A Collaborative Approach for FCA-Based Knowledge Extraction

My Thao Tang and Yannick Toussaint

Loria-Campus Scientifique,
BP 239 - 54506 Vandoeuvre-les-Nancy Cedex, France
{My-Thao.Tang,Yannick.Toussaint}@loria.fr

Abstract. In this paper, we propose an approach for an FCA-based knowledge extraction process relying on the collaboration between humans and machines. Evaluation of the results is performed on the lattice by experts through an interactive process where they may specify their wishes for changes using a set of predefined operations. Thus, the system then may suggest several strategies to reach their goal. In such an interactive and iterative process, the system converges towards a knowledge model close to the experts' needs. We illustrate the process on a small preliminary experiment.

Keywords: Formal Concept Analysis, Knowledge Extraction, Expert Interaction

1 Introduction

Several approaches ([1–4]) showed how powerful is Formal Concept Analysis (FCA) ([5]) for knowledge modelling and ontology design, and the lattice is usually considered as the knowledge model. This paper focuses on providing domain experts with capabilities to control and customize the lattice using retro-engineering operations. Our approach is motivated by a desire of keeping a trace between text sources and concepts of the resulting ontology. Objects and properties in texts are annotated and used to build the formal context. Annotations and the lattice evolved simultaneously thanks to retro-engineering. In this way, we can keep a trace between the linguistic level and the conceptual level making one two separated processes in literature, namely **semantic annotation** which identifies concepts from an ontology in texts and **ontology building** which builds concepts from texts.

FCA is a bottom-up process which builds concepts from a set of instances of objects and properties (the formal context). To improve the lattice and to make it closer to expert needs, we want to define an iterative and interactive process where experts are asked, at each loop, to evaluate the "quality" of the concepts. Unfortunately, in Knowledge Engineering, building ontology is not a straightforward process and usually results from trial and error process. There are several reasons for experts to ask for changes in the lattice: (1) there may be noise in resources or in the information extraction processes and thus, some

concepts result from this noise, (2) experts are not satisfied with the resulting knowledge model and wish it to be more in accordance with their needs, *i.e.* the application that will use the knowledge model.

The rest of the paper is structured as follows. Firstly, section 2 explains how the system and experts collaborate in building and changing the lattice, illustrates the approach by removing a concept from a lattice. Next, section 3 presents experimental results. Finally, section 4 concludes with a summary and draws perspectives.

2 Collaboration for Changing A Lattice

In our approach, experts do not have access to the formal context; they can browse concepts, run through subsumption paths, look at extents and intents of concepts. Then, they express their wishes to change the lattice using operations. An operation on the lattice is a kind of retro-engineering: experts select an operation on the lattice (ex: remove this concept), the system assesses the impact of the change, suggests different strategies and then, for the chosen strategy, calculates what to change in data for the new lattice to meet the requirements. The lattice is then built accordingly. Experts repeat the process until they reach an agreement between the model and their knowledge.

2.1 Defining Operations on a Lattice

We list basic changes on a lattice to add or remove one element from the lattice (Table 1). There could be more complex operations such as merging two concepts into one, splitting one concept into two sub-concepts, creating a common concept for a set of concepts... For each operation, we need to define an algorithm to compute the set of possible strategies to perform the change, to identify related changes in the lattice, to rank them according to a cost function and to keep a trace in a history. Adding or removing objects or properties in a formal context motivated the development of incremental algorithms to update a lattice [6–9]. Concept removing is somehow complex: several strategies are possible and impacts on other concepts in the lattice (side effects). We detail this operation to explain our approach in the next part.

2.2 Removing a Concept from a Lattice

Removing a concept while keeping the lattice structure means moving it down to one of its children or up to one of its parents. These solutions correspond to what we call different *strategies*.

Strategy 1 moves concept C down to one of its children, C_{child}. The relation I of the formal context should be modified to I^* as follows:

$I^* = I \cup \{(g,m) : m \in \{\texttt{Intent}(\texttt{C}_{\texttt{child}}) \setminus \texttt{Intent}(\texttt{C})\}, g \in \{\texttt{Extent}(\texttt{C}) \setminus \texttt{Extent}(\texttt{C}_{\texttt{child}}), gIm\}$.

