Concepts in Application Context

Steffen Staab[1,2][0000−0002−0780−4154]

[1] Institute for Web Sciencer and Technologies, Universitt Koblenz-Landau, Germany
staab@uni-koblenz.de
https://west.uni-koblenz.de
[2] WAIS Research Group, University of Southampton, UK
s.r.staab@soton.ac.uk
https://wais.ecs.soton.ac.uk

Abstract. Formal concept analysis (FCA) derives a hierarchy of concepts in a *formal context* that relates objects with attributes. This approach is very well aligned with the traditions of Frege, Saussure and Peirce, which relate a signifier (e.g. a word/an attribute) to a mental concept evoked by this word and meant to refer to a specific object in the real world. However, in the practice of natural languages as well as artificial languages (e.g. programming languages), the *application context* often constitutes a latent variable that influences the interpretation of a signifier. We present some of our current work that analyzes the usage of words in natural language in varying application contexts as well as the usage of variables in programming languages in varying application contexts in order to provide conceptual constraints on these signifiers.

Keywords: FCA · Semantics · Programming · Word Embeddings

1 Introduction

In this talk we want to bridge between and explore several research communities that consider the conceptual analysis of various kinds of objects: First, the natural language processing research community which has seen a rapid growth and enormous successes derived from better conceptual understanding of words. Second, the programming language research community which has always used conceptual analysis, including formal concept analysis, in the past, but has not fully exploited its potential in the past decade.

In both research areas we observe that there is a need for conceptual analysis that varies and adapts according to the *context of use* of signifiers/attributes. We try to derive such need from the fundamental consideration of how we use words in communication situations, that arise both in the use of natural language and the use of programming languages.

It is germane to this undertaking that we cannot and do not want to claim completeness of our exploration and our knowledge of the above mentioned fields. However, we hope to stimulate a fruitful discussion for extending methods of FCA and furthering its uptake in various fields.

2 Concept Analysis as a Foundational Means in Communication

The core model of Formal Concept Analysis [15] represents objects and attributes in a *formal context*:

Definition 1 (Formal Context). *A formal context* $\mathbb{K} := (G, M, I)$ *is a triple comprised of a set of objects* G*, a set of attributes* M *and an indicidence relation* $I \subseteq G \times M$ *encoding that* g *has attribute* m *iff* gIm*.*

It computes the formal concepts of this context by:

Definition 2 (Formal Concept). *For* $A \subseteq G$ *and* $B \subseteq M$*, define* $A' := \{m \in M | \forall g \in A : gIm\}$*,* $B' := \{g \in G | \forall m \in B : gIm\}$*. Then a pair* (A, B) *with* $A' = B$ *and* $B' = A$ *is a formal concept.* A *is the extent,* B *is the intent of the concept.*

We may bridge between formal concept analysis and the usage of words, or more generally *signifiers*, in communication. Following the tradition of Frege, Saussure or Pierce, we may illustrate the usage of signifiers in Figure 1.[3]

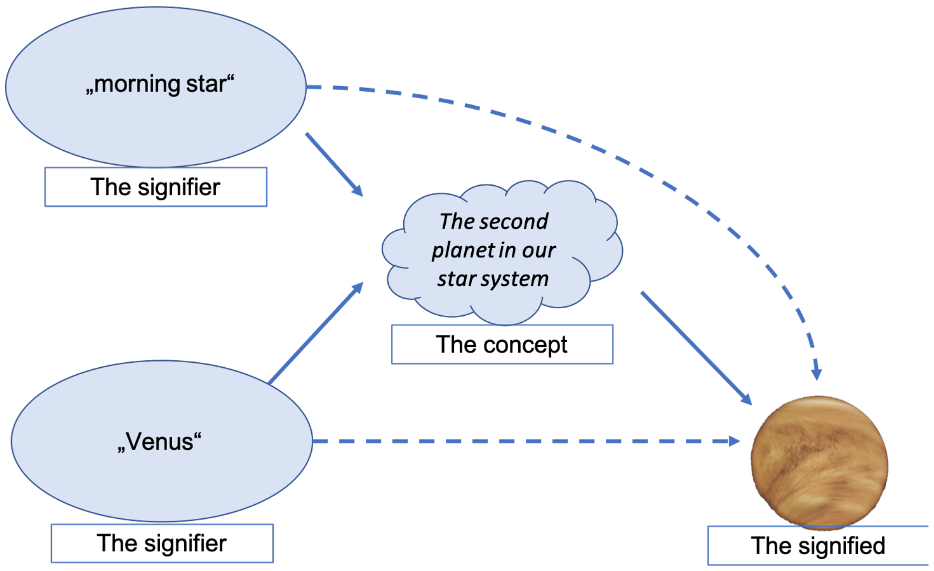

Fig. 1. A signifier evokes a concept in the mind of a recipient of communication allowing her to identify the signified object.

