Interactive Query Refinement using Formal Concept Analysis

Tim Pattison

tim.pattison@dst.defence.gov.au
Defence Science & Technology Group
West Ave, Edinburgh
South Australia 5111

Abstract. Formal Concept Analysis (FCA) of a document corpus yields a concept lattice uniting the powersets of corpus terms and documents. Structural navigation of the directed graph connecting neighbours in this lattice affords interactive query refinement. The starting point for navigation is typically the closure of the conjunctive Boolean query with respect to the corpus. This paper describes the special treatment of "closure" terms – those in this closure but which are not specified by the user – and its implications for *implicit* structural navigation of the digraph through query editing. This approach, in which the user's choice of each additional query term is constrained to avoid the null result set, is contrasted with explicit structural navigation. If a term is added which does not co-occur with the terms already specified, the user must either remove the new term or choose which other term(s) to remove. A novel technique is used to present the user with options for those other terms.

1 Introduction

1.1 Formal Concept Analysis for Interactive Query Refinement

Formal Concept Analysis (FCA) has been used in corpus-based information retrieval for the construction and interactive refinement of conjunctive Boolean queries (see e.g. [15,14,16,6]). Here, a query is a set of terms, all of which must be present in each document retrieved in response to the query. The derivation operator, which maps the query onto the set of matching documents, induces a Galois connection between the set of potential queries – the powerset of terms appearing in the corpus – and the set of potential results – the powerset of documents in the corpus. This choice elegantly combines the query and result spaces into a unified structure – the concept lattice – which can be navigated either explicitly or implicitly by the user for interactive refinement of the query.

The set of documents returned by a query, and the set of query terms, augmented by any other terms – known as *closure* terms – which are common to all documents in the result set, constitute a *formal concept*. The association between terms present in the corpus and the documents in which they occur is known as a *formal context*, and is often specified in the form of a binary matrix. A simple example formal context is shown in Figure 1a.

ISBN 978–80–244–5328–6, Department of Computer Science, Palacký University Olomouc, 2018.

Given a formal context, FCA enumerates the set of formal concepts, partially orders them by document set subsumption, and computes the neighbour relation as the transitive reduction of this ordering relation. The result is a single-source, single sink, directed acyclic graph (DAG), whose vertices represent formal concepts and whose arcs connect neighbours.

A lower [upper] neighbour of a concept represents a minimal conjunctive expansion [contraction] of the corresponding set of query terms with respect to the corpus, and the consequently contracted [expanded] result set[1]. These neighbours constitute the available options for incremental modification of the current query. The dominant paradigm for FCA-based query refinement involves explicit navigation via (undirected) edges in this graph – a process which we refer to as *structural navigation*. Navigation is typically constrained to neighbours of the query concept, although concepts beyond neighbours are sometimes also offered as candidates [12,5]. The graph can be presented to the user in its entirety [8], or, more usually, as a keyhole view of the neighbourhood of the current concept [2,4,3,7].

1.2 Query Editing with Ranked Refinement Options

Lindig [12] instead presented users with a list of terms in the current query, any one of which could be selected for removal, and a second list of terms which, when selected and thereby added to the query, would return a non-empty result set. The user is also shown the result set as a list of identifiers, and this list is updated with each term selection. This approach supports implicit navigation of the concept lattice to super- and sub-concepts – not necessarily neighbours – without explicitly exposing the user to the concept lattice or requiring them to comprehend it. To refine their query, the user is thereby focused on editing the query itself rather than navigating the concept lattice.

In contrast with structural navigation, the use of term lists as the primary interface for interactive query refinement naturally supports any scheme which ranks terms to assist the user's choice. Using Latent Semantic Analysis, the SORTED prototype [13] ranks the lists of query and unused terms according to their "semantic" relevance to the result set of a conjunctive Boolean query. With the exception of term substitution, as described in Section 4, SORTED implements the query editing techniques described in this paper.

1.3 Contribution

This paper describes a method by which the user explicitly edits a conjunctive Boolean query, subject to the constraint that the edited query must have a non-empty result set. To respect this constraint, the addition of a disjunctive term entails the removal of one or more terms previously entered by the user.

