Document Retrieval for Email Search and Discovery using Formal Concept Analysis

Richard Cole[1] and Peter Eklund[1] and Gerd Stumme[2]

[1] School of Information Technology and Electrical Engineering
The University of Queensland, St. Lucia QLD 4072, Australia

[2] Institute of Applied Informatics and Formal Description Methods (AIFB)
University of Karlsruhe
D–76128 Karlsruhe, Germany

Abstract. This paper discusses an document discovery tool based on conceptual clustering by formal concept analysis. The program allows users to navigate email using a visual lattice metaphor rather than a tree. It implements a virtual file structure over email where files and entire directories can appear in multiple positions. The content and shape of the lattice formed by the conceptual ontology can assist in email discovery. The system described provides more flexibility in retrieving stored emails than what is normally available in email clients. The paper discusses how conceptual ontologies can leverage traditional document retrieval systems and aid knowledge discovery in document collections.

1 Introduction

Traditional email management systems provide the user with a tree structure of folders which is an analog of the directories provided in traditional file systems and folders in filing cabinets. The advantage of such a system is that, by analogy, any user familiar with lexical ordering and filing cabinets is also familiar with a file-system. The disadvantage is that documents are placed in a single folder only. To be retrieved, the user must determine the correct folder, and then locate the document within that folder.

Placing an email in a folder while using an email-reading program is an act of classification. Its purpose is to aid in the later retrieval of that email. It makes sense to classify an email document according to multiple orthogonal criteria, for example according to the author, the recipients, the content of the document, the type of the email (incoming or outgoing, read/unread etc) and the context of the email. Traditional file handling systems are often accompanied by a text search capabilities based on keyword retrieval via an index or via *grep* style stream search. As well as classifying email we are also required to search it.

For example, consider looking for an email from *Carol Goble* about the program committee of the WWW 11 conference. Neither the folder of all emails from *Carol Goble* nor the folder of all emails about WWW 11 are good places to look, as they contain too many irrelevant emails. Rather, the intersection of these two folders is likely to recover your email. By encouraging classification and the creation of a large number of folders, rather than discouraging it by forcing the user to select a single folder, one increases the chance that a folder will correspond to a user's query. This is the aim of our HierMail system[1]. The system also provides the facility to create folders whose content is decided by information retrieval techniques like keyword search and machine learning classifiers developed for automatic classification of texts [4,2]. Such folders can be dynamically created and combined with existing folders to aid in email search. HierMail associates email documents with descriptors arranged in a multiple inheritance hierarchy (ontology) using an inverted file index. Descriptors replace the concept of folders. More than one descriptor can (and should) be assigned to an email.

In the example earlier, the client need not decide where on the file-system to store the email from *Carole Goble* about the program committee of WWW 11. He simply assigns all relevant descriptors to the email. HierMail displays folder contents (i. e., sets of emails related to a selection of descriptors) as *conceptual views* via concept lattices.

Concept lattices are defined in the mathematical theory of *Formal Concept Analysis* [13]. Formal Concept Analysis is an unsupervised KDD technique which allows for conceptual clustering [18,3,16]. Cluster Analysis in general comprises a set of unsupervised machine learning techniques which split sets of objects into clusters (subsets) such that objects within a cluster are as similar as possible while objects from different clusters are as different as possible. Conceptual Clustering techniques additionally aim at determining not only clusters but to provide at the same time intensional descriptions of these extensions [15].

[1] see *http://www.hiermail.com*

In our case, the clusters are considered as the content of an email folder, while the intensional description provides means for locating and retrieving it within the multiple cluster hierarchy (i. e., the concept lattice). The concept lattice is derived from the binary relation which assigns descriptors like ‘conferences’, ‘goble’, or ‘organization’ to emails. The exploitation of the resulting model (the concept lattice) can then be used for discovery of new knowledge out of the emails (in particular discovering interesting associations between email content), but can also be exploited for information retrieval purposes.

There are a number of related approaches. For instance, the concept of a *virtual folder* was introduced in a program called View Mail (VM) [14]. A virtual folder is a collection of email documents retrieved in response to a query. The virtual folder concept has more recently been popularized by a number of open-source projects [17]. Other search tools for email are also available, 80-20 is one such product.[2]

HierMail differs from those systems in the understanding of the underlying structure – via formal concept analysis – and in the details of its implementation. It therefore extends the virtual or associative file system idea to email management and discovery. HierMail is designed with an open architecture to accommodate information extraction techniques for text, images or multimedia content. The regular expressions used in HierMail to associate emails to labels in an ontology can therefore be swapped for neural networks or other classifiers

[2] see *http://www.80-20.com*

that identify features in the source documents [1, 2, 4]. Our approach is thus complementary to these machine learning techniques.

In our approach, emails can be stored any place, in a relational database, in a legacy file hierarchy or on different distributed file systems later federated according to the context of an inquiry. In HierMail, email retrieval is independent of the physical organization of the file-system.

In this paper, we profile HierMail (previously referred to in various stages of development as Cem, ECA or Warp9) that follows from earlier work in medical document retrieval reported in [9, 6]. We consolidate elements of work presented previously in various conference papers [10, 8, 7] in this treatment. In the next section, we will give an introduction to the basic notions of Formal Concept Analysis. In Section 3, we describe the mathematical structures employed in HierMail. Requirements for their maintenance are discussed in Section 4. We describe how they are fulfilled by our implementation in Section 5. Section 6 illustrates HierMail by an application scenario, and Section 7 concludes the article.

2 Mathematical Structure

Formal Concept Analysis (FCA) [13] has a long history as a technique for data analysis. Two software tools, Toscana [21] and Anaconda embody a standard methodology for data-analysis based on FCA. Recently, a Java-based open-source version of Toscana, called ToscanaJ has also been developed[3]. Following

[3] see *http://toscanaj.sourceforget.net*

	Conference Related	Conferences with papers	Conference Organisation	Program Committee
WWW 11 submission	×	×		
hi!				
ICCS location, PKDD reference	×		×	
Re: WWW 11 submission	×	×		
ICCS room reservation	×		×	
ICCS camera ready/thanks to PC	×	×	×	×
Un petit garcon !!!				
Re: Summary of ICCS 2001 Arrangements	×		×	×
Re: hi!				
Invitation to IIP ’02 PC	×		×	×

Fig. 1. Descriptors assigned to emails in a formal context.

the FCA methodology, data is organized as a table in a relational database and is modeled mathematically as a formal context.

Definition 1. *A (formal) context is a triple* $\mathbb{K} := (G, M, I)$ *where* G *and* M *are sets and* I *is a relation between* G *and* M*. The elements of* G *and* M *are called objects and attributes, respectively, and* $(g, m) \in I$ *is read "object* g *has attribute* m*".*

In the relational database table there is one row for each object and one column for each (Boolean) attribute.

We illustrate the definition by an example (see Fig. 1). The set G of objects consists of 10 emails, the set M of attributes of four descriptors. The binary relation I indicates which descriptors are assigned to each email.

Definition 2. *For $A \subseteq G$, we define $A' := \{m \in M \mid \forall g \in A\colon (g,m) \in I\}$. Dually, for $B \subseteq M$, we define $B' := \{g \in G \mid \forall m \in B\colon (g,m) \in I\}$. Now a (formal) concept is a pair (A,B) such that $A \subseteq G$, $B \subseteq M$, $A' = B$ and $B' = A$. (This is equivalent to A and B being maximal with $A \times B \subseteq I$.) The set A is called the extent and the set B the intent of the concept (A,B).*

Resulting from the context in Fig. 1 there are in total six formal concepts. One of these concepts has for instance intent equal to the set of descriptors {Conference Related, Program Committee}, and extent equal to the set of documents {Re: Summary of ICCS 2001 arrangements, Invitation to IIP '02 PC, ICCS camera ready/thanks to PC.}

Definition 3. *Each formal context $\mathbb{K}$ gives rise to a conceptual hierarchy, called concept lattice of $\mathbb{K}$, and denoted by $\underline{\mathfrak{B}}(\mathbb{K})$. The hierarchical subconcept–super-concept–relation on the concepts is formalized by*

$$(A,B) \leq (C,D) :\Longleftrightarrow A \subseteq C \quad (\Longleftrightarrow B \supseteq D) \ .$$

The concept lattice of the context of Fig. 1 is shown in Fig. 2. Each node stands for a formal concept, and the subconcept-superconcept hierarchy can be read by following ascending paths of straight line segments. The intent of each concept is given by all attributes reachable from that concept by ascending paths of straight line segments, and its extent is given by all objects reachable by descending paths of straight line segments. The concept mentioned above is the one labeled by 'Program Committee'. In the diagram, one can find above it the descriptors 'Conference Related' and 'Program Committee', hence its intent.

