FCA-Based Browsing and Searching of a Collection of Images

Jon Ducrou[1], Björn Vormbrock[2], and Peter Eklund[3]

[1] School of Information Technology and Computer Science, The University of Wollongong, Northfields Avenue, Wollongong, NSW 2522, Australia
jrd990@uow.edu.au

[2] AG Algebra und Logik, FB Mathematik, Technische Universität Darmstadt, Schloßgartenstr. 7, D–64289 Darmstadt, Germany
vormbrock@mathematik.tu-darmstadt.de

[3] School of Economics and Information Systems, The University of Wollongong, Northfields Avenue, Wollongong, NSW 2522, Australia
peklund@uow.edu.au

Abstract. This paper introduces ImageSleuth, a tool for browsing and searching annotated collections of images. It combines the methods of Formal Concept Analysis (FCA) for information retrieval with the graphical information conveyed in thumbnails. In order to use thumbnails of images to represent concept extents, line diagrams can not be efficiently utilised and thus other navigation methods are necessary. In addition to established methods like search and upper/lower neighbours, a query by example function and the possibility to restrict the attribute set are included. Moreover, metrics on conceptual distance and similarity are discussed and applied to automated discovery of relevant concepts. This paper describes the FCA base of ImageSleuth which formed the basis for its design and the implementation which followed.

1 Motivation

Formal Concept Analysis (FCA) has been successfully applied in Information Retrieval for browsing and searching text documents ([CS01], [KC00]). The richer structure of the concept lattice has advantages over simple keyword search or tree structures. For keyword search, the user has to remember or guess the correct keywords. For searching in trees, the names of nodes serve as keywords, but there is a unique path leading to the desired information. Moreover, once a categorisation scheme for the documents is chosen, this hierarchy is enforced for every search. In concept lattices multiple paths can lead to a result, so the user may guide the search via the addition of required properties step by step without the restriction imposed by a single inheritance hierarchy. The order of these properties is irrelevant.

This paper illustrates how ImageSleuth uses FCA methods for information retrieval within a collection of images. Any such approach has to take into consideration the graphical nature of this information. The established method for browsing collections of images is to display all images as *thumbnails*. A thumbnail is a smaller version of the original image, small enough to view many images simultaneously but large enough to distinguish features of the full size image. Within a collection of thumbnails, each thumbnail is usually the same size and displayed in a two dimensional layout, sorted by a simple feature of the image (e.g. name, date, filesize, etc). The desired outcome is to combine thumbnails as the technique that best conveys the content of an image with the advantages of FCA information retrieval for the annotated information associated with the image. This requires that the concept lattice representation has a different presentation and navigation paradigm compared to that of text documents.

This paper contains four more sections. In Section 2, a description of the FCA-background of ImageSleuth is presented. Section 3 explains the implementation, while Section 4 describes an example. Finally, Section 5 contains concluding remarks.

2 Using FCA to Browse Images

In this section, the mathematical structure underlying ImageSleuth and the resulting search and browse options are described. We assume that the reader is familiar with the basic notions of Formal Concept Analysis such as context, formal concept and conceptual scaling. For an introduction to FCA we refer to [GW99].

Following the approach used for browsing and searching of text documents, ImageSleuth computes concept lattices of contexts having the collection of images as objects and their annotated features as attributes. These features may be information about the depicted object annotated by hand as well as automatically extracted graphical information. In contrast to most approaches for FCA document retrieval, no line diagram of the lattice is displayed. Instead, following [KC00], the user is always located at one concept of the concept lattice. This allows thumbnails of the images to be shown as the extent of the present concept and thus to convey most of the graphical information characterising this concept. The intent is represented as a list of attributes. As no line diagram of the lattice is shown, lists of upper and lower neighbours are the only representation of the lattice structure around the present concept. Searching and browsing in the image collection then corresponds to moving from concept to concept in the lattice. By including new attributes in the intent, the user moves to a smaller concept where all images in the extent have these features. ImageSleuth offers the following possibilities to navigate in the concept lattice:

- Restriction of the set of attributes in consideration
- Move to upper/lower neighbour
- Search by attributes
- Search for similar objects (Query by example)
- Search for similar concepts

The possibility to restrict the set of attributes in consideration allows focus on the features that are relevant for the current navigation needs of the user.