[1] Square brackets are used throughout this paper to indicate that a sentence is true both when read without the bracketed terms and when read with each bracketed term substituted for the term which precedes it.

Here we describe the term most recently entered by the user as *disjunctive* if its addition to the existing conjunctive Boolean query would return an empty result set and *conjunctive* otherwise. The user is alerted upon entry of a disjunctive term, and prompted to choose, from amongst automatically generated options, which term(s) should be removed. Removal of the disjunctive term is one of those options. An FCA-based technique is described for identifying those options which minimise the number of user-specified terms removed.

This approach to refining conjunctive Boolean queries is compared and contrasted with the explicit structural navigation of the concept lattice digraph. In particular, it demonstrates that: special treatment of closure terms is appropriate; not all neighbours of the query concept are reachable through the addition or removal of a single user-specified query term; and navigation is not constrained to neighbours.

1.4 Organisation

This paper is organised as follows. Section 2 commences with a brief introduction to the application of Formal Concept Analysis for interactive query refinement. It describes a query editing approach to interactive query refinement, comparing and contrasting it with structural navigation of the lattice digraph. Section 3 canvasses the computational implications of the query editing approach. A novel scheme is then described in Section 4 for computing term substitution options upon user entry of a disjunctive query term. Following a discussion in Section 5 of related work, the contributions of this paper are summarised in Section 6.

2 FCA for Interactive Query Refinement

2.1 Introduction

Formal Concept Analysis (FCA) derives multiple-inheritance class hierarchies from empirical data, and is commonly applied to information retrieval (see e.g. [6] for a recent survey). For classical information retrieval, each class or *concept* is a set of documents and a set of terms such that each document contains each of the terms [3]. A concept is maximal in the sense that there are no other documents containing all of the specified terms, and no other terms which are common to all of those documents. A concept is a *sub-concept* [*super-concept*] of another if it corresponds to a proper subset [superset] of its documents – or equivalently to a proper superset [subset] of its terms. The term "multiple-inheritance" refers to the fact that a concept can be a sub-concept of two or more others, between which a sub-concept relationship does not exist.

For a given document corpus, the corresponding set of concepts, partially ordered by the sub-concept relation between them, forms a *complete lattice*. This lattice can be represented as a directed graph – or *digraph* – whose vertices are the concepts and whose directed arcs are from concepts to their upper neighbours. A super-concept of a given concept is an *upper neighbour* if none of its

sub-concepts are super-concepts of the given concept. Layered drawings of this digraph exist in which all arcs of the digraph are upwards, allowing arrows to be omitted from the arcs. The resultant graph drawing is known as a *Hasse diagram* (see e.g. [10]). Figure 1b shows the Hasse diagram corresponding to the formal context in Figure 1a.

At the top of the lattice digraph is the (sole) sink vertex, representing the *supremum* concept, which corresponds to the entire corpus and any terms common to all documents; at the bottom of the lattice digraph is the (sole) source vertex, representing the *infimum* concept, which corresponds to the set of all terms in the corpus and any documents which contain all of them. Each graph vertex is labelled with any term(s) for which the extent of the corresponding concept is exactly the set of documents which contain it, and with any document identifier(s) for which the intent is exactly the set of terms in that document. A concept is an attribute [object] concept for each attribute [object] with which the corresponding vertex is labelled.

