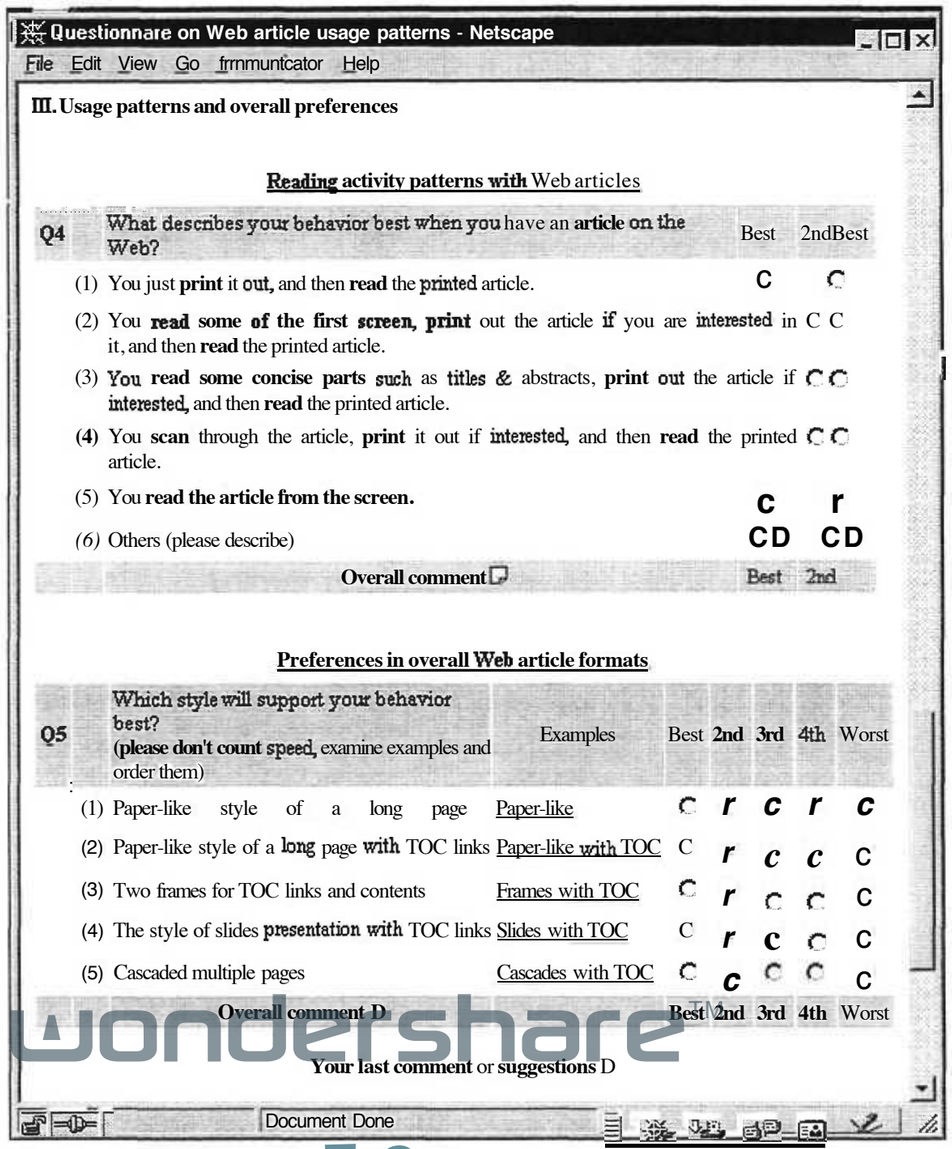
http://www.acm.org/pubs/contents/journals/tochi/1998-5/
[13]Alertbox, 1997. http://www.useit.com/papers/webwriting/writing.html
[14] International Journal of Human-Computer Studies, 1997. http://ijhcs.open.ac.uk
[15] Kristensen, A., "Formsheets and the XML Forms Language," WWW8 Proceedings, 1999. http://www8.org/w8-papers/1c-xml/formsheets/formsheets.html
[16]Rho, Y., Gedeon, TD, "A Link-Click Lifecycle on the screen," APWeb'98 Proceedings, Beijing, 1998. http://www.cse.unsw.edu.au/~yrho/Publications/apweb98/apweb98_C-tiled_main.htm
[17] WWW6 Conference, 1996. http://www.scope.gmd.de/info/www6/technical/paper003/paper3.html
[18]WWW8 Conference, 1999. http ://ww w8. org/fullpaper.html


