *Maxima* computes the *Cover* set from *Candidate* set which is {*cd, bc*}, and then, the connections (*bc, bcd*) and (*cd, bcd*) are added to $\leq_L$ in line 6. In line 7 the sets *cd* and *bc* are removed from *Border* set, and *bcd* is added into the *Border* set. So after this iteration the *Border* set of the algorithm is {bcd, abc}.

The complexity of Border is: [20] [39]

$$|C| \times w(L) \times |M|^2 .$$

2. iPred algorithm.

To explain the iPred algorithm, I need to define the ***face***, ***set of faces*** and ***accumulation of faces*** at first.

Definition 9: The face of an element $c \in C$ for its each immediate successor *c'* is the difference between these two sets. The set of faces is:

$$faces(c) = \{c' - c \mid c' < c\}$$

I take the concept *a* in figure 1 as an example. The immediate successors of *a* are *abc* and *bc*, and the *faces*(*a*) = {*bc, de*}.

Before I give the definition for the accumulation of faces, I need to define an ***enumeration*** of C, which will be used in accumulation of faces.

Definition 10: An enumeration of C is the set:

$$enum(C) = \{c_1, c_2, ..., c_n\}$$

Such that $\forall i, j \leq n : i \leq j \Rightarrow |Inten(c_i)| \leq |Inten(c_j)|$

Actually, the enumeration is just a size wise sorting of the elements of a set.

Definition 11: the accumulation of faces of an element $c \in C$ for each element in *enum(C)* is:

$$\Delta_c^i = \bigcup\{c_j - c \mid c_j \in enum(C) \cap c_j < c \cap j < i\}$$

I also take the concept lattice L in figure 1 as an example. The *enum*(C) is:

{Φ, c, d, a, bc, cd, de, abc, bcd, ade, cde}

The accumulation of faces for a up to 9 is $\Delta_a^9 = bc$ and the accumulation of faces for a up to 11 is $\Delta_a^{11} = bcde$. In table 5 I give the pseudo code for the iPred algorithm.

Table 5. The iPred algorithm

Input: $C = \{c_1, c_2 \ldots c_l\}$
Output: $L =< C, \leq_L >$
1 Sort(C);
2 **foreach** $i \in \{2, l\}$ do
3 $\Delta[c_i] \leftarrow \Phi$
4 **end**
5 *Border* $\leftarrow \{c_1\}$;
6 **foreach** i $\in \{2, l\}$ do
7 *Candidate* $\leftarrow \{c_i \cap c' \mid c' \in Border\}$
8 **foreach** $c' \in$ *Candidate* do
9 **if** $\Delta[c'] \cap c_i = \Phi$ then
10 $\leq_L \leftarrow \leq_L \cup (c_i, c')$;
11 $\Delta[c'] = \Delta[c'] \cup (c_i - c')$;
12 *Border* $\leftarrow$*Border* –*c'*;
13 **end**
14 **end**
15 *Border* $\leftarrow$*Border* $\cup$ c_i;
16 **end**

The iPred algorithm is quite similar to Border algorithm. It computes the sets Border and Candidate, but the Cover set is not used in this algorithm anymore. A new structure Δ is introduced in this algorithm, which store the accumulation of faces for all the elements of the lattice. The notation $\Delta[c]$ show the access to the accumulated faces of the set of attributes c. The complexity for access the element in it should be $|M|$.

I still use the example in 4.1.3 to explain how this algorithm works. For convenience I just do the calculation for the intension here. The input is:

{*Φ, c, d, a, bc, cd, de, abc, bcd, ade, cde, abcde*}

We assume the elements *Φ, c, d, a, bc, cd* and *de* have been insert into to L, and the current process element is *abc*. Figure 3 show the current situation.

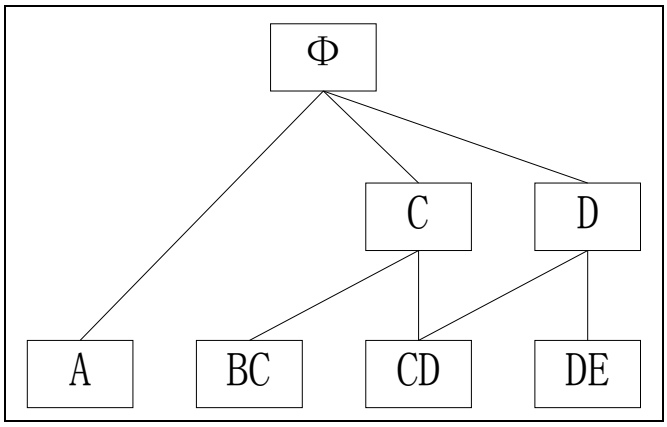

Figure 3. Iceberg for current situation

The *Border* set now is the elements {*a, bc, cd, de}*, which are generated when insert *de* into the *L*. The *Candidate* set computed in line 7 will get the following set: {*Φ, a, c, bc*}, which is the intersection of *abc* with border set. At that time, the accumulations of faces for all elements in Candidate set are:

$$\Delta[\Phi] = acd,\ \Delta[c] = bd,\ \Delta[a] = \Phi,\ \Delta[bc] = \Phi.$$

Because $\Delta[a]$ and $\Delta[bc]$ intersect with *abc* are equal to Φ. According line 9, we add (*a, abc*) and (*bc, abc*) into the relation set $\leq_L$ in line 10, and then accumulation of faces for *a* and *bc* are changed to $\Delta[a] = bc$, $\Delta[bc] = a$ in line 11. *Border* set remove element *a* and *bc* in 12, and added *abc* in line 15. So the *Border* set after this iteration of the algorithm is {*cd, de, abc*}.

The complexity of iPred algorithm is [19]

$$|C| \times w(L) \times |M|.$$

4.2.2 Description of Adapted algorithms

I adapted above algorithms for getting better efficiency when I apply these two algorithms on query expansion system. During I do research on query expansion based on formal concept analysis, I find that when do the query expansion, the system just care of the concept lattice which have high cardinality of its extension set. (I describe this in section 4.3.) Therefore, we can set a threshold for the cardinality of the concept lattice extension, when construct the concept lattice diagram. If the cardinality of the concept lattice extension is bigger than the threshold, algorithm adds it into the concept lattice diagram. Otherwise the concept lattice node will be skipped. In this way, we can save a lots calculation time for construct concept lattice diagram.

1. Adapted Border algorithm.

Table 6. Adapted Border algorithm

Input: $C = \{c_1, c_2 \ldots c_l\}$
Output: $L = < C, \leq_L >$
1 Sort(*C*);
2 *Border* ← {c_1};
3 **Foreach** i ∈ {2, *l*} do
4 **if** *Cardinality*(*Exten*(c_i)) > *threshold* **then**

```
5       Candidate ← {c_i ∩ c' | c' ∈ Border}
6       Cover ← Maxima (Candidate);
7       ≤_L ← ≤_L ∪{(c_i, c') | c' ∈ Cover} ;
8       Border ← (Border – Cover) ∪ c_i ;
9   end
10 end
```

The adapted algorithm is quite similar to the original one, the only difference of these two algorithms is in line 4. In line 4, adapted algorithm will check whether the cardinality of extension in concept lattice c_i is bigger than the threshold. If the cardinality is bigger than the threshold, algorithm will insert this concept lattice into the concept lattice diagram. Otherwise this concept lattice will be jump over.

The value of the threshold can be set by the user. In my experiment I set the threshold as 1.

Because the adapted Border algorithm and original one are similar and I have given the explanation in 4.2.1, I do not describe too much here.

2. Adapted iPred algorithm.

Table 7. Adapted iPred algorithm

```
Input: C = {c_1, c_2... c_l,}
Output: L =< C, ≤_L >
1   Sort(C);
2   foreach i ∈ {2, l} do
3       if Cardinality(Exten(c_i)) > threshold then
4           Δ[c_i] ← Φ
5       else
6           delete Δ[c_i]
7       end
8   end
9   Border ← {c_1};
10  foreach i ∈ {2, l} do
11      if Cardinality(Exten(c_i)) > threshold then
12          Candidate ← {c_i ∩ c' | c' ∈ Border}
13          foreach c' ∈ Candidate do
14              if Δ[c'] ∩ c_i = Φ then
15                  ≤_L ← ≤_L ∪(c_i, c') ;
16                  Δ[c'] = Δ[c'] ∪ (c_i − c') ;
17                  Border ← Border –c';
18              end
19          end
20      end
21      Border ← Border ∪ c_i ;
22  end
```

The algorithm works as following:

1. In line 1, it sorts the elements of the lattice by size. After this step the sequence

is an enumeration as in Definition 10.