Otherwise large sets of irrelevant attributes would increase the number of concepts and make search unnecessarily complex. ImageSleuth offers predefined sets of attributes (called *perspectives*) covering different aspects of the images. The user may combine these perspectives and include or remove perspectives during the search. Scale attributes are natural candidates for such attribute sets but other sets are allowed (for example, overlapping perspectives and perspectives which are subsets of other perspectives).

The option to search for similar concepts requires a similarity measure. In order to use this similarity together with the normal search or query-by-example, (where the user may describe the searched concept with attribute or object sets which are not intent or extent of a concept) we want the similarity measure to be defined for semiconcepts as introduced in [LW91] as a generalisation of concepts:

Definition 1. *A* semiconcept *of a context $\mathbb{K} := (G, M, I)$ is a pair (A, B) consisting of a set of objects $A \subseteq G$ and a set of attributes $B \subseteq M$ such that $A = B'$ or $B = A'$. The set of all semiconcepts of $\mathbb{K}$ is denoted by $\mathfrak{H}(\mathbb{K})$.*

Note that every concept is a semiconcept. The underlying structure of ImageSleuth is thus:

1. A context $\mathbb{K} := (G, M, I)$ with a collection of images as object set G, possible features as attribute set M and an incidence relation I assigning features to objects.
2. A collection $\mathcal{P}$ of subsets of M called perspectives. Every subset $\mathcal{A} \subseteq \mathcal{P}$ defines a subcontext $\mathbb{K}_{\mathcal{A}} := (G, \bigcup\mathcal{A}, I_{\mathcal{A}})$ with $I_{\mathcal{A}} := I \cap (G \times \bigcup\mathcal{A})$ of $\mathbb{K}$.
3. A similarity measure
$$s : \bigcup_{\mathcal{A}\subseteq\mathcal{P}} \mathfrak{H}(\mathbb{K}_{\mathcal{A}})^2 \to [0, 1]$$
assigning to every pair of semiconcepts of a subcontext $\mathbb{K}_{\mathcal{A}}$ a value between 0 and 1 which indicates the degree of similarity.

Since for every $\mathcal{A} \subseteq \mathcal{P}$ the contexts $\mathbb{K}_{\mathcal{A}}$ and $\mathbb{K}$ have the same object set and every attribute of $\mathbb{K}_{\mathcal{A}}$ is an attribute of $\mathbb{K}$ it follows for every $m \in \bigcup\mathcal{A}$ that $m^I = m^{I_{\mathcal{A}}}$. Since for $(A, B) \in \mathfrak{B}(\mathbb{K}_{\mathcal{A}})$ we have

$$A = B^{I_{\mathcal{A}}} = \bigcap\{m^{I_{\mathcal{A}}} \mid m \in B\} = \bigcap\{m^I \mid m \in B\}$$

it follows that A is the extent of a concept of $\mathfrak{B}(\mathbb{K})$. Therefore, $\phi(A, B) := (A, A^I)$ defines a map $\phi : \underline{\mathfrak{B}}(\mathbb{K}_{\mathcal{A}}) \to \underline{\mathfrak{B}}(\mathbb{K})$ and the image of ϕ is a $\wedge$-subsemilattice of $\underline{\mathfrak{B}}(\mathbb{K})$. In the following, the different navigation means based on this structure are described.