Questionnare on Web article usage patterns - Netscape

File Edit View Go frnmuntcator Help

**III. Usage patterns and overall preferences**

**Reading activity patterns with** Web articles

| Q4 | What describes your behavior best when you have an **article on the Web?** | Best | 2ndBest |
|---|---|---|---|
| | (1) You just **print** it **out,** and then **read** the printed article. | C | C |
| | (2) You **read some of the first screen, print** out the article **if** you are **interested** in it, and then **read** the printed article. | C | C |
| | (3) **You read some concise parts such** as **titles &** abstracts, **print out** the article if **interested,** and then **read** the printed article. | C | C |
| | **(4)** You **scan** through the article, **print** it out if **interested,** and then **read** the printed article. | C | C |
| | (5) You **read the article from the screen.** | C | r |
| | *(6)* Others (please describe) | CD | CD |
| | **Overall comment** | Best | 2nd |

**Preferences in overall Web article formats**

| Q5 | Which style will support your behavior best? **(please don't count speed,** examine examples and order them) | Examples | Best | 2nd | 3rd | 4th | Worst |
|---|---|---|---|---|---|---|---|
| | (1) Paper-like style of a long page | Paper-like | C | r | c | r | c |
| | (2) Paper-like style of a **long** page **with** TOC links | Paper-like with TOC | C | r | c | c | C |
| | (3) Two frames for TOC links and contents | Frames with TOC | C | r | C | C | C |
| | (4) The style of slides **presentation with** TOC links | Slides with TOC | C | r | c | C | C |
| | (5) Cascaded multiple pages | Cascades with TOC | C | c | C | C | C |
| | **Overall comment D** | | Best | 2nd | 3rd | 4th | Worst |

**Your last comment** or **suggestions** D

Document Done

**Figure 6: A partial screen shot of the Web-based questionnaire [6]**



# DYNAMIC HYPER-LINKING BY QUERYING FOR A FCA-BASED QUERY SYSTEM

*Bernd Groh*
School of Information Technology
Griffith University
PMB 50 Gold Coast Mail Centre
QLD 9726, Australia
*b.groh@gu.edu.au*

*Peter Eklund*
School of Information Technology
Griffith University
PMB 50 Gold Coast Mail Centre
QLD 9726, Australia
*p.eklund@gu.edu.au*

## Abstract

*This paper presents a* ***mechanism*** *for hyper-linking documents by search-terms. Search-terms are selected by the user interactively building a formal concept lattice. In order to explain this interface we give some background to Formal Concept Analysis and an example is developed which illustrates the use of the concept lattice. Selected search-terms are used to create hyper-links, based on term repetition.*

*As the* ***search-terms*** *differ between queries, we need a mechanism from which to dynamically create the target hyper-linked HTML documents. Therefore, documents are stored in a structure which is based on a word-list rather than plain text format. The documents are represented as links between the individual words within the* ***word-list.*** *In so doing the word-list becomes a full-text-retrieval index into each word in each of those documents and therefore provides a good basis for the fast creation of an HTML document set from specific queries by keywords.*

*To have the words in a word-list from which the documents are created also allows easy classification of words which should be hyper-linked within specific HTML documents. Furthermore, both documents and hyper-linking keywords are stored as well in this structure since any word in any document is indexed by the word-list.*

Keywords: Document Databases, WWW **and** Internet.

