Noname manuscript No.
(will be inserted by the editor)

Concept Similarity and Related Categories in Information Retrieval using Formal Concept Analysis

Frithjof Dau · Jon Ducrou · Peter Eklund

Received: March 31, 2009 / Accepted: date

Abstract The application of Formal Concept Analysis to the problem of information retrieval has been shown useful but has so far lacked any real analysis of the idea of relevance ranking of search results. SearchSleuth is a program developed to experiment with the automated local analysis of Web search using Formal Concept Analysis. SearchSleuth extends a standard search interface to include a conceptual neighborhood centered on a formal concept derived from the initial query. This neighborhood of the concept derived from the search terms is decorated with its upper and lower neighbors representing more general and special concepts respectively. SearchSleuth is in many ways an archetype of search engines based on Formal Concept Analysis with some novel features. In SearchSleuth, the notion of related categories – which are themselves formal concepts – is also introduced. This allows the retrieval focus to shift to a new formal concept called a sibling. This movement across the concept lattice needs to relate one formal concept to another in a principled way. This paper presents the issues concerning exploring, searching and ordering the space of related categories. The focus is on understanding the use and meaning of proximity and semantic distance in the context of information retrieval using Formal Concept Analysis.

Keywords Concept Lattice · Information Retrieval · Relevance Ranking · Semantic distance · Conceptual Similarity

P. Eklund and J. Ducrou
School of Information Systems and Technology
The University of Wollongong
Northfields Avenue, NSW, Australia
Tel.: +61-242-213874
Fax: +61-242-214843
E-mail: peklund@uow.edu.au, jonducrou@gmail.com

F. Dau
SAP Research CEC Dresden
Chemnitzer Strae 48
D-01187 Dresden, Germany
Tel.: +49-351-4811-6152
Fax: +49-6227-78-51425
E-mail: frithjof.dau@sap.com

1 Introduction

SearchSleuth is one of several Formal Concept Analysis-based Web metasearch engines that provide local analysis of search results for query refinement and labeled clustering [1,2]. These metasearch engines work via the creation of a conceptual space from polled search results – using search API's such as Yahoo – which are then organised via Formal Concept Analysis and displayed in a variety of ways. SearchSleuth is novel because previous efforts do not create a formal concept representing the query itself within the information space – meaning the conceptual space induced is representative of the results returned from the query, not the query terms. SearchSleuth overcomes this problem by creating a conceptual space as a neighborhood of the *search concept*: the formal concept derived from the search terms. The resulting neighborhood is comprised of "Generalizations" (upper neighbors), "Specializations" (lower neighbors) and "Related Categories" (also called siblings). Fig. 1 shows the SearchSleuth interface and these components.

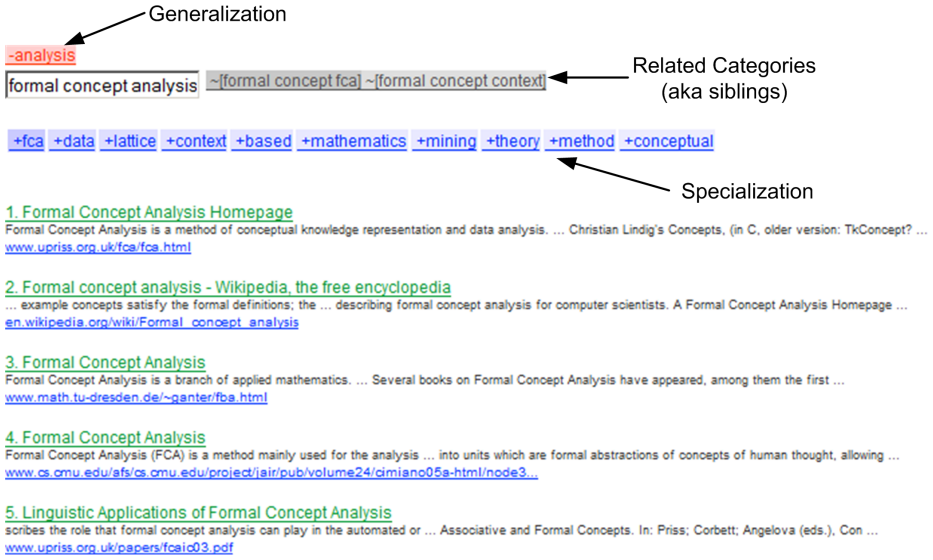

Fig. 1 The SearchSleuth interface, including the top five results after searching for `formal concept analysis`. Generalization/Specialization of the formal concept is shown above/below the text-box. "Related Categories" (or siblings) are to the right of the text-box. The relevance rankings of the result set conform to those supplied by the Yahoo search API.

By centering the conceptual space around the search concept, the resulting query refinement operations are closely coupled to the search terms used in the creation of the space. SearchSleuth was first presented at the Concept Lattices and their Applications Conference (CLA 2007) in October 2007 [3] but was subsequently improved and refined. At the International Conference on Conceptual Structures (ICCS 2008) [4] the authors elaborated the idea of "Related Categories" (siblings) and explored the notion of how "close" to the current formal concept the siblings concepts were. This paper combines both presentations in a complete, improved and extended form.

At this point there is no definitive study of the relative performance of SearchSleuth, compared to other metasearch engines – usability trials are currently being

conducted – but SearchSleuth is promising, its design and architecture sufficiently novel compared to other information retrieval efforts based on Formal Concept Analysis to be reported.

However, the main contribution of this paper is that it assays proximity in the topological neighborhood of a concept lattice and relates this idea to distance and similarity metrics. We seek to answer the question, how "close" or "similar" are the formal concepts in the local lattice neighborhood and how widely throughout the lattice do we need to search to find "relevant" formal concepts to the current search concept? This raises many questions: the definitions of topological proximity in a concept lattice; a study of the metrics for measuring concept similarity and distance; and an understanding of relevance for a formal concept.

The results of our study condition the design of SearchSleuth, and are also relevant to other document browsing and information retrieval that use Formal Concept Analysis. However, these results will also apply to many other applications of Formal Concept Analysis. This paper is therefore suited to a general readership in Artificial Intelligence and Mathematics with an interest in concept lattices. Situating our analysis within the content of browsing an information space using concept lattices motivates the study of local proximity and similarity of concepts in Formal Concept Analysis.

2 Navigation and Conceptual Neighborhoods: Background

SearchSleuth is the culmination of 10 years of research and development in applications of concept lattices [5–9, **?**]. It combines results from three disciplines: Formal Concept Analysis, information retrieval and document browsing using Formal Concept Analysis and usability and interface design, involving a iterative design science information systems methodology that evaluates how Formal Concept Analysis can be used effectively in an applied context.

2.1 Information Retrieval and Formal Concept Analysis

Web search is a difficult problem given the WWW's immeasurably large and constantly changing content and structure. Traditionally, Web search is initiated by entering one or more keywords into a search engine, then by reviewing the results until the appropriate site is found. This process requires a good understanding of the link between the search terms and the result set and the iterative nature of the retrieval on the Web is a well-recognised interaction paradigm for search [10].

