Conceptual Relevance Feedback

F.A. Grootjen and Th.P. van der Weide

University of Nijmegen, Toernooiveld 1, Nijmegen, 6525 ED, The Netherlands
sparky@cs.kun.nl, tvdw@cs.kun.nl

Abstract— **Capturing the user's information need may be seen as a major challenge search engines are confronted with. This paper proposes a way to interpret the results of a user's initial query. This is done by positioning this preliminary result into a semantical structure called concept lattice. The resulting substructure is used for relevance feedback.**

Keywords— **relevance, feedback.**

I. Introduction

Formulating a query is not an easy task. Web search engines observe users spending large amounts of time reformulating their queries to accomplish effective retrieval [1]. Precise query formulation is difficult:

- *Do I know what I am looking for?* This often neglected aspect of information retrieval can be best explained by the fact that information need is created by a *knowledge gap*. This gap can range from being fairly specific to very broad. During the searching process users may learn things about their *knowledge gap* and even may discover aspects of this gap they were initially not aware of [6]). Search methods like *Query By Navigation* [3] may help users to find out what they need.
- *How do I formulate what I am looking for?* As in human dialogs, the participants must know each other's language and somehow predict the impact of the words they use. The same holds for query formulation. Good query formulation requires that a user can somehow predict which terms appear in documents relevant to the information need. Accurate term prediction requires extensive knowledge about the document collection. Such knowledge may be hard to obtain, especially in large document collections.

Experiments show that users usually submit short (one or two word) queries that result in large inaccurate document sets, apparently preferring recall above precision.

Relevance feedback, introduced over 30 years ago, is a well known approach to deal with this problem. This method treats the user's first query as an initial attempt: a rough representation of the user's information need hopefully covering (part of) the knowledge gap. The documents resulting from this initial query (the initial set) may be analyzed for relevance, to get an impression of the document collection, and used to formulate a new improved query. Usually query reformulation methods are grouped in three categories:

1. User feedback approaches. A drawback of this approach is that users are not inclined in providing this feedback. There is no point in blaming the user for this, providing feedback might be not cost-effective.
2. Local approaches, based on information obtained from the initial set of documents.
3. Global approaches that incorporate knowledge of the document collection.

II. Conceptual Query Reformulation

Since information need is a internal mental state of the user (*intension*) it is obviously difficult to grasp. Suppose a user is able to browse through all the documents of a collection and pick out all relevant ones. One might claim that this set of relevant documents is the *representation* of the information need within the context of this collection. This representation is also referred to as the *extension* of the user's information need.

Elaborating on this idea, one might argue that the outcome of a query, that is a set of documents, can be viewed upon as an approximation of (the extension of) the information need. Approximation, because it may not be the same as the extension: it probably contains irrelevant documents and some relevant documents will be missing. It would be useful if we could help users (interactively or automatically) to pinpoint the right extension of their information need. In order to give this kind of support we have created a semantical structure of interconnected nodes. Each node contains a set of documents and is a possible candidate to reflect the user's information need. Of course not all combination of documents are present in the structure: only those that form a semantically useful group: a so called *concept*. Due to the concepts' implicit ordering this semantical structure called *concept lattice* has nice navigational properties. Note that the used structure does not necessarily has to be derived from the same document collection as the target collection. It is feasible that conceptual query reformulation is done on a domain specific concept lattice and eventually carried out on the web.

A. Running example

To illustrate our ideas we will use an example presented in [2]. This example collection (see table I) contains 17 documents and 16 terms.

III. Concept lattice theory

In order to support relevance feedback efficiently, we need a model that somehow captures the 'meaning' of terms and documents. Using the theory called *Formal Concept Analysis* introduced by [7] we can create a mathematical structure which can be used to semantically classify documents in formal concepts. This structure (called lattice) has nice mathematical properties and is a starting point for navigational systems [3].