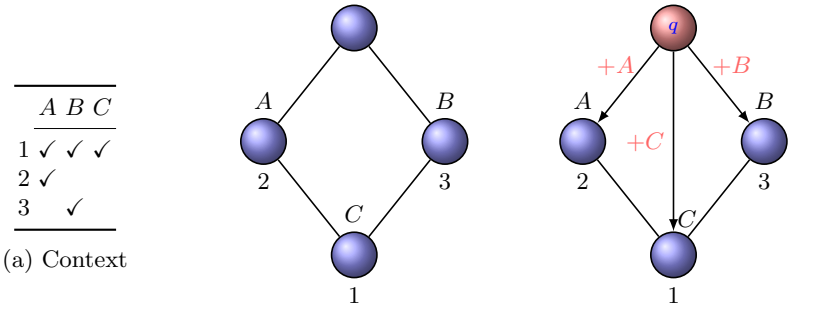
2.2 Closure Terms

In corpus-based information retrieval applications of FCA, the user starts by specifying a set of terms which is to be used as a conjunctive query. The result set – the set of documents containing all of those terms – is easily determined, along with any additional terms which those documents have in common. To distinguish these additional terms from the query terms, we refer to them as *closure terms*. Closure terms, if any, are identified automatically and provide additional information about the result set – and indeed the corpus – which the user acquires without reading the constituent documents. Codocedo and Napoli [6] refer to the addition of closure terms as query "extension", and note the "exhaustive" exploitation of closure terms to provide feedback to the user on the corpus-specific context of their query.

While closure terms are exposed to the user in SORTeD, however, they are quarantined from user interaction during subsequent query editing, since the result set is unaffected by either: explicitly extending the user-specified query with closure terms; or removing closure terms from a conjunctive Boolean query which has been extended in this manner. The user must understand the definition of closure terms to properly interpret not only the information they impart about the corpus and the result set, but also their different treatment and behaviour in the user interface.

2.3 Query Refinement

Having specified a query which has a non-empty result set, the user can specialise or generalise the query as required. Query generalisation [specialisation] is typically implemented as the explicit selection of a super- [sub-] concept in the concept lattice, potentially via upward [downward] structural navigation. Some authors [3,14] refer to navigation to upper and lower neighbours as query "enlargement" and "refinement" respectively.



(a) Context (b) Hasse diagram (c) Specialisation options

Fig. 1: An example formal context, its corresponding Hasse diagram, and options for specialisation from query concept q.

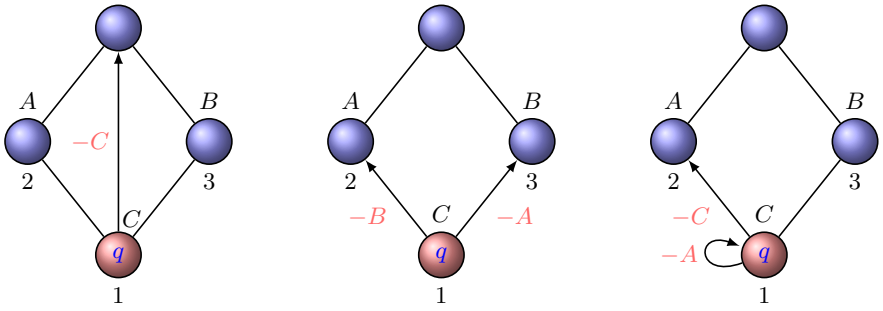
Alternatively, individual terms can be manually added to, or removed from, the query [12]. Query generalisation is achieved by removing one or more of the specified query terms. Query specialisation, on the other hand, involves adding conjunctive terms – those which occur in proper[2], non-empty subsets of the current result set of documents. In addition to requiring the user to select only from query terms in the case of generalisation, and only from conjunctive terms in the case of specialisation, the SORTED user interface provides advanced knowledge of the effect each choice would have on the size of the result set. Constraining the choice of additional terms to conjunctive terms during query refinement also provides the user with explicit information about the terms and term combinations occurring in the corpus, which they would otherwise need to infer from failed queries.

2.4 Contrast with Structural Navigation

Section 2.3 described the interactive refinement of a conjunctive Boolean query through explicit editing of the query. Like explicit structural navigation of the concept lattice digraph, this approach to query refinement constrains the user's choice to the set of super- and sub- concepts of the current concept. Nevertheless, there are significant differences between the behaviour of these two approaches, and these are explained in this section.

The manual addition [removal] of a single term implicitly selects a sub- [super-] concept – but not necessarily a neighbour – of the current concept. The following example illustrates this point. Consider a corpus consisting of three documents $\{1, 2, 3\}$ containing combinations of three terms $\{A, B, C\}$. Its formal context is shown in Figure 1a and the corresponding Hasse diagram in Figure 1b. Starting at the supremum, as shown in Figure 1c, the user specifies

[2] this excludes both current query and closure terms, which occur in an improper subset.