## 1 Introduction

There have been several developments in automatically generating hyper-linked documents, from hyper-linking by term repetition to semantic approaches based on similarity measures between documents. One approach by Green [7, 8] uses lexical chains [9] to measure the similarity between documents. This mechanism is based on WordNet [5] and generates groups of related words within documents. Those groups have specific meanings and can be related to groups with equivalent or similar meanings in other documents.

Our approach is query-driven and focuses on keywords. We use concept lattices as a visual interface so that the user can navigate to a specific subset of documents with given combinations of search-terms (or keywords) of interest. The hyper-links will be created based on repetition of those terms and will exist only between documents in the specific subset that satisfies the query. The technique to create the concept lattices is called Formal Concept Analysis (FCA) [6].

We structure the paper with a brief introduction to FCA. Next, based on an example of a medical document-set, we demonstrate how to read a concept lattice and how it helps the user to identify documents of interest. As document-sets differ, depending on the terms appearing in the concept lattice and on the selected concepts within the lattice, hyper-links have to be created dynamically.

In Section 4, we present a mechanism to dynamically generate a hyper-linked document-set, wrt to a specific query, based on term repetition. The approach uses a memory structure in which the entire document collection is stored and an algorithm to create the hyper-linked document-subset. The structure is based on a word-list, containing entire words and links between the words. These links, when followed, constitute the document-set.

As our interest lies in complete words which precisely match a pre-defined list of terms, in our example medical terms, fast string searching algorithms based on substrings, such as Suffix Trees, cannot be used. Our approach is explained on a small example in Section 5 and the algorithm is described in Section 6. The last sections conclude the paper and provide an outline of current and future work.

Proceedings of the Fourth Australasian Document Computing Symposium, Coffs Harbour, Australia, December 3, **1999.**

## 2 Formal Concept Analysis

Wille introduced in [10] the metaphor of "Landscape of Knowledge", to describe the processes by which knowledge is explored using formal concept analysis. The processes of knowledge exploration is seen as a dynamic one, in which the computer is used as a medium through which aspects of the knowledge can be investigated.

Central to the idea of formal concept analysis is the understanding that a fundamental unit of thought is a concept. The concept is constituted by its intention and its extension. This understanding had been formalised by starting with a *(formal)* context, $\mathbb{K}$ defined by a triple $(G, \mathrm{M}, I)$ where $G$ and $M$ are sets and $/$ is a binary relation between $G$ and $M$ (i.e. $I \subseteq G \times MgIm$ is read as *"the object g has the attribute m"*. The *(formal)* concepts of $\mathbb{K}$ are the pairs $(A,B)$ with $A \subseteq G$ and $B \subseteq M$ such that $(A,B)$ is maximal with respect to the property $Ax\ B\ CI$. The set $A$ is called the extent and the set $B$ is called the intent of the concept $(A, B)$. The set "under" $\mathcal{B}(\mathbb{K})$ of all concepts of a context $\mathcal{K}$ with order $(A_1, B_1) \leq (A_2, B_2) :\leftrightarrow A_1 \subseteq A_2$ is always a complete lattice, and is called the *concept lattice* of the context $\mathbb{K}$.

In our example of a medical document collection, $G$ is the set of all documents in the collection and $M$ is the set of all MeSH-terms[1] within the MeSH-hierarchy that appear in at least one of the documents. $gIm$ is therefore read *"the document g contains the MeSH-term m "*.

The process of creating a concept lattice must be performed using expert knowledge from the domain from which the data is taken. Often concept lattices are created by hand. This includes the selection of terms to form a lattice and the layout of the lattice. In work by Cole [3, 4, 2] concept lattices are created by the manipulation of a view of a taxonomy of terms, i.e. the medical taxonomy of the MeSH-hierarchy. Manipulation of the view both defines the concept lattice and conditions its layout. This project was the motivation for hyper-linking documents using a concept lattice as the basis for a query interface.