Traditional Web search returns a ranked list of results. The result set takes the form of snippets: URL, title, a short summary and various details such as date last accessed. It is the text-based components of the snippet that aid in creating a conceptual information space of the results. One problem transitioning Web search to Formal Concept Analysis is that document ranking is often ignored when Formal Concept Analysis performs concept clustering. All results are treated equally when the formal concept is induced, and this issue is usually addressed by reproducing the rank ordering on any result set realized as the formal concept object's extent.

Another difficulty is that ranking methods use techniques such as link structure, popularity and referring pages. As such, it cannot be assumed that all results of a multiple term query will contain all queried terms. Even a single query term may yield a page that does not contain that term. This may seem counter intuitive, but if enough Web pages link to a result page that *does* contain the search term, and the term is very infrequent, that page's rank may be inflated to feature in the result set.

Carpineto and Romano's metasearch engine *CREDO* [1] (shown in Fig. 2) uses an iceberg lattice [11] to generate clusters for given search terms. Search terms are submitted to the Yahoo search API and the results returned input into a formal context. The formal context is built with the results as objects and the terms found in the result summaries as attributes. The most general concepts of the concept lattice are computed and displayed as a tree. The user can then interact with a DHTML tree widget (shown in Fig. 2 (left)) to view named clusters of the results.

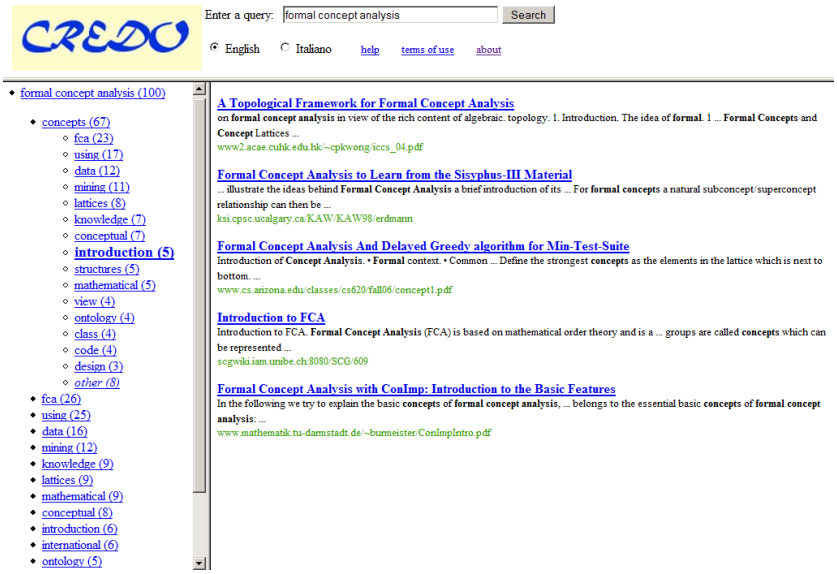

Fig. 2 The *CREDO* Web Search Application [1]: *CREDO* displays the uppermost concepts derived from keywords featured in search results.

CREDO does not include the search terms themselves when creating the context because it expects that all results contain all search terms. As Web search does not behave exclusively as a boolean search for keywords in document this assumption may not hold (as previously discussed) and the original search terms may (in some cases) be missing from the context. *CREDO* displays two levels of the lattice with users initially placed at the top-most concept. This pseudo-tree display is initially restricted to a single level, with user interaction a single label can be expanded at a time. This reduces clutter and confusion; labels that may appear in multiple branches are not shown because expansion of a label occurs one branch at a time.

Another FCA Web search application is Koester's *FooCA*[2]. *FooCA* provides one of the richest interfaces for the construction of a formal context based on Web search results. By taking result snippets as objects and terms found in the results as attributes, a formal context is created and presented to the user. A screenshot of *FooCA* is shown in Fig. 3. Like *CREDO* and *FooCA*, SearchSleuth uses the 'result has term' representation for building a formal context. This means each result from the search is considered an object, and all terms contained in the results title and summary are considered attributes. In the case of SearchSleuth (and optionally in *FooCA*) all words in the result set are stop-word filtered and stemmed to their lexical root. This reduces the size of the formal context and the complexity of the conceptual space. It

also reduces redundant terms with common lexical roots; it assumes that the terms 'wood' and 'woods' can be consolidated into a single stem. *FooCA* builds and displays the entire context derived from the search results. The context is build without query terms and the program provides powerful controls to influence the formation of the formal context. The user is shown the entire formal context in one cross-table as shown in Fig. 3 (lower). By viewing the entire information space as a cross-table, the user is never positioned within the conceptual space, rather a perspective of the entire space is defined by the query, its parameters and the resulting cross-table.

There are four alternative ways of presenting an information space of a concept lattice. The first is to reveal the entire conceptual space as a cross-table (as in *FooCA*). The second is to partially reveal the concept lattice with a pseudo-tree widget as in *CREDO*. The third is to display a line diagram of a complete concept lattice as in *CEM* [12]. A forth way is to situate its user in the information landscape, with a perspective centered on the query used to create it. The design logic is that this should give more insight into the meaning of the query with respect to other terms that appear in the result set (or surrounding it). This is the approach adopted by SearchSleuth.

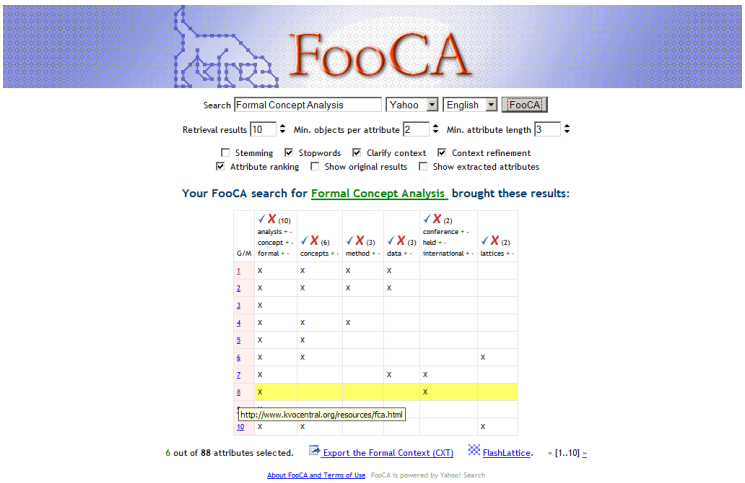

Fig. 3 Koester's *FooCA* Web Search Application [2]: An example of *improvement*. By allowing users to see and access the quality of the various terms in the current search results, successive search results increase retrieval quality.

2.2 Usability and Interface Design

The interface design of SearchSleuth is driven by the idea that we should adopt a local view of the concept lattice and this is supported by usability testing [13]. Kim and Compton [14, 15] presented a document navigation paradigm using Formal Concept Analysis and a neighborhood display. Their program, *KANavigator* uses annotated documents that can be browsed by keyword and displays the direct neighborhood (in particular the lower neighbors) as its interface. At the time, *KANavigator* emphasis on the use of labels as representations of single formal concepts (as opposed to a line diagram of the concept lattice) broke with many Formal Concept Analysis traditions. However, subsequent usability studies have shown that an interface based on a conceptual neighborhood has significant merit [16], simplifying the interaction and enabling

users non-expert in concept lattices to interact with a concept lattice without any instruction in lattice theory or Formal Concept Analysis.