2.1 Restriction of the Attribute Set

By including different perspectives the user defines a subcontext of $\mathbb{K}$ in which all operations are performed. She may change this subcontext while browsing, thus obtaining at the present concept further information and search options. If at the concept $(A, A^{I_{\mathcal{A}}})$ the perspective $S \in \mathcal{P}$ is included (i.e. the set of attributes in consideration is increased), then ImageSleuth moves to the concept $(A^{I_{\mathcal{A}\cup\{S\}}I_{\mathcal{A}\cup\{S\}}}, A^{I_{\mathcal{A}\cup\{S\}}})$ of $\mathfrak{B}(\mathbb{K}_{\mathcal{A}\cup\{S\}})$. Since for $\mathcal{A} \subseteq \mathcal{P}$ and $S \in \mathcal{P}$ the extent of every concept of $\mathbb{K}_{\mathcal{A}}$ is an extent of $\mathbb{K}_{\mathcal{A}\cup\{S\}}$ we have $A = A^{I_{\mathcal{A}\cup\{S\}}I_{\mathcal{A}\cup\{S\}}}$ and the set of images shown does not need to be updated when a further perspective is included. This allows the addition of perspectives during the search without losing information. A similar strategy is known from Toscana (cp. [TJ02]) where the user moves through different scales. At every point the user may also remove a perspective S which takes her to the concept $(A^{I_{\mathcal{A}\setminus\{S\}}}, A^{I_{\mathcal{A}\setminus\{S\}}I_{\mathcal{A}\setminus\{S\}}})$. If in this way an attribute of $A^{I_{\mathcal{A}}}$ is removed from the current subcontext then the extent may be increased since $A^{I_{\mathcal{A}}} \subseteq A^{I_{\mathcal{A}\setminus\{S\}}}$.

2.2 Moving to Upper and Lower Neighbours

ImageSleuth uses most of its interface to show thumbnails of images in the extent of the chosen concept. As a result the user never sees the line diagram of a lattice. Instead, the lattice structure around the current concept is represented through the list of upper and lower neighbours which allow the user to move to super- or subconcepts. For every upper neighbour (C, D) of the current concept (A, B) the user is offered to remove the set $B \setminus D$ of attributes from the current intent. Dually, for every lower neighbour (E, F) the user may include the set $F \setminus B$ of attributes which takes her to this lower neighbour. By offering the sets $B \setminus D$ and $F \setminus B$ dependencies between these attributes are shown. Moving to the next concept not having a chosen attribute in its intent may imply the removal of a whole set of attributes. In order to ensure that the extent of the given concept is never empty it is not possible to move to the minimal concept.

2.3 Search and Query-by-Example

Browsing of the image collection is achieved by moving to neighbouring concepts. In many cases the user will want to go directly to images having a certain set of attributes $B \subseteq \bigcup \mathcal{A}$. This is offered by the search function which computes, for the selected attributes, the concept $(B^{I_{\mathcal{A}}}, B^{I_{\mathcal{A}}I_{\mathcal{A}}})$. Its extent is the set of all images having these attributes, its intent contains all attributes implied by B.

Another type of search is performed by the query-by-example function. Instead of defining a set of attributes, a set of objects A is defined as the sample set. The query-by-example function then computes the common attributes of these images (in the selected subcontext) and returns all other images having these attributes by moving to $(A^{I_{\mathcal{A}}I_{\mathcal{A}}}, A^{I_{\mathcal{A}}})$. In this way, query-by-example is the dual of the search function. While the search for images having certain attributes is not affected by the removal or addition of perspectives to the subcontext, query-by-example depends strongly on the selected subcontext. The more attributes taken into consideration, the smaller the set of images that have exactly the same attributes as the examples.

2.4 Similarity

The aim of query-by-example is to find objects which are similar to the objects in a given sample set. This is a narrow understanding of similarity implying equivalence in the considered subcontext; for the query-by-example function two objects g, h are "similar" in a subcontext $\mathbb{K}_{\mathcal{A}}$ if $g^{I_{\mathcal{A}}} = h^{I_{\mathcal{A}}}$. If the objects are uniquely described by the attributes in the chosen subcontext then query-by-example seldom yields new information. A more general approach is to define a similarity measure. In [Le99] several similarity measures on attribute sets are investigated. Similarity of two objects g and h is then described as the similarity of the attribute sets g' and h'. In order to use the grouping of objects provided by the formal concepts, ImageSleuth works with a similarity measure on semiconcepts which allows the return of a ranked list of similar concepts. We use semiconcepts since the set of sample images chosen by the user is not necessarily the extent of a concept. The similarity measure is derived from the following metric:

Definition 2. *On the set $\mathfrak{H}(\mathbb{K})$ of semiconcepts of a context $\mathbb{K} := (G, M, I)$ the metric $d : \mathfrak{H}(\mathbb{K}) \times \mathfrak{H}(\mathbb{K}) \to [0, 1]$ is defined as*

$$d((A, B), (C, D)) := \frac{1}{2}\left(\frac{|A \setminus C| + |C \setminus A|}{|G|} + \frac{|B \setminus D| + |D \setminus B|}{|M|}\right).$$

This definition formalizes the idea that two semiconcepts are close if there are few objects and attributes belonging to only one of them. In order to compare the number of objects and the number of attributes where they differ, these numbers are set in relation to the total number of objects or attributes. Semiconcepts with small distance are considered similar. ImageSleuth uses $1 - d((A, B), (C, D))$ as the similarity of (A, B) and (C, D).

For a similar purpose Saquer and Deogun introduced in [SD01] a related similarity measure as

$$s((A, B), (C, D)) := \frac{1}{2}\left(\frac{|A \cap C|}{|A \cup C|} + \frac{|B \cap D|}{|B \cup D|}\right).$$

This definition of similarity extends to semiconcepts (A, B), (C, D) if $A \cup C \neq \emptyset$ and $B \cup D \neq \emptyset$. In particular, the similarity $s((A, A'), (C, D)))$ is defined for every nonempty set A of objects and every concept $(C, D) \neq (G, \emptyset)$. For a sample set A of images, ImageSleuth uses a combination of both measures to return a ranked list of concepts similar to the semiconcept $(A, A^{I_{\mathcal{A}}})$.

The given metric on semiconcepts has two advantages. First, it allows the return of a list of similar concepts rather than just a list of images. This provides a reasonable grouping of the similar images and, since the attributes of the concepts are displayed, it shows in which way the images relate to the sample set.

Second, in contrast to other approaches such as graph distance, the number of different objects of two concepts is taken into account. Instead of counting only the attributes in which two concept intents differ, we assume that the significance of this difference is reflected in the difference of their corresponding attribute sets. If (A, B) is a concept and (C, D), (E, F) are upper neighbours of (A, B) with $|C| \leq |E|$ then the attributes in $B \setminus F$ are considered as more characteristic for the concept (A, B) than the attributes in $B \setminus D$. Thus, if $|D| = |F|$ then (C, D) is closer to (A, B) than (E, F) even though they differ from (A, B) in the same number of attributes. In this way, even an incomparable concept may be the closest. This contradicts the intuition that, for a concept, its sub- and superconcepts should be closest. Yet upper and lower neighbours are directly accessible by other navigation means. The advantage of the search for similar concepts for a given concept is that it offers a selection of (in the lattice order) incomparable but close concepts which are otherwise invisible.

As the original query-by-example function described above is the dual of a search this approach can be used for the search function, too. If a search is carried out for a set of attributes B, and if B' is empty, then the concept (B', B'') contains only the information that these attributes do not occur together. No images are returned as a result of this search, since there are no images having the required attributes. In this case, the user may be shown a list of concepts similar to or with small distance to the semiconcept (B', B).

3 Implementation

This section introduces the application ImageSleuth. Focus is placed on the dataset used for testing, its history, navigation overview and a method for resolving the empty extent search result.

3.1 Image Collection

The dataset used is taken from the popular computer game "*The Sims 2*". It features 412 objects of household funiture and fittings, described by 120 attributes which include in-game properties, suggestions for use and automatically extracted colour information. There are 7,516 concepts in the complete context. Each attribute of the context is assigned to one or more perspectives. In this dataset, 10 perspectives have been constructed.