	Add	**Remove**
Object	*Add an object to a lattice*	*Remove an object from a lattice*
	Add an object to a concept	*Remove an object from a concept*
Attribute	*Add an attribute to a lattice*	*Remove an attribute from a lattice*
	Add an attribute to a concept	*Remove an attribute from a concept*
	Add an attribute to an object	*Remove an attribute from an object*
Concept	*Add a concept to a lattice*	*Remove a concept from a lattice*
	Add a supper concept to a concept	*Remove a supper concept from a concept*
	Add a sub concept to a concept	*Remove a sub concept from a concept*

Table 1: Basic changes in a lattice.

Strategies	**NOMC**	**NODC**	**NONC**
1.1 Move c_4 down to c_7	2	2	2
1.2 Move c_4 down to c_9	2	2	2
2.1 Move c_4 up to c_0	0	3	0

Table 2: Strategies for removing concept c_4 associated with the number of modified concepts (NOMC), the number of deleted concepts (NODC) and the number of new concepts (NONC) between the current lattice and the new one.

Strategy 2 moves concept C up to one of its parents, C_{parent}. The relation I of the formal context should be modified to I^* as follows:

$I^* = I \setminus \{(g, m) : m \in \{\texttt{Intent(C)} \setminus \texttt{Intent}(\texttt{C}_{\texttt{parent}})\}, g \in \{\texttt{Extent(C)}\}, gIm\}$.

Table 2 shows an example of strategies for removing concept c_4 from the initial lattice given in Fig. 1. Here, we have three strategies, two for moving c_4 down and one for moving c_4 up. In *strategy 1.1*, $c_4 = (\{\texttt{F}, \texttt{HD}\}, \{\texttt{CoWi\_D}\})$ and $c_7 = (\{\texttt{F}\}, \{\texttt{Ca\_RAS}, \texttt{CoWi\_D}\})$. To move concept c_4 down to concept c_7, the attribute in set $\{\texttt{Intent}(\texttt{c}_7) \setminus \texttt{Intent}(\texttt{c}_4)\} = \{\texttt{Ca\_RAS}\}$ is added to the object in set $\{\texttt{Extent}(\texttt{c}_4) \setminus \texttt{Extent}(\texttt{c}_7)\} = \{\texttt{HD}\}$. Similarly, in *strategy 2.1*, to move concept c_4 up to concept c_0 with $\texttt{Intent}(\texttt{c}_0) = \emptyset$, the attribute CoWi_D is removed from the objects in set $\{\texttt{F}, \texttt{HD}\}$.

We benefit from incremental algorithms that have been implemented to build lattices ([6], [8]) not only to avoid complete recalculation of the lattice but also for identifying the different possible strategies to perform a given change in the current lattice. Incremental algorithms, at the i^{th} step, produce the concept set or diagram graph for i first objects of the context. The new object $i + 1^{th}$ is dynamically added by modifying the existing lattice without recomputing the whole lattice from scratch. ([8])

Let us explain our approach through the algorithm for moving concept down - *Strategy 1*. In *Strategy 1*, the set of objects and the set of properties remain unchanged. Only the relation I is enriched. Moreover, a new closed set $(\texttt{Extent(C)}, \texttt{Intent}(\texttt{C}_{\texttt{child}}))$ is a formal concept of the new lattice $\mathcal{L}^*$.

From the new closed set $(\texttt{Extent(C)}, \texttt{Intent}(\texttt{C}_{\texttt{child}}))$ and the existing lattice $\mathcal{L}$, we generate the new lattice $\mathcal{L}^*$ by identifying the categories of concepts. The new lattice is obtained from the existing lattice by taking, deleting or modifying some concepts and creating new concepts. Thus, concepts are classified into four