ImageSleuth [17] is a tool for navigating collections of annotated images and for content-based retrieval using Formal Concept Analysis. It combines methods of Formal Concept Analysis for information retrieval with the graphical information of the concept's extent conveyed in image thumbnails. In addition to established methods such as keyword search and inclusion of attributes that move the user to upper/lower neighbors, a query-by-example function and restrictions to attribute sets are included. Metrics on formal concept similarity are discussed and applied to ranking and automated discovery of relevant concepts: from both concepts and semi-concepts. ImageSleuth (shown in Fig. 4), used and extended this interface idea to explore image collections using Formal Concept Analysis. By showing upper and lower neighbors of the current concept and allowing navigation to these concepts, users can refine or generalise their position in the information space. This is aided by the use of pre-defined conceptual scales (called perspectives) that can be combined to define the attribute set of the lattice which forms the conceptual space (see Fig. 4 (left)). In Fig. 4 we see that two perspectives are activated, `SimpleGameplay` and `Advancedcolours` and the current formal concept consists of 6 objects (rendered as thumbnail images) which all share the attributes {*Environment, Needs, red*}. Two upper neighbor formal concepts can be reached by removing `Red` or `Environment` and the numbers represented on each attribute are the number of objects in the extent of each, i.e., removing `Environment` will induce a formal concept with 7 images in its extent and removing `Red` will induce a formal concept with 221 images in its extent.

Fig. 4 ImageSleuth presents only the extent of the current concept as thumbnails and generalizations/specializations by removal/addition of attributes to reach the upper and lower neighbors (top/bottom). Pre-defined scales (perspectives) are displayed on the left.

Likewise, three lower neighbor concepts are available through the addition of any of `Orangered`, `Maron,Brown` and `Coral`, the numbers on each represent the cardinality of the extent. A minor association rule is evident since all images with `Maron` also contain `Brown` and visa versa so `Maron,Brown` decorates the downward navigation link.

As is evident from Fig. 4, ImageSleuth uses most of its interface to show thumbnails of images in the extent of the current concept. As a result the user never sees a line diagram of a concept lattice. Instead, the lattice structure around the current concept is represented through the list of upper and lower neighbors that allow navigation to super- or sub-concepts.

SearchSleuth employs the same conceptual neighborhood paradigm for interaction. Unlike ImageSleuth, SearchSleuth's formal context is not static, so the space is rebuilt with each navigation step. This is because computing the entire domain, the Web, as a conceptual neighborhood would be computationally prohibitive and so the refinement process has to incrementally change the current formal context based on subsequent refinements. Incrementality in this way also maintains diversity in the search context by allowing the user to navigate to "alternative" formal concepts, other than the one induced by the search terms. These are called "Related Categories" or siblings and are shown to the right of the text-box in Fig. 1.

3 The Domain of SearchSleuth: Context Building

SearchSleuth uses the 'result has term' representation to build a formal context. The "search" formal concept for SearchSleuth (A_s, B_s) is created on demand for each query; this suits the dynamic nature of the Web but it can be expensive to compute. The formal objects are the individual snippets, and the formal attributes are the terms contained in the title and summary of each result. But before the search formal concept (A_s, B_s) is computed we need to reduce the context to something manageable.

Recall, a *formal context* $\mathbb{K} := (G, M, I)$ is a triple where G is a set of formal *objects*, M is a set of *attributes* and I is an *incidence* relation between the objects and the attributes. $I \subseteq G \times M$ is a binary relation where $(g, m) \in I$ is read "object g has attribute m" and written as gIm. The formal context $\mathbb{K}$ can be pictured as a cross-table and since the complexity of computing the concept lattice is $O(|C||M|(|G| + |M|)$ [18], so too a fundamental concern in computing concept lattices is reducing $|M|$. Also, since the cardinality of the object set $|G|$ is fixed by the Yahoo search API, progress can only be made by reducing $|M|$, the size of the set of attributes in $\mathbb{K}$.

This is achieved in two ways, firstly attributes (terms) are extracted from the title and summary after stemming and stop-word filtering has been performed. Stemming reduces words to their lexical root (e.g. `jump`, `jumping` and `jumps` are all reduced to `jump`). Stop-word filtering also removes words without individual semantic merit, for example `a`, `the` and `another`. Both stemming and stop-word removal are standard techniques in information retrieval and have the effect of reducing $|M|$ without a noticeable reduction in semantic quality.

The formal context $\mathbb{K}$ is is then further reduced by removing attributes (terms) with low support in the result set. Every attribute with less than 5% of the objects in the incidence relation is removed. This is called *context reduction* and differs from *clarification* [19] where both attributes and objects are removed. Reduction decreases the computational overhead involved in computing the concept lattice by reducing the cardinality of the attribute set $|M|$. Experience shows that reduction rarely effects the computed conceptual neighborhood as the terms removed are scarce within the conceptual space. In the example given in Fig 1 for the query *Formal Concept Analysis*, $|M|$ is reduced by an order a magnitude.

For a subset of the objects, $A \subseteq G$ we define the set of common attributes $A^{\uparrow}$ as:

$$A^{\uparrow} := \{m \in M \mid (g, m) \in I, \forall g \in A\}$$

and dually, for a subset of attributes, $B \subseteq M$ we can define the set $B^{\downarrow}$ of objects having all the attributes from B as:

$$B^{\downarrow} := \{g \in G \mid (g, m) \in I, \forall m \in B\}$$

For convenience $A^{\uparrow}$ and $B^{\downarrow}$ are often written as A' and B'.

Once the formal context is reduced as $\mathbb{K}'$, the search concept (A_s, B_s) can computed. This is done by taking the query terms as attributes and deriving the formal concept using the reduced context $\mathbb{K}'$, namely, (A_s, B_s) is a *formal concept* of (G, M, I) iff:

$$A_s \subseteq G,\ B_s \subseteq M,\ A_s = B_s,\ \textit{and}\ B_s = A_s.$$

The set A is called the *extent* and B the *intent* of the formal concept (A, B). The upper neighbors of this formal concept are then derived and used to expand the reduced context $\mathbb{K}'$ to $\mathbb{K}''$. This is done by querying the Yahoo search API with the attributes of each upper neighbor and inserting the results into a new context $\mathbb{K}''$. Results for these ancillary/expanded searches to produce $\mathbb{K}''$ are limited to fewer results, namely 50% of the size of the initial result set. The process of building the new context $\mathbb{K}''$ increases the number of terms in the conceptual space based on a single level of generalization. It makes the conceptual space larger and richer with objects in the immediate neighborhood of the current search concept. (A_s, B_s) is then recomputed from $\mathbb{K}''$.

In summary, the original context is reduced via the removal of attributes via stemming and stop-word removal and low support attributes, then the context is expanded by those attributes of the upper neighbors to give more meaningful focus to the neighborhood of the current formal concept (A_s, B_s). The amount of time spent reducing and expanding the context, and computing and recomputing (A_s, B_s), is incidental to the latency of the search API, namely this process occurs in 1/10th of the time it takes to transfer the results from the Yahoo search API.