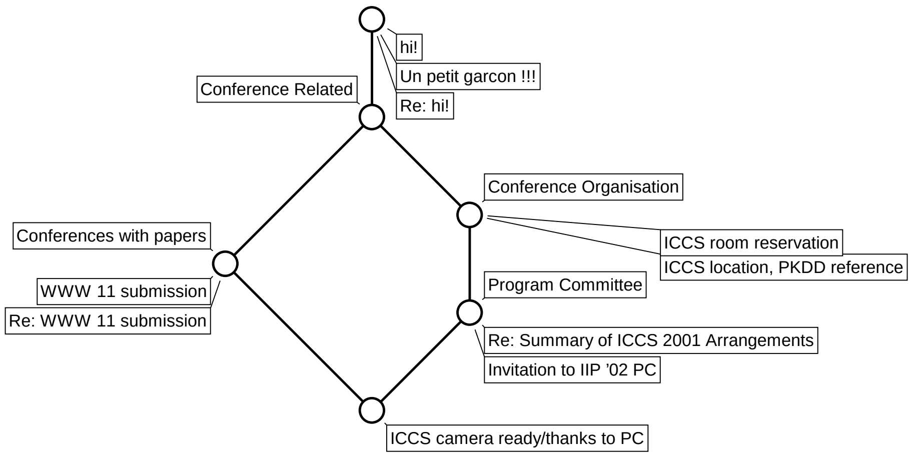


Fig. 2. Concept lattice of the formal context in Fig. 1

Its extent are the three emails found below: namely the emails with the subjects "Re: Summary of ICCS 2001 Arrangements", "Invitation to IIP '02 PC" and "ICCS camera ready/thanks to PC". In the diagram one can also see that two of the three emails have exactly the descriptors of the intent, while the third email additionally has the descriptor 'Conference with papers' and so appears at the base of the lattice.

3 Mathematical Structures for Conceptually Managing Emails

Mixed initiative is a process from human-computer interaction involving humans and machines sharing tasks best suited to their individual abilities [12]. The computer performs computationally intensive tasks and prompts human-clients to intervene when either the machine is unsuited to make a decision or resource limitations demand human intervention. Mixed initiative requires that the client be able to determine trade-offs between different descriptors and alter search constraints to locate objects that satisfy an information requirement. Nesting and zooming [21, 22] are two well established techniques used in FCA (see Figures 9 and 10). Together these techniques allow users to wander around in a conceptual landscape [23] attempting to find concepts that satisfy their constraints.

Given this background we now describe the system on a structural level; we abstract from implementation details which are discussed in Section 4. We distinguish three fundamental structures:

1. a *formal context* that assigns to each email a set of descriptors;
2. a *hierarchy* over the set of descriptors in order to define more general descriptors;
3. a mechanism for creating *conceptual scales* used as a graphical interface for email retrieval.

3.1 Assigning Descriptors to Emails

In HierMail, we use a *formal context* (G, M, I) for storing email and assigning descriptors. The set G contains all emails stored in the system, the set M contains all descriptors. For the moment, we consider M to be unstructured — in the next subsection we will introduce a hierarchy over it.

The relation I indicates the assignment of descriptors to emails. In the example given in the introduction, the client might want to assign all the descriptors 'goble, 'www11', 'program_committee', 'conferences', 'www', and 'organization' to a new email. The incidence relation is generated in a semi-automatic process: (i) a string-search algorithm recognizes words within sections of an email and suggests relations between email attributes, (ii) the client may accept the suggestion of the string-search algorithm or otherwise modify it, and (iii) the client may attach his own attributes to the email. In Section 4, we discuss how the user is supported in this assignment. For the moment, we suppose that the relation between a document and its attributes is already provided.

A major interest in the information retrieval learning community is how this assignment of labels to emails can occur automatically. In HierMail, we simplify this problem by assuming these assignments are given by the user. It is however possible using the system to construct complex regular expressions to classify incoming email. However, the quality of the label assignment task is a critical feature of the process we described.

Instead of a tree of disjoint folders and sub-folders, we consider the concept lattice $\underline{\mathfrak{B}}(G, M, I)$ as the navigation space. The formal concepts replace the folders. In particular, this means that the same email can appear in different concepts and therefore in different folders. The most general concept contains all emails in the collection. The deeper the user moves into the multiple inheritance hierarchy, the more specific are the concepts, and the fewer emails they contain.

3.2 Hierarchies of Descriptors

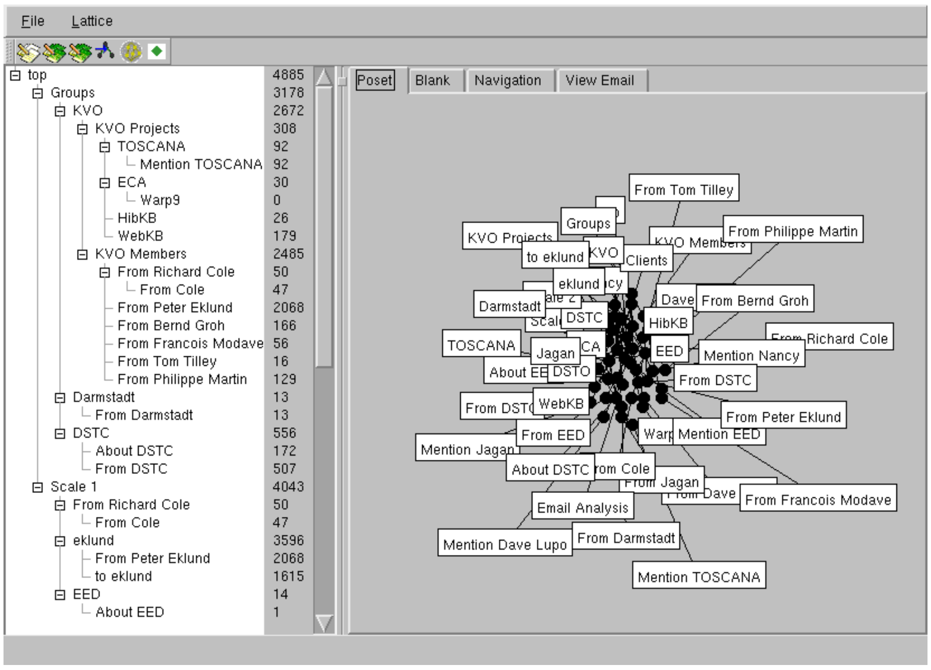
To support the semi-automatic assignment of descriptors to emails, we provide the set M of descriptors with a partial order $\leq$. For this *subsumption hierarchy*, we assume that the following *compatibility condition* holds:

$$\forall g \in G,\ m,n \in M\colon \quad (g,m) \in I,\ m \leq n \ \Rightarrow\ (g,n) \in I \qquad (\ddagger)$$

i.e., the assignment of descriptors respects inheritance along the hierarchy. Hence, when assigning descriptors to emails, it is sufficient to assign the most specific descriptors only. The more general descriptors are automatically added.

For instance, the hierarchy may contain the fact that 'www' is a more specific descriptor than 'conferences', and that 'www11' is more specific than 'www' (i. e., 'www 11'≤'www'≤'conferences'). An email concerning the creation of a paper for the WWW 11 conference is assigned to 'www 11' only (and possibly also to some additional descriptors like 'cole', 'eklund' and 'stumme'). When the client wants to retrieve this email, she is not required to recall the complete pathname. Instead, the email also appears under the more general descriptor 'conferences'. If 'conferences' provides too large a list of email, the client can refine the search, by choosing a sub-term like 'www', or adding a new descriptor, for instance 'cole'.