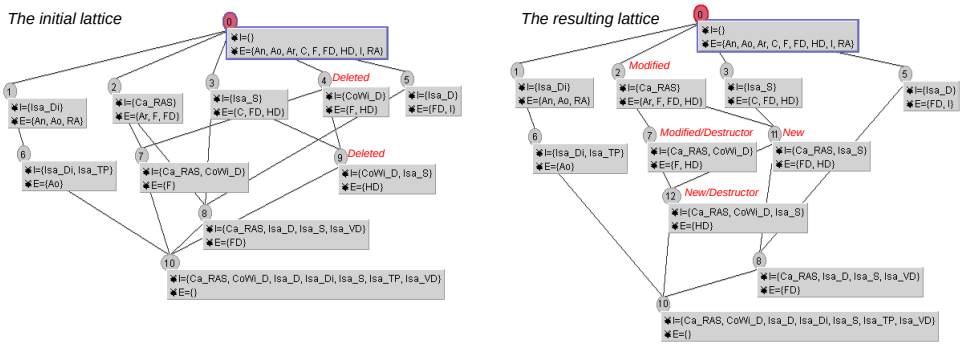

Fig. 1: The initial lattice and the resulting lattice after removing concept c_4 according to the strategy moving concept c_4 down to concept c_7.

categories, *old* concepts, *deleted* concepts, *modified* concepts and *new* concepts as follows. Let C be a concept in $\mathcal{L}^*$:

- C is an *old* concept if there exists a concept with the same `Extent(C)` and `Intent(C)` in $\mathcal{L}$.
- C is a *modified* concept if there exists a concept with the same `Intent(C)` in $\mathcal{L}$ but the extent is different from `Extent(C)`.
- C is a *new* concept if `Intent(C)` doesn't exist in $\mathcal{L}$.
- C in $\mathcal{L}$ is a *deleted* concept if `Intent(C)` doesn't exist in $\mathcal{L}^*$.

Moreover, we identify *destructors*. A *destructor* is a concept that a *deleted* concept will jump to. Thus links to the *deleted concepts* should be reported to *destructors*.

Fig. 1 shows an example of removing concept c_4 according to the strategy *moving it down to concept* c_7. Here, two concepts, c_4 and c_9, in the initial lattice, are *deleted*; two concepts, c_2 and c_7, are *modified*, adding the object `HD` to their extent; two *new* concepts, c_{11} and c_{12}, are created. In addition, concept c_7 is *destructor* of the *deleted* concept c_4 and concept c_{12} is *destructor* of the *deleted* concept c_9.

The algorithm calculating the new lattice is based on the fundamental property of lattices ([10]). For any closed set (X, X') in the lattice:

$$X' = f(X) = \bigcap_{x \in X} f(\{x\}) \qquad \text{and } X = g(X') = \bigcap_{x' \in X'} g(\{x'\})$$

Moreover, for any set of $f(x)$ (resp. $g(\{x'\})$), their intersection should be in the lattice.

The main process of the algorithm is to update the set of intersections of extents. We perform a top-down traversal of the lattice $\mathcal{L}$ to identify whether a concept is *old*, *modified* or *deleted*, to create *new* concepts and to keep the information of *destructors*.

Thanks to the categories of concepts, the resulting lattice can be calculated and we can keep a trace from the previous lattice. By this way, we can know the impact of a change on the lattice (lists of modified, deleted and new concepts).

Finally, all the strategies are suggested to the expert with their impacts (Table 2) so that the expert can consider choosing one strategy. Once the expert makes a choice, the system will update the lattice accordingly.

3 Experiment of Removing A Concept

We experimented our approach for extracting knowledge from a real text corpus of *Fibromuscular Dysplasia* extracted from PubMed[1]. The corpus consists of 400 texts.

In the preprocessing step, we used SemRep ([11]) to annotate the texts. 2402 annotation objects were identified from the corpus, of which only 668 objects are shown in the right or left side of a triplet relationship and 481 objects on the right side. 36 different relationships are identified in these texts. Given an annotation (`object1`, `relationship`, `object2`), we consider `object1` as an object and the name of the relation is concatenated with `object2` to become a binary attribute of `object1`. For example, `Fibromuscular Dysplasia` is an object and `ISA_Disease` is one of its attributes. SemRep annotation process can be noisy but, of course, no automatic tool is perfect. Moreover, any annotation process could annotate the texts, including manual annotation. This is one of our goals to take the advantage of expert interaction in the construction of the knowledge model.