TABLE I
DATABASE OF TITLES FROM BOOKS REVIEWED IN SIAM

d_1	A Course on *Integral Equations*
d_2	Attractors for Semigroups and Evolution *Equations*
d_3	Automatic Differentiation of *Algorithms*: *Theory*, *Implementation*, and *Application*
d_4	Geometrical Aspects of *Partial Differential Equations*
d_5	Ideals, Varieties, and *Algorithms* - An *Introduction* to Computational Algebraic Geometry and Commutative Algebra
d_6	*Introduction* to Hamiltonian Dynamical *Systems* and the N -Body *Problem*
d_6	Knapsack *Problems*: *Algorithms* and Computer *Implementations*
d_8	*Methods* of Solving Singular *Systems* of *Ordinary Differential Equations*
d_9	*Nonlinear Systems*
d_{10}	*Ordinary Differential Equations*
d_{11}	*Oscillation Theory* for Neutral *Differential Equations* with *Delay*
d_{12}	*Oscillation Theory* of *Delay Differential Equations*
d_{13}	Pseudodifferential Operators and *Nonlinear Partial Differential Equations*
d_{14}	Sinc *Methods* for Quadrature and *Differential Equations*
d_{15}	Stability of Stochastic *Differential Equations* with Respect to Semi-Martingales
d_{16}	The Boundary *Integral* Approach to Static and Dynamic Contact *Problems*
d_{17}	The Double Mellin-Barnes Type *Integrals* and Their *Application* to Convolution *Theory*

TABLE II
CONTEXT RELATION OF SIAM REVIEW

	algorithms	application	delay	differential	equations	implementation	integral	introduction	methods	nonlinear	ordinary	oscillation	partial	problem	systems	theory
d_1					×		×									
d_2					×											
d_3	×	×				×										×
d_4				×	×								×			
d_5	×							×								
d_6								×						×	×	
d_7	×					×								×		
d_8				×	×				×		×				×	
d_9										×					×	
d_{10}				×	×						×					
d_{11}			×	×	×							×				×
d_{12}			×	×	×							×				×
d_{13}				×	×					×			×			
d_{14}				×	×				×							
d_{15}				×	×											
d_{16}							×							×		
d_{17}		×					×									×

A. Context

We denote the collection of documents with the letter $\mathcal{D}$. Individual members of this collection (documents) are written with small letters like d, d_1, d_2, while subsets are written in capitals (D, D_1, D_2). During the indexing process, descriptors (attributes) are attached to documents. We write $\mathcal{A}$ to denote the set of all attributes, a, a_1, a_2 for individual attributes and A, A_1, A_2 for attribute sets (subsets of $\mathcal{A}$). The result of indexing process is reflected in the binary relation $\sim$: we write $a \sim d$ iff attribute a describes document d. The tuple $(\mathcal{D}, \mathcal{A}, \sim)$ is called a *context*. The context relation $\sim$ is overloaded to cover set arguments.

Example 1
In our running example $\mathcal{D} = \{d_1, d_2, \ldots, d_{17}\}$, $\mathcal{A} =$ {algorithms, application, delay, differential, equations, implementation, integral, introduction, methods, nonlinear, ordinary, oscillation, partial, problem, systems, theory}. The context relation $\sim$ is depicted in table II.

B. Properties of contexts

Using the context relation a classification of documents and attributes can be generated such that each class can be seen as a *concept* in terms of properties of the associated documents and attributes. In our interpretation, documents and attributes assign meaning to each other via the context relation: within the limits of this view, we can not distinguish between document with identical properties, while attributes having the same extensionality are assumed to be identical. Sharing document meaning thus can be seen as sharing attributes:

Definition 1
The common attributes of a set of documents are found by the right polar function $\mathbf{ComAttr} : \mathcal{P}(\mathcal{D}) \to \mathcal{P}(\mathcal{A})$ defined as follows:

$$\mathbf{ComAttr}\ (D) = \{a \in \mathcal{A} \mid a \sim D\}$$

Documents may also be shared by attributes:

Definition 2
The documents sharing properties are captured by the left polar function $\mathbf{ComDocs} : \mathcal{P}(\mathcal{A}) \to \mathcal{P}(\mathcal{D})$ defined by:

$$\mathbf{ComDocs}\ (A) = \{d \in \mathcal{D} \mid A \sim d\}$$

Example 2
In the context of the running example we have:

1. $\mathbf{ComAttr}\ (\{d_1, d_5\}) = \emptyset$
2. $\mathbf{ComAttr}\ (\{d_{10}, d_{11}\}) = \{\mathsf{differential}, \mathsf{equations}\}$
3. $\mathbf{ComDocs}\ (\{\mathsf{differential}, \mathsf{equations}\})$
   $= \{d_4, d_8, d_{10}, d_{11}, d_{12}, d_{13}, d_{14}, d_{15}\}$
4. $\mathbf{ComAttr}\ (\{d_4, d_8, d_{10}, d_{11}, d_{12}, d_{13}, d_{14}, d_{15}\})$
   $= \{\mathsf{differential}, \mathsf{equations}\}$

B.1 Concepts

A special situation is when the duality of meaning between a set D of documents and a set A of attributes is symmetric: $A \sim D$. It is easily verified that $D \subseteq \mathbf{ComDocs}\ (A)$ and $A \subseteq \mathbf{ComAttr}\ (D)$. When the sets D and A are maximal, then this combination is referred to as a concept:

Definition 3
A *concept* is a pair $(D, A) \in \mathcal{P}(\mathcal{D}) \times \mathcal{P}(\mathcal{A})$ such that D and A are their mutual meaning:

$$\begin{aligned} \mathbf{ComAttr}\ (D) &= A \\ \mathbf{ComDocs}\ (A) &= D \end{aligned}$$

The set of documents for concept c is referred to as $\mathbf{Docs}\ (c)$, the set of attributes associated as $\mathbf{Attr}\ (c)$. Let further $\mathbf{DocsClass}\ (D) = \mathbf{ComDocs}\ (\mathbf{ComAttr}\ (D))$ and $\mathbf{AttrClass}\ (A) = \mathbf{ComAttr}\ (\mathbf{ComDocs}\ (A))$.

Note that with each document set at most one concept can be associated. The same is true for each set of attributes. The lattice has a bottom element, corresponding with overspecification, and a top element corresponding with underspecification:

Lemma 1
$(\mathcal{D}_u, \mathcal{A})$ is a concept. where $\mathcal{D}_u = \mathbf{DocsClass}\ (\emptyset)$, This concept corresponds to overspecification, and is called the top concept ($\top$) of the collection. $(\mathcal{D}, \mathcal{A}_u)$ also is a concept. where $\mathcal{A}_u = \mathbf{AttrClass}\ (\emptyset)$. This concept, the bottom concept ($\bot$) corresponds to underspecification.

Our intention is to apply Formal Concept Analysis in the context of Information Retrieval. Therefore we will be interested what concept can be associated with an initial retrieval result. The minimal concept that can be associated with a set of documents is obtained by:

Lemma 2
Let D be a set of documents, then the pair $(\mathbf{DocsClass}\ (D), \mathbf{ComAttr}\ (D))$ is a concept. This concept is referred to as $\mathbf{Concept}\ (D)$.

So, given a set of documents, the associated concept is readily computed by intersecting all document characterizations, and then uniting all attribute extensions.

This operator is useful when the smallest concept that can be associated with a query result is to be obtained. Analogously, given a set of attributes (a query, say), we might be interested in the smallest concepts containing these attributes:

Lemma 3
Let A be a set of attributes, then the pair $(\mathbf{ComDocs}\ (A), \mathbf{AttrClass}\ (A))$ is a concept. This concept is referred to as $\mathbf{Concept}\ (A)$.

For each document d in the collection the smallest concept containing this document is called the base concept associated with that document, denoted as $\mathbf{Base}\ (d)$. Then obviously: $\mathbf{Base}\ (d) = \mathbf{Concept}\ (\{d\})$. The base concept for an attribute is introduced analogously.

The base document concepts in our running example are given in table III. The concepts are displayed in figure 2. Note that base concepts not necessarily are positioned directly above the bottom element.

Base concepts are special as they may be seen as the concepts introducing the associated document or attribute in the concept lattice. Base concepts, so to say, are the independent components that build the concept lattice. In the next subsection, this is elaborated further.

We will need some insight in the distribution of elementary set operations over the **Concept** -function.