3.2 History

The version of ImageSleuth presented here is the second version. The original prototype used concept neighbourhoods and include/remove attributes, but was limited to traversal between three mutually exclusive subcontexts via single objects. It underwent user-evaluation to test functionality and opinion of ImageSleuth's navigation paradigm. 29 honours level university students (from various disciplines) were asked to perform tasks and provide feedback on ImageSleuth v1. Results are overviewed in [DE05]. Results indicated that concept neighbourhoods offered a useful navigation method, users liked the "*grouping*

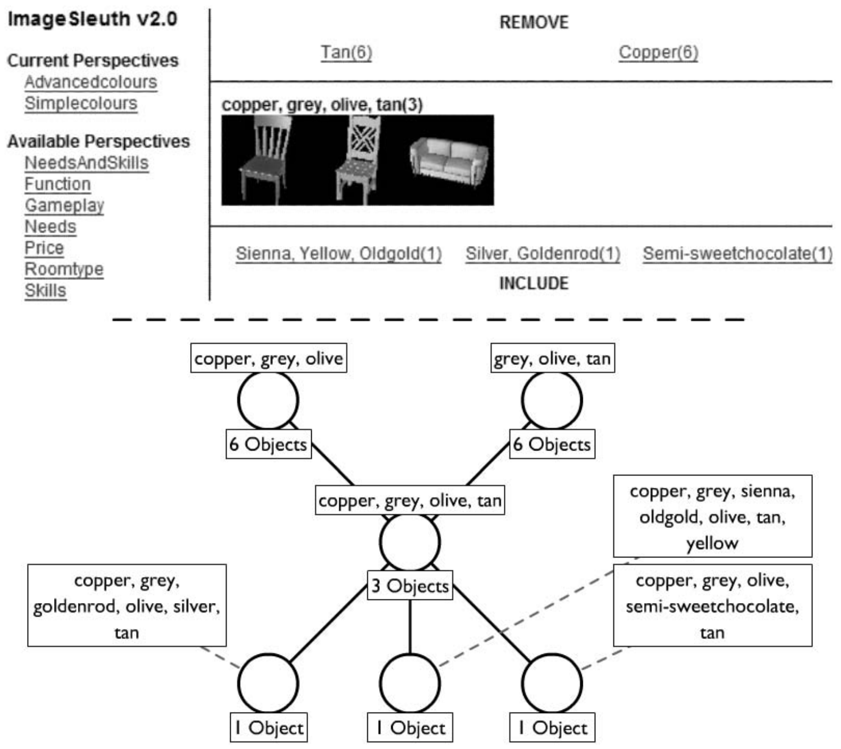


Fig. 1. An example screenshot of ImageSleuth and the lattice representation of the corresponding neighbourhood. The screenshot shows the four primary navigation functions of ImageSleuth. On the left is the listings of current and available perspectives (currently, advanced and simple colour perspectives are selected). Top and bottom show the remove and include functions respectively. The central pane shows the current concept; with intent listed as textual attributes and extent as thumbnailed images. The lattice neighbourhood shows the current concept at its centre.

of similar objects"[1] (concept extents) and the efficient searching by selection of defined attributes. Negative feedback included complaints about the interface and the systems performance. Analysis of the task results revealed the biggest problem: if a search included mutually exclusive attributes, it returned an empty extent, which left users confused. According to [Co99], making a user feel stupid is the worst possible software interaction fault.

The second version of ImageSleuth addressed the primary problems experienced by participants in the user testing sessions. These included interface layout, slow performance, inability to combine contexts and the empty extent search result problem. In the first version, *include* and *search* functionality was listed after the thumbnails, and users needed to scroll to the bottom of the page to continue navigation. This was repaired by partitioning the page into frames with

[1] A term used by more than one of the participants.

each frame assigned a set amount of screen space and function. This means a given functionality is always found in the same location regardless of conceptual position in, or content of, the dataset.

To address performance issues, the entire system (which was implemented as a single Perl script) was rewritten in C++ as a set of executables. The database was ported to PostGreSQL to take advantage of performance advantages for FCA systems outlined in [Ma06]. This process lead to a system that is roughly 10,000% faster.