Next, we built a Java script to transform the set annotations into a formal context. The lattice was produced by Galicia[2] and exported as the XML document. The context built from our corpus contains 481 objects, 545 attributes formed by the association of the relationship of the object with which it relates. The lattice contains 523 concepts and the longest path from the *Top* to the *Bottom* is 7.

Then, we implemented a system with operation removing a concept from a lattice. When an expert, during the evaluation phase, asks for removing a concept, the system suggests a list of strategies and makes explicit their impacts on the lattice (lists of modified, deleted and new concepts) to the expert. Once the expert chooses one strategy, the system will update the lattice accordingly.

The required time for the expert to remove concepts has not yet been evaluated. Our experience shows that a judicious choice of a strategy in order to remove concepts has a strong impact on the speed of convergence towards a satisfactory knowledge model.

4 Conclusion and Perspective

We presented a collaborative approach for FCA-based knowledge extraction. Our study shows how domain experts and machines can collaborate in building and changing a lattice. The system handles changes in a lattice and keeps a trace

[1] http://www.ncbi.nlm.nih.gov/pubmed
[2] http://sourceforge.net/projects/galicia/

from the previous lattice. In that way, experts know the impact of a change. Our iterative and interactive approach takes advantage from FCA and reduces limitations due to the use of a formal method to model a complex cognitive process.

This approach opens several perspectives. We can do more to help the expert. Refining the cost function associated to changes can make easier the choice of one strategy. Tagging concepts that the expert agrees with and wants to keep unchanged all along the process could reduce the number of the suggested strategies. Performing several changes at once needs also more investigations.

References

1. Bendaoud, R., Napoli, A., Toussaint, Y.: Formal concept analysis: A unified framework for building and refining ontologies. In Gangemi, A., Euzenat, J., eds.: EKAW. Volume 5268 of Lecture Notes in Computer Science., Springer (2008) 156–171
2. Obitko, M., Snásel, V., Smid, J.: Ontology design with formal concept analysis. In: Proceedings of the CLA 2004 International Workshop on Concept Lattices and their Applications, Ostrava, Czech Republic, September 23-24, 2004. (2004)
3. Cimiano, P., Völker, J.: Text2onto - a framework for ontology learning and data-driven change discovery. In Springer, ed.: Proceedings of Natural Language Processing and Information Systems (NLDB 2005). Number 3513 in Lecture Notes in Computer Science (LNCS) (June 2005) 227–238
4. Huchard, M., Napoli, A., Rouane, M.H., Valtchev, P.: A proposal for combining formal concept analysis and description logics for mining relational data. In: proceeding of the 5th International Conference Formal Concept Analysis (ICFCA'07), Clermond-Ferrand, France (2007)
5. Ganter, B., Wille, R.: Formal Concept Analysis, Mathematical Foundations. Springer (1999)
6. Godin, R., Missaoui, R., Alaoui, H.: Incremental concept formation algorithms based on Galois (concept) lattices. Computational Intelligence **11**(2) (May 1995) 246–267
7. Valtchev, P., Missaoui, R.: Building galois (concept) lattices from parts: Generalizing the incremental approach. In Delugach, H., Stumme, G., eds.: Proceedings of the ICCS 2001. Volume 2120 of LNCS., Springer Verlag (2001) 290–303
8. Kuznetsov, S.O., Obiedkov, S.A.: Comparing performance of algorithms for generating concept lattices. J. Exp. Theor. Artif. Intell. **14**(2-3) (2002) 189–216
9. Merwe, D., Obiedkov, S., Kourie, D.: Addintent: A new incremental algorithm for constructing concept lattices. In Eklund, P., ed.: Concept Lattices. Volume 2961 of Lecture Notes in Computer Science. Springer, Berlin/Heidelberg (2004) 205–206
10. Barbut, M., Monjardet, B.: Ordre et classification, Algèbre et combinatoire. Hachette, Paris (1970)
11. Rindflesch, T.C., Fiszman, M.: The interaction of domain knowledge and linguistic structure in natural language processing: Interpreting hypernymic propositions in biomedical text. Journal of Biomedical Informatics **36**(6) (2003) 462–477