Notice that even though we impose no specific structure on the subsumption hierarchy $(M, \leq)$, it naturally splits three ways. One relates the contents of the emails, e. g., if an email is related to 'conference' (or not) or classified to 'organization' etc. A second relates to the sender or receiver of the email. The third describes aspects of the emailing process (for instance if it is inbound or outbound mail, or if it is read and answered). An example of a hierarchy is given in Fig. 3. The partially ordered set is displayed both in the style of a folding editor and as a connected graph. The hierarchy displayed in Fig. 3 is a forest



```
⊟ top                           4885
  ⊟ Groups                      3178
    ⊟ KVO                       2672
      ⊟ KVO Projects            308
        ⊟ TOSCANA               92
          └ Mention TOSCANA     92
        ⊟ ECA                   30
          └ Warp9               0
        - HibKB                 26
        └ WebKB                 179
      ⊟ KVO Members             2485
        ⊟ From Richard Cole     50
          └ From Cole           47
        - From Peter Eklund     2068
        - From Bernd Groh       166
        - From Francois Modave  56
        - From Tom Tilley       16
        └ From Philippe Martin  129
    ⊟ Darmstadt                 13
      └ From Darmstadt          13
    ⊟ DSTC                      556
      - About DSTC              172
      └ From DSTC               507
  ⊟ Scale 1                     4043
    ⊟ From Richard Cole         50
      └ From Cole               47
    ⊟ eklund                    3596
      - From Peter Eklund       2068
      └ to eklund               1615
    ⊟ EED                       14
      └ About EED               1
```

Fig. 3. Partially ordered set of descriptors: as a folding editor and connected graph.

(i. e., a union of trees), but the resulting concept lattice — used as the search space — is by no means a forest: it can be any lattice. Consider for example the concept generated by the conjunction of the two descriptors 'WWW 11' and 'conference organization'. It will have at least two incomparable super-concepts, namely the one generated by the descriptor 'WWW 11' and the one generated by the descriptor 'conference organization'. In general, all we know is that the resulting concept lattice is embedded as a join-semi-lattice in the lattice of all order ideals (i. e., all subsets $X \subseteq M$ s. t. $x \in X$ and $x \leq y$ imply $y \in X$) of $(M, \leq)$.

3.3 Conceptual Scales and Navigating Email

Conceptual scaling deals with many-valued attributes [11]. Often attributes are not one-valued, as are the string descriptors given above, but allow a range of values. This is modeled by a *many-valued context* which is roughly equivalent to a relation in a relational database with one field being a primary key. As one-valued contexts are special cases of many-valued contexts, conceptual scaling can also be applied to one-valued contexts to reduce the complexity of the visualization.

In this paper, we only deal with one-valued formal contexts. Readers who are interested in many-valued contexts and the use of conceptual scaling in the general case are referred to [13]. Applied to one-valued contexts, conceptual scales are used to determine the concept lattice that arises from one vertical 'slice' of a large context:

Definition 4. *A* conceptual scale *for a subset $B \subseteq M$ of attributes is a (one-valued) formal context $\mathbb{S}_B := (G_B, B, \ni)$ with $G_B \subseteq \mathfrak{P}(B)$. The scale is called* consistent *wrt $\mathbb{K} := (G, M, I)$ if $\{g\}' \cap B \in G_B$ for each $g \in G$. For a consistent scale $\mathbb{S}_B$, the context $\mathbb{S}_B(\mathbb{K}) := (G, B, I \cap (G \times B))$ is called its* realized scale.

Conceptual scales are used to group together related attributes. They can be considered as different points of view, which can be combined by the user when she is browsing/exploring the email collection. Conceptual scales can be derived automatically (as described at the end of this section) or can be defined without any restriction by the user.

Realized scales are derived from the conceptual scales when a diagram is requested by the user. One of them is the scale Conference Related which is

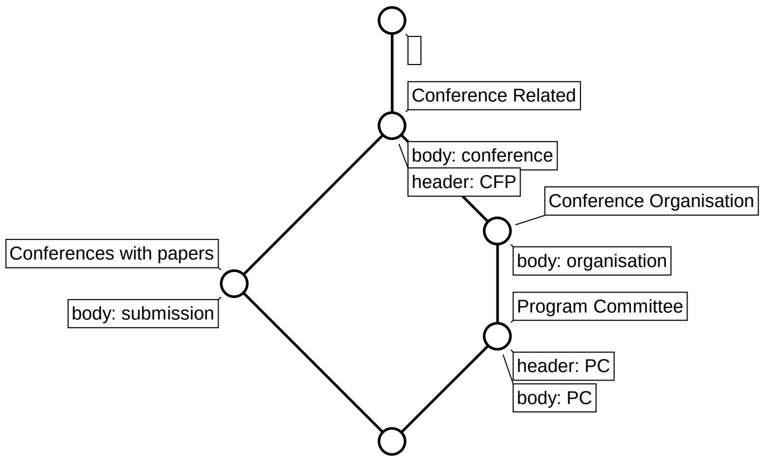


Fig. 4. Conceptual Scale 'Conference Related'.

displayed in Fig. 4. Its realized scale is the one we already saw in Fig. 2. In the conceptual scale, the attributes are strings that are displayed later as attribute names in the realized scale. The objects are queries (see Section 5.3) which determine the set of emails to be displayed in the realized scale ('body: PC' is for instance a query that selects emails containing the string 'PC' in their body).

HierMail stores all scales that the client has defined in previous sessions. For each scale the client assigns a unique name. This is modeled by a mapping:

Definition 5. *Let $\mathcal{S}$ be a set, whose elements are called scale names. The mapping*

$$\alpha\colon \mathcal{S} \to \mathfrak{P}(\mathcal{M})$$

defines for each scale name $s \in \mathcal{S}$ a scale $\mathbb{S}_s := \mathbb{S}_{\alpha(s)}$.

For instance, the user may introduce a new scale which classifies emails according to being related to a conference by adding a new element 'Conference' to $\mathcal{S}$ and by defining:

$$\alpha(\text{Conference}) := \{\text{WWW 11}, \text{IJCAI '01}, \text{K-CAP '01}, \text{ADCS '01 }, \text{PKDD 2000}\}$$

Observe that $\mathcal{S}$ and M need not be disjoint. This allows the following default construction deducing conceptual scales directly from the subsumption hierarchy: Let $\mathcal{S} := M$, and define, for $s \in \mathcal{S}$, $\alpha(s) := \{m \in M | m \prec s\}$ (with $x \prec y$ if and only if $x < y$ and there is no z s.t. $x < z < y$ in the partially ordered set $(M, \leq)$ defined in Section 3.2). This means that all descriptors $m \in M$ are considered as the name of scale $\mathbb{S}_m$ and as a descriptor of another scale $\mathbb{S}_n$ (where $m \prec n$) [20]. A result of this definition is that descriptors with no lower covers lead to trivial scales containing no other descriptors.

4 Requirements of HierMail

In this section, we discuss requirements for conceptual email management with HierMail. In the following section we will then explain how our implementation responds to these requirements. The requirements may be divided along the underlying mathematical structures defined in Section 3:

1. assist the user in editing and browsing a descriptor hierarchy;
2. help the client visualize and modify the scale function α;
3. allow the client to manage the assignment of descriptor to emails;

4. assist the client search the conceptual space of emails for both individual emails and conceptual groupings of emails.

Additional to the requirements stated above, a good email client needs to be able to send, receive and display emails, process the various email formats and interact with the current email protocols. Since these requirements are already well understood and implemented by existing email programs they are not discussed further. This does not mean they are not important, rather that the normal feature set of an email client is implicit in the description of HierMail.

4.1 Editing and Modifying the Descriptor Hierarchy

The descriptor hierarchy is a partially ordered set $(M, \leq)$ where each element of M is a descriptor. The requirements for editing and browsing the descriptor hierarchy are:

– graphically display the structure of the the partially ordered set. The ordering relation must be evident to the client.
– make accessible to the client a series of direct manipulations to alter the ordering relation. It should be possible to create any partial order to a reasonable size limit.

4.2 Visualizing and Modifying the Scale Function α

The program must be able to visualize the scale function, α, explained in Section 3, and to support its modification. It must allow an overlap between the set of scale labels, $\mathcal{S}$, and the set of descriptors M.

4.3 Managing the Descriptor Assignments

The program should store the formal context (G, M, I) and ensure that the compatibility condition is always satisfied. It is inevitable that the program will have to sometimes modify the formal context in order to satisfy the compatibility condition after a change is made to the descriptor hierarchy.

The program must support two mechanisms for the association of descriptors to emails. Firstly, a mechanism is needed in which emails are automatically associated with descriptors based on the email content. Secondly, the user should be able to view and modify the association of descriptors with emails.