2. All the $\Delta[c_i]$ in each element whose cardinality of extension is bigger than threshold is initialized to the empty set. Other elements whose cardinality of extension is smaller than the threshold do not have $\Delta[c_i]$. (line 2-8) This $\Delta[c_i]$ will contain the accumulation of faces fore these elements.
3. Put the first element in the sequence into the border. (line 9)
4. The rest elements in the input sequence are processed in the order in which they are in the enumeration. (line 10-22)
5. The element whose cardinality of extension is bigger than threshold is processed. (line 11)
6. The current element c_i intersect with all the elements in the border and generate the candidate set. (line 12)
7. Check if the current element has no intersection with the accumulation of faces for the elements which are in the candidate set. If this test result is positive, that means the element *c'* in the candidate set is the immediate predecessor for current element c_i. (line 14)
8. Connect c_i and *c'*. (line 15). Update the accumulated of faces for *c'*. (line 16).
9. Remove the *c'* from the *Border* set and add c_i into the *Border* set. (line 17 and line 22)

3. Running example

I use the example in section 4.1.3 to run the adapted algorithm, and set the threshold as 1. The following variable will be show:

1. The current processed element.
2. The candidate set.
3. The relation set.
4. The accumulation of faces. Only show the changed element in it.
5. The border set.

After step 1 (line 1) we can get the enumeration of *C*.

$\{\Phi, a, c, d, bc, cd, de, abc, bcd, ade, cde, abcde\}$

Table 8 shows how the adapted iPred algorithm works for *C*.

Table 8. Running example for adapted iPred algorithm

	1	2	3
Current element	*a*	*c*	*d*
Candidate set	$\{\Phi\}$	$\{\Phi\}$	$\{\Phi\}$
Relation set $\leq_L$	(Φ, a)	(Φ, c)	(Φ, d)
Accumulation of faces	$\Delta[\Phi] = a$	$\Delta[\Phi] = ac$	$\Delta[\Phi] = acd$
Border set	$\{a\}$	$\{a,c\}$	$\{a,c,d\}$
	4	5	6
Current element	*bc*	*cd*	*de*
Candidate set	$\{\Phi, c\}$	$\{\Phi, c, d\}$	$\{\Phi, d\}$
Relation set $\leq_L$	(c, bc)	(c, cd) (d, cd)	(d, de)
Accumulation of faces	$\Delta[c] = b$	$\Delta[c] = bd, \Delta[d] = c$	$\Delta[d] = ce$
Border set	$\{a, bc, d\}$	$\{a, bc, cd\}$	$\{a, bc, cd, de\}$

The rest elements (*abc, bcd, ade, cde, abcde*) will be skipped in adapted iPred algorithm, because the cardinality of their extension is not bigger than the threshold 1. The concept lattice diagram shows in figure 4.

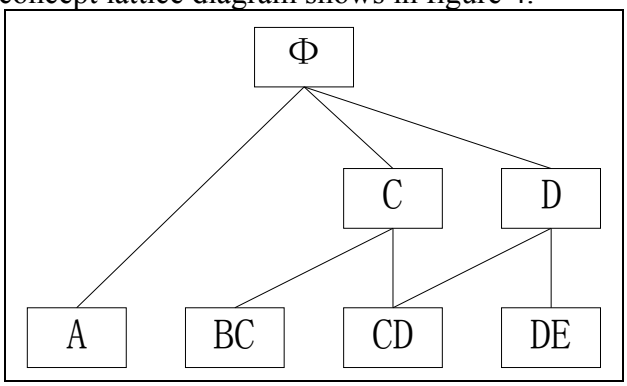

Figure 4. Result of adapted iPred algorithm

4.2.3 Complexity analysis

In article [19], authors give a complexity analysis for Border algorithm and iPred algorithm in detail. For Border algorithm the complexity is

$$|C| \times w(L) \times |M|^2$$

The complexity for iPred algorithm is

$$|C| \times w(L) \times |M|$$

The complexities of the adapted algorithms are the same of original algorithm. However in adapted algorithm the size of $|C|$ and size of $w(L)$ are changed. I use the running example in section 4.2.2 to explain how these two factors changed.

When using the adapted iPred algorithm, because the elements (*abc, bcd, ade, cde, abcde*) in enumeration *C* are skipped, the $|C|$ is changed from 11 to 7. Although $w(L)$ is still 4 for the adapted iPred algorithm, it will change when G and M are big. Especially when we apply formal concept analysis on the query expansion system, the documents set (*G*) and the words set (*M*) are very big, and a lot of word just appeared in few document. Therefore, if we set a property threshold, we can shrink the size of $|C|$ and $w(L)$ dramatically.

Above all it is very hard to compare the complexity for the adapted algorithm and original algorithm by theory analysis, thus I do the experiment in next chapter. So we can see how the adapted algorithm works for the query expansion prototype system.

4.3 Expansion word generated from concept lattice diagram

After we have the concept lattice diagram *L* of the query result page, we can apply the association rules mining on the concept lattice diagram to generate the query expansion word. Nowadays, most commercial search engine do the query expansion based on the global user habit. That method has high efficiency at text match, but can not provide the concept level query expansion. We take the Google as an example. If

user type “laptops” as the query keyword, Google will give the user some suggestion. Like “laptops for sale”, “laptops cheap” and “laptops deal” etc… We can see these suggestions are the most popular keywords for normal user. Most people want to buy cheap laptops through the Internet. However, if the user just want to know the popular laptops brand, these suggestions does not so useful. In this time, the user need the system give a suggestion about the sub concept of their query keywords.

The concept lattice can solve the above problem very well. Appling concept lattice on query expansion system is one kind of dynamic cluster query expansion methods. It will not generate the query expansion word by the global user data set, but using the data form the query result at the first time. This query expansion system select the main concept in the first time query result pages, and then give these main concepts as the suggestion for the user. In this way the query expansion system not only give the user expansion word in the same level of their query keywords, but also show the user sub concept of their query keywords.

First I give some basic knowledge of association rule mining, which will be used in query expansion later.

4.3.1 Association rule mining:

Association rule mining with the concept lattice diagram is similar to Apriori. Both of them create the frequent item sets, and then calculate the confidence coefficient between each frequent set. According to the above data, system can mine the association rule from these frequent item set.

Here gives the Apriori property. It is the foundation of the association rule mining. You can find the detail in article [40].

Apriori property: all the sub sets of the frequent item set are frequent item set.
The Apriori property based on following observation: according to the definition. If an item set I can not meet the minimum support threshold s, I is not the frequent item set, that is $P(I) < s$. if add item A into I, the new item set ($I \cup A$) is not more frequent than item set I. Therefore, $I \cup A$ also is not frequent item set, that is $P(I \cup A) < s$.

Based on the concept lattice diagram (Hasse diagram), we can do the query expansion word mining. The whole process has two steps: **First,** do the association rules mining for the concept lattice diagram. **Second,** generate the query expansion words form these association rules.

4.3.2 Mining association rule from concept lattice diagram

In the general association rule mining, the system first finds the frequent item sets, and then use those frequent item sets create the association rule. But if we have the concept lattice and generate its concept lattice diagram, the system will be very easy to find the frequent item sets.

In the query expansion system, we define the frequent item sets like this: for the

concept lattice node N (D_N , W_N) in concept lattice diagram, D_N is the document sets (extension) of this node, and W_N is the word sets (intension) of this node. If the document quantity (the cardinality of extension set) in D_N is bigger than or equal to the minimum support times quantity of all documents, this concept lattice node N is a frequent item node.

We take figure 5 as an example to explain how to select the query expansion words. We assume the minimum support threshold is 0.4. In figure 3 the node (135, *AB*) includes 3 documents. The quantity of the all document is 5, so the support of this node is 3/5 = 0.6. It is bigger than the minimum support threshold 0.4, thus it is a frequent item set. In the same way, the nodes (12345, *A*), (12, *AC*) and (45, *AD*) are the frequent item set.

From the figure 5 we find that we use the simple breadth first traversal of the concept lattice diagram instead of other complex methods to find the frequent item set. If the support of the node is bigger than or equal to the minimum support threshold, the intension of this node is a frequent item set.

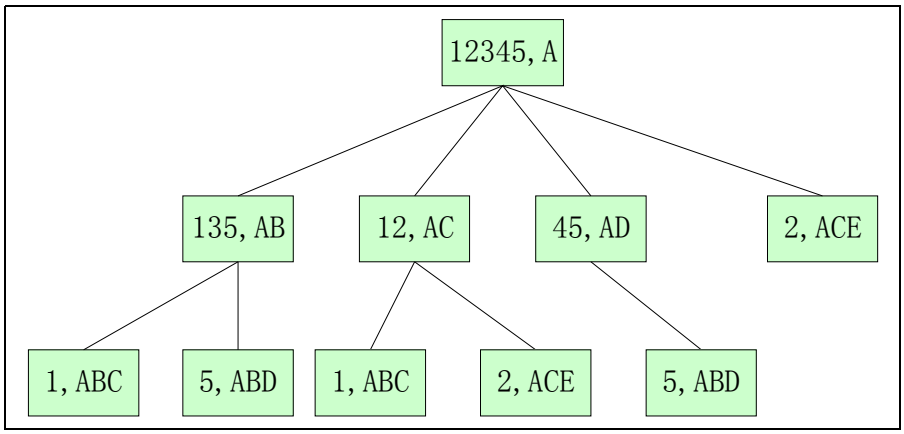

Figure 5. Example for generate query expansion words

How to find the association rule form the frequent item set? We first give the definition of it.