## 3 Concept Lattices Over Medical Documents

Figure 1 shows a concept lattice for some given attributes out of a medical taxonomy, the MeSH-hierarchy. The attributes or keywords within a document are placed as labels above a node, which represents a concept in the formal context. Documents are placed as labels, containing the number of documents with those specific attributes, or keywords, below the node. The

[1] MeSH is the Medical subjects Headings [1].

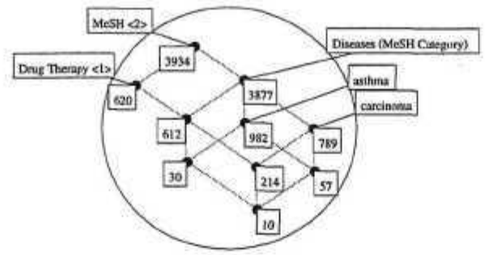



Figure 1: Lattice diagram.

lattice is best read bottom up. The node with 214 documents associated to it, for example, has the attributes "carcinoma", "Drug Therapy <1>"[2], "Diseases (MeSH Category)" and "MeSH <2>". As the documents are patient records, this means that 214 patients have "carcinoma", which falls under "Diseases (MeSH Category)" and is a term of the MeSH-Hierarchy, classified by "MeSH <2>" and are undergoing a "Drug Therapy <1>", which is also a term of the MeSH-Hierarchy. The node with 789 documents associated with it, has all those attributes, except "Drug Therapy <1>". This, in conjunction with the node with 214 documents, can be read as 789 patients do have "carcinoma" of which 214 are undergoing a "Drug Therapy <1>". The node with 10 documents also has the attribute "asthma", which can be read as 214 patients who have "carcinoma" and are undergoing a "Drug Therapy <1>" also have "asthma". 57 patients have "carcinoma" and "asthma" and are not undergoing a "Drug Therapy <1>".

The way in which the attributes are selected is done in an interactive way by the user, where she/he can select her/his area of interest, see Cole [3, 4, 2]. We are interested in the document sets on specific nodes. Each node, for every possible attribute combination, contains a specific number of documents, containing exactly those attributes or keywords.

## 4 Dynamic Hyper-linking Medical Documents

The process by which MeSH-terms are selected by the user is a metaphor for a query theme. This theme can then be used as the basis for generating a concept lattice showing how the documents distribute across combinations of terms in the theme. It would be useful, on the basis of creating a theme

[2] Suffixes such as <I> on a MeSH-term indicate that the term has several occurrences, in this case <I> indicates that this is the first occurrence in the MeSH-hierarchy for this term.



(a conceptual scale in the FCA literature), if we could use the theme to dynamically hyper-link the document collection. The mechanism for hyper-linking the documents has been applied to the same document collection as mentioned above. An example document is shown in Figure 2.

```
{{ Problem List )
 { 0. Small cell ca ca left lung, 8 cycles chemotherapy 1990, plus
   radiotherapy 1991. 1. Pulmonary embolus. 2. Glaucoma.
   3. Peptic ulcer. 4. Cholecystectomy. E. Appendicectomy.
   6. Oophorectomy. 7. Right sided pneumonia and neutropaenia. »
<{ Discharge Treatment >
 { Coloxyl with senna 3 taba bd, Ventolin 90 no 4 hrly prn,
   Ranitidine 300mg bd, Mylanta 20 mls tds prn,
   Nifedipine 10mg tds, Panadeine forte 1-2 tabs 4 hrly pra,
   Dipiverine Hydrochloride .1% 2 dpa bd both eyes. »
{{ Information to Patient >
 < Patient aware of diagnosis and limited prognosis.
   Knows to present to LMO or RAH with any problem). )>
{< Summary of Admission }
 < 68 year old woman, well known to S2. Discharged one week ago.
   Day altar discharge, developed increasing 508 with
   yellow/white sputum production. Felt unveil, but denies
   rigors or chills. Using Ventolin regularly with no
   improvement. Transferred by ambulance to RAH. )>
{< Examination }
 { ; mildly tachypnoeac, RE 30, not cyanosed, looks unwell,
   febrile 38.9, dehydrated, HR 120/reg, BP 140/90, J7P NR,
   HS dual * nil. no ankle availing, peripheral pulses all
   present, TML, PN dull left base, BS vesicular, reduced
   at left base, crackles right anterior chest. Abdominal
   and neurological examination unremarkable. >)
{{ Investigation >
 < »
{{ Progress )
 { a steady improvement made. Freely mobile around the ward
   without oxygen on discharge. >)
{{Follow Up }
 ( Cheat Clinic/Or- Holmes LMO to perform MBA20 prior to OPD
   appt. >)
{{ Copies >
 { lmo,file,ur. »
```