4 Building the Information Space

As explained, after the context is expanded, the search concept (A_s, B_s) is recomputed as it has been enhanced by the production of $\mathbb{K}''$. The upper and lower neighbors of (A_s, B_s) are computed next. A concept C is said to be the upper neighbor (or cover) of Z iff we have $C > Z$, and/but there is no concept Y with $C > Y > Z$. A concept C is said to be the lower neighbor of (or covered by) a concept Z iff we have $C < Z$, and/but there is no concept Y with $C < Y < Z$. The sets of upper and lower neighbors of a concept C are written as $UN(C)$ and $LN(C)$ respectively.

The DownSet (DS) and UpSet (US) of a formal concept C are defined as follows:

$$DS(C) := \{y \mid y \leqslant x \text{ for an } x \in C\} \qquad US(C) := \{y \mid y \geqslant x \text{ for an } x \in C\}$$

Consider now a set of concepts X, then $UN(C)$ is defined as the union of all upper neighbors of the concepts in X. Dually, consider the set of concepts Y, $LN(C)$ is defined as the union of all lower neighbors of the concepts in Y, i.e., we set:

$$UN(X) := \bigcup\{UN(C) \mid C \in X\} \qquad UN(Y) := \bigcup\{LN(C) \mid C \in Y\}$$

The next step is to compute the related categories or sibling concepts. In fact, we define a hierarchy of three different notions of siblings called Type I, Type II and Type III siblings. Child Siblings (CS) and Parent siblings (PS) are defined as follows:

$$CS(C) := UN(LN(C))\backslash\{C\} \qquad PS(C) := LN(UN(C))\backslash\{C\}$$

Put another way, siblings constitute formal concepts created by the removal of an attribute (or attributes) that define an upper neighbor (UN), and the inclusion of an attribute (or attributes) that define a lower neighbor (LN).

Exact Siblings (ES or Type I siblings) are those which are both Parent and Child siblings, since they represent a stricter version of the notion of siblings they are referred to as Type I siblings and PS and CS are termed Type II siblings:

$$ES(C) := [LN(UN(C)) \cap UN(LN(C))] \backslash \{C\}$$

General Siblings (GS – Type III siblings) define an even broader set of sibling concepts and are defined:

$$GS(C) := [DS(UN(C)) \cap US(LN(C))] \backslash (\{C\} \cup UN(C) \cup LN(C))$$

namely, anything strictly between some lower and some upper neighbor.

Child Siblings (CS), Parent Siblings (PS), and Exact Siblings (ES) form anti-chains, but General Siblings (GS) do not.

An example is shown in Fig. 5 in each of the two lattices presented the ES, CS, PS and GS of the concept marked C are listed.

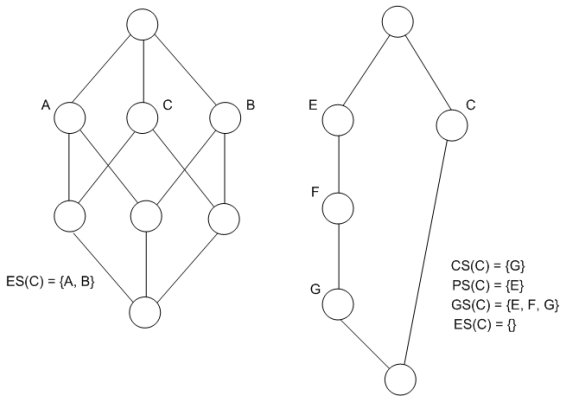

Fig. 5 Diagram demonstrating the *Parent Sibling (PS), Child Siblings (CS), Exact Sibling (ES)* and *General Siblings (GS)* concepts of the concept labeled with a C in two lattices.

Apply the same idea in ImageSleuth and using the full intent as labels of sibling concepts, a display is rendered for the user (shown in Fig. 1). The primary feature of the display is the text-box which the query is entered. It is considered representative of the search concept, and thus is centered in the display. Upper neighbors are shown above this text entry box, displayed as text labels (shown in Fig. 1). The labels are the attributes which would be removed to navigate to that upper neighbor. These labels are preceeded by a minus symbol (`-`) to reinforce the notion of *removal*, in this case there is a single label `-analysis` which leads to the upper neighbor.

Lower neighbors are similarly displayed (also indicated with arrows in Fig. 1), but placed below the text-box. These labels are the attributes which would be added to navigate to that lower neighbor. Like upper neighbor labels, these labels are preceeded by a symbol to reinforce the labels meaning, namely the plus symbol (`+`) and the notion of *inclusion*, these are the labels `+fca, +data ,+lattice, +context, +based, +mathematics, +mining, +theory +method` and `+conceptual`.

The display order of the upper and lower neighbors is defined by extent size, larger extents displayed first (left-most). Extent is representative of the 'importance' or prominence of the labels within the current information space. Extent is also used to aid in

the coloring of the label's background. The higher the extent on a lower neighbor, the deeper the (blue) block shade behind that concepts label. Upper neighbors are displayed with the same principle but with (red) block shading.

One method for dealing with the return of empty-extents from term-based searching is to provide users with a list of the terms entered so that they can incrementally remove terms to unconstrain the search. SearchSleuth explores an approach based on variations on defined distance and similarity metrics in the Formal Concept Analysis literature [20,21] in order to find similar relevant concepts.

Exact Siblings (ES) or "Related Categories" are shown to the right of the text entry box (indicated by the arrow in Fig. 1) and are indicative of related concepts. The complete intent of these concepts is displayed within square brackets preceded by a tilde (`~[...]`). This helps group the concept intents and aids distinguishing between related concepts. Unlike upper and lower neighbors, Exact Siblings are ordered by *similarity*. The similarity metric is based on work by Lengnink [20] and was initially adapted for use in ImageSleuth. The metric uses the size of the common objects and attributes of the concepts. For two concepts (A, B) and (C, D), we set:

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

The similarity measurement in Eqn (1) is used to order the exact sibling concepts, while highlighting remains based on extent size. Coloring on sibling labels is based on grey block shading, the ES's included in Fig. 1 are the labels, ∼`[formal concept fca]` and ∼`[formal concept context]`.

By clicking any of the possible concept labels, the query is set to the intent of the selected concept and the query process is restarted. This is an important restructuring step as a change in the query will change the result set, and in order for the information to be valid a new context needs to be recomputed.

Looking back to Fig. 1, we see the search concept shown is based on the query `Formal Concept Analysis`. It shows a single upper neighbor `-analysis` which interestingly shows an implication that `+formal` and `+concept` are implied by `+analysis`, so `+analysis` always appears when `+formal` and `+concept` appear. The first of the lower neighbors is the acronym `+fca`. This is followed by the terms `+lattice`, `+mathematics` and `+theory`. These labels are good examples of specialization from the search concept.

This conceptual neighborhood induced in Fig. 1 is based on 115 formal objects. The initial number of attributes for this example was 623, after reduction of the context through the removal of stop words, stemming and low support attributes, the number of attributes is in $\mathbb{K}''$ is reduced to 40. This offers a tremendous reduction in context complexity, and therefore computation time but these numbers also reflect the need to search a subset of conceptual neighborhood to maintain relevance and search diversity.

A main question in the design of SearchSleuth is whether the definition of Exact Siblings (ES) provides sufficient space for proximity search of neighboring categories. Should we also be looking at Child Siblings (CS), Parent Siblings (PS) and General Siblings (GS) siblings as well? The remainder of the paper addresses this issue in detail.