Definition 12: There are concept node C_1 and C_2 in concept lattice diagram. C_1=($g(A)$, $f(g(A))$, C_2=($g(A \cup B)$, $f(g(A \cup B))$, besides C_1 and C_2 are the frequent item sets. If $|g(A \cup B)|/|g(A)| \geq min\_conf$ (minimum confidence threshold), then $R: A \Rightarrow B$ is the rule which generate by the pair (C_1 , C_2). We call (C_1 , C_2) as the generate pair for this rule, and $C_1 \geq C_2$. The support and the confidence of this rule can be calculated by the pair (C_1 , C_2) dirtily.

Definition 13: The support of rule R:

$$support(A \Rightarrow B) = |g(A \cup B)|/|U| = |Extension(C_2)|/|U|$$

Definition 14: The confidence of rule R:

$$confidence(A \Rightarrow B) = |g(A \cup B)|/|g(A)| = |Extension(C_2)|/|Extension(C_1)|$$

We also take Figure 5 as the example. We have find the frequent item set in it: (12345 , *A*) , (153 , *AB*) , (12 , *AC*) and (45 , *AD*) . The concept lattice

node (12345, A) is the precedence node for another tree nodes, that is （12345，A）≥（153，AB）. Now we can calculate the confidence between those nodes. For concept lattice node （12345，A）and （153，AB）, the confidence of them is 3/5=0.6. But the confidence of other node with (12345, A) is 0.4. If the minimum confidence threshold is 0.5, then the rule in the concept lattice diagram is just $A \Rightarrow B$.

4.3.3 Get the query expansion word from association rule.

To get the query expansion word from association rule is based on the definition of association rule and confidence. If the confidence of rule $A \Rightarrow B$ is bigger than the threshold, that means when there is word A the word B has c percent probability appeared at the same place. Therefore, if the confidence c is very high, that means A and B have the close relation. When the confidence of the rule is bigger than the minimum confidence threshold, word B is a query expansion word for the word A.
Because the query expansion system extract the association rule form the frequent item sets, the support of all association rule is bigger than the minimum support threshold. Meanwhile, when generating these association rules, the system has calculated the confidence of each rule. System just need set the confidence of words in consequent of rule equal to the confidence of the rule. After that select several words which have the biggest confidence value as the expansions word.

In my experiment, I will choose the top 10 high confidence rule to generate the expansion word.

Chapter 5: Empirical study

The experiment is employed to examine the calculation cost of four algorithms in my thesis work. To do the experiment I implement a prototype query system, and implement four algorithms on it.

The experiments are carried out on a Genuine Inter CPU 1.66 GHz laptop with 1.5 GB RAM running under Windows XP. The prototype system is implemented by C# in Microsoft Visual Studio 2008 under .Net Framework 3.5. Google is used as original search engine on which system do the query expansion.

The following paragraphs describe how this prototype system works.

5.1 Prototype main flow

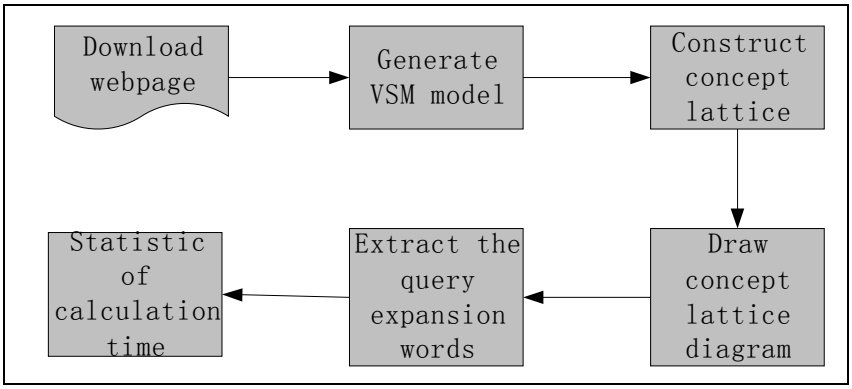

Figure 6. Main flow of prototype system.

The query expansion prototype system consisted of six main steps: Download webpage, retrieval the webpage and generate VSM model, construct concept lattice, Draw concept lattice diagram, extract the query expansion words and statistic calculation time.

To analysis the result returned by search engine, first we need to download that result page. After that we need extract the useful information form this page. The prototype system filters out images, commercials and the frame of this page. Then the prototype system gets the text information of that page, and it deletes the common exclude words from this text information. The common exclude words are some words which are useless for query expansion. Like link verb, auxiliary verb etc. when system finish above steps, the VSM model is generated. In VSM, one search result is represented by several words. It can be expressed like this: $Result = R(w_1, w_2, w_3 \ldots w_n)$. Here w_n represents the n-th word appearing in result abstract.

System uses the VSM model as the formal context. One search result in the VSM represents an object, and the words for it represent attributes. The four concept lattice construction algorithms which have been described in section 4.2 are applied on this formal context to create the concept lattice diagram.

After we get the concept lattice diagram, system draws it on the user interface, and does the association rule mining on it to get the expansion words. The calculation time for the query expansion of each algorithm is recorded in a log file. These calculation times are considered as the result of our experiment, which can evaluate the efficiency of four construction algorithm.

5.2 Data structure design

For implement the prototype system, I designed several data structures for it.

5.2.1 Data structure for Set

I use the template `List<string>` as a Set. Every string in the list represent the element in the set. The union operation for two set can be down by `Union(IEnumerable(UTP))` funciton. The intersect operation for two set can be down by the `Intersect(IEnumerable(UTP))` funciton.

5.2.2 Class for concept lattice node

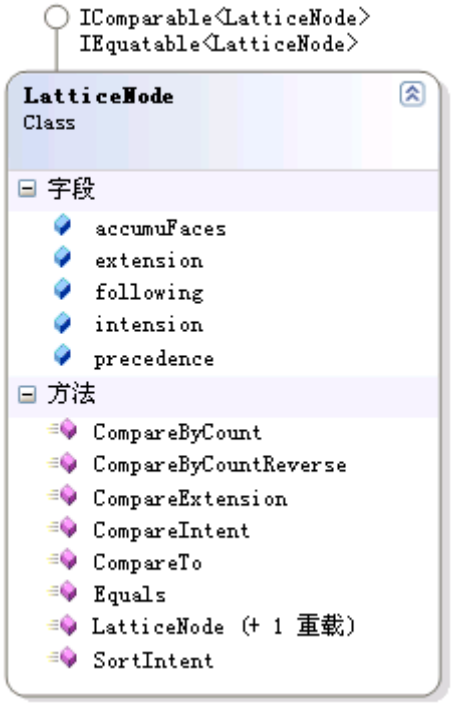

Figure 7. Class diagram for Lattice Node

The members of Lattice Node class are:

1. accumuFaces. It stores the accumulation of faces for this node. It will be used in iPred algorithm.
2. extension. It stores the extension of this lattice node.
3. following. It stores the immediate successor of this lattice node
4. intension. It stores the intension of this lattice node.
5. precedence. It stores the immediate predecessor of this node.

Figure 8. Interface IComparable and IEquatable

This class also implements the interfaces `IComparer<T>` and `IEquatable<T>`. These interfaces are required when sort the list of concept lattice node. Functions CompareTo and Equals are override to implement these two interfaces.

The delegate `int Comparison<T> (T x, T y)` is implemented by function CompareByCount, CompareByCountReverse, CompareIntent and CompareExtension. With these functions system can sort the concept lattice list in different ways.

5.2.3 Data structure for concept lattice

The concept lattice *C* is presented by a `List<LatticeNode>`. After concept lattice construction algorithm add precedence relation to all the concept lattice node in this list, this list present the concpet lattice diagram *L*.

5.3 Generate VSM model

In prototype system, we use query result page of Google as the information resource. The system requires Google return 100 results in one page. In this way the prototype system can get enough objects and attributes to evaluate concept lattice construction algorithm. As the input of prototype system, the system downloads it first. In this result page there are much useless information, like images, commercials and the frame of this page. System need to eliminate them first.