Figure 2: Medical document as used.

If we are interested in documents that contain both the terms "carcinoma" and "Drug Therapy <1>" for example, we select the node in Figure 1 with 214 documents. The first document in this set of 214 is shown in Figure 3.

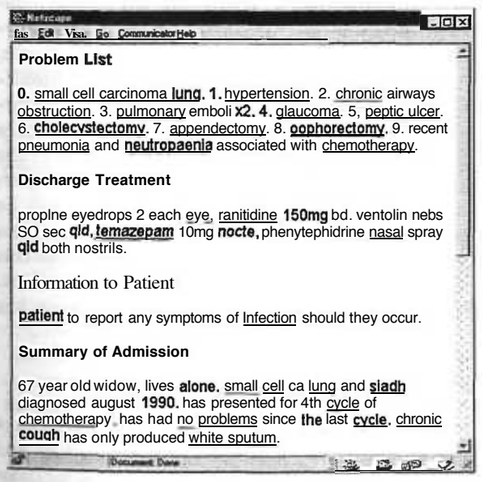



Figure 3: First of 214 generated HTML documents.

Every term that is in the hierarchy (in the example the MeSH-Hierarchy) is either linked to the next document in the retrieved document set, containing that term, or is boldfaced to indicate there is no other document in the retrieved set that contains that term. In Figure 3, all terms that are contained by at least one other document in the retrieved set are linked to the next document with that specific term occurrence. The last occurrence of that term in the last document, containing it, refers back to the first occurrence of that term. The search terms "carcinoma" and "Drug Therapy <1>" appear in every document and it is therefore possible to view every document in the retrieved set by following those links. The problem is then to produce a fast and efficient way to generate these hyper-linked HTML documents.

## 5 Efficiently Representing Documents

As the retrieved document sets differ each time, depending on the query (or the conceptual scale) and the hyper-links within the documents differ each time, it is not desirable to create all possible HTML documents in advance. Therefore, the documents have to be stored in a way that is as easy and efficient as possible to create specific HTML documents on demand. As we work with text documents, where attributes are words (or phrases), the easiest thing is to store documents as linked lists of words. The first thing to do is to create a list of all the words in all the documents — the index. Given we have the following three documents:

```
The house is blue.

My house is nice.

My pen is blue.
```

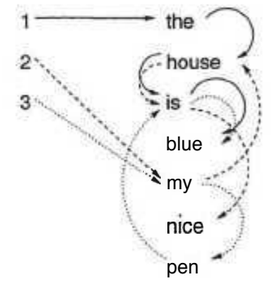



Figure 4: Linked list of the 3 documents.

Figure 4 shows the representation of the linked lists for the three documents. How should this be stored physically?

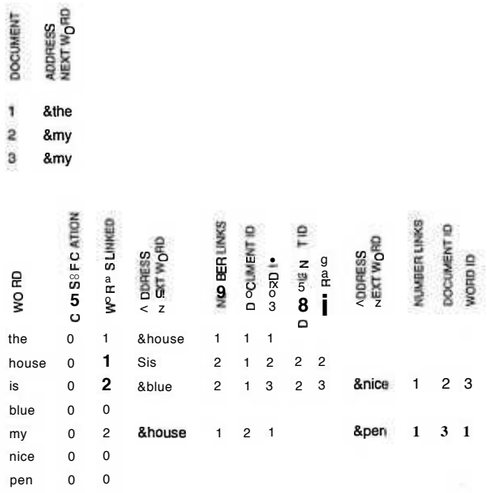



Figure 5: Memory structure of the linked documents.