ImageSleuth is accessed as a web site which allows simple access via a web browser. This also means that ImageSleuth is platform independent for users as all code is run on the server. Another reason for centralising the running of ImageSleuth is to allow logging of users' activities during usability testing sessions for analysis.

3.3 Empty Extent Search Result

The most common solution to concept searches in FCA, that result in an empty extent, is to offer attributes that can be removed from the search to supply a more general answer that meets a majority of search attributes. Most other forms of search (for example, text search) do not work this way - instead they supply the user with a list of results that are ranked by a relevance to the query. ImageSleuth tries to address this using the semiconcept search result and a combination of distance and similarity measures (see section 2.4). When a search is performed that would return the concept with an empty extent, the user can opt to allow the system to find and rank conceptually relevant concepts. This process is achieved by finding possible neighbours of the semiconcept and performing a bounded traversal which ranks the traversed concepts. These possible neighbours (Fig. 3, Line 3.) become the first concepts traversed. Each concept visited has its relevance calculated and stored. A test is applied to each concept visited to calculate whether it is to be used for further traversal. The test condition is based on the distance metric compared to a weighted average of the query concepts intent and extent size (Fig. 3, Line 8.). The condition is represented as:

$$\mathit{Dist}((A,B),(C,D)) \times \mathit{SearchWidth} < \tfrac{1}{2}(|A|/|G| + |B|/|M|)$$

where (A,B) is the query concept and (C,D) is the current concept of the traversal. $SearchWidth$ is a modifier to allow the search to be made wider or narrower. If the traversal is to continue, the concept's neighbourhood is added to the traversal list, the concept is marked as visited and the process continues (Fig. 3, Lines 9-11.).

Relevance is calculated as the average of the similarity scores which is presented to the user as a percentage.

4 Empty Extent Search Result Example

The following is a simple example of how ImageSleuth's semi-concept searching works. This example uses two perspectives, *Function* and *RoomType* which have

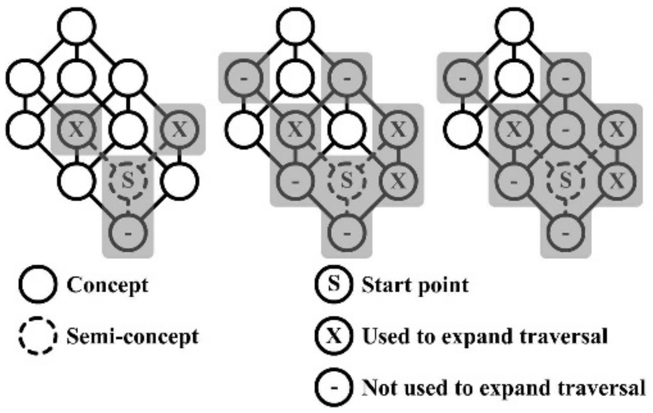


Fig. 2. An example of lattice traversal starting from a semi-concept. The traversal in this example is complete in 3 steps. The shaded area shows the computed concepts at each step.