5.3.1 Filter out useless information

To filter out the useless information in the result pages, the prototype system uses the Regular Expressions to select the information what we want.

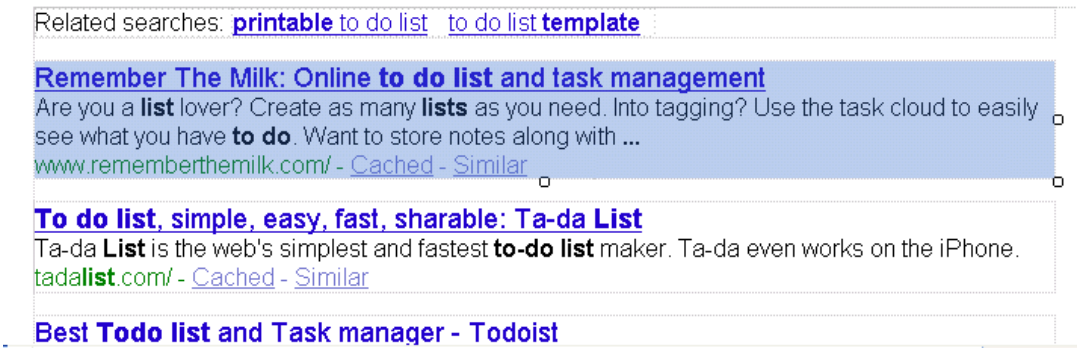

Figure 9. One result in the result page

In figure 9 show one result in the result page. The prototype system use “to do list”

as a query keyword, and the blue part in the figure 9 is the first result in the result page which returned by Google.

```
<li class="g">
    <h3 class="r">
        <a href="http://www.rememberthemilk.com/" class="l">Remember The Milk: Online <em>to
            do list</em> and task management</a></h3>
    <div class="s">
        Are you a <em>list</em> lover? Create as many <em>lists</em> as you need. Into tagging?
        Use the task cloud to easily see what you have <em>to do</em>. Want to store notes
        along with <b>...</b><br>
        <cite>www.rememberthemilk.com/ - </cite><span class="gl"><a href="http://74.125.77.132/
            Cached</a> - <a href="/search?hl=en&amp;lr=lang_en&amp;ie=UTF-8&amp;q=related:www.r
                Similar</a></span></div>
```

Figure 10. The HTML code for one result

In figure 10, there is the HTML code for the first result. Because we just need the information for these results, we can use the regular expression to get this information. The pattern used in the prototype system is `"<li class=g>.*?href="(.*?)".*?(<br>|<script>|<span class=a>)"`. After match the download result page with the pattern, system get the informaiont show in figure 11.

```
---page start---
<body> <li class=g><h3 class=r><a href="http://www.rememberthemilk.com/"
class=l>remember the milk: online <em>to do list</em> and task
management</a></h3><div class="s">are you a <em>list</em> lover? create as many
<em>lists</em> as you need. into tagging? use the task cloud to easily see what you
have <em>to do</em>. want to store notes along with <b>...</b><br> </body>
```

Figure 11. The select information for one result

After that systme use another patter `"<body.*?>(.*?)</body>"` to get the text informatio of this result. Figure 12 show the text information of this result. In this stpe system get the required information from the web pages.

```
    remember the milk: online  to do list  and task management   are you a  list
lover? create as many  lists  as you need. into tagging? use the task cloud to
easily see what you have  to do . want to store notes along with  ...
```

Figure 12. The text information for one result

5.3.2 Create VSM model

After the prototype system get the text information of result page, it start create the VSM model. First, the system removes the commoner morphological and inflexion endings from words in text information of result page. Here I use the Porter stemming algorithm to implement it [41]. Then, the system eliminate the exclusive words (like a, an, the etc.) and create VSM. In VSM, one search result is represented by several words. It can be expressed like this: *Result = R* (w_1, w_2, w_3 ...w_n). Here w_n represents the n-th word appearing in result abstract. The red part in Figure 12 presents in this way: *Result1 = R (remember, milk, online…, along)*. "the" between "remember" and "milk" is considered as exclusive word and is eliminated.

5.4 Concept lattice construction

After VSM of results page is created, system uses VSM to generate the formal context. One search result is mapped as Object, and the words in it will be mapped as attribute. Then, four concept lattice construction algorithms are applied on this formal context, and create the concept lattice diagram. First, all the concept lattice

notes are create. In Figure 13, every two line represent a concept lattice node, the intent represents the intension and the extent represents extension.

```
intent:
extent: -1
------------------
intent: busi
extent: 19 63 75
------------------
intent: probabl
extent: 20 88
------------------
intent: offic
extent: 19 44 50
------------------
intent: microsoft
extent: 19 57
------------------
intent: todo
extent: 2 18 28 29 38 45 73 85 37 49 78 82 56 88 40 52
------------------
intent: easi
extent: 1 14 21 31 33 36 53 2 24 78
------------------
intent: simpl
extent: 1 9 16 34 40 49 67 2 78 24
------------------
intent: due
```

Figure 13. Part of All concept lattice nodes

The number after extension represents the index of each result in query result pages. Take “Microsoft” as an example, it appeared in result 19 and 57. Figure 14.

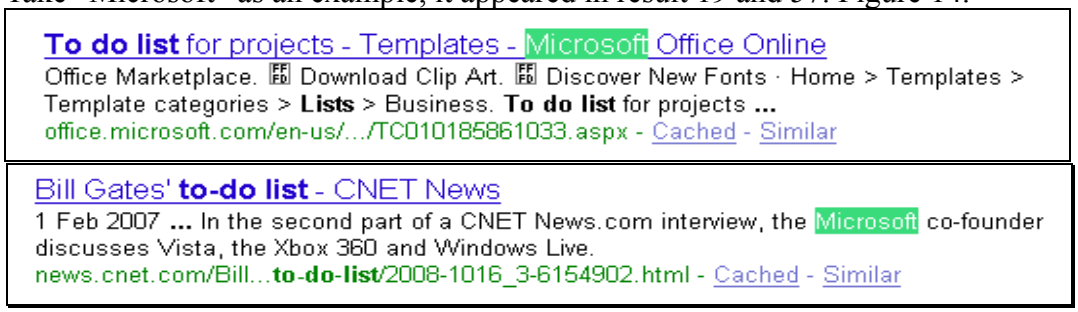

Figure 14. Result 19 and 57

The intension of the first node is query key word “to do list” and the extension is all. For convenience to calculate, I set its intension to “null” and extension to “-1”.

After all concept lattices are generated, the construction algorithm creates the precedence relation between them. That means construction algorithm connects these concept lattices *C* and create concept lattice diagram *L*.

5.4.1 Draw the concept lattice diagram

Once the concept lattice diagram *L* is generated, the prototype system uses the Treeveiw control to display concept lattice diagram. The depth-first traversal recursion algorithm is employed to draw this diagram. I sort the immediate successor of every node by their extension. In this way, the user gets the node which has more extension at top of diagram. Figure 15 shows the tree view in the prototype system. In this time, “to do list” is the query keyword.

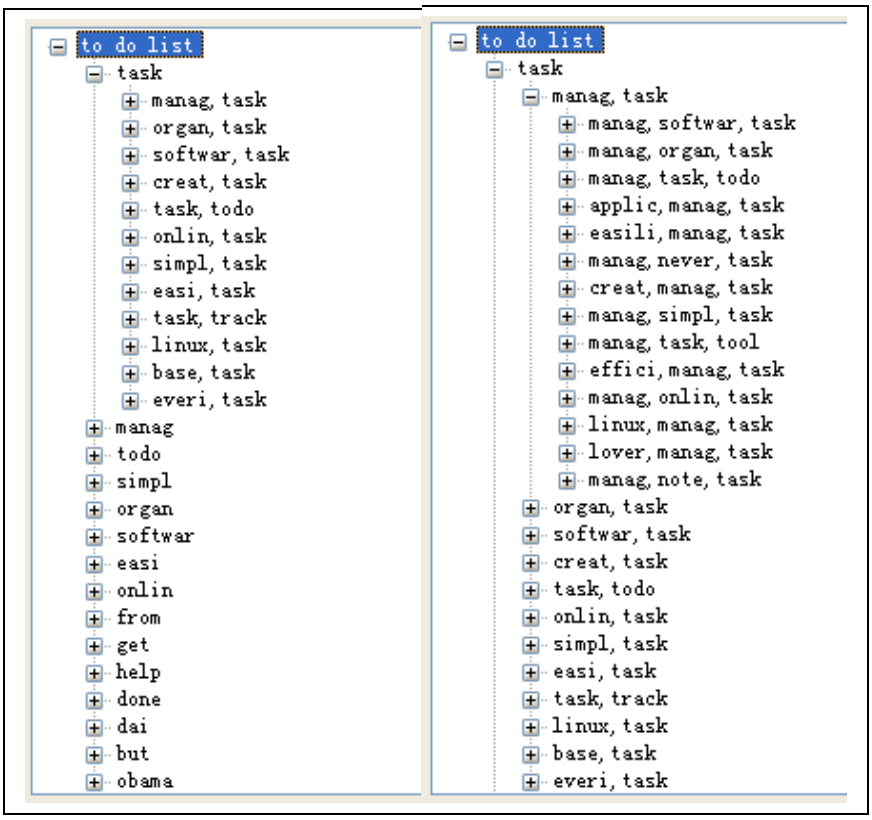

Figure 15. Tree view of concept lattice diagram

5.4.2 Generate expansion words

Once we have the concept lattice diagram, it is very easy to generate the expansion words form it. The prototype system just needs to do the breadth-first traversal for this diagram, and it need to find the concept nodes have high cardinality extension. These nodes which have high cardinality extension are the high frequent item. According assassination rule mining theory, the intension of these nodes are the expansion words. In figure 16 give the expansion words for the query key word as "to do list".