Lemma 4
Let c_1 and c_2 be concepts, then:
$\mathbf{Concept}\ (\mathbf{Docs}\ (c_1) \cup \mathbf{Docs}\ (c_2))$
$= \mathbf{Concept}\ (\mathbf{Attr}\ (c_1) \cap \mathbf{Attr}\ (c_2))$
$\mathbf{Concept}\ (\mathbf{Attr}\ (c_1) \cup \mathbf{Attr}\ (c_2))$
$= \mathbf{Concept}\ (\mathbf{Docs}\ (c_1) \cap \mathbf{Docs}\ (c_2))$

B.2 The join and meet operator

Let $\mathcal{C}$ be the set of all concepts within the collection $\langle \mathcal{D}, \mathcal{A}, \sim \rangle$. Two special concepts are $\top$ and $\bot$. In terms of their associated document sets, the top concept is clearly more general than the bottom concept. In terms of documents, the associated attribute collections are more less restrictive. Focusing on documents, being more specific is formalized by the relation $\subseteq$ over concepts, introduced as follows:

Definition 4
Let c_1 and c_2 be concepts, then:

$$c_1 \subseteq c_2 \equiv \mathbf{Docs}\ (c_1) \subseteq \mathbf{Docs}\ (c_2)$$

The duality between documents and attributes als is present in this relation of specificity:

Lemma 5
$c_1 \subseteq c_2 \Longleftrightarrow \mathbf{Attr}\ (c_1) \supseteq \mathbf{Attr}\ (c_2)$

Note that this relation is reflexive, anti-symmetric and transitive. The relation of specificness is a partial ordering of concepts (which is a direct consequence of the subset relation being a partial order of $\mathcal{P}(\mathcal{D})$). A lowerbound of a set C of concepts is a common subconcept. If there exists a greatest element in the set of lowerbounds of C, then this is called the *greatest lower bound*. Likewise the smallest element in the set of upper bounds is called the *smallest upper bound*.

Lemma 6
Let C be a set of concepts, then:
$\vee(C) = \mathbf{Concept}\ (\cup_{c \in C} \mathbf{Docs}\ (c))$ is the least upperbound of C.
$\wedge(C) = \mathbf{Concept}\ (\cap_{c \in C} \mathbf{Docs}\ (c))$ is the largest lowerbound of C.

Proof: We will prove this lemma only for the upperbound of a class of concepts C. The result immediately follows via the attribute view on the concept specificness relation (see lemma 5) and the observation: $\mathbf{Attr}\ (\vee(C)) = \mathbf{ComAttr}\ (\cup_{c \in C} \mathbf{Docs}\ (c)) = \cap_{c \in C} \mathbf{ComAttr}\ (\mathbf{Docs}\ (c)) = \cap_{c \in C} \mathbf{Attr}\ (c)$. ■

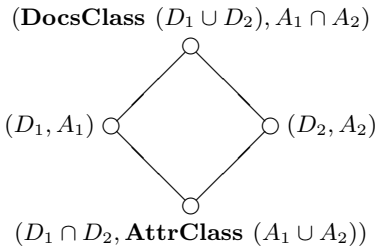

Fig. 1. Binary join and meet

The join operator takes a set of concepts as an argument. However, each join can also be seen as a succession of binary joins.

Lemma 7

1. $\vee(\emptyset) = \emptyset$
2. $\vee(C_1 \cup C_2) = \vee(\{\vee(C_1), \vee(C_2)\})$
3. $c \in C \implies \vee(C) = \vee(C - \{c\}) \vee \{c\}$

Proof:

1. $\vee(\emptyset) =$ **Concept** $(\emptyset_D) = \top$.
2. The concept $\vee(\{\vee(C_1), \vee(C_2)\})$ has associated the attributes

$$\textbf{Attr}\ (\vee(C_1)) \cap \textbf{Attr}\ (\vee(C_2))$$

which can be split as

$$(\cap_{c \in C_1} \textbf{Attr}\ (c)) \cap (\cap_{c \in C_2} \textbf{Attr}\ (c)) = \cap_{c \in C_1 \cup C_2} \textbf{Attr}\ (c)$$

This latter attribute set also is the attribute set of the concept $\vee((C_1 \cup C_2)$.