5 Conceptual Distance, Similarity and Siblings

We have two measures to consider the proximity of formal concepts. In addition to similarity (s) defined in Eqn. (1) we also have for two formal concepts (A, B), (C, D),

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

where dg is the *distance* of the concepts (A, B), (C, D) [20]. To ease comparison between the two measures, let

$$dl((A,B),(C,D)) := 1 - s((A,B),(C,D))$$

Let us first observe that dl and dg are metrics in the mathematical understanding, i.e., dg satisfies for arbitrary concepts x, y, z: $dg(x, y) \geqslant 0$ and $dg(x, y) = 0 \Leftrightarrow x = y$ (non-negativity and identity of indiscernibles), $dg(x, y) = dg(y, x)$ (symmetry), and $dg(x, z) \leqslant dg(x, y) + dg(y, z)$ (triangle inequality). The triangle inequalities can easily be shown by straight-forward computations, and the remaining properties are easily to be seen. Next, note that we have

$$dl((A,B),(C,D)) = \frac{1}{2}\left(\frac{|A \cup C| - |A \cap C|}{|A \cup C|} + \frac{|B \cup D| - |B \cap D|}{|B \cup D|}\right) \tag{2}$$

$$dg((A,B),(C,D)) = \frac{1}{2}\left(\frac{|A \cup C| - |A \cap C|}{|G|} + \frac{|B \cup D| - |B \cap D|}{|M|}\right) \tag{3}$$

Comparing Eqns. (2) and (3), we see that they differ in that in (2), we divide through $|A \cup C|$ and $|B \cup D|$, whereas in (3), we divide through $|G|$ and $|M|$. Therefore, dl is a local distance, focusing on the shared attributes and objects of the two formal concepts being compared, and dg is a global distance, using all the attributes and objects in the formal context (hence the names dl and dg). The choice of measurement to use therefore depends on the sensitivity of the proximity measure required. The preferred approach for SearchSleuth is proximity in the conceptual neighborhood to the current formal concept. Therefore, the local measure (dl) is considered most suitable.

One can however easily combine the two measures. Let $l \in [0, 1]$, measuring the desire of a local point of view ($l = 0$ means the user wants a purely global point of view, and $l = 1$ means the user wants a purely local point of view). Then the resulting distance ($dist_l$) measure is,

$$dist_l((A,B),(C,D)) := l \cdot dl((A,B),(C,D)) + (1-l) \cdot dg((A,B),(C,D)). \tag{4}$$

6 Relationships between Metric and Sibling Types Explored

The design approach of SearchSleuth is to explore the 'conceptual neighborhood' of a given concept. To grasp the 'conceptual neighborhood' idea SearchSleuth takes advantage of two fundamentally different notions of neighboorhood. On the one hand, we use the lattice-theoretic notions of sibling, which does not take the sizes of the concept-extents or intents into account. On the other hand, similarity and distance are set-theoretic notions (they do not take the lattice-order into account). In the next two paragraphs, we investigate the different types of siblings and then the two similarity metrics.

As we have seen earlier, the notion of a siblings is finely-grained divided into exact siblings (ES= Type I), parent- and child siblings (PS or CS = Type II), and general siblings (GS = Type III). This provides a hierarchy of sibling types: Each Type I sibling is a Type II sibling, and each Type II sibling is a Type III sibling.

Apart from this hierarchy of type inclusion, we cannot generalize about the number of siblings in each of the different siblings types. To be more precise: If $n_{\mathrm{I}}, n_{\mathrm{II}}, n_{\mathrm{III}} \in \mathbb{N}_0$ are three numbers with $n_{\mathrm{I}} \leqslant n_{\mathrm{II}} \leqslant n_{\mathrm{III}}$ and $n_{\mathrm{II}} \neq 1$, then there exists a lattice with an element c that has n_{I} Type I siblings, n_{II} Type II siblings, and n_{III} Type III siblings. An example for such a lattice is given below in Fig 6. In Fig 6, for each sibling of C, the most special sibling type is inscribed on the vertex. That is in the diagram, n_{I} vertices are labelled with 'I', $n_{\mathrm{II}} - n_{\mathrm{I}}$ vertices are labelled with 'II', and $n_{\mathrm{III}} - n_{\mathrm{II}} - n_{\mathrm{I}}$ vertices are labelled with 'III'.

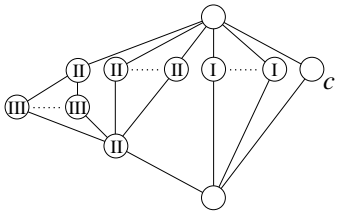

Fig. 6 Example lattice showing the spread of Type I, II and III siblings from the concept C.

For the notions of local (dl) and global (dg) distance, a somewhat similar consideration applies. Due to Eqns. (2) and (3), in each lattice, for any concepts x, y, we have

$$dl(x, y) \geqslant dg(x, y) \quad .$$

On the other hand, there are examples (one is given in the following subsection) of lattices where there are two concepts c, n which are arbitrary close with respect to the local metric, but arbitrary distant with respect to the global distance metric.

The question remains whether there are dependencies between the lattice-theoretic (i.e., siblings) and the set-theoretic (i.e., metrics) notions of conceptual neighborhood. We will investigate some examples in the following sections. As these examples will show, the two notions are somewhat orthogonal but help in determining the most appropriate related categories in SearchSleuth.

6.1 Exact Siblings (ES) and Proximity Metrics

We first consider an example where we have an exact sibling n of a concept c, and we investigate whether we can draw some conclusions about the local or global distance between c and n. The example we consider is the following concept-lattice:

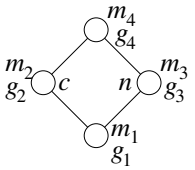

Fig. 7 Example for comparing the local and global distance for an Exact Sibling.

In this diagram and the following diagrams in Figs. 8, 9, 10, the g_i and m_i do *not* denote objects or attributes, but the *numbers* of objects resp. attributes which generate the concept, i.e., the g_i and m_i are the numbers of objects and attributes in the common diagrams of concept lattices. For example, if (G_1, M_1) is the bottom concept and for $c = (G_2, M_2)$, we have $g_2 = |G_2| - |G_1|$. Two concepts from Fig. 7 are given names, namely c and n. We have:

$$s(c,n) = \frac{1}{2}\left(\frac{g_1}{g_1+g_2+g_3} + \frac{m_4}{m_2+m_3+m_4}\right)$$
$$dg(c,n) = \frac{1}{2}\left(\frac{g_2+g_3}{g_1+g_2+g_3+g_4} + \frac{m_2+m_3}{m_1+m_2+m_3+m_4}\right)$$

For fixed $g_2, g_3, g_4, m_1, m_2, m_3$ (e.g., $g_2 = g_3 = g_4 = m_1 = m_2 = m_3 = 1$), we have

$$\lim_{\substack{g_1\to\infty \\ m_4\to\infty}} dl(c,n) = 1 - \frac{1}{2}(1+1) = 0 \qquad \text{and} \qquad \lim_{\substack{g_1\to\infty \\ m_4\to\infty}} dg(c,n) = \frac{1}{2}(0+0) = 0$$

i.e., c and n can be arbitrarily similar with respect to both dl and dg.