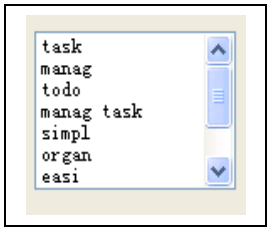

Figure 16. Expansion words for "to do list"

We can find there are some useful query expansion words like "task", "manag" and "todo" (manag is the mistake of manage, which are caused be the wrong extract word stem). Meanwhile consider my prototype is designed to evaluate the efficiency of the concept lattice construction algorithm, the expansion results are acceptable.

5.5 User interface design

Figure 17 gives the interface of prototype system. The text box in the top right corner

is the input text box for user typing the query keywords. Now in figure 17 I type “to do list” as the query keyword. The tree view in the left shows the concept lattice diagram for query keyword “to do list”. In the right there are group of radio button. User can choose which algorithm will be used to create the concept lattice diagram. In the figure I choose the iPred algorithm. The calculation time are counted as Milliseconds below. We can see the prototype system use 125 ms to generate the concept lattice diagram. Meanwhile this calculation time, query keywords and the name of algorithm are recorded in a log file every time when this system works. At the bottom right corner the expansion words are given in a list box.

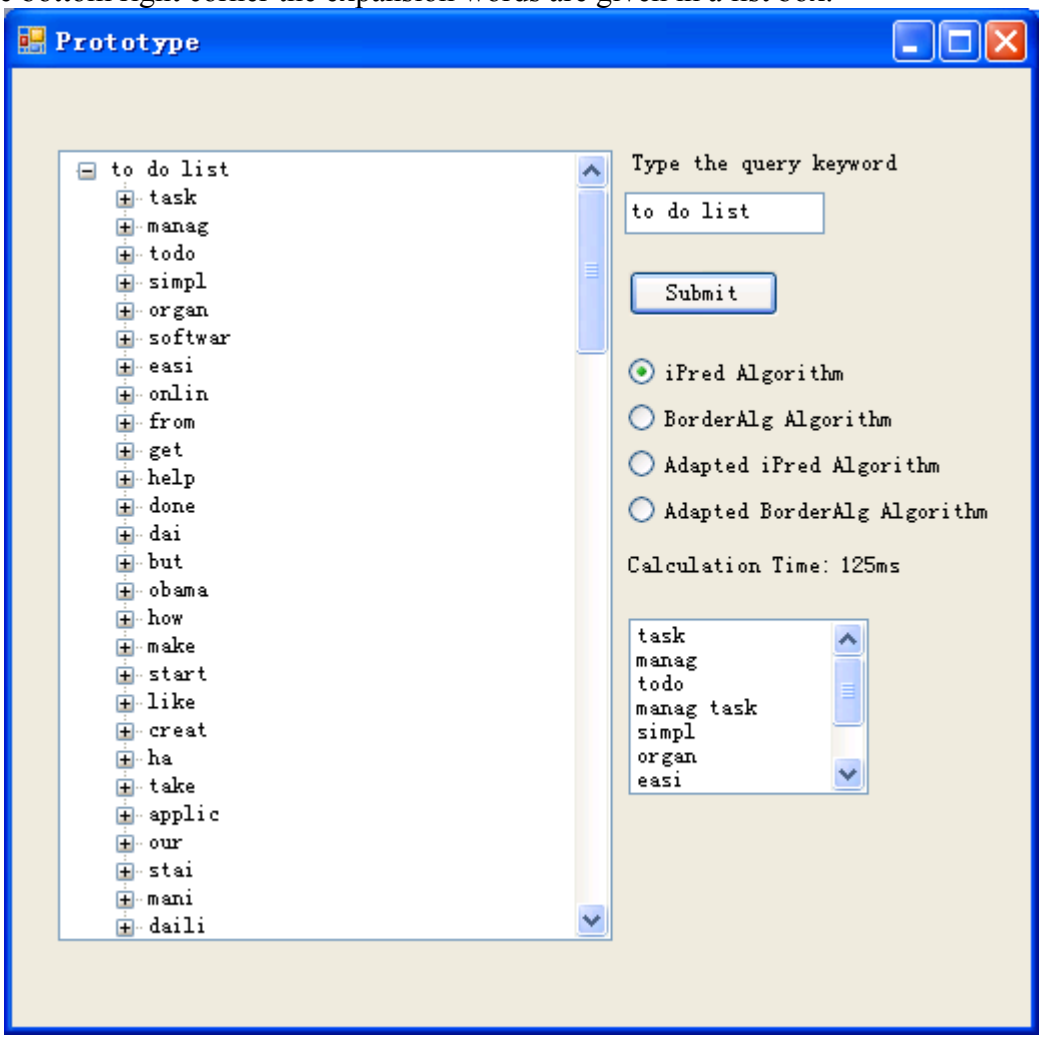

Figure 17. The interface of the prototype system

Chapter 6: Results and analysis

6.1 Data set

The input of the experiment is the search keyword. To make my experiment more objective, I use the hot trends from Google Trends [42] as the search keyword. I choose top 25 terms form 2009-07-27 to 2009-07-30 at Google trends USA. So I use 100 query keywords to do the experiment. Here, I select the top 25 terms in Google Trends 2009-07-29 in USA as the search keyword to analysis. These 25 terms are the input. See Table 9.

Table 9. The input of the query expansion prototype system

keywords	madonna biceps	tim alderson	freddy sanchez	jack wilson	fwm
	kathy wetherell	richard leroy walters	inkblot test	gossip cop	jeff clement
	rorschach blots	stephon marbury web show	path train	natalie smith	patricide
	ben francisco	cliff lee	rorschach test online	madonna photos	wladimir balentien
	gongoozler	west side story gangs	jane skinner	jason knapp	change blindness

6.2 Result

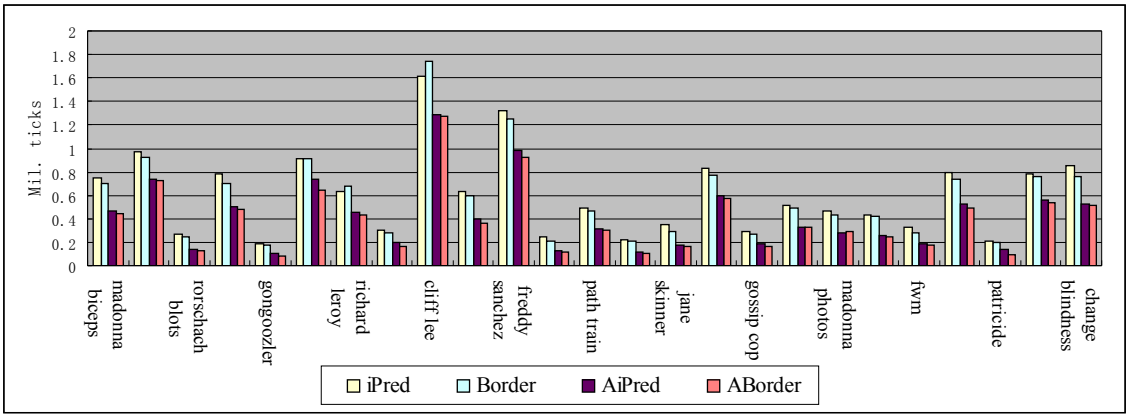

Figure 18. Example of experiment results

Figure 18 show the result of experiment for these terms. In figure 18, iPred, Border, Aipred and ABorder stand for iPred algorithm, Border algorithm, Adapted iPred algorithm and Adapted Border algorithm respectively. The unit of the *y*-axis is million ticks. It represents how many timer ticks for each algorithm constructing concept lattice diagram. The two adapted algorithm have high efficiency, adapted iPred algorithm reduce 31.02% calculation times and adapted Border algorithm reduce 31.13%.