4.4 Navigating the Conceptual Space

The program must allow the navigation of the conceptual space of the emails by drawing line diagrams of concept lattices derived from conceptual scales [13]. These line diagrams should extend to (locally [19]) nested line diagrams. The program must allow for retrieval and display of emails forming the extension of concepts displayed in the line diagrams.

5 Specification and Implementation of HierMail

This section presents, in mathematical terms, the specification of the behavior of the HierMail system, and discusses specific aspects of its implementation. It is divided into a similar structure as Section 4.

5.1 Classifier Hierarchy

Browsing the Hierarchy. The user is presented with a view of the hierarchy $(M, \leq)$ as a tree widget shown in Fig. 3. The tree widget has the advantage that most users are familiar with its behavior and it provides a compact representation (in the sense of screen space used) of a tree structure. The descriptor hierarchy, being a partially ordered set, allows for multiple inheritance. Although the example given in Fig. 3 is a forest, no limitation is placed by the program on the structure other than that it must be a partial order.

The following is the definition of the tree derived from the descriptor hierarchy for the purpose of visualization in the tree widget. Let $(M, \leq)$ be a partially ordered set and denote the set of all sequences of elements from M by M^* (including the empty sequence ε). Then the labeled tree derived from the descriptor hierarchy is comprised by $(T, \sqsubseteq, \texttt{label})$ where $T := \{(m_1, \ldots, m_n) \in M^* \mid m_i \prec m_{i+1},\ m_n \in \max(M)\} \cup \{\varepsilon\}$, $w_1 \sqsubseteq w_2$ iff w_2 is a suffix of w_1, and $\texttt{label}: T \setminus \{\varepsilon\} \to M$ is the function defined by $\texttt{label}(m_1, \ldots, m_n) := m_n$.

Each tree node is hence identified by a path from a descriptor to the top of the descriptor hierarchy. An example of a tree is given on the left side for Figure 3. The hierarchy displayed is just the one encoded in $(T, \sqsubseteq)$, and the function `label` provides the labeling.

The tree representation has the disadvantage that elements with multiple parents occur multiple times in the tree and the tree can become large. The tree is however manageable if the user keeps the number of elements with multiple parents in the partial order small.

Modifying the Hierarchy $(M, \leq)$. The program provides four operations for modifying the hierarchy: *insert & remove descriptor* and *insert & remove ordering*. More complex operations provided to the client, for instance moving an item in the taxonomy, are resolved internally to a sequences of these basic operations. In this section we denote the *order filter* (or *up-set*) of m as $[m) := \{x \in M \mid m \leq x\}$, the *order ideal* (or *down-set*) of m as $(m] := \{x \in M \mid x \leq m\}$, the lower cover of m as $\prec_m := \{x \in M \mid x \prec m\}$, and the upper cover of m as $\succ_m := \{x \in M \mid x \succ m\}$.

The operation of *insert descriptor* adds a new descriptor to M, and leaves the $\leq$ relation unchanged. The *remove descriptor* operation takes a single parameter $a \in M$ for which the lower cover is empty, and removes a from M and $[a) \times \{a\}$ from the ordering relation.

The operation of *insert ordering* takes two parameters $a, b \in M$ and inserts into the relation $\leq$, the set $[a) \times (b]$. The operation of remove ordering takes two parameters $a, b \in M$ where a is an upper cover of b. The operation has been drawn in the left diagram in Fig. 5. The shading gives an indication of the up-sets and down-sets of a and b before and after the insert operation. The *remove ordering* operation removes from $\leq$ the set $((b] \setminus (\prec_a \setminus \{b\}]) \times ([a) \setminus [\succ_b \setminus \{a\}))$. The right diagram in Fig. 5 shows the remove ordering operation.

Inserting the ordering $b \leq a$ into $\leq$ requires the insertion of set $([a) \setminus [b)) \times \{g \in G \mid (g, b) \in I\}$ into I. Such an insertion into an inverted file index is $O(nm)$ where n is the average number of entries in the inverted index in the shaded region, and m is the number of elements in the shaded region. The real

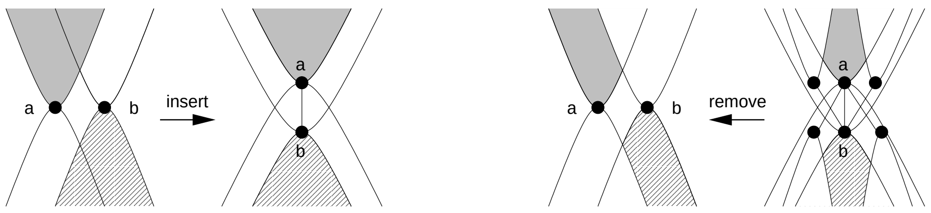


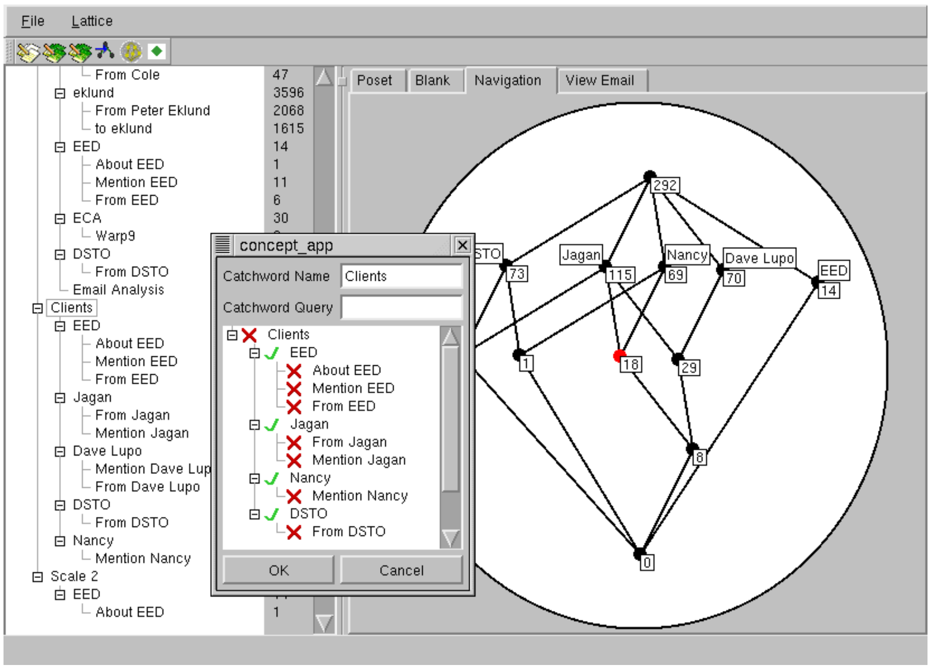
Fig. 5. `Insert` and `removal ordering` operation

complexity of this operation is best determined via experimentation with a large document sets and a large user defined hierarchy [5]. Similarly the removal of the ordering $b \leq a$ from $\leq$ will require a re-computation of the inverted file entries for elements in $[a)$.

5.2 Visualization and Maintenance of the Scale Function α

The set of scales $\mathcal{S}$, according to the mathematization in Section 3, is not disjoint from M, thus the tree representation of M already presents a view of a portion of $\mathcal{S}$. In order to reduce the complexity of the graphical interface, we make $\mathcal{S}$ equal to M, i. e., all descriptors are scale names, and all scale names are descriptors. Such an assumption is made possible by the definition of the default scale for a descriptor given in Section 3.

The function α maps each descriptor m to a set of descriptors. The program displays this set of descriptors, when requested by the user, using a dialog box (see Fig. 6). The dialog box contains, for all descriptors in the down-set of m, an icon (either a tick, or a cross) to indicate membership in the set of descriptors given by $\alpha(m)$. Clicking on the icon changes the membership of $\alpha(m)$.



Catchword Name: Clients
Catchword Query:

- ✗ Clients
  - ✓ EED
    - ✗ About EED
    - ✗ Mention EED
    - ✗ From EED
  - ✓ Jagan
    - ✗ From Jagan
    - ✗ Mention Jagan
  - ✓ Nancy
    - ✗ Mention Nancy
  - ✓ DSTO
    - ✗ From DSTO

Fig. 6. Scale function and concept lattice. The central dialogue box shows how α can be edited.