On the other hand, for fixed g_1, g_3, m_3, m_4, we have

$$\lim_{\substack{g_2\to\infty \\ m_2\to\infty}} dl(c,n) = 1 - \frac{1}{2}(0+0) = 1 \qquad \text{and} \qquad \lim_{\substack{g_2\to\infty \\ m_2\to\infty}} dg(c,n) = \frac{1}{2}(1+1) = 1$$

(similar for g_2, m_3, and g_3, m_2, and g_3, m_3), i.e., c and n can be arbitrarily different (again with respect to s and to dg).

Now let $\varepsilon_1, \varepsilon_2 > 0$. Let g_3, g_4, m_1, m_3 be fixed. By *first* choosing g_2 and m_2 sufficiently large, we can obtain $s(x,n) < \varepsilon_1$, and by *then* choosing g_1, m_4 sufficiently large (which does not affect $s(c,n)$), we can obtain $dg(c,n) < \varepsilon_2$, i.e., we can achieve in a local understanding (i.e., w.r.t. dl), the concepts c and n are very similar, whereas in a global understanding ((i.e., w.r.t. dg), the concepts c and n are very distant. To summarize, even for the most special case of being an exact sibling n of a given concept c, we cannot draw any conclusion about the local or global distance between c and n.

6.2 The Proximity of Type I Siblings versus Non-Siblings Concepts

A concept n which is a sibling for a given concept c belongs, from a lattice-theoretic point of view, to the conceptual neighborhood of c; a concept x which is not a sibling of c does not belong to the conceptual neighborhood. Is this property reflected by the distances dl and dg? We consider again an example where n is an exact sibling of c.

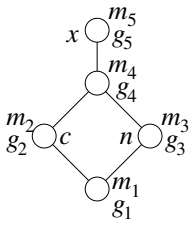

Fig. 8 Example for comparing Exact Siblings to Non-Siblings.

6.2.1 Similarity

In terms of similarity we have:

$$s_n := s(c,n) = \frac{1}{2}\left(\frac{g_1}{g_1+g_2+g_3} + \frac{m_4+m_5}{m_2+m_3+m_4+m_5}\right)$$
$$s_x := s(c,x) = \frac{1}{2}\left(\frac{g_1+g_2}{g_1+g_2+g_4+g_5} + \frac{m_5}{m_2+m_4+m_5}\right)$$

Note that we can have $g_1 = 0$, $m_1 = 0$, $g_4 = 0$, and $m_5 = 0$, but all other numbers must be $\geqslant 1$. Now, n could be more similar to c than x, equally similar, or less similar, as the following examples show.

g_1	g_2	g_3	g_4	g_5	m_1	m_2	m_3	m_4	m_5	$2 \cdot s_n = 2 \cdot s(c,n)$	$2 \cdot s_x = 2 \cdot s(c,x)$
1	1	1	0	1	1	1	1	1	1	$1/3 + 2/4 = 5/6$	$2/3 + 1/3 = 1$
1	1	1	1	1	0	1	1	1	1	$1/3 + 2/4 = 5/6$	$2/4 + 1/3 = 5/6$
1	1	1	1	1	0	1	1	1	0	$1/3 + 1/3 = 2/3$	$2/4 + 0/2 = 1/2$

Running a computer-program checking all values for g_i and m_i with a threshold of 8 yields:

$s(c,n) > s(c,x)$	$s(c,n) < s(c,x)$	$s(c,n) = s(c,x)$
804.068.208	913.112.127	2.746.449

Therefore, the cases in which $s(c,n) > s(c,x)$ and $s(c,n) < s(c,x)$ do not significantly differ and we cannot conclude that siblings are *generally* more similar than non-siblings. We therefore conclude by counterexample that siblings are not generally more similar to a given formal concept than non-siblings.

6.2.2 Distance

Similarly, repeating the analysis in terms of the distance metric dg in Fig 8, we have:

$$d_n := s(c,n) = \frac{1}{2}\left(\frac{g_2+g_3}{g_1+\cdots+g_5} + \frac{m_3+m_4}{m_1+\cdots+m_5}\right)$$
$$d_x := s(c,x) == \frac{1}{2}\left(\frac{g_4+g_5}{g_1+\cdots+g_5} + \frac{m_2+m_4}{m_1+\cdots+m_5}\right)$$

Now the question becomes: is n neccessarily less distant to s than x? Now the examples are:

g_1	g_2	g_3	g_4	g_5	m_1	m_2	m_3	m_4	m_5	$2 \cdot d_n = 2 \cdot dg(c,n)$	$2 \cdot d_x = 2 \cdot dg(c,x)$	result
0	1	1	0	2	0	1	1	2	0	$2/4 + 2/4 = 1$	$2/4 + 3/4 = 5/4$	$d_1 < d_2$
0	1	1	1	1	0	1	1	1	0	$2/4 + 2/3 = 7/6$	$2/4 + 2/3 = 7/6$	$d_1 = d_2$
0	1	1	0	1	0	1	1	1	0	$2/3 + 2/3 = 4/3$	$1/3 + 2/3 = 1$	$d_1 > d_2$

Now the computer program output for a threshold of 8 is:

$dg(c,n) > dg(c,x)$	$dg(c,n) < dg(c,x)$	$dg(c,n) = dg(c,x)$
908.328.121	788.136.280	23.462.383

Again the cases $dg(c,n) > dg(c,x)$ and $dg(c,n) < dg(c,x)$ do not significantly differ.

To summarize, even for the most special case of being an exact sibling n of a given concept c, we cannot draw any conclusion that n is closer to c compared to a non-sibling using distance or similarity measures.

6.3 The Proximity of Type II versus Type III Siblings Concept

There are clearly different strengths of being a sibling depending on sibling type. We still could hope that this is reflected in the metrics. In the example provided in Fig 9, we consider Type II siblings of a concept c with the more general Type III siblings and check whether the Type II siblings are closer to c than the Type III siblings. In terms of similarity we have:

$$s_1 := s(c, n_1) = \frac{1}{2}\left(\frac{g_1}{g_1+g_2+g_3} + \frac{m_6}{m_2+m_3+m_4+m_5+m_6}\right)$$
$$s_2 := s(c, n_2) = \frac{1}{2}\left(\frac{g_1}{g_1+g_2+g_3+g_4} + \frac{m_6}{m_2+m_4+m_5+m_6}\right)$$
$$s_3 := s(c, n_3) = \frac{1}{2}\left(\frac{g_1}{g_1+g_2+g_3+g_4+g_5} + \frac{m_6}{m_2+m_5+m_6}\right)$$

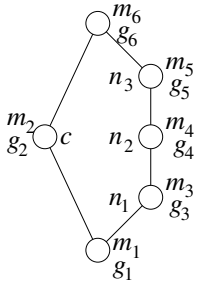

Fig. 9 Example for comparing Type II and Type III Siblings.