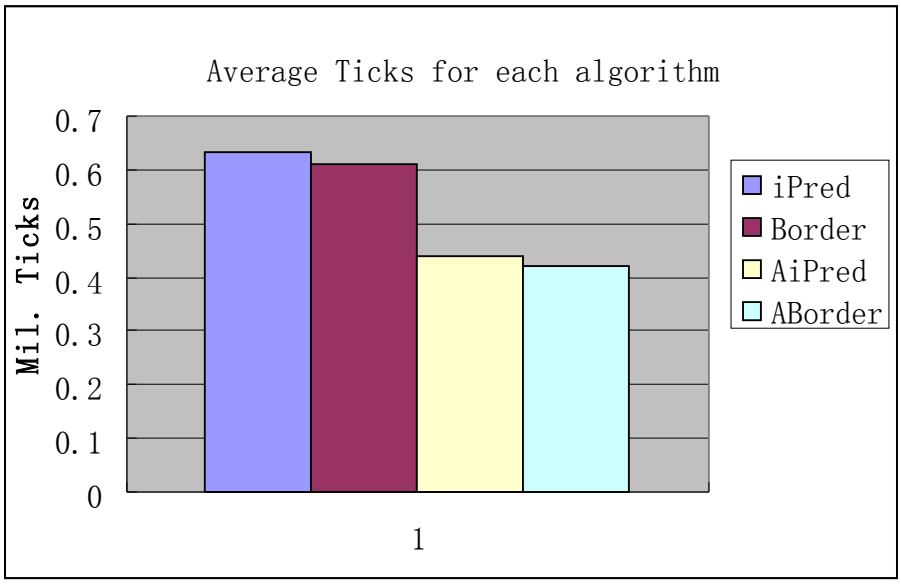

Figure 19. Average Ticks for each algorithm

6.3 Analysis and discussion

From figure 19 we can find the Border algorithm is little bit better than iPred algorithm in my prototype system. However, according to the complexity analysis, the iPred algorithm is supposed to be better. After analysis all the result of my experiment, I find that: if the number of concept nodes is quite large, the iPred algorithm has high efficiency.

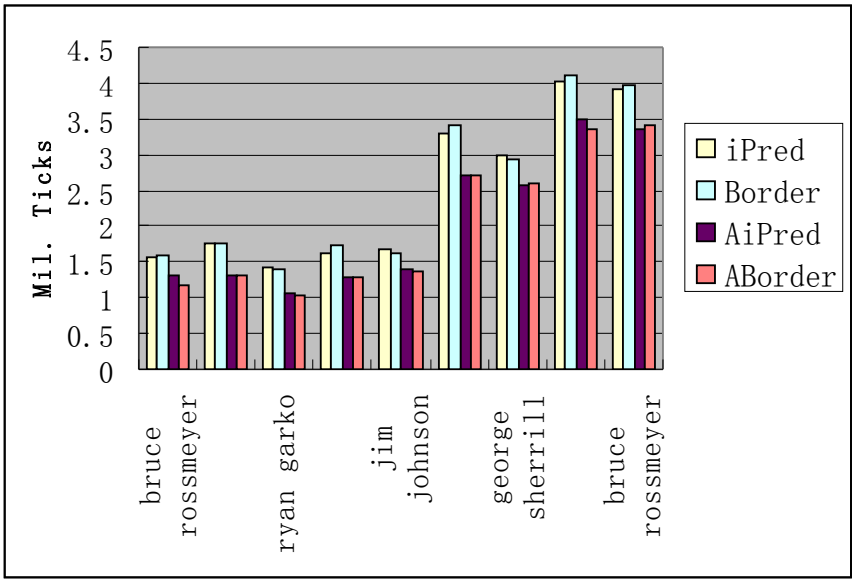

Figure 20. Efficiency for lager number of concept nodes.

In figure 20, there are the all results whose concept lattice nodes are more than 500. At this time, iPred algorithm works well. This is consistent with the article [19]. J. Baixeries et al. also test their algorithm on the synthetic dataset tT20I6D100K and T25I10D10K, which are create by IBM Almaden generator. The efficiency of the iPred algorithm is not satisfied on these datasets. It is because these datasets is a low min_supp datasets, and these datasets just can generate few concept lattice nodes.

The min_supp datasets means the sparse dataset. In article [19], authors also admit "in the case of sparse datasets (T20 and T25), the difference is not that spectacular."

In my prototype system the VSM model is a typical sparse dataset. The average number of total attributes is 138.09. But the total attributes for one object normally is just 8.

Moreover, because J. Baixeries et al. test their algorithm on the big dataset, they can generate thousands of concept lattice nodes (normally more than 10 thousands). However, when they use the minimum number of concept lattice nodes which they generated is 4711, iPred does not work very well. But for my query expansion prototype system, even the maximum number of concept lattice nodes is 753. This can explain why the iPred does not have high efficiency. Also we can not increase the scale of the datasets, because the prototype system needs to give the users expansion words in an acceptable time. No one can wait for more than 30 seconds for the expansion words.

We can have a look of the pseudo code in Maximum function in Border algorithm and iPred algorithm in table 10. They can give us some clues.

Table 10. Maximum function and iPred algorithm

Maximum function:
Input: *Candidate* = $\{c_1, c_2 \ldots c_l\}$
Output: *Cover*
1 *Cover* $\leftarrow \Phi$
2 **Foreach** *c'* $\in$ *reverse(Intents)*
3 *ismin* = 1
4 **foreach** $c \in$ *Cover*
5 **if** $c' \cap c = c'$ **then**
6 *Ismin* = 0
7 **end**
8 **end**
9 **if** *ismin* = 1 **then**
10 $Cover \leftarrow Cover \cup c'$
11 **end**
12 **end**

iPred algorithm
Input: $C = \{c_1, c_2 \ldots c_l\}$
Output: $L = < C, \leq_L >$
1 Sort(C);
2 **foreach** $i \in \{2, l\}$ do
3 $\Delta[c_i] \leftarrow \Phi$
4 **end**
5 *Border* $\leftarrow \{c_1\}$;
6 **foreach** i $\in \{2, l\}$ do
7 $Candidate \leftarrow \{c_i \cap c' \mid c' \in Border\}$
8 **foreach** $c' \in Candidate$ do
9 **if** $\Delta[c'] \cap c_i = \Phi$ then
10 $\leq_L \leftarrow \leq_L \cup (c_i, c')$;
11 $\Delta[c'] = \Delta[c'] \cup (c_i - c')$;
12 $Border \leftarrow Border - c'$;
13 **end**
14 **end**
15 $Border \leftarrow Border \cup c_i$;
16 **end**

The red codes in table 10 iPred algorithm do the similar work as Maximum function does in Border algorithm. The iPred algorithm maintains a set for each concept lattice node to store the accumulation of faces. With this set iPred algorithm do not need to do the loop in line 4-8 Maximum function. That is why iPred algorithm has low complexity. However, when apply these two algorithms on a sparse dataset which has few concept lattice nodes, the above advantage in iPred algorithm do not give the help of efficiency.

When work on a small sparse dataset, we just can generate a “simple” and “low” concept lattice diagram. That means the concept lattice node do not have so much relation with other nodes. Take my prototype system as an example. The concept lattice normally is 4-6 levels and the connections for one node normally are 2-3. For the Maximum function in Border algorithm, that means the *Cover* set is very small, and the loop in it just cost little time. At the same time, iPred algorithm still need to maintain the accumulation of faces set, and it cost more than the loop in *Cover* set according to the experiment result.

My query expansion prototype system works on a sparse datasets and has few concept lattice nodes. In this situation iPred algorithm is not suitable. Instead, the Border algorithm works well.

For the adapted algorithms, because they decrease the depth and width of the concept lattice diagram, they save 31% calculation time. However, the adapted iPred algorithm is little bit worse than the adapted Border algorithm. Due to the result and analysis above the query expansion prototype system should choose the adapted Border algorithm.

According to the experiment result and the conclusion in article [18], we can find the content of dataset has a great impact for the efficiency of a concept lattice construction algorithm. If we want to use concept lattice construction algorithm in our system, we can not just compare the efficiency of these algorithm by the complexity analysis. These algorithms must be implemented on a system, and test the real efficiency by the real dataset. Only in this way we can find out which algorithm is suitable for our system.

In Appendix A I give all experiment information

7. Conclusion and future work

7.1 Conclusion of study

At the beginning of this paper I gave a short introduction about query expansion methods and the formal concept analysis theory. The current researches for applying formal concept analysis on query expansion system were presented after that.