Figure 5 shows how the data is stored in memory The figure is a table representation for an easier understanding of the structure. Each document is stored with just the address of the first word. Document 1, for example, points to "the", which is the first word of document 1. Of course other information such as filename can be stored as well, but there is just one pointer into the document itself. For every word is stored:

1. an id which represents the classification of the word or its place in the hierarchy;
2. the number of links to a following word;
3. for each link to a following word;
   (a) the address of the next word;
   (b) the number of links to that specific word;
   (c) for each link to that specific word;
      i. the document number;
      ii. the word number.

From this data structure the documents can be reproduced. In Figure 5 document 1 points to "the". From "the", there exists just one possible link, to "house", and to word 1 in document 1 (we count "the" as 0). From "house" there is one possible link, to "is". There are two possible links and one of them is to word 2 in document 1. From "is" there are two possible links, to "blue" and to "nice". One of the links to "blue" is to word 3 in document 1, the next word therefore is "blue". From "blue" there are no further links and the end of the document is reached.

The restored words "the", "house", "is", "blue" constitute document 1. In the implementation, the difference between upper- and lowercase as well as special characters and set signs have been considered and are stored in additional entries, but we will not talk about that in this paper. As we are creating HTML documents, we are simply storing information to reproduce the text, formatting may be lost. In our testing we use plain text-documents. Nevertheless this mechanism is extendible to accommodate further text-options that can be represented by HTML.

## 6 Building the HTML Documents

After the user has made a selection of the document-subset of interest, each document in the subset will be regenerated from the memory structure.

Figure 6 shows the algorithm, which loops over all documents in the subset. For each document the algorithms run through the word-list, collecting all the words within the document. If the classification id is non-zero, this is a term of the hierarchy or the beginning of a phrase within the hierarchy, in our example the MeSH-hierarchy. If the classification id is zero, the word is output to the current HTML document and the algorithm can continue.

When the current word is classified, the algorithm tests whether it is a one-word term or the beginning of a phrase within the MeSH-hierarchy. Words collect, until the phrase is maximal - long phrases have higher priority than short phrases, short phrases a higher priority than single words. The maximal phrase in the MeSH-hierarchy constitutes the term. If the words "Small cell carcinoma" collected through the links, for example, "Small" will not be used, even if it is a MeSH-term. Next, the algorithm looks up the document-word pairs that contain the first word occurrence (or phrase occurrence) in the next document in the specified subset. That is the position the hyper-link is construct to. Now the hyper-link will be output into the HTML document, where the name of the hyper-link is the word-number, so that this link can be linked directly from another document. This is repeated, until the end of the document has been reached.

## 7 Future Work

We plan to investigate different implementations of the algorithm to create the memory structure and examine differences in memory management/time complexity issues. We also want to investigate different implementations of physically storing the links and see whether we can achieve better memory management without changes to time



Inputs
  Let $G$ be the array of the document collection
  Let *AID* be a set of document ids
  Let $B$ be a set of MeSH-terms
Outputs
  Let *HTA* be an array of HTML documents
Variables
  Let *next* be the address of a word within the word-list
  Let *term* be a MeSH-term
  Let *dn* be the id of a document
  Let *wn* be the position of a word within a document
  Let *wc* be the counter of the words within the current document
Algorithm

```
FOR EACH (aid ∈ AID)
  PrintHeader(hta[aid], g[aid])
  wc = 0
  next=g[aid]. First Address
  WHILE (next)
    IF (next →Classification = 0) THEN
      Print(hta[aid], next →Word)
    ELSE
      GetMeshTerm(term, next, wc)
      IF (GetNextOccurrence(dn, wn, term) = 0)
        PrintNoLink(hta[aid], term)
      ELSE
        IF (term ∈ B) THEN
          PrintActiveLink(hta[aid],
                wc, dn, wn, term)
        ELSE
          PrintPassiveLink(hta[aid],
                wc, dn, wn, term)
        END IF
      END IF
    END IF
    next = GetNextWord(aid, wc)
  END WHILE
  PrintFooter(hta [aid])
END FOR
```

Figure 6: Algorithm to create the HTML documents from the memory structure.

complexity. In doing so, we will also consider compression techniques.