(a) C (b) AB (c) AC

Fig. 2: Generalisation options for three query strings which select the infimum in Figure 1b. This query concept, q, is shown red. Directed edges from q, labelled with the attribute to be removed, enumerate the query generalisation options.

term A, B or C as their first search term. Term A selects the left lower neighbour of the supremum, since there is a document in the corpus containing only that term. Similarly, term B selects the right lower neighbour. However, if the user enters C as the first search term, the infimum is selected, since the only document in the corpus which contains term C also contains the (closure) terms A and B. Here the addition of a search term to the (empty) query implicitly navigates to a sub-concept of the current concept which is not a lower neighbour. As illustrated using the directed edge labelled "-C" in Figure 2a, its subsequent removal returns directly to the supremum, which is a super-concept but not upper neighbour of the infimum.

A given concept in the lattice can be reached using different sequences of query terms. For the example context of Figure 1, the infimum can be reached not only with the query C, but also with the queries AB, AC, BA and BC. Query sequences such as CA and ABC are not possible, since the last or last few terms are rendered closure terms – and therefore unavailable for subsequent user entry – by the entry of earlier terms in the sequence. In the case of CA, for example, the query C selects the infimum, at which point A becomes a closure term whose entry by the user would be nugatory.

Since only the query (vice closure) terms are available for removal, the query sequence used clearly affects the subsequent options for generalisation. Figure 2 illustrates the options for query generalisation for three of these queries. Figure 2b shows that for the query AB, either of the upper neighbours of the infimum is reachable by removing one of the search terms. In contrast, Figure 2c shows that the right-most concept cannot be reached by removing either term from the query AC. Whereas removing C from the search AC selects the left-most concept, removing A instead results in no change in the selected concept. In this case, A simply changes category from a user-specified query term to a closure term. Such nugatory interaction could be prevented by removing A from the set of terms which can be removed until such time as C has been removed.

A query editing approach to query refinement was described in Section 2.3. In this subsection its behaviour has been compared and contrasted with structural navigation of the concept lattice digraph. In particular, while all lower neighbours are reachable in the next move, the query used to reach the current concept may render some upper neighbours unreachable in the user's next move. Furthermore the concepts reachable in the next move may include super- [sub-] concepts which are not upper [lower] neighbours.

3 Computation for Query Editing

The query editing scheme described in Section 2.3 involves the addition or removal of a single user-specified query term. In order to inform the user of the anticipated effect on the size of the result set, and in the case of query specialisation, to constrain the choice of additional terms to those which produce a non-empty result set, it is necessary to calculate the set of concepts which can be reached in the user's next move.

A range of algorithms exist for the enumeration of concepts in a formal context (see e.g. [11,1]), and these can be readily modified to enumerate only those concepts which can be reached from the current concept by adding or removing a single query term. Reachable sub-concepts can be calculated by adding to the intent of the current concept each unused term in turn and calculating the result set. Here, an "unused" term is one which is not already in the intent of the current concept. Terms giving rise to an empty result set are candidates for term substitution, a technique for which is described in Section 4. Similarly, the reachable super-concepts can be calculated by removing each term in turn from the set of user-specified terms, calculating the result set, and optionally disallowing the removal of any terms for which the result set is not a proper superset of the extent of the current concept.

The on-demand computation scheme described in this section has the benefit that only those concepts required to inform the user's next move are computed at each step. The required set of concepts must be computed in interactive timescales, so that the user is not required to wait before contemplating their next move. Given that the computation is triggered by the user's move, however, and the user must assess the consequences of that move – viz. the intent and extent of the new query concept – before contemplating the next, this requirement is not onerous. The computational complexity of enumerating the reachable sub-concepts is $\mathcal{O}(t^2 d)$ Boolean AND operations, where t is the number of unique terms in the formal context and d is the number of documents. Whilst this has the potential to become prohibitive for large corpora, the computation is highly parallelisable.