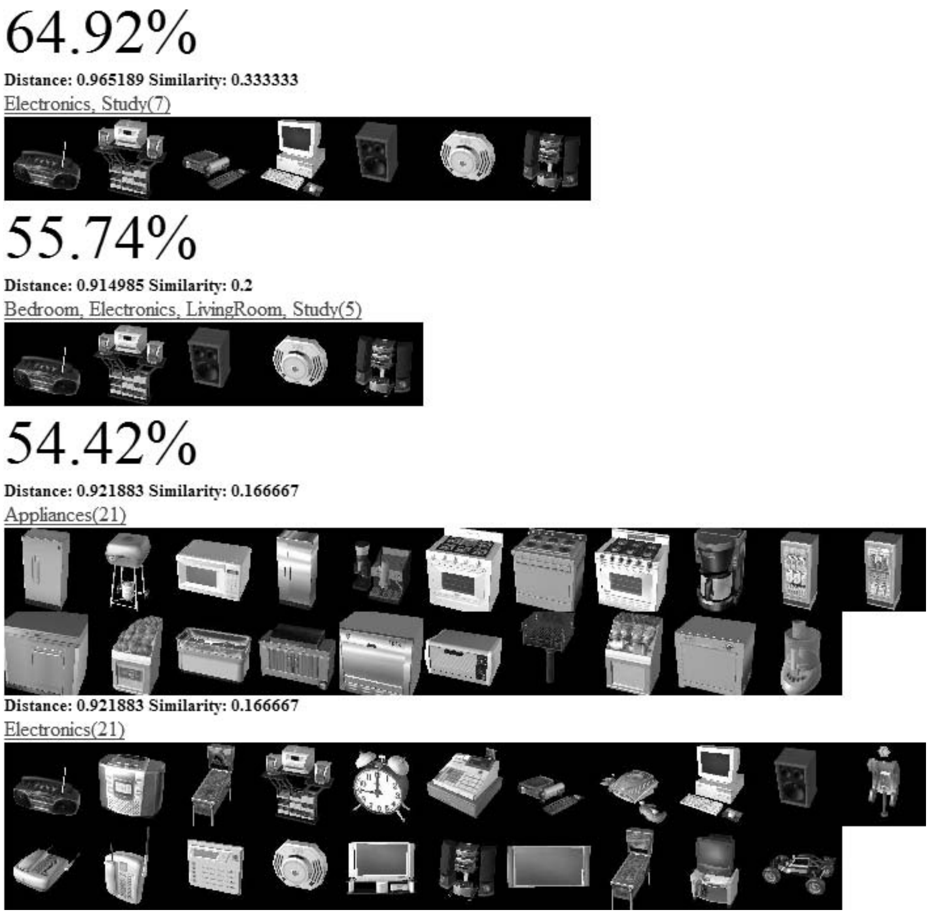
```
find_similar ( Concept: input, Number: width )
    input_size = size ( input.intent ) + size ( input.extent )
    candidate = upperNeigh ( input ) ∪ lowerNeigh ( input )
    exclude = ( input )
    while ( size ( candidate ) > 0 )
        concept = pop ( candidate )
        exclude = exclude ∪ concept

        compute and store rank information for concept.

        if ( distance ( input , concept ) × width
                < weightedAverage( input ) )
            candidate = candidate ∪ upperNeigh ( concept )
            candidate = candidate ∪ lowerNeigh ( concept )
            candidate = candidate / exclude
        end if
    end while
end
```

Fig. 3. Pseudocode representation of search traversal. Parameters are the starting concept or semiconcept (*input*) and a numeric value used to modify the width of the search (*width*).

20 attributes in total. The *Function* perspective is a simple nominal scale with each object having one *function* attribute. The *RoomType* perspective, on the other hand, is more complex with each object having zero or more *room type* attributes. With this context the complete lattice has 194 concepts.



64.92%

Distance: 0.965189 Similarity: 0.333333

Electronics, Study(7)

55.74%

Distance: 0.914985 Similarity: 0.2

Bedroom, Electronics, LivingRoom, Study(5)

54.42%

Distance: 0.921883 Similarity: 0.166667

Appliances(21)

Distance: 0.921883 Similarity: 0.166667

Electronics(21)

Fig. 4. Results of a concept traversal from the query "*Applications*, *Electronics*, *Study*" using the perspectives "*Function*, *RoomType*"

The query for this example will be "*Applications*, *Electronics*, *Study*", the first two attributes from the *Function* perspective and the remaining one from *RoomType*. *Function* being nominally scaled, the inclusion of two attributes from this perspective means that if the concept was completed it would result in the empty extent concept or $(\emptyset, M)$. Although this result is technically correct, it does not suit the query's intention.