I analyzed and described the main characters of existing concept lattice construction algorithm, because I wanted to select some of them to adapt to improve the efficiency of query expansion system. The concept lattice construction algorithm can be categorized into to tree groups. The batch algorithm has the potential to improve the efficiency of query expansion system, so I choose two new algorithms to be adapted. These two new algorithms are iPred algorithm and Boarder algorithm. They are proposed in 2009 and 2008 separately.

Meanwhile, I explained how formal concept analysis theory worked on query expansion system. The iPred algorithm and the Border algorithm were explained in detail in this paper. I adapted these two algorithms by skip some concept lattice nodes which are unnecessary for query expansion system. Moreover, I have implemented a query expansion prototype system to test the efficiency of these four algorithms.

The efficiency of adapted algorithms was consistent with what it was supposed. Because the adapted algorithms decrease the depth and width of the concept lattice diagram, they reduce 31% calculation time compare with the original algorithms. Moreover, I find the Border and its adapted algorithms have better efficiency on my prototype system. That is because the Border algorithm is suitable for the sparse dataset. Coincidently, the data set of my prototype system is a sparse dataset. Meanwhile I analysis and discussed why Border algorithm works better on spars dataset.

7.2 Future work

Normally the commercial query expansion system uses more than one method to do the query expansion, therefore besides formal concept analysis we can apply other methods on my query expansion prototype system in the future. In this way we can improve the efficiency, precision and recall of this system.

From result of my experiment, we can find that the efficiency of algorithm is not always consistent with its complexity analysis result. When apply an algorithm on the query expansion system, the efficiency of this algorithm is great affected by the dataset structure. Therefore the theory analysis of complexity just can be a reference when we selected algorithm. The complexity of algorithm is not the real criteria.

Instead we should test the algorithms which have good complexity in the real system, and find out which algorithm is the best for specific query expansion system. So, in the future we can try some other algorithm which not mentioned in this paper. Those algorithms may give us a surprised result.

Acknowledgements

At the beginning, I thank Mr. Guohua Bai and Mr. Yang Liu for leading me to do the research. And I also appreciate that BTH library provides me with the powerful databases of adequate literatures, which are the important foundation for me to hunt more useful references. Also I have a word to my girl friend thanks for sharing with me all the happy time. You are so sweet. Finally, I am deeply grateful to my families and friends. Thanks them for supporting me no matter what adventurous plans I had and will have.

References

[1] R. Wille and F. Mathematik, "Restructuring lattice theory: an approach based on hierarchies of concepts," *Boston: Dordrecht*, 1982, pp. 445-470.

[2] R. Kraft and J. Zien, "Mining anchor text for query refinement," *Proceedings of the 13th international conference on World Wide Web*, ACM New York, NY, USA, 2004, pp. 666-674.

[3] N. Stojanovic, "On analysing query ambiguity for query refinement: The librarian agent approach," *Lecture notes in computer science*, 2003, pp. 490-505.

[4] C. Buckley, G. Salton, J. Allan, and A. Singhal, "Automatic query expansion using SMART: TREC 3," *Overview of the Third Text REtrieval Conference (TREC-3)*, Diane Pub Co, 1995, pp. 69-80.

[5] Y. Qiu and H.P. Frei, "Concept based query expansion," *Proceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval*, ACM New York, NY, USA, 1993, pp. 160-169.

[6] R. Attar and A.S. Fraenkel, "Local Feedback in Full-Text Retrieval Systems," *J. ACM*, vol. 24, 1977, pp. 397-417.

[7] Jie Wei, S. Bressan, and Beng Chin Ooi, "Mining term association rules for automatic global query expansion: methodology and preliminary results," *Proceedings of WISE 2000: 1st International Conference on Web Information Systems Engineering, 19-21 June 2000*, Los Alamitos, CA, USA: IEEE Comput. Soc, 2000, pp. 366-73.

[8] Y. Hai, Y. Du, and H. Li, "A new strategy of query expansion using formal concept analysis," *6th IEEE International Conference on Computer and Information Technology, CIT 2006, September 20,2006 - September 22,2006*, Seoul, Korea, Republic of: Inst. of Elec. and Elec. Eng. Computer Society, 2006.

[9] W. Cho and D. Richards, "Improvement of precision and recall for information retrieval in a narrow domain: Reuse of concepts by formal concept analysis," *Proceedings - IEEE/WIC/ACM International Conference on Web Intelligence, WI 2004, September 20,2004 - September 24,2004*, Beijing, China: IEEE Computer Society, 2004, pp. 370-376.

[10] Bing Zhang, YaJun Du, HaiMing Li, and YuTing Wang, "Query expansion based on topics," *2008 Fifth International Conference on Fuzzy Systems and Knowledge Discovery (FSKD), 18-20 Oct. 2008*, Piscataway, NJ, USA: IEEE, 2008, pp. 610-14.

[11] N. Spyratos and C. Meghini, "Preference-based query tuning through refinement/enlargement in a formal context," *Foundations of Information and Knowledge Systems. 4th International Symposium, FoIKS 2006. Proceedings, 14-17 Feb. 2006*, Berlin, Germany: Springer-Verlag, 2006, pp. 278-93.

[12] S. Ferre and O. Ridoux, "Searching for objects and properties with logical concept analysis," *Conceptual Structures: Broadening the Base. 9th International Conference on Conceptual Structures, ICCS 2001, 30 July-3 Aug. 2001*, Berlin, Germany: Springer-Verlag, 2001, pp. 187-201.

[13] F. Grootjen and T. van der Weide, "Conceptual query expansion," *Data & Knowledge Engineering*, vol. 56, Feb. 2006, pp. 174-193.

[14] T.I. Wang, T.C. Hsieh, K.H. Tsai, T.K. Chiu, and M.C. Lee, "Partially constructed knowledge for semantic query," *Expert Systems with Applications*, vol. In Press, Corrected Proof.
[15] N. Stojanovic, "On the query refinement in the ontology-based searching for information," *Information Systems*, vol. 30, Nov. 2005, pp. 543-563.
[16] B. Safar and H. Kefi, "Domain ontology and Galois lattice structure for query refinement," *Tools with Artificial Intelligence, 2003. Proceedings. 15th IEEE International Conference on*, 2003, pp. 597-601.
[17] J. Ducrou, B. Wormuth, and P. Eklund, "Dynamic schema navigation using formal concept analysis," *Data Warehousing and Knowledge Discovery. 7th International Conference, DaWaK 2005. Proceedings, 22-26 Aug. 2005*, Berlin, Germany: Springer-Verlag, 2005, pp. 398-407.
[18] S. Kuznetsov and S. Obiedkov, "Algorithms for the Construction of Concept Lattices and Their Diagram Graphs," *Principles of Data Mining and Knowledge Discovery*, 2001, pp. 289-300.
[19] J. Baixeries, L. Szathmary, P. Valtchev, and R. Godin, "Yet a Faster Algorithm for Building the Hasse Diagram of a Concept Lattice," *Formal Concept Analysis*, 2009, pp. 162-177.
[20] B. Martin and P. Eklund, "From Concepts to Concept Lattice: A Border Algorithm for Making Covers Explicit," *Formal Concept Analysis*, 2008, pp. 78-89.
[21] F. Dau, "The Advent of Formal Diagrammatic Reasoning Systems," *Formal Concept Analysis*, 2009, pp. 38-56.
[22] J. Boulicaut and J. Besson, "Actionability and Formal Concepts: A Data Mining Perspective," *Formal Concept Analysis*, 2008, pp. 14-31.
[23] F. Valverde-Albacete and C. Peláez-Moreno, "Galois Connections Between Semimodules and Applications in Data Mining," *Formal Concept Analysis*, 2007, pp. 181-196.
[24] F. Valverde-Albacete and C. Peláez-Moreno, "Towards a Generalisation of for Data Mining Purposes," *Formal Concept Analysis*, 2006, pp. 161-176.
[25] Carlo Meghini and N. Spyratos, "Computing Intensions of Digital Library Collections," *Formal Concept Analysis*, 2007, pp. 66-81.
[26] O. Hoeber, X. Yang, and Y. Yao, "Conceptual Query Expansion," *Advances in Web Intelligence*, 2005, pp. 190-196.
[27] W.C. Cho and D. Richards, "Improvement of Precision and Recall for Information Retrieval in a Narrow Domain: Reuse of Concepts by Formal Concept Analysis," *Proceedings of the 2004 IEEE/WIC/ACM International Conference on Web Intelligence*, IEEE Computer Society, 2004, pp. 370-376.
[28] R. Wille, "Formal Concept Analysis as Mathematical Theory of Concepts and Concept Hierarchies," *Formal Concept Analysis*, 2005, pp. 1-33.
[29] B. Ganter, G. Stumme, and R. Wille, *Formal concept analysis*, Springer, 2005.
[30] B. Kitchenham, *Procedures for performing systematic reviews*, 2004.
[31] M. Kamber, L. Winstone, Wan Gong, Shan Cheng, and Jiawei Han, "Generalization and decision tree induction: efficient classification in data mining," *Research Issues in Data Engineering, 1997. Proceedings. Seventh International Workshop on*, 1997, pp. 111-120.
[32] G. Snelting and F. Tip, "Reengineering class hierarchies using concept analysis," *Proceedings of the 6th ACM SIGSOFT international symposium on Foundations of software engineering*, Lake Buena Vista, Florida, United States: ACM, 1998, pp. 99-110.