The set of concepts reachable in the next move could be stored in case subsequent changes to the query return to the same concept. However, since the set of reachable super-concepts is dependent not only on the current concept, but also on the query used to reach it, more than one set of reachable super-concepts may need to be stored for a given concept.

4 Substituting Disjunctive Terms

Let the partially ordered set $\langle \mathfrak{B}, < \rangle$ be the concept lattice corresponding to a formal context constructed from the terms and documents of a nominated corpus. The elements of $\mathfrak{B}$ are the formal concepts and the relation $<$ between concepts corresponds to set inclusion between concept extents. $\langle \mathfrak{B}, < \rangle$ is a complete lattice, for which the greatest lower and least upper bounds, $\inf \mathfrak{S}$ and $\sup \mathfrak{S}$, respectively, on any $\mathfrak{S} \subseteq \mathfrak{B}$ exist and are unique [9].

Definition 1 *Let* $\mathfrak{X}(\mathfrak{S}) \triangleq \{x \in \mathfrak{B} | x \leq z, \forall z \in \mathfrak{S}\}$. *Then* $\inf \mathfrak{S} \in \mathfrak{X}(\mathfrak{S})$ *and* $\inf \mathfrak{S} \geq x, \forall x \in \mathfrak{X}(\mathfrak{S})$.

Definition 2 *Let* $\mathfrak{Y}(\mathfrak{S}) \triangleq \{y \in \mathfrak{B} | y \geq z, \forall z \in \mathfrak{S}\}$. *Then* $\sup \mathfrak{S} \in \mathfrak{Y}(\mathfrak{S})$ *and* $\sup \mathfrak{S} \leq y, \forall y \in \mathfrak{Y}(\mathfrak{S})$.

Let $q \in \mathfrak{B}$ denote the current query concept, $\alpha \in \mathfrak{B}$ an ancestor of $q < \alpha$, and $t \in \mathfrak{B}$ the attribute concept of a nominated disjunctive term. By definition, a disjunctive term is one which is not in the intent of q or any of its descendants, so that

$$q \not< t \tag{1a}$$

$$\inf \{q, t\} = \inf \mathfrak{B} \tag{1b}$$

Assume the existence of a procedure which identifies all distinct pairs (α', ω) such that

$$\alpha > q \tag{2a}$$

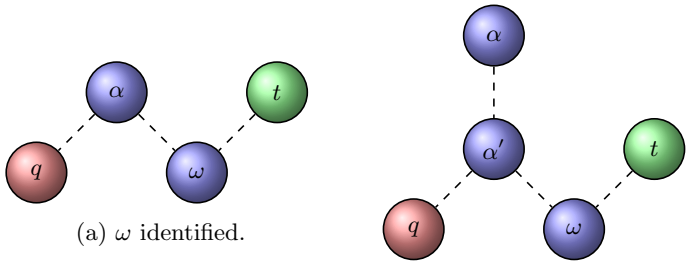
$$\omega \triangleq \inf\{\alpha, t\} \in \mathfrak{B} \tag{2b}$$

$$\alpha' \triangleq \sup\{\omega, q\} \in \mathfrak{B} \tag{2c}$$

Figure 3a shows the Hasse diagram for the poset $\langle \{q, \alpha, \omega, t\}, < \rangle$. The edges are shown dashed to indicate that they need not correspond to neighbour relations in the Hasse diagram for $\langle \mathfrak{B}, < \rangle$. From Equation 2b and Definition 1, $\omega \in \mathfrak{X}(\{\alpha, t\})$, and for any $x \in \mathfrak{X}(\{\alpha, t\})$, $x \leq \omega$. The intent of any $x < \omega$ is a superset of that of ω, and hence differs from those of α and t by more terms. In particular, Equation 2b ensures that the new query, ω, differs from $\alpha > q$ by the smallest number of (added) terms consistent with the constraint $\omega \in \mathfrak{X}(\{\alpha, t\})$. We note in passing that Equation 1a eliminates the possibility that $\omega = \alpha$.