By only displaying the down-set of m in the dialog box, the program restricts the definition of α to $\alpha(m) \subseteq (m]$. This has an effect on the "remove ordering operation" defined on $(M, \leq)$. When the ordering of $a \leq b$ is removed, the image of attributes in $[a)$ under α must be checked and possibly modified.

5.3 Associating Emails with Classifiers

Recall that G is the set of email documents, M the set of descriptors, and I the binary relation between them. We make use of four intermediate relations Q, R, R^+, and R^- as defined below for managing the different ways in which email documents may be associated with descriptors in relation I. The relations Q and R associate emails with descriptors via an automatic process based on content and queries attached to descriptors, R^+ and R^- associate email based on user input, and I combines these two sources with the hierarchy defined over the descriptors. By separating the relations for automatic associations of descriptors to emails from the relations for user defined associations, the program maintains a pure keyword index into the email collection. Relations R and I are derived from Q, R^+, and R^-, and so need not be stored. However, computing storing I greatly reduces the time complexity of the program.

Each member of $(M, \leq)$ is associated with a query term,: in this application is a set of *section/word pairs.* A *section* of an email is either a header field, e. g. the "From:" field, or the section "body" which is composed of the parts[4] of the email directly encoding text. More formally stated: Let H be the set of sections found in the email documents, W the set of words found in the email documents, then the function `query`: $M \to \mathfrak{P}(H \times W)$ attaches to each attribute a set of section/word pairs.

$Q \subseteq G \times (H \times W)$ is a relation between documents and section/word pairs. The relation member $(g, (h, w)) \in Q$ indicates that document g has word w in section h. Q is stored via an inverted file index and is only updated when new email is presented to the system. The relation $R \subseteq G \times M$ is derived from the relation Q and the function `query` via: $(g, m) \in R$ iff $(g, (h, w)) \in Q$ for some

[4] The MIME extension to the email format allows an email document to have multiple parts. These multiple parts are sometimes referred to as attachments.

$(h, w) \in \texttt{query}(m)$. The relation R is only used as an intermediate step and is calculated from Q as required by the program.

The relations R^+ and R^- store user judgments saying whether or not an email should have a descriptor m. These judgments will "over-rule" the relation R. We impose the constraint

$$\forall g \in G,\ m, n \in M\colon\ m \geq n \Longrightarrow \neg((g, m) \in R^- \wedge (g, n) \in R^+)) \quad (\#)$$

on the two relations R^+ and R^-, saying that a user is not allowed to contradict himself, i. e., he is not allowed, for $m \geq n$, to assign (g, m) to R^- and (g, n) to R^+. The handling of the user judgments is discussed below.

Maintaining the Compatibility Condition. The relation I has to respect the compatibility condition ($\ddagger$). It is derived from the relations R, R^+ and R^- using the following operator: For any relation $J \subseteq G \times M$, we define $J^{\ddagger} := \{(g, m) \in G \times M \mid \exists n \in M\colon (g, n) \in J,\ n \leq m\}$. We obtain the binary relation I by $I := ((R \setminus R^-) \cup R^+)^{\ddagger}$.

Processing new Emails. When a batch of new emails, G_b, is presented to the program, the relation Q is updated automatically by inserting new pairs, Q_b, into the relation. The modification of Q into $Q \cup Q_b$ will cause an insertion of pairs R_b into R according to $\texttt{query}(m)$ and then subsequently an insertion of new pairs I_b into I. The definitions are:

$$Q_b \subseteq G_b \times (H \times W)$$

$$R_b = \{(g, m) \mid \exists\,(h, w) \in \text{query}(m) \text{ and } (g, (h, w)) \in Q_b\}$$

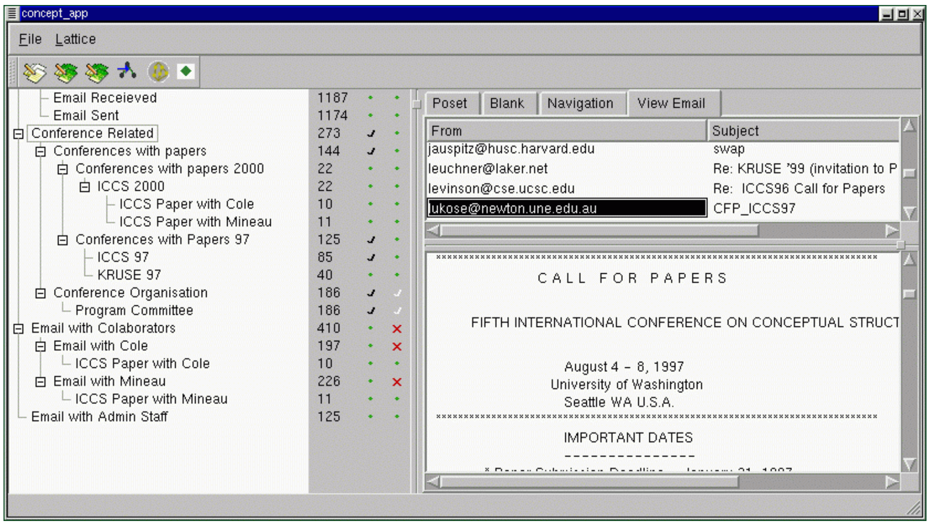
$$I_b = \{(g,m) \mid \exists m_1 \leq m \text{ with } (g,m_1) \in R_b\}$$

When new emails presented to the system for indexing, the modification to the inverted file index consists of inserting new entries which is a very efficient operation. Each pair inserted is $O(1)$.

Integrating User Judgments. The user can modify the association of emails with descriptors in two ways. Firstly, by changing the relations R^+ and R^-, and secondly, by making modifications to the `query` function. In order to explain the user interface for making modifications to R^+ and R$^-$ we introduce the following notation. For an email $g \in G$, we define the restriction of any relation $J \subseteq G \times M$ to this email by $J_g := J \cap (\{g\} \times M)$. For the purpose of brevity we say m belongs to J_g if $(g,m) \in J_g$.

The user is able to view individual emails as shown in Fig. 7. In this mode, icons are attached to descriptors in the tree widget displayed to the left of the email. These icons indicate to the user how each of the descriptors is related to the displayed email by R, R^-, and R^+. The user is able to change the relations R^- and R^+ by toggling the icons.

1. If m is not in $R_g^\ddagger$, $(R_g^+)^\ddagger$, or R_g^- then no icon is displayed. This is the case when the descriptor m is not assigned to the email g and this non-assignment was not forced by the user.
2. If m is in $R_g^\ddagger$ then a yellow tick (shown as white in Fig. 7) is displayed. It indicates that m was assigned to the email automatically.



Email Receieved	1187	●	●
Email Sent	1174	●	●
Conference Related	273	✓	●
Conferences with papers	144	✓	●
Conferences with papers 2000	22	●	●
ICCS 2000	22	●	●
ICCS Paper with Cole	10	●	●
ICCS Paper with Mineau	11	●	●
Conferences with Papers 97	125	✓	●
ICCS 97	85	✓	●
KRUSE 97	40	●	●
Conference Organisation	186	✓	✓
Program Committee	186	✓	✓
Email with Colaborators	410	●	✗
Email with Cole	197	●	✗
ICCS Paper with Cole	10	●	●
Email with Mineau	226	●	✗
ICCS Paper with Mineau	11	●	●
Email with Admin Staff	125	●	●

From	Subject
jauspitz@husc.harvard.edu	swap
leuchner@laker.net	Re: KRUSE '99 (invitation to P
levinson@cse.ucsc.edu	Re: ICCS96 Call for Papers
lukose@newton.une.edu.au	CFP_ICCS97

```
*****************************************************************************
                    C A L L   F O R   P A P E R S

      FIFTH INTERNATIONAL CONFERENCE ON CONCEPTUAL STRUCT

                         August 4 - 8, 1997
                       University of Washington
                         Seattle WA U.S.A.
*****************************************************************************
                         IMPORTANT DATES
                         ---------------
```

Fig. 7. Interface for viewing emails and associating them with descriptors

3. If m is in R_g^- then a red cross is displayed. It means that the user explicitly decided not to assign the descriptor to the email (overruling eventually the automatic assignment).

4. If m is in $(R_g^+)^\ddagger$ then a green (shown as black in Fig. 7) tick is displayed. This is the case when the user explicitly decided to assign the descriptor to the email (overruling eventually the automatic non-assignment).