What takes up most memory are the document/word pairs and the average/max length of those depend on the number of documents in the collection and the number of words within a single document. Given different cases we will compare the presented structure with other techniques, inverted file indexing for example, to see how it compares with other techniques we could use for this purpose. Finally, we want to build a tool that is general enough for most document-sets. It should automatically create the memory structure out of the document collection, allow to plug-in any hierarchy, on which the hyperlinks are based, and that then creates the HTML documents from any query.

## 8 Conclusion

A mechanism has been presented that allows a user to define terms (attributes) and use them to generate hyper-linked HTML documents. It has been shown that a structure to store these documents results in easy and efficient access to the documents.

Apart from fast composition of HTML documents this structure also provides a full-text retrieval index into each word in each document. Furthermore, it allows easy classification of the single words or phrases and an easy embedding of a hierarchy over those words or phrases. It also allows easy access to information about the documents, such as word-count, number of a specific link or a specific word combination. Finally, the resulting structure might require less memory than the plain documents itself. This, of course, is dependent on the document collection, but it was valid for our test-set. Once this structure has been created, it is no longer necessary to keep the original documents, as the plain text documents can be fully restored from that structure.

To compare the memory needs: The 4,000 documents in our test-set took up storage space of 7.23 MB. The newly created structure consumes 5.12 MB. Given that reduction of storage space was not the aim of this work it is nevertheless an appreciated side-effect. Just to mention the compression results we got using gzip$^{TM}$. The Zip file of the 4,000 documents, using maximum compression, has a size of 4.13 MB, the size of the Zip file of our structure is 3.88 MB. Of course, it would be possible to use further compression techniques, but our aim was fast creation of HTML documents and we therefore have decided not to use any. In the current version we work with full byte-lengths, even when we need only a few bits because access to full bytes is easier to implement. With most compression techniques it would be difficult to apply techniques, such as binary search, to reduce the time for creation of HTML documents.

## References

[1] *1998 MeSH, Annotated Alphabetic List*. National Technical Information Service, U.S. Department of Commerce, Springfield, VA 22161, 1998.

[2] Richard Cole and Peter Eklund. Scalability in formal concept analysis. *Computational Intelligence: An International Journal*, Volume 15, Number 1, pages 11-27, February 1999.

[3] Richard Cole, Peter Eklund and Bernd Groh. Dealing with large contexts in formal concept analysis. In *Second International Symposium on Knowledge Retrieval, Use and Storage for Efficiency,* pages 151-164, Vancouver, B.C., Canada, August 1997.

[4] Richard Cole, Peter Eklund and Don Walker. Using conceptual scaling in formal concept analysis for knowledge and data discovery in medical texts. In *Second Pacific Asian Conference on Knowledge Discovery and Data Mining,* 1998.

[5] Christiane Fellbaum (editor). *WordNet: An Electronic Lexical Database*. MIT Press, 1998.

[6] Bernhard Ganter and Rudolf Wille. *Formal Concept Analysis: Mathematical Foundations*. Springer-Verlag, 1999.

[7] Stephen Green. *Automatically generating hypertext by computing semantic similarity*. University of Toronto, Canada, 1997.

[8] Stephen Green. Automated link generation: Can we do better than term repitition? In *Proceedings of the Seventh International World Wide Web Conference,* pages 75-84, Brisbane, Australia, April 1998.

[9] Jane Morris and Graeme Hirst. Lexical cohesion computed by thesaural relations as an indicator of the structure of text. *Computational Linguistics,* Volume 17, Number 1, pages 21-48, 1991.