Figure 3b shows the Hasse diagram for the poset $\langle \{q, \alpha, \alpha', \omega, t\}, < \rangle$, which differs from that in Figure 3a by the addition of α'. From Equation 2c and Definition 2, $\alpha' \in \mathfrak{Y}(\{q, \omega\})$ and for any $y \in \mathfrak{Y}(\{q, \omega\})$, $y \geq \alpha'$. From Equations 2a and 2c and Definition 2, $\alpha \in \mathfrak{Y}(\{q, \omega\})$ and hence

$$\alpha \geq \alpha' \tag{3}$$



(a) ω identified.

(b) α' identified.

Fig. 3: Hasse diagrams showing the ordering of concepts q, α, t, ω and α'.

Equation 2c ensures that α' differs from q by the smallest number of (removed) terms, and from ω by the smallest number of (added) terms, both consistent with the constraints $\alpha' \in \mathfrak{Y}(\{q, \omega\})$.

A procedure for enumerating all pairs satisfying Equation 2 is as follows. For each $\alpha > q$ encountered during an upwards, breadth-first traversal of $\langle \mathfrak{B}, < \rangle$ from q, find ω and then α' using Equations 2b and 2c. If $\alpha' \neq \alpha$, discard (α', ω). The upwards, breadth-first traversal ensures that the algorithm encounters α' as a candidate ancestor of q before – or strictly speaking not later than – $\alpha \geq \alpha'$. By an upwards, breadth-first traversal, we mean that all upper neighbours of a concept are visited before any more distant ancestors, *and* that a vertex is not visited until each of its lower neighbours has been. The latter requirement is necessitated by the possibility that the lattice digraph may contain parallel directed paths of unequal path length. If ω has been previously encountered, and hence also α' from Equation 2c, the pair (α', ω) has already been generated, and traversal can progress immediately to the next $\alpha > q$.

Proposition 1.

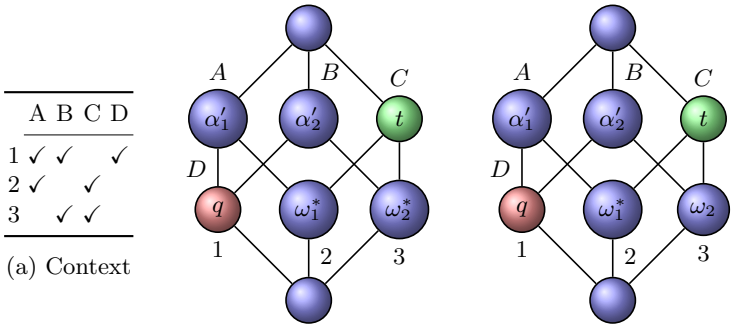
$$\omega = \inf \{\alpha', t\} \tag{4}$$

Proof. Define

$$\omega' \triangleq \inf \{\alpha', t\} \tag{5}$$

Substituting Equation 3 into Definition 1 yields $\omega' \leq t$ and $\omega' \leq \alpha' \leq \alpha$, to which Definition 1 can be reapplied to give $\omega' \leq \inf \{\alpha, t\} = \omega$. But $\omega \leq \alpha'$ from Equation 2c and $\omega \leq t$ from Equation 2b, so $\omega \leq \inf \{\alpha', t\} = \omega'$. Hence $\omega' = \omega$.

Proposition 1 shows that evaluating the right-hand side of Equation 5 in the hope of finding a tighter lower bound ω' on α' and t is nugatory. Subject to the initial choice of α, the pair (α', ω) minimises the number of terms removed from and added to the current query q to include the disjunctive term for which t is the attribute concept. Having enumerated the possible choices for $\alpha > q$ to identify all unique pairs (α', ω), the problem reduces to choosing from amongst these the ω differing from q by the smallest number of terms. The cardinality of the set



(a) Context (b) Query AB (c) Query AD

Fig. 4: Substitution of term C for queries AB and AD in example context.

difference between the intents of q and α' gives the number of terms removed from the existing query, while that of the set difference between the intents of ω and α' gives the number of terms added to it. The most straightforward approach is to minimise the sum of these two numbers. If preserving terms of the current query is a priority, then a weighted sum might be used in which the number of removed terms is weighted more highly than the number of added terms. The possible choices for the new query ω could be presented to the user ranked in order of increasing (weighted) term difference.