All combinations that exclude simultaneous red cross and green tick (i. e., which do not contradict the (#) constraint) are possible.

The user can then determine that the displayed email has a descriptor in I if there is either a green tick or a yellow tick in the absence of a red cross. The program provides two basic operations, `associate attribute` and `disassociate attribute` from which more complex operations for use in the user interface may be constructed. The `associate attribute` operation takes two parameters, an email document, and a descriptor m. The operation inserts the pair (g, m) into R^+, and removes, for all $n \geq m$, (g, s) from R^-. Similarly the operation `disassociate attribute` takes two parameters, an email and a descriptor. The operation inserts (g, m) in R^- and removes, for all $n \leq m$, (g, n) from R^+. The construction of the two operators guarantees that the constraint (#) is always satisfied.

The user is also able to influence the way that R is derived from Q by modifying the `query` function. The user is able to modify the query using the Scale Query field in the dialog box shown in Fig. 6. After any such modification to the `query` function the relations R and I are modified accordingly.

When the user makes a judgment that an indexed email should be associated with a descriptor m, then an update must be made to R^+, which in turn causes updates to all attributes in the order filter of m to be updated in I. The expense of such an update depends on the implementation of the inverted file index and could be as bad as $O(n)$ where n is the average number of documents per descriptor. The case that a client retracts a judgment, saying that an email is no longer be associated with an attribute m, requires a possible update to each attribute n in the order filter of m.

It is useful for the system to maintain the relation R^+ for special descriptors dependent on observation by the program of the users behavior. Two examples of such descriptors are "read" for emails that the user has already read, and "new" for emails that the user has not yet been notified of.

5.4 Navigating the Conceptual Space

When the user requests that the concept lattice derived from the scale with name $s \in \mathcal{S}$ be drawn, the program computes $\mathbb{S}_{\alpha(s)}$ from Definition 1 via the algorithm reported in [5]. In the case that the user requests a diagram combining two scales with names labels s and t, then the scale $\mathbb{S}_{B \cup C}$ with $B = \alpha(s)$ and $C = \alpha(t)$ is calculated by the program and its concept lattice $\underline{\mathfrak{B}}(\mathbb{S}_{B \cup C})$ is drawn as a projection into the lattice product $\underline{\mathfrak{B}}(\mathbb{S}_B) \times \underline{\mathfrak{B}}(\mathbb{S}_C)$.

To assist the user in navigating the conceptual space of emails, the program draws simple line diagrams and (locally) nested line diagrams. A simple line diagram is used to visualize a single scale (see Fig. 8), while nested line diagrams are used to visualize combinations of scales (see Fig. 9). The concept lattices, from which the nested line diagrams are drawn, are computed from the contexts given by $\mathbb{S}_{\alpha(s)}$. Figure 10 shows a locally nested line diagram according to [19], where the complexity of the visualization is reduced by displaying the inner scale only when it refines the concept of the outer scale.

In the next Section, the navigation in the conceptual space by means of (nested) line diagrams is illustrated by two examples.

6 Two Application Scenarios for HierMail

The user may navigate the conceptual space of emails documents for different purposes:

1. to retrieve previously stored emails; and

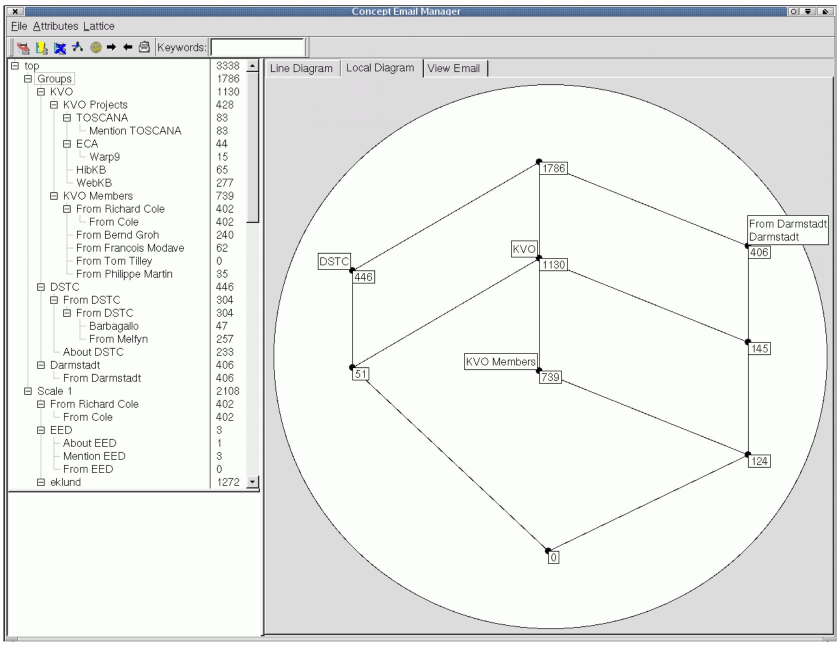
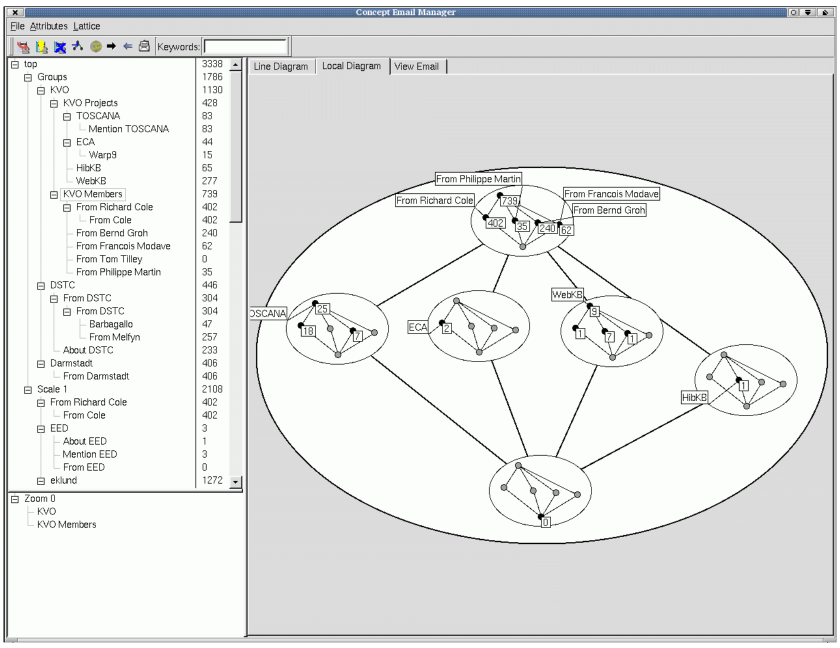


Fig. 8. Top-left-center shows the text input field allowing keywords to be entered in the style of a classic search engine. The explorer-like structure to the left is the user-defined descriptor hierarchy (ontology). To the right is the concept lattice.

2. to discover knowledge in the email collection, for instance to find collections of emails thematically linked (i. e., conceptual clustering), to discover patterns of communication between different groups, or to detect upcoming topics.