3. This is a direct consequence of the previous property. ■

The binary join operator is both commutative and associative. As a consequence, each join can be written as a serious of binary joins, where no concept occurs more than once, and the order of joins may be taken arbitrarily. In fact, basic concepts are sufficient:

Lemma 8
Each concept c can be written as $c = \vee(B)$ where B is a set of basic concepts.

Base concepts can thus be seen as a base for the lattice, in the sense that each non-base concept can be derived from those concepts by the join-operator.

IV. Concept lattice generation

This section deals with the problem how to find all concepts of a concept lattice. Key point here is that definition 3 is descriptive, telling what a concept is, not how to construct it. A straightforward method is to generate all possible subsets D from $\mathcal{D}$ and A from $\mathcal{A}$ and check whether (D, A) is a concept. Assuming a number of n documents and m attributes, this method has a complexity of 2^{n+m}.

The number of concepts obviously is bounded by $\max(2^n, 2^m)$. In practice, the number of concepts will be far less. As a consequence, it will be profitable to bound the complexity of the generation algorithm to the actual number (c say) of concepts. The generation of the concept lattice starts with the generation of the base concepts. The complete lattice then is obtained by calculating the closure with respect to the join-operator.

TABLE III
Base concepts

d_i	d_1	d_2	d_3	d_4	d_5	d_6	d_7	d_8	d_9
Concept (d_i)	c_{11}	c_{12}	c_0	c_5	c_1	c_{14}	c_2	c_6	c_{15}
d_i	d_{10}	d_{11}	d_{12}	d_{13}	d_{14}	d_{15}	d_{16}	d_{17}	
Concept (d_i)	c_7	c_4	c_4	c_8	c_9	c_{10}	c_{13}	c_3	

Example 3
Using the context presented in figure II we find the the concept lattice from table IV. This lattice is also shown in figure 2. In this figure the base concepts are bold.

V. Conceptual Relevance Feedback

As seen before, the base concepts of a lattice can be used to find all concepts of that lattice. Especially for large collections, it can be very time consuming to calculate all concepts. For some applications however, we are not interested in the complete lattice, but in a relatively small sublattice.

A. Fingerprint

Given a query result (in the form of a set of documents D) what are the concepts we are interested in? The answer is simple: at least all the concepts that share a document with D. We call this set of concepts the *fingerprint* of D:

Definition 5
Let $D \subseteq \mathcal{D}$ be a query result, then the associated fingerprint is:

$$\textbf{Fp}\ (D) = \{c \in \mathcal{C}\ \mid\ \textbf{Docs}\ (c) \cap D \neq \emptyset\}$$

Note that the set of fingerprints is closed under the join operator. However, in general a fingerprint is *not* a sublattice, since this set of concepts may not be closed under the $\wedge$-operator.

Example 4
For our running example, let D be $\{d_1, d_6\}$. The corresponding fingerprint is $\{c_{11}, c_{12}, c_{14}, c_{20}, c_{21}, c_{22}, c_{23}, c_{24}\}$. Note that $c_{22} \wedge c_{23} = c_{13}$, and thus the fingerprint is not closed under the $\wedge$-operator.

B. Weighting the concepts

Some concepts in the fingerprint are more relevant than others. We introduce the traditional precision/recall figures for concepts:

Definition 6

$$\textbf{Precision}\ (c) = \frac{|\textbf{Docs}\ (c) \cap D|}{|\textbf{Docs}\ (c)|}$$

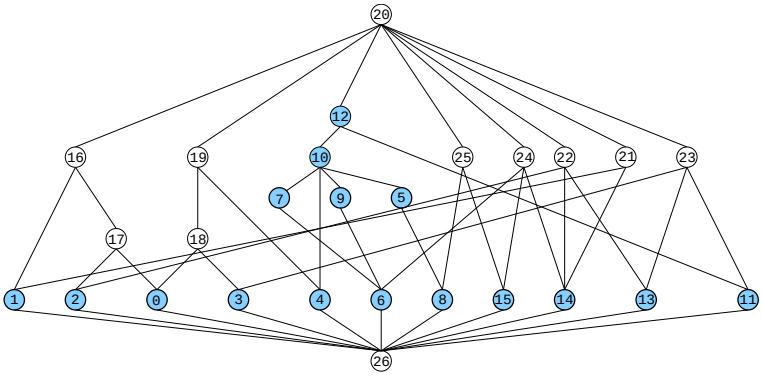

Fig. 2. Concepts in lattice

TABLE IV
CONCEPTS (INTENSION/EXTENSION)