Applying this approach to the example context in Figure 4a with query AB yields the two jointly-optimal solutions ω_1^* and ω_2^* shown in Figure 4b, corresponding to ancestors α'_1 and α'_2, respectively, of the query concept q (shown red). Each involves removing one query term and adding the (formerly) disjunctive term C, for which the attribute concept t is shown green. The user might be asked to choose between them, possibly aided by additional information such as extent cardinality, or the semantic relevance of the additional terms.

In this section, we have so far ignored the distinction between user-specified and closure terms. This distinction should be taken into account to privelege options minimising the number of user-specified terms which must be removed to accommodate the new term. If, for example, the user-specified query against the formal context of Figure 4a were AD, then the removal of closure term B from the intent of the query concept should be preferred over that of query term A. Descendant ω_1^* of α'_1 is therefore preferred over ω_2 of α'_2, as shown in Figure 4c, even though both involve the removal of two terms from the query concept.

Like conjunctive terms, disjunctive terms should be offered in a progressively-constrained list for addition to the query. Upon user selection of a disjunctive term, options would be presented showing which query (vice closure) terms must be removed to accommodate the new term. The terms to be removed might be shown struck out, and the terms to be added – other than the nominated disjunctive term – highlighted, but otherwise treated as closure terms.

5 Discussion and Related Work

If the user is not aided in their initial choice of query terms, it is not possible to guarantee that each term occurs in the corpus and that at least one document contains all terms. The query editing scheme described in this paper is therefore used *ab initio*, with the initial query being the set of terms associated with the supremum. In contrast, Carpineto and Romano [2] permitted unconstrained entry of an intial set of query terms, and described a composite method whereby the resultant query could be mapped onto the concept lattice. *Ab initio* computer-assisted query refinement requires the initial enumeration of all attribute concepts in the formal context and user interaction with the list of all terms in the corpus. Manual entry of terms, aided by term completion, partially alleviates the challenge posed by large corpora of finding terms in a long list.

6 Conclusion

This paper has described a method by which the user explicitly edits a conjunctive Boolean query, subject to the constraint that the edited query has a non-empty result set. Adoption of this query-editing approach, and in particular its use of lists to present options for term addition and removal, has allowed the SORTED prototype to offer the user ranked options for interactive query refinement. The addition of a disjunctive term entails the removal of one or more terms previously entered by the user. The user is alerted upon entry of a disjunctive term, and prompted to choose, from amongst automatically generated options, which term(s) should be removed. An FCA-based technique has been described for automatically identifying options minimising the number of user-specified terms removed.

This approach to refining conjunctive Boolean queries has been compared and contrasted with the explicit structural navigation of the concept lattice digraph. In particular: special treatment of closure terms is appropriate; not all neighbours of the query concept are reachable through the addition or removal of a single user-specified query term; and navigation is not constrained to neighbours.

References

1. Andrews, S.: A 'best-of-breed' approach for designing a fast algorithm for computing fixpoints of Galois connections. Information Sciences **295**, 633–649 (2015). https://doi.org/10.1016/j.ins.2014.10.011
2. Carpineto, C., Romano, G.: ULYSSES: a lattice-based multiple interaction strategy retrieval interface. In: Blumenthal, B., Gornostaev, J., Unger, C. (eds.) Human-Computer Interaction (EWHCI 1995). pp. 91–104. No. 1015 in LNCS, Springer, Berlin, Heidelberg (1995). https://doi.org/10.1007/3-540-60614-9_7
3. Carpineto, C., Romano, G.: Exploiting the potential of concept lattices for information retrieval with CREDO. Journal of Universal Computing **10**(8), 985–1013 (2004). https://doi.org/10.3217/jucs-010-08-0985