To identify a concept that is more representative, a concept traversal is started using the semiconcept, $(\emptyset, (\mathit{Applications}, \mathit{Electronics}, \mathit{Study}))$. In this example, the traversal visits 12 concepts, four of which are conceptually close enough to extend the traversal. Consequently, only 6.19% of the total lattice is computed. The first three of five rankings are shown in Fig. 4. Relevance is shown as a large percentage, while individual distance and similarity scores are displayed below. Each result is displayed as a list of attributes representing the intent and a collection of thumbnails representing the extent. The highest ranking concept, with relevance 64.92%, has the intent (*Electronics*, *Study*), which is two of the three original query attributes. Following that, at 55.74%, is the concept with the intent (*Bedroom*, *Electronics*, *LivingRoom*, *Study*). The third ranking, at 54.42% relevance, has two concepts, with the intents (*Applications*) and (*Electronics*), which represent the mutually exclusive elements of the original query.

5 Conclusion

Presented is an image based navigation paradigm combining the methods of Formal Concept Analysis for information retrieval with the graphical information conveyed as thumbnails. This paradigm is formalised and realised via the ImageSleuth application which uses a collection of images taken from the game, *The Sims 2*.

It was required that the concept lattice representation used in ImageSleuth had a different presentation and navigation paradigm compared to that of text documents; in contrast to most approaches for FCA document retrieval, no line diagram of the lattice is displayed. In our approach, the user chooses perspectives of interest and is always located at one concept of the concept lattice, with the extent of the current concept displayed as thumbnails. Query-by-example and a method for ranking attribute search results when an exact match is not to be found are also described and exemplified in ImageSleuth. Also shown is how ImageSleuth has been improved from the previous version after testing and user evaluation.

References

[Co99] A. Cooper: The Lunatics are Running the Asylum, SAMS, 1999.

[CS01] R. Cole, G. Stumme: CEM – A conceptual email manager. In: B. Ganter, G. W. Mineau (eds.): Conceptual structures: Logical, linguistic, and computational issues. Proc. ICCS 2000. LNAI **1867**. Springer, Heidelberg 2000, 438–452.

[DE05] J. Ducrou, P. Eklund: Browsing and Searching MPEG-7 Images using Formal Concept Analysis. To Be Published, Feb 06 in: ACTA: IASTED AIA.

[GW99] B. Ganter, R. Wille: Formal concept analysis: mathematical foundations. Springer, Heidelberg 1999.

[KC00] M. Kim, P. Compton: Developing a Domain-Specific Document Retrieval Mechanism. In: Proc. of the 6th pacific knowledge acquisition workshop (PKAW 2000). Sydney, Australia.

[Le99] K. Lengnink: Ähnlichkeit als Distanz in Begriffsverbänden. In: G. Stumme, R. Wille (eds.): Begriffliche Wissensverarbeitung: Methoden und Anwendungen. Springer, Heidelberg 2000, 57–71.

[LW91] P. Luksch, R. Wille: A mathematical model for conceptual knowledge systems. In: H. H. Bock, P. Ihm (eds.): *Classification, data analysis, and knowledge organisation.* Springer, Heidelberg 1991, 156 – 162.

[Ma06] B. Martin, P. Eklund: Spatial Indexing for Scalability in FCA. In: Formal Concept Analysis: 4th International Conference (ICFCA 2006), Lecture Notes in Computer Science, Volume 3874, 2006, 205–220.

[SD01] J. Saquer, J. S. Deogun: Concept aproximations based on rough sets and similarity measures. In: Int. J. Appl. Math. Comput. Sci., Vol.11, No.3, 2001, 655 – 674.

[TJ02] P. Becker, J. Hereth, G. Stumme: ToscanaJ - An Open Source Tool for Qualitative Data Analysis,In: Advances in Formal Concept Analysis for Knowledge Discovery in Databases. Proc. Workshop FCAKDD of the 15th European Conference on Artificial Intelligence (ECAI 2002), 2002.

[TJ] The ToscanaJ Homepage. `<http://toscanaj.sourceforge.net>`

[VW95] F. Vogt, R. Wille: TOSCANA - a graphical tool for analyzing and exploring data. In: Proceedings of the DIMACS International Workshop on Graph Drawing (GD'94), 1995, 226 – 233.