[10] Rudolf Wille. Landscapes of knowledge: A pragmatic paradigm for knowledge processing. In *Second International Symposium on Knowledge Retrieval, Use and Storage for Efficiency,* pages 2-13, Vancouver, B.C., Canada, August 1997.





# TML: A Thesaural Markup Language

*Maria Lee*
Mathematical & Information Sciences
CSIRO
Locked Bag 17, North Ryde 1670
Australia
Maria.Lee@cmis.csiro.au

*Stewart Baillie*
Mathematical & Information Sciences
CSIRO
723 Swanston Street, Carlton 3053
Australia
Stewart.Baillie@cmis.csiro.au

*Jon Dell 'Oro*
Mathematical & Information Sciences
CSIRO
723 Swanston Street, Carlton 3053
Australia
Jon.Delloro@cmis.csiro.au

## 1 Abstract

*Thesauri are used to provide controlled vocabularies for resource classification. Their use can greatly assist document discovery because thesauri man date a consistent shared terminology for describing documents. A particular thesaurus classifies documents according to an information community's needs. As a result, there are many different thesaural* ***schemas.*** *This has led to a proliferation of schema-specific thesaural systems. In our research, we exploit schematic regularities to design a generic thesaural ontology and specify it as a markup language. The language provides a common representational framework in which to encode the idiosyncrasies of specific thesauri. This approach has several advantages: it offers consistent syntax and semantics in which to express thesauri; it allows general purpose thesaural applications to leverage many thesauri; and it supports a single thesaural user interface by which information communities can consistently organise, store and retrieve electronic documents.*

Keywords: Electronic Documents, Metadata, Ontology, Thesaurus, XML

## 2 Introduction

Many problems common to electronic document systems are often not new, but well-known problems occurring in a new medium. In a search for solutions to problems in the electronic medium, we can often learn from the experience of traditional media. This is true for resource discovery in large electronic information repositories. The solutions offered by search engines have evolved rapidly to fill a need for resource discovery in the electronic storage medium. But, in a managed information environment, their free text search approach can be a poor substitute to thesaurally organised metadata approaches.

The use of metadata search can complement and enrich the text matching approach of search engines. Metadata is data which describes data. It provides a conceptual description of a resource's content, context, and function. The keywords list at the head of this paper is an example of **metadata—** it describes something about the document content. In document management systems, metadata is often used to index a document by describing what it is about and its catalogue detail. However, this metadata content, when used for search, can run into a similar problem to that of document content: it lacks a consistent shared vocabulary. A traditional solution to this problem is to use a thesaurus to control metadata content.

In the terminology of the record keeping community, a thesaurus is a fixed vocabulary of approved and unapproved terms, their functions and meanings, and their inter-term relationships. A thesaurus can provide *accuracy* of description through explicit classification by approved terms; *consistency* through controlled terminologies; and *efficiency* in retrieval through the use of the right terminologies [Lancaster 1972].

A thesaurus is valuable if its vocabulary acts as a *lingua franca* that reflects the culture of a user community and purposes the information repository schema. This often means that a different thesaurus is necessary for each user community. The result, in the electronic storage medium to date, has been many incompatible thesaural applications each one designed about its particular thesaurus. In our research we have sought a generic ontology in which to represent the idiosyncrasies of these many specific thesauri. This would allow a single application to work with many different thesauri. In this paper, we describe this ontology, a markup language used to express it, and introduce a general purpose Thesaural Explorer application based upon them.

## 3 Generic Thesaural Ontology

An ontology, in computer science, has come to denote an explicitly specified conceptualisation of part of the world. In software, an ontology is implemented as a data structure. What distinguishes the ontology from the data structure is semantics: that it talks about something in the world. An ontology provides users with a representation which is essential to effective communication and coordination.

Proceedings of the 4th Australasian Document Computing Symposium,
Coffs Harbour, Australia,
December 3, 1999.