concept	documents	attributes
c_0	d_3	algorithms, application, implementation, theory
c_1	d_5	algorithms, introduction
c_2	d_7	algorithms, implementation, problem
c_3	d_{17}	application, integral, theory
c_4	d_{11}, d_{12}	delay, differential, equations, oscillation, theory
c_5	d_4, d_{13}	differential, equations, partial
c_6	d_8	differential, equations, methods, ordinary, systems
c_7	d_8, d_{10}	differential, equations, ordinary
c_8	d_{13}	differential, equations, nonlinear, partial
c_9	d_8, d_{14}	differential, equations, methods
c_{10}	$d_4, d_8, d_{10}, d_{11}, d_{12}, d_{13}, d_{14}, d_{15}$	differential, equations
c_{11}	d_1	equations, integral
c_{12}	$d_1, d_2, d_4, d_8, d_{10}, d_{11}, d_{12}, d_{13}, d_{14}, d_{15}$	equations
c_{13}	d_{16}	integral, problem
c_{14}	d_6	introduction, problem, systems
c_{15}	d_9	nonlinear, systems
c_{16}	d_3, d_5, d_7	algorithms
c_{17}	d_3, d_7	algorithms, implementation
c_{18}	d_3, d_{17}	application, theory
c_{19}	$d_3, d_{11}, d_{12}, d_{17}$	theory
c_{20}	*all documents*	*no attributes*
c_{21}	d_5, d_6	introduction
c_{22}	d_6, d_7, d_{16}	problem
c_{23}	d_1, d_{16}, d_{17}	integral
c_{24}	d_6, d_8, d_9	systems
c_{25}	d_9, d_{13}	nonlinear
c_{26}	*no documents*	*all attributes*

$$\textbf{Recall}\ (c) = \frac{|\textbf{Docs}\ (c) \cap D|}{|D|}$$

Example 5
Suppose a user formulates the initial query *differential*, and a traditional search engine comes up with $D = \{d_4, d_8, d_{10}, d_{13}\}$. The corresponding fingerprint is $\textbf{Fp}\ (D) = \{c_5, c_6, c_7, c_8, c_9, c_{10}, c_{12}, c_{20}, c_{24}, c_{25}\}$.

Calculating the base concepts yields $\{c_5, c_6, c_7, c_8\}$ (See table III). To get smallest sublattice that contains the fingerprint we have to add the following concepts to the base: c_9 (to get c_9), c_{11} (to get c_{12}) and c_{15} (to get c_{24} and c_{25}). See figure 4 for the sublattice and figure 3 for the corresponding precision/recall graph.

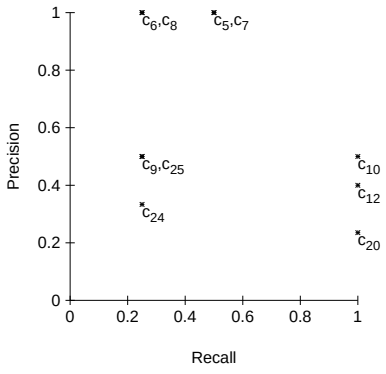

Fig. 3. Precision/Recall graph

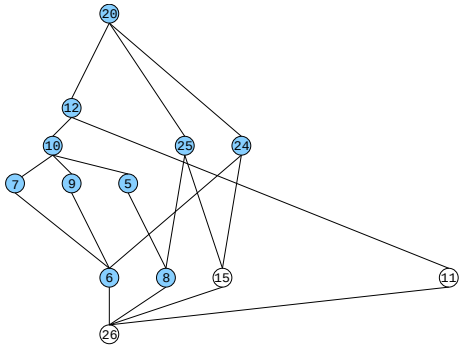

Fig. 4. Sublattice for differential query

C. TREC example

To demonstrate how the system works for a real life situation, the next example is taken from the TREC collection [4] (Associated Press newspaper articles, year 1988).