4. Carpineto, C., Romano, G.: Effective reformulation of Boolean queries with concept lattices. In: Andreasen, T., Christiansen, H., Larsen, H. (eds.) Flexible Query-Answering Systems (FQAS 1995). LNCS, vol. 1495, pp. 83–94. Springer, Berlin, Heidelberg (1998). https://doi.org/10.1007/BFb0055993
5. Codocedo, V., Lykourentzou, I., Napoli, A.: A semantic approach to concept lattice-based information retrieval. Annals of Mathematics and Artificial Intelligence **72**(1-2), 169–195 (2014). https://doi.org/10.1007/s10472-014-9403-0
6. Codocedo, V., Napoli, A.: Formal Concept Analysis and Information Retrieval – a survey. In: Baixeries, J., Sacarea, C., Ojeda-Aciego, M. (eds.) Formal Concept Analysis, Lecture Notes in Computer Science, vol. 9113, pp. 61–77. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-19545-2_4
7. Eklund, P., Wray, T., Goodall, P., Bunt, B., Lawson, A., Christidis, L., Daniels, V., Van Olffen, M.: Designing the digital ecosystem of the Virtual Museum of the Pacific. In: 3rd IEEE Intl. Conf. Digital Ecosystems and Technologies. pp. 377–383. IEEE (2009). https://doi.org/10.1109/DEST.2009.5276744
8. Eklund, P.W., Ducrou, J., Brawn, P.: Concept lattices for information visualization: Can novices read line diagrams? In: Eklund, P. (ed.) Proc. 2nd Intl. Conf. on Formal Concept Analysis (FCA'04). LNCS, vol. 2961, pp. 57–73. Springer, Berlin, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24651-0_7
9. Ganter, B., Wille, R.: Formal Concept Analysis: Mathematical Foundations. Springer, Berlin, Heidelberg (1999). https://doi.org/10.1007/978-3-642-59830-2
10. Gross, J.L., Yellen, J., Zhang, P. (eds.): Handbook of Graph Theory. Chapman and Hall/CRC Press, New York, second edn. (2013)
11. Kuznetsov, S.O., Poelmans, J.: Knowledge representation and processing with Formal Concept Analysis. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery **3**(3), 200–215 (2013). https://doi.org/10.1002/widm.1088
12. Lindig, C.: Concept-based component retrieval. In: Working notes of the IJCAI-95 workshop: Formal Approaches to the Reuse of Plans, Proofs, and Programs. pp. 21–25. Montreal, Canada (1995)
13. Pattison, T.: Interactive visualisation of formal concept lattices. In: Burton, J., Stapleton, G., Klein, K. (eds.) Joint Proc. 4th Intl. Workshop on Euler Diagrams (ED 2014) and the 1st Intl. Workshop on Graph Visualization in Practice (GViP 2014). CEUR Workshop Proceedings, vol. 1244, pp. 78–79 (Jul 28 – Aug 1 2014), `http://ceur-ws.org/Vol-1244/GViP-paper6.pdf`
14. Poelmans, J., Ignatov, D.I., Viaene, S., Dedene, G., Kuznetsov, S.O.: Text mining scientific papers: A survey on FCA-based information retrieval research. In: Perner, P. (ed.) Advances in Data Mining: Applications and Theoretical Aspects (ICDM 2012). pp. 273–287. Springer, Berlin, Heidelberg (2012). https://doi.org/10.1007/978-3-642-31488-9_22
15. Priss, U.: Formal Concept Analysis in Information Science. Annual Review of Information Science and Technology **40**, 521–543 (2006). https://doi.org/10.1002/aris.1440400120
16. Valverde-Albacete, F.J., Peláez-Moreno, C.: Systems vs. methods: an analysis of the affordances of Formal Concept Analysis for information retrieval. In: Carpineto, C., Kuznetsov, S.O., Napoli, A. (eds.) Proc. Workshop Formal Concept Analysis Meets Information Retrieval (FCAIR 2013). vol. 977 (2013), `http://ceur-ws.org/Vol-977/paper13.pdf`