6.1 Email Retrieval with HierMail

Imagine a researcher who was in the Program Committee (PC) of ICCS ’97 and was at that time co-authoring with other members of the PC for the same confer-



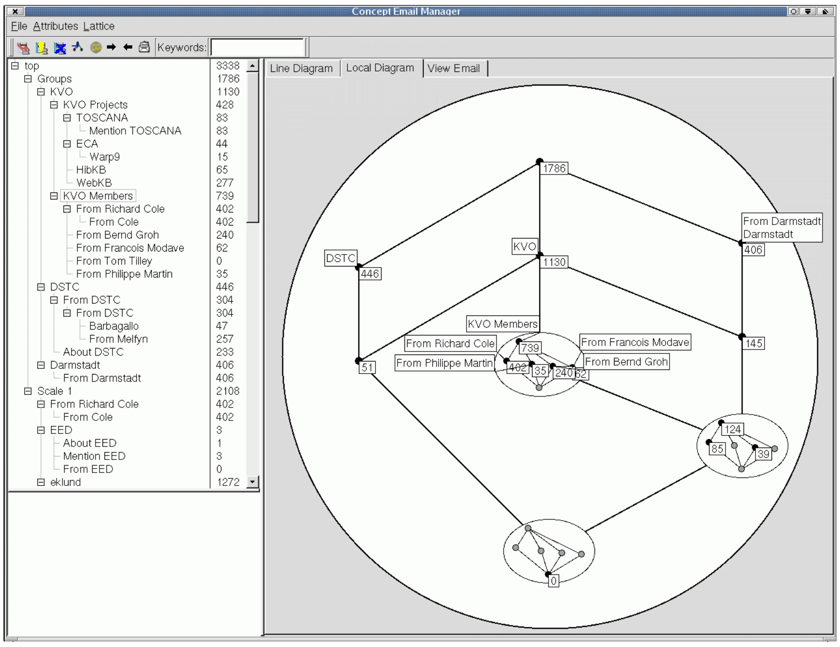
top	3338
Groups	1786
KVO	1130
KVO Projects	428
TOSCANA	83
Mention TOSCANA	83
ECA	44
Warp9	15
HibKB	65
WebKB	277
KVO Members	739
From Richard Cole	402
From Cole	402
From Bernd Groh	240
From Francois Modave	62
From Tom Tilley	0
From Philippe Martin	35
DSTC	446
From DSTC	304
From DSTC	304
Barbagallo	47
From Melfyn	257
About DSTC	233
Darmstadt	406
From Darmstadt	406
Scale 1	2108
From Richard Cole	402
From Cole	402
EED	3
About EED	1
Mention EED	3
From EED	0
eklund	1272

Zoom 0
KVO
KVO Members

Fig. 9. This screenshot shows a nested concept lattice with a conceptual scale nested within the outer scale.

ence. For the organization of a conference in the year 2002, she wants to retrieve some facts about the organization of ICCS '97. However, she only remembers that she exchanged this information with one of the people she was co-authoring with for ICCS '97, and that it was only one small part of a correspondence covering various topics.

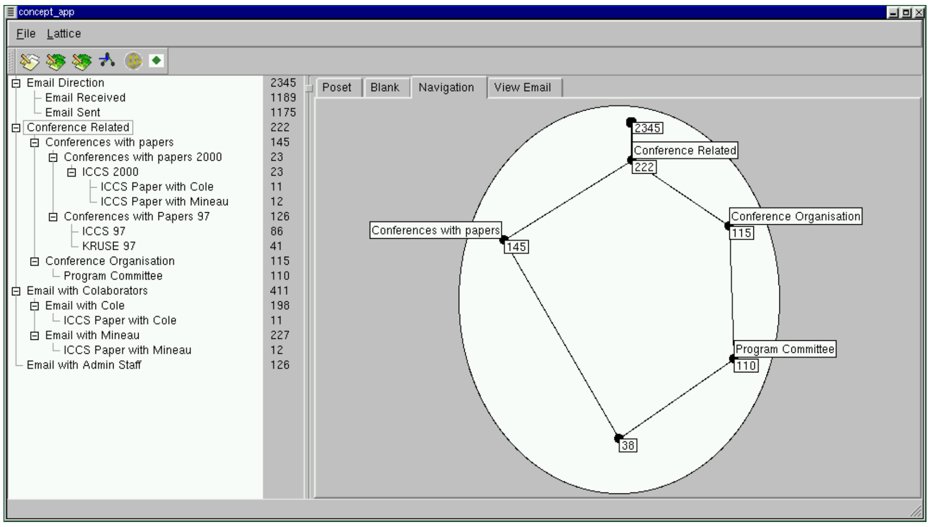
The researcher may begin her search by requesting a line diagram for the scale named "Conference Related". This scale is shown in Fig. 11. It shows that from her 2345 emails in total, there are 222 emails related to conferences, 145 of which are related to conferences with papers submitted, 115 related to



```
top                          3338
  Groups                     1786
    KVO                      1130
      KVO Projects           428
        TOSCANA              83
          Mention TOSCANA    83
        ECA                  44
          Warp9              15
        HibKB                65
        WebKB                277
      KVO Members            739
        From Richard Cole    402
          From Cole          402
        From Bernd Groh      240
        From Francois Modave 62
        From Tom Tilley      0
        From Philippe Martin 35
    DSTC                     446
      From DSTC              304
        From DSTC            304
          Barbagallo         47
          From Melfyn        257
      About DSTC             233
    Darmstadt                406
      From Darmstadt         406
  Scale 1                    2108
    From Richard Cole        402
      From Cole              402
    EED                      3
      About EED              1
      Mention EED            3
      From EED               0
    eklund                   1272
```

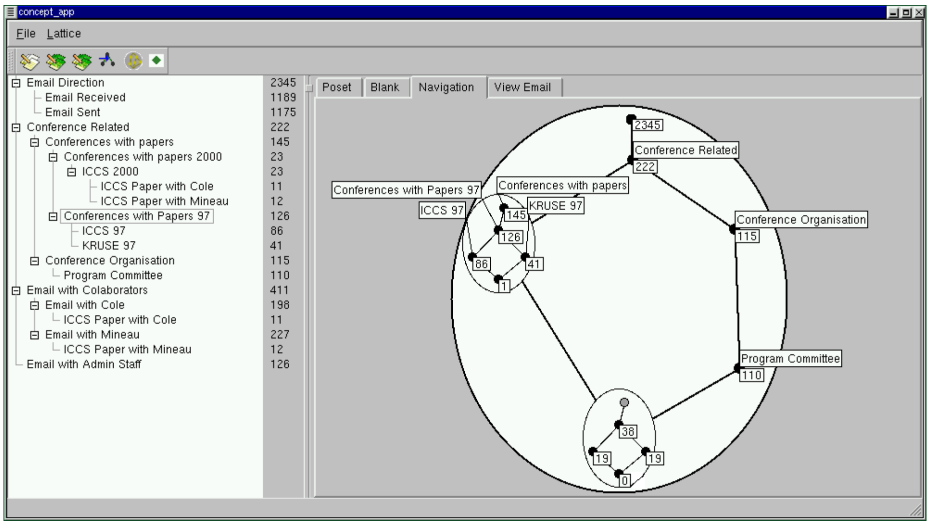
Fig. 10. Nesting and Zooming. Here we see local scaling as a way of reducing complexity. The inner scale is displayed only when it differentiates the concept of the outer scale.

conference organization, 110 of which in their turn are related to both conference organization and program committees. 38 emails are related to all those topics. The researcher decides that the email she is looking for is likely to be under the descriptor "Conferences with Papers". As there are too many emails in its extent to be read through, she may for instance want to expand the concept and narrow the search. By choosing the scale $\mathbb{S}_{\text{Conferences 1997}}$, she obtains Fig. 12. Now the researcher can for instance check the 19 mails related to "ICCS '97" and



```
Email Direction                      2345
  Email Received                     1189
  Email Sent                         1175
Conference Related                   222
  Conferences with papers            145
    Conferences with papers 2000     23
      ICCS 2000                      23
        ICCS Paper with Cole         11
        ICCS Paper with Mineau       12
    Conferences with Papers 97       126
      ICCS 97                        86
      KRUSE 97                       41
  Conference Organisation            115
    Program Committee                110
Email with Colaborators              411
  Email with Cole                    198
    ICCS Paper with Cole             11
  Email with Mineau                  227
    ICCS Paper with Mineau           12
Email with Admin Staff               126
```

Fig. 11. Concept Lattice derived from the Scale for "Conference Related".



```
Email Direction                      2345
  Email Received                     1189
  Email Sent                         1175
Conference Related                   222
  Conferences with papers            145
    Conferences with papers 2000     23
      ICCS 2000                      23
        ICCS Paper with Cole         11
        ICCS Paper with Mineau       12
    Conferences with Papers 97       126
      ICCS 97                        86
      KRUSE 97                       41
  Conference Organisation            115
    Program Committee                110
Email with Colaborators              411
  Email with Cole                    198
    ICCS Paper with Cole             11
  Email with Mineau                  227
    ICCS Paper with Mineau           12
Email with Admin Staff               126
```

Fig. 12. Concept Lattice derived from the Scales for "Conference Related" and "Conferences with Papers".

"Conference Organization/Program Committee" as in a conventional mailing program (see Fig. 7).