In this example, there is no order relationship between s_1, s_2, and s_3. We can have $s_1 < s_2 < s_3$ or $s_3 < s_1 < s_2$ or $s_1 = s_2 < s_3$ etc. Any order combination is possible. The following table shows examples for all possible strict orders of s_1, s_2, s_3 (examples for cases like $s_1 = s_2 < s_3$ are left out due to space limitations).

g_1	g_2	g_3	g_4	g_5	g_6	m_1	m_2	m_3	m_4	m_5	m_6	$2 \cdot s_1$	$2 \cdot s_2$	$2 \cdot s_3$	result
1	1	1	1	1	1	1	1	2	1	1	2	$1/3+2/7$	$1/4+2/5$	$1/5+2/4$	$s_1 < s_2 < s_3$
1	1	2	1	2	1	1	2	2	1	2	2	$1/4+2/9$	$1/5+2/7$	$1/7+2/6$	$s_1 < s_3 < s_2$
1	1	1	1	1	1	1	1	1	1	1	2	$1/3+2/6$	$1/4+2/5$	$1/5+2/4$	$s_2 < s_1 < s_3$
1	1	1	1	1	1	1	1	1	1	2	2	$1/3+2/7$	$1/4+2/6$	$1/5+2/5$	$s_2 < s_3 < s_1$
2	2	2	1	2	1	1	1	2	1	2	1	$2/6+1/7$	$2/7+1/5$	$2/9+1/4$	$s_3 < s_1 < s_2$
1	1	1	1	1	1	1	2	1	1	2	1	$1/3+1/7$	$1/4+1/6$	$1/5+1/5$	$s_3 < s_2 < s_1$
1	1	1	1	1	1	1	1	2	1	1	1	$1/3+1/6$	$1/4+1/4$	$1/5+1/3$	$s_1 = s_2 < s_3$
1	1	1	1	2	1	1	1	2	1	2	2	$1/3+2/8$	$1/4+2/6$	$1/6+2/5$	$s_3 < s_1 = s_2$
1	1	2	1	2	1	1	1	2	1	2	1	$1/4+1/7$	$1/5+1/5$	$1/7+1/4$	$s_1 = s_3 < s_2$
1	1	1	1	1	1	1	1	1	1	1	1	$1/3+1/5$	$1/4+1/4$	$1/5+1/3$	$s_2 < s_1 = s_3$
1	1	1	1	1	1	1	1	1	1	2	1	$1/3+1/6$	$1/4+1/5$	$1/5+1/4$	$s_2 = s_3 < s_1$
2	1	2	1	2	1	1	1	2	1	1	1	$2/5+1/6$	$2/6+1/4$	$2/8+1/3$	$s_1 < s_2 = s_3$
1	1	1	1	2	1	1	1	2	1	1	1	$1/3+1/6$	$1/4+1/4$	$1/6+1/3$	$s_1 = s_2 = s_3$

Similarly, repeating the analysis in terms of the distance metric, we have:

$$d_1 := dg(c, n_2) = \frac{1}{2}\left(\frac{g_2 + g_3}{g1 + \cdots + g_6} + \frac{m_2 + m_3 + m_4 + m_5}{m_1 + \cdots + m_6}\right)$$
$$d_2 := dg(c, n_2) = \frac{1}{2}\left(\frac{g_2 + g_3 + g_4}{g1 + \cdots + g_6} + \frac{m_2 + m_4 + m_5}{m_1 + \cdots + m_6}\right)$$
$$d_3 := dg(c, n_3) = \frac{1}{2}\left(\frac{g_2 + g_3 + g_4 + g_5}{g1 + \cdots + g_6} + \frac{m_2 + m_5}{m_1 + \cdots + m_6}\right)$$

Again here is no relationship between d_1, d_2, and d_3, and any order combination is possible, as the following table shows:

$g_1 g_2 g_3 g_4 g_5 g_6$	$m_1 m_2 m_3 m_4 m_5 m_6$	$2 \cdot d_1$	$2 \cdot d_2$	$2 \cdot d_3$	result
1 1 1 1 1 1	1 1 1 1 2 1	$2/6 + 5/7$	$3/6 + 4/7$	$4/6 + 3/7$	$d_1 < d_2 < d_3$
1 1 1 1 1 1	1 2 1 2 2 2	$2/6 + 7/10$	$3/6 + 6/10$	$4/6 + 4/10$	$d_1 < d_3 < d_2$
1 1 1 1 1 1	1 2 2 1 2 2	$2/6 + 7/10$	$3/6 + 5/10$	$4/6 + 4/10$	$d_2 < d_1 < d_3$
1 1 1 1 1 1	1 1 2 1 1 1	$2/6 + 5/7$	$3/6 + 3/7$	$4/6 + 2/7$	$d_2 < d_3 < d_1$
1 1 1 1 1 1	1 1 1 2 1 1	$2/6 + 5/7$	$3/6 + 4/7$	$4/6 + 2/7$	$d_3 < d_1 < d_2$
1 1 1 1 1 1	1 1 2 2 1 1	$2/6 + 6/8$	$3/6 + 4/8$	$4/6 + 2/8$	$d_3 < d_2 < d_1$
1 1 1 1 2 1	1 1 1 1 1 2	$2/7 + 4/7$	$3/7 + 3/7$	$5/7 + 2/7$	$d_1 = d_2 < d_3$
1 1 1 1 1 2	1 1 1 2 1 1	$2/7 + 5/7$	$3/7 + 4/7$	$4/7 + 2/7$	$d_3 < d_1 = d_2$
1 1 1 1 1 1	1 1 1 2 2 2	$2/6 + 6/9$	$3/6 + 5/9$	$4/6 + 3/9$	$d_1 = d_3 < d_2$
1 1 1 1 1 1	1 1 2 1 2 2	$2/6 + 6/9$	$3/6 + 4/9$	$4/6 + 3/9$	$d_2 < d_1 = d_3$
1 1 1 1 1 2	1 1 2 1 1 1	$2/7 + 5/7$	$3/7 + 3/7$	$4/7 + 2/7$	$d_2 = d_3 < d_1$
1 1 1 2 1 1	1 1 1 1 1 2	$2/7 + 4/7$	$4/7 + 3/7$	$5/7 + 2/7$	$d_1 < d_2 = d_3$
1 1 1 1 1 1	1 1 1 1 1 1	$2/6 + 4/6$	$3/6 + 3/6$	$4/6 + 2/6$	$d_1 = d_2 = d_3$

To summarize the analysis: we cannot draw any conclusion that Type II siblings of a concept c are any closer to c, using dl or dg, than the weaker Type III siblings, i.e., we cannot say that Type II siblings better represent related categories than Type III siblings.