[33] J. Han, “Towards on-line analytical mining in large databases,” *SIGMOD Rec.*, vol. 27, 1998, pp. 97-107.
[34] J.P. Bordat, “Calcul pratique du treillis de Galois d'une correspondance.”
[35] R. Godin, R. Missaoui, and H. Alaoui, “Incremental concept formation algorithms based on Galois (concept) lattices,” *Computational intelligence*, vol. 11, 1995, pp. 246-267.
[36] J.F.D. Kengue, P. Valtchev, and C.T. Djamegni, “A Parallel Algorithm for Lattice Construction,” *Formal Concept Analysis*, 2005, pp. 249-264.
[37] S.O. Kuznetsov and S.A. Obiedkov, “Comparing efficiency of algorithms for generating concept lattices,” *Journal of Experimental & Theoretical Artificial Intelligence*, vol. 14, 2002, pp. 189-216.
[38] B. Ganter and R. Wille, *Formal Concept Analysis: Mathematical Foundations*, Springer, 1998.
[39] P. Valtchev, M. Hacene, and R. Missaoui, “A Generic Scheme for the Design of Efficient On-Line Algorithms for Lattices,” *Conceptual Structures for Knowledge Creation and Communication*, 2003, pp. 282-295.
[40] L. Nourine and O. Raynaud, “A fast algorithm for building lattices,” *Information Processing Letters*, vol. 71, 1999, pp. 199-204.
[41] M. Porter, “The porter stemming algorithm,” *Accessible at http://www. tartarus. org/martin/PorterStemmer*.
[42] “Google Trends: Jul 29, 2009.”

Appendix A

	Query keywords	NCL	NAt	iPred	Border	AiPred	Aborder
1	sommelier	388	178	704411	708508	468588	411178
2	berres brothers coffee	262	136	404107	397354	236129	213863
3	oh calcutta	344	157	362868	327735	207254	186266
4	movits	451	158	850189	825165	620851	606677
5	adam rubin	415	140	758283	727275	521780	479845
6	everyone calls you amazing but i just call you mine	233	97	242642	237713	154560	161282
7	sandstone retreat	217	136	172269	161943	85842	85386
8	ryan garko	557	159	1414488	1383748	1065248	1024702
9	jillian and ed	395	135	663732	645664	419186	401922
10	alexis cohen death	710	100	3291047	3405626	2723208	2702839
11	coxcomb	259	146	213260	188652	111518	91223
12	cash for clunkers program	453	159	923353	857101	639606	591430
13	fairytale brownies	327	161	518662	471468	295367	286532
14	chris masters	366	168	789374	736215	484624	459455
15	cash for clunkers program 2009	347	151	541831	493771	303318	297688
16	the bachelorette finale	446	116	945363	912171	670471	657229
17	tom skerritt	370	144	454414	398739	285908	266312
18	morgan haley	435	113	1030992	1016136	840922	840131
19	culture of corruption	303	167	560736	541511	345048	317565
20	wire rope express	235	146	163807	144922	80777	73350
21	are jillian and ed still together	229	100	214363	205260	115886	115085
22	alexis cohen season 8	336	111	591583	566144	385592	382262
23	princeton review	384	115	507682	470988	312916	303960
24	chukkas	315	166	405758	383263	237960	211534

25	chrystee pharris	281	113	349009	328798	194022	189339
26	cash for clunkers eligible vehicles	270	113	477317	456889	282487	264335
27	jim johnson	415	171	682996	645996	471720	460820
28	thelma and louise last names	217	120	168927	157882	88978	79472
29	apollo alliance	426	169	775059	753224	527064	511638
30	brian anderson	311	143	316725	303957	188618	167871
31	josh willingham	462	144	855020	849696	624534	571518
32	mark kotsay	370	145	644518	633115	443505	426021
33	santa cruz city council	293	160	339977	302486	185421	170246
34	the room	252	153	204648	193716	95238	86340
35	holly letchworth	373	113	605782	618926	493367	479233
36	sommelier	384	177	650503	593810	412393	371872
37	the room movie	315	149	448171	404777	245438	241520
38	harridan	258	131	203827	189974	102977	98653
39	jillian harris and ed swiderski	309	112	413916	414586	269350	247344
40	jillian and ed still together	204	104	163523	155445	93344	82067
41	andrew kinard	345	178	402302	376866	246352	231198
42	jim johnson eagles	582	122	1666426	1629664	138977	137110
43	cash for clunkers program 2009	345	149	520398	486437	304014	280740
44	shana martin	305	133	292416	262864	165281	151743
45	harridan definition	222	116	185963	174688	101159	92596
46	movits	445	155	826584	816855	598605	584594
47	mortgage apple cake	392	137	796622	764177	576871	591362
48	bridge to terabithia	385	151	653330	614713	415162	392030
49	bachelorette finale	374	109	555012	544789	377016	359608
50	brett favre	340	149	474033	442956	287871	259655
51	madonna biceps	398	158	747155	702505	465421	449092
52	kathy wetherell	434	116	976300	920832	732278	722437
53	rorschach blots	261	115	264236	241877	140812	130813
54	ben francisco	427	137	788589	705525	499999	482568
55	gongoozler	232	129	190647	179295	101753	83962
56	tim alderson	477	139	909178	913716	732489	637625
57	richard leroy	362	118	631499	675232	452378	436781

	walters						
58	stephon marbury web show	288	109	309758	284225	200085	166796
59	cliff lee	580	158	1614723	1738642	1283720	1278374
60	west side story gangs	346	147	626079	592434	394951	362999
61	freddy sanchez	474	141	1323570	1256790	980858	923845
62	inkblot test	266	143	240599	215848	124724	112296
63	path train	384	148	489077	470116	318344	307995
64	rorschach test online	239	114	225781	208083	116667	104652
65	jane skinner	301	143	348822	289841	175401	158614
66	jack wilson	456	180	827672	774637	600305	567657
67	gossip cop	301	144	291940	268817	181895	163710
68	natalie smith	325	136	512360	494958	322987	326856
69	madonna photos	347	140	463792	435300	279974	297322
70	jason knapp	360	145	434651	418278	259534	239828
71	fwm	321	172	323990	286213	184606	178733
72	jeff clement	480	149	801027	740539	531260	490625
73	patricide	247	114	210896	198778	137430	94693
74	wladimir balentien	451	131	787270	762491	559926	536607
75	change blindness	384	148	857662	765296	527335	509472
76	quick weight loss center	301	132	319558	285993	194369	178432
77	buckler beer	336	133	415904	381157	264427	260600
78	cash for clunkers suspended	319	110	530675	533212	348161	322989
79	david ortiz	380	131	534253	513393	328464	301163
80	home with the kids	319	149	487003	395897	240216	230923
81	qwl	260	150	233968	208785	112063	98093
82	cash for clunkers list	397	128	759529	729701	508285	525826
83	thunderpants	357	167	529198	492215	333515	344583
84	bruce rossmeyer	754	120	3909734	3962719	3347985	3417433
85	thundercrack	286	150	250757	236735	135589	128995
86	samantha burke	728	118	4034624	4114258	3484001	3361695
87	omer bhatti.	548	96	1765598	1772568	1304929	1307468

88	joan walsh	348	159	429741	375018	248325	234537
89	virtual assistant jobs	257	138	289497	265658	149808	142656
90	george sherrill	720	141	2988958	2928484	2577823	2610588
91	bruce rossmeyer harley davidson	542	108	1564261	1599451	1304257	1174350
92	omer bhatti s mother	394	120	709478	634900	467694	452640
93	big papi	377	163	542985	513966	339496	370446
94	iphone virus	413	160	803368	751165	541785	522446
95	amy kucsmas	309	138	418909	396702	276282	272418
96	bryan texas	352	152	498532	449071	287568	270044
97	international virtual assistants association	292	118	471617	458596	283535	267555
98	rush limbaugh diet	258	131	211076	192266	119872	105301
99	qwlc diet	173	97	92648	75617	33975	32048
100	sheree whitfield	426	151	961948	952359	660242	633186

Third column specifies the number of concept lattice nodes. Forth column is the total number of attributes for concept lattice. Column 5, 6, 7 and 8 are the calculation time for iPred, Border, Adapted iPred and Adapted Border algorithm respectively. The unit for calculation time is tick.