If she still doesn't find the email she is looking for there, then she has to check either the 86 papers related to "ICCS '97" or even all 115 emails under the descriptor "Conference Organization". Before doing this, however, she might want to differentiate these concepts further, e. g., by zooming into them with the scale "Members of ICCS '97 Program Committee". If this scale doesn't exist yet, then she can create it on the fly using the widget for modifying the scale function (see Fig. 6) and eventually store the new scale for further use.

Note that with a classical, tree-structured search hierarchy, for instance one where the names of the correspondents are on the highest level, one would be forced to search all branches individually.

6.2 Knowledge Discovery with HierMail

We now look at how HierMail supports knowledge discovery based on the lattice metaphor, uncovering relationships inherent in the data. Suppose we want insights about the working behavior of the KVO research group and analyze a collection of emails of the group leader. We are particularly interested in the emails exchanged between the group leader, the PhD students and Research Scientists in the group. In order to select relevant emails, we consider the conceptual scale 'Groups'. Fig. 8 shows that among his 1,786 emails about 'Groups', there are 1,130 related to the KVO group (Fig. 8 centre) and 739 emails from members of the KVO group. Interestingly, 124 of those mails are also assigned to the descriptor 'From Darmstadt'. This is due to PhD students traveling to Europe, and indicates extensive exchange between the KVO group and Darmstadt University of Technology.

This corpus of 739 emails will be subject to further analysis. We zoom into the middle concept, using the scale KVO group. The result is shown in Fig. 10. It reveals that Richard Cole is the group member who exchanged the most emails with the group leader. In order to derive new knowledge out of this information, further implicit background knowledge of the group leader is needed, as there may be different reasons for this observation. For instance, it may be that Cole is the most active student, or he simply prefers email as a means of communication, or it may be that he is the longest serving member of the research group and has therefore accumulated the most email traffic.

Let us now analyze how the group members are related to projects. We zoom into the middle of Fig. 10 with the scale KVO Projects, and swap inner and outer scales. The result is shown in Fig. 9. The inner scale distinguishes the emails by team members and the outer scale by project names. The effect is that we can immediately see from the number of emails attached to each node how team members' interests are partitioned across projects. The gray nodes in the middle left ellipse show that only Richard Cole shows interest in the project named ECA; there is no email traffic on ECA from any other team member.

A tool like HierMail might have been useful for instance for the law suit of the U. S. government against a major software company in 2000, where the public prosecutor had to check the company's emails on eventually unlawful cartel agreements. Another possible use is the discovery of new trends on mailing lists and newsgroup servers. As in most other KDD applications, of course data privacy regulations have to be considered for such applications.

7 Conclusion

This paper gives a mathematical description of the algebraic structures used to create a lattice-based view of email. The structure, its implementation, and its operation aid the process of knowledge discovery in large collections of emails. The paper illustrates the mathematical treatment through the development of a useful document management, retrieval and knowledge discovery tool for emails. By using a conceptual multi-hierarchy, the content and shape of the lattice view is varied via the process of mixed-initiative interaction. An efficient implementation of the index promotes client iteration. The work shows that the principles of Formal Concept Analysis can be supported by an inverted file index and that a useful and scalable email management system results.

Acknowledgment. The work reported in this paper forms part of the GoDa (Gold-Coast/Darmstadt) project and is supported by the Australian Research Council and the *Deutsche Forschungsgemeinschaft*. This research also benefits from the support of the Distributed Systems Technology Research Centre (DSTC Pty Ltd) which operates as part of the Australian Government's CRC program.

References

1. I. Androutsopoulus, J. Koutsias, K.V. Chandrinos, D. Spryropoulos: An Experimental Comparison of Naive Bayesian and Keyword-Based Anti-Spam Filtering, In *Proceedings of the 23rd SIGIR Conference*, ACM Press, 2000, 160–167
2. J. Brutlag, J. Meek: Challenges of the email domain for text classification, In *Proceedings of the 17th International Conference on Machine Learning*, Morgan Kaufmann, 2000, 103–110
3. C. Carpineto, G. Romano: GALOIS: An Order-Theoretic Approach to Conceptual Clustering. *Machine Learning.* Proc. ICML 1993, Morgan Kaufmann Publishers 1993, 33–40
4. W. Cohen: Learning Rules that Classify E-mail, In *Proceedings of the AAAI Spring Symposium on Machine Learning in Information Access*, Standford, CA, 1996.
5. R. Cole, P. Eklund: Scalability in Formal Concept Analysis: A Case Study using Medical Texts. *Computational Intelligence*, **15**(1), 1999, 11–27
6. R. Cole, P. Eklund: Analyzing an Email Collection using Formal Concept Analysis. *Proceedings of the European Conf. on Knowledge and Data Discovery* (PKDD99), LNAI 1704, Springer, Heidelberg 1999, 309–315
7. R. Cole, P. Eklund: Browsing Semi-Structured Web Texts using Formal Concept Analysis, in Proceedings of the 9th International Conference on Conceptual Structures (ICCS'2001), LNAI 2120, Springer, Heidelberg 2001, 319–332
8. R. Cole, P. Eklund, G. Stumme: CEM — A Program for Visualization and Discovery in Email, In D.A. Zighed, J. Komorowski, J. Zytkow (Eds), in *Proceedings of the European Conf. on Knowledge and Data Discovery* (PKDD00), LNAI 1910, Springer, Heidelberg 2000, 367–374

9. R. Cole, P. W. Eklund, D. Walker: Using Conceptual Scaling in Formal Concept Analysis for Knowledge and Data Discovery in Medical Texts, *Proceedings of the Second Pacific Asian Conference on Knowledge Discovery and Data Mining* (PAKDD98), World Scientific, 1998, 378–379

10. R. Cole, G. Stumme: CEM: A Conceptual Email Manager, in Proceedings of the 8th International Conference on Conceptual Structures (ICCS'2000), LNAI 1867, Springer, Heidelberg 2000, 438–453

11. B. Ganter, R. Wille: Conceptual scaling. In: F.Roberts (ed.): *Applications of combinatorics and graph theory to the biological and social sciences.* Springer, New York 1989, 139–167

12. E. Horvitz: Uncertainty, Action and Interaction: In pursuit of Mixed-initiative Computing *Intelligent Systems* IEEE, September, 1999, 17–20 http://research.microsoft.com/~horvitz/mixedinit.HTM

13. B. Ganter, R. Wille: *Formal Concept Analysis: Mathematical Foundations.* Springer, Heidelberg 1999

14. K. Jones: View Mail Users Manual. *http://www.wonderworks.com/vm.* 1999

15. R. S. Michalski: Knowledge acquisition through conceptual clustering: a theoretical framework and an algorithm for partitioning data into conjunctive concepts. *Policy Analysis and Information Systems* **4**(3), 1980, 219–244

16. G. Mineau, R. Godin: Automatic Structuring of Knowledge Bases by Conceptual Clustering. *IEEE Transactions on Knowledge and Data Engineering* **7**(5),1995, 824–829

17. W. Schuller: *http://gmail.linuxpower.org/.* 1999

18. S. Strahringer, R. Wille: Conceptual clustering via convex-ordinal structures. In: O. Opitz, B. Lausen, R. Klar (eds.): *Information and Classification.* Springer, Berlin-

Heidelberg 1993, 85–98

19. G. Stumme: Local Scaling in Conceptual Data Systems. In *Proceedings of the 5th International Conference on Conceptual Structures* (ICCS96), LNAI 1115, Springer, Heidelberg 1996, 308–320
20. G. Stumme: Hierarchies of Conceptual Scales. *Proceedings of the Knowledge Acquisition Workshop: Modeling and Management* (KAW99). Banff 1999. Vol. 2, 78–95
21. F. Vogt, R. Wille: TOSCANA: A Graphical Tool for Analyzing and Exploring Data. In: R. Tamassia, I. G. Tollis (eds.): *Graph Drawing '94*, LNCS 894, Springer, Heidelberg 1995, 226–233
22. F. Vogt, C. Wachter, R. Wille: Data Analysis based on a Conceptual File, In: Hans-Hermann Bock, W. Lenski and P. Ihm (eds.): *Classification, Data Analysis and Knowledge Organization*, Springer, Heidelberg Berlin, 1991, 131–140
23. R. Wille: Conceptual Landscapes of Knowledge: A Pragmatic Paradigm for Knowledge Processing In: W. Gaul, H. Locarek-Junge (eds.): *Classification in the Information Age*, Springer, Heidelberg 1999.