6.4 Type I versus Type II versus Type III Siblings Concepts

Fig. 10 is a combination of the first and the third example. In terms of similarity we have:

$$s_n := s(c, n) = \frac{1}{2}\left(\frac{g_1}{g_1 + g_2 + g_7} + \frac{m_6}{m_2 + m_6 + m_7}\right)$$
$$s_1 := s(c, n_1) = \frac{1}{2}\left(\frac{g_1}{g_1 + g_2 + g_3} + \frac{m_6}{m_2 + m_3 + m_4 + m_5 + m_6}\right)$$
$$s_2 := s(c, n_2) = \frac{1}{2}\left(\frac{g_1}{g_1 + g_2 + g_3 + g_4} + \frac{m_6}{m_2 + m_4 + m_5 + m_6}\right)$$
$$s_3 := s(c, n_3) = \frac{1}{2}\left(\frac{g_1}{g_1 + g_2 + g_3 + g_4 + g_5} + \frac{m_6}{m_2 + m_5 + m_6}\right)$$

Note that changing g_7 and m_7 does not affect the similarity measures between c and n_1, n_2, n_3, resp. According to Section 6.1, for high values of g_7 and m_7,

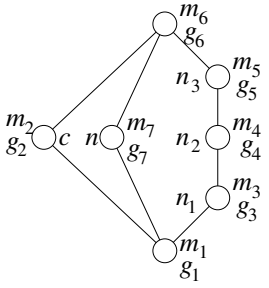

Fig. 10 Example for comparing all three types of siblings

the similarity between c and n (i.e., dg) decreases. We can easily use the values for $g_1, \ldots, g_6, m_1, \ldots, m_6$ of the last example to get all possible orderings of s_1, s_2, s_3, and choose g_7 and m_7 such that $d < s_1, s_2, s_3$, i.e. Type II siblings, are not necessarily more similar to c than Type III siblings.

In fact, we have again that for s, all 24 strict orders of c, n, s_1, s_2, s_3 can appear. And the same holds for dg. (As we have both for s and dg 24 such strict orders, thus 48 examples, these examples are not provided due to space limitations). In short, no general statements which render some preference for siblings used as Related Categories in terms of similarity and distance.

7 Conclusion

The notion of Type I, II and II siblings are purely lattice-theoretic notions, whereas the notion of distance and similarity are purely set-theoretic notions. In order to find similar concepts to a given concept (related categories), there is no hint that the closest siblings contain the most similar concept. This might sound disappointing at a first glance, but in practice our observations lead to an important design feature in SearchSleuth. Computationally, the neighboring siblings to the current formal concept (whether of Type I, II or III) are the easiest concepts to compute and therefore represent natural candidates for related category search. Our design question was should we extend our search of the conceptual neighborhood from exact siblings (ES) to other sibling types? The answer is no because there is no guarantee that formal concepts with the best distance and similarity will be found in any of the sibling spaces. We have shown that to find the optimally close formal concepts involves searching and computing all formal concepts in the entire concept lattice. Such a search would render SearchSleuth intractable. Ranking the ES space on similarity gives the diversity we need in the SearchSleuth interface without any great overhead.

SearchSleuth extends current Formal Concept Analysis Internet search engines by positioning the user within the information space, rather than placing the user arbitrarily or presenting the entire space. This allows generalization and categorization operations to be performed against the current query concept. SearchSleuth overcomes a number of practical difficulties in the use of Formal Concept Analysis for Internet Search, namely a practical approach to the construction of a sparse context and the categorization operation, where the conceptual focus is moved to a sibling concept of the search concept. These paper explains how related categories are derived using a combination of order-theoretic notions of neighborhood in combination of set-theoretic

definitions of concept similarity, the approach has clear limitations in the sense that there is no guarantee that the most similar formal concepts to the current concept are in its immediate conceptual neighborhood. This finding allows us to constrain the search for similar concepts to an arbitrary search horizon that is easy to compute.

References

1. Carpineto, C., Romano, G.: Exploiting the potential of concept lattices for information retrieval with credo. Journal of Universal Computer Science **10**(8) (2004) 985–1013
2. Koester, B.: Conceptual knowledge processing with google. In: Contributions to ICFCA 2006. (2005) 178–183
3. Ducrou, J., Eklund, P.: Searchsleuth: the conceptual neighbourhood of an internet query. In: 5th International Conference on Concept Lattices and Their Applications (CLA 2007), http://CEUR-WS/Vol-331/Ducrou.pdf (2007) 249–260
4. Ducrou, J., Eklund, P., Dau, F.: Concept similarity and related categories in searchsleuth. In: 16th Int. Conf. on Conceptual Structures (ICCS 2008), Springer (2008) 249–260
5. Carpineto, C., Romano, G.: A lattice conceptual clustering system and it's application to browsing retrieval. Machine Learning **24** (1996) 95–122
6. Wille, R.: Conceptual landscapes of knowledge: A pragmatic paradigm for knowledge processing. In Gaul, W., Locarek-Junge, H., eds.: Classification in the Information Age, Springer (1999) 344–356
7. Eklund, P., Groh, B., Stumme, G., Wille, R.: A contextual-logic extension of toscana. In: Conceptual structures: logical, linguistic and computational issues. LNAI 1867, Springer (2000) 453–467
8. Cole, R., Eklund, P., Stumme, G.: CEM - a program for visualization and discovery in email. In: Principles of Data Mining and Knowledge Discovery: 4th European Conference, PKDD 2000, Lyon, France, September 2000. Proceedings. (2000) 367 – 374
9. Ducrou, J., Eklund, P.: An intelligent user interface for browsing and search MPEG-7 images using concept lattices. International Journal of Foundations of Computer Science **19**(2) (2008) 359–381
10. Bruza, P., McArthur, R., Dennis, S.: Interactive internet search: keyword, directory and query reformulation mechanisms compared. In: SIGIR '00: Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval, New York, NY, USA, ACM (2000) 280–287
11. Stumme, G., Taouil, R., Bastide, Y., Pasquier, N., Lakhal, L.: Computing iceberg concept lattices with Titanic. Data & knowledge engineering **42**(2) (2002) 189–222
12. Cole, R.J., Eklund, P.W., Stumme, G.: Document retrieval for email search and discovery using formal concept analysis. Applied Artificial Intelligence **17**(3) (2003) 257–280
13. Ducrou, J., Eklund, P.: Faceted document navigation using conceptual structures. In Hitzler, P., Schärf, H., eds.: Conceptual Structures in Practice, CRC Press (2009) 251–278
14. Kim, M., Compton, P.: Formal Concept Analysis for Domain-Specific Document Retrieval Systems. In: Australian Joint Conference on Artificial Intelligence. (2001) 237–248
15. Kim, M., Compton, P.: The perceived utility of standard ontologies in document management for specialized domains. International Journal of Human-Computer Studies **64**(1) (2006) 15–26
16. Eklund, P., Ducrou, J., Brawn, P.: Concept lattices for information visualization: Can novices read line diagrams. In Eklund, P., ed.: Proceedings of the 2nd International Conference on Formal Concept Analysis - ICFCA'04, Springer (2004)
17. Ducrou, J., Vormbrock, B., Eklund, P.: FCA-based Browsing and Searching of a Collection of Images. In: Proceedings of 14th International Conference on Conceptual Structures. LNAI4068, Springer (2006) 203–214
18. Carpineto, C., Romano, G.: Concept Data Analysis: Theory and Applications. John Wiley & Sons (2004)
19. Wille, R.: Methods of conceptual knowledge processing. In: Formal Concept Analysis. (ICFCA 2006). LNAI, (Springer)
20. Lengnink, K.: *"Ahnlichkeit als Distanz in Begriffsverbänden.* In G Stumme, R.W., ed.: *Begriffliche Wissensverarbeitung: Methoden und Anwendungen*, Springer (2001) 57–71
21. Saquer, J., Deogun, J.S.: Concept aproximations based on rough sets and similarity measures. In: Int. J. Appl. Math. Comput. Sci. Volume 11. (2001) 655 – 674