The collection consists of 78.131 documents[1] (newspaper articles) and has been indexed by 2.779.380 terms (or phrases). This collection comes with a set of topics. We will consider topic 63:

Number: 063
Domain: Science and Technology
Topic: Machine Translation
Description: Document will identify a machine translation system.
Summary: Document will identify a machine translation system.
Narrative: A relevant document will identify a machine translation system which is being developed or marketed in any country. It will identify the developer or vendor, name the system, and identify one or more features of the system.
Concept(s): machine translation system, language, dictionary, font, batch, interactive, process, user interface

According to the TREC results database the following documents are relevant to this query[2]:

document	TREC code
d_{2970}	AP880425-0284
d_{11845}	AP880718-0242
d_{16768}	AP880906-0198

Creating the sublattice out of the corresponding base concepts yields the following concepts: (for a graphics representation see figure 5). Note that the attributes listed in this figure are cumulative: for example, node 3 'inherits' all attributes from node 6. A catching result is that the top node (precision and recall 1) has attributes that are *not* part of the query: the semantical structure concluded that (in the context of this collection) documents about machine translation should contain the word computer.

[1] actually AP88 has slightly more documents, this number of documents remains after cleaning up the collection

[2] Document AP880906-0202, although assessed, is a mangled document: the first few lines are those of AP880906-0198, the rest is a medical article.

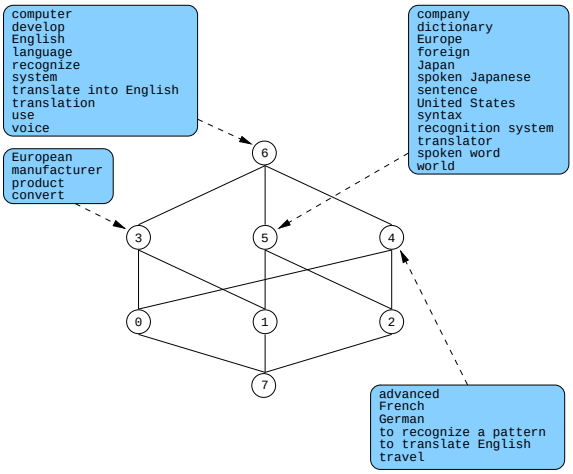

Fig. 5. TREC concept lattice for query 63

concept	documents
c_0	d_{2970}
c_1	d_{11845}
c_2	d_{16768}
c_3	d_{2970}, d_{11845}
c_4	d_{2970}, d_{16768}
c_5	d_{11845}, d_{16768}
c_6	d_{2970}, d_{11845}, d_{16768}
c_{13}	*no documents*

VI. Conclusions

This paper presents an approach to use a concept lattice to interpret a user query for relevance feedback. An algorithm is proposed to generate the lattice. Finally a TREC experiment on query expansion shows promising results. Further research should investigate the automatic application of this approach on larger query sets and compare the results to other feedback mechanisms and conceptual methods [5].

References

[1] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. Addison-Wesley, 1999.

[2] M. W. Berry, S. Dumais, and G. O'Brien. Using linear algebra for intelligent information retrieval. *SIAM Review*, 37(4):573–595, 1995.

[3] P. Bruza and Th. P. van der Weide. Stratified hypermedia structures for information disclosure. *The Computer Journal*, 35(3):208–220, 1992.

[4] D. Harman. Overview of the first TREC conference. In *Proceedings of 16th ACM SIGIR Conference*, 1993.

[5] M. Montes-y Gómez, A. López-López, and A. Gelbukh. Information Retrieval with Conceptual Graph Matching. In *Proceedings of the 11th International Conference and Workshop on Database and Expert Systems Applications, (DEXA-2000)*, Greenwich, England, Sept. 2002. Lecture Notes in Computer Science.

[6] H. A. Proper and P. D. Bruza. What is information discovery about. *Journal of the American Society for Information Science*, 50(9):737–750, July 1999.

[7] R. Wille. Restructuring lattice theory: An approach based on hierarchies of concepts. In I. Rival, editor, *Ordered sets*, pages 445–470. D. Reidel Publishing Company, Dordrecht–Boston, 1982.