[3] NASA - NSSDC Photo Gallery Version:ftp://nssdcftp.gsfc.nasa.gov/photo_gallery/hi-res/planetary/venus/pvo_uv_790226.tiff, Public Domain, https://commons.wikimedia.org/w/index.php?curid=10914

Indeed, different signifiers such as “Venus” or “morning star” may be used to evoke the same mental concept in the receiver of a message, which allows him to dereference, i.e. to point, to the actual object, which might be the corresponding planet in our solar system. We may understand formal concept analysis as a means that simulates mental concept formation. Investigating the *object*, which is referred to as planet Venus, we may come up with its attributes, such as its orbit around the sun, but also its naming attributes “Venus” and “morning star”. Based on these attributes and the set of objects concerned, which is a singleton in this specific example, we may form the concept in our mind and use the naming attributes, “Venus” or “morning star”, to point to the intended object.

3 Concepts in Natural Language Processing Using Word Embeddings

3.1 Global Representations of Signifiers in Concept Space

Methods of natural language processing usually come with very limited knowledge of objects in the world, but rather with massive text corpora based on which the usage of signifiers may be analysed. In this vein, we have analysed the usage of signifiers in their *textual contexts* using formal concept analysis in order to determine concepts and conceptual relationships [3]. Being based on standard formal concept analysis our approach mapped each signifier into a binary vector representation indicating the words that would co-occur (or not) with the signifier in a given corpus.

Likewise approaches had pursued such representation of signifiers in word space [7] much earlier. However, they could not preserve structural information (e.g. *isa*-relationships) and they suffered from the very high dimensionality of these representations, as the length of vectors representing signifiers was given by the number of words that occurred in a text corpus (i.e. in the order of 10^4 to 10^6).

Word2Vec [9] has been a turning point in such research, efficiently and effectively mapping these vectors into a lower dimensional space with vector lengths significantly below 1000. Even some structural information is partially preserved by the offset between different representations of signifiers in this *concept space*. Thus, the vector offset between concept referred to by signifiers such as “king” and “queen” resembles the vector offset between “man” and “woman”.

3.2 Context-dependent Representations of Signifiers in Concept Space

Figure 2[4] illustrates the problem with all of these approaches. Depending on their textual contexts signifiers like “morning star” and “Venus” may alternatively

[4] Sketch of mace from https://commons.wikimedia.org/wiki/File:Boeheim_Morgenstern_01.jpg#/media/File:Boeheim_Morgenstern_01.jpg, Public domain

refer to weapons (cf. Table 1, rows 3 and 4) or a Greek goddess, rather than to the planet Venus (cf. Table 1, rows 1 and 2).

Table 1. Various textual contexts of "morning star" evoking varying concepts of planet Venus (rows 1,2) and mace (3,4).

1	Morning star, most commonly used as a name for the planet Venus when it appears in the east before sunrise
2	The Egyptians knew the morning star as Tioumoutiri and the evening star as Ouaiti.
3	A morning star is any of several medieval club-like weapons consisting of a shaft with an attached ball adorned with one or more spikes
4	The Morning star is normally considered to be a one handed weapon but it was also a polearm weapon

More recent and most successful approaches such as ELMo [10] and BERT [4] also analyse the words that co-occur with signifiers, however they map each individual occurrence of a signifier into an individual vector in concept space. A signifier like "Venus" is no longer represented by a single vector in concept space, but rather by samples from a probability distribution that reflects in which word contexts it has been observed. We posit that the analysis of such a probability distribution lends itself also to an abstraction that discretizes thousands (or more) observations of a signifier into a more manageable set of concept representations [11], though further experiments are still to be performed.

4 Concepts in Programming Languages Using Queries

Formal concept analysis has been successfully used in several stages of the software engineering lifecycle. For example, Hesse and Tilley [5] describe the use of formal concept analysis in the early stages of the software development process. Core is the elucidation and formation of important programming concepts, such as classes and components:

> *"FCA allows a "crossing of perspectives' – between the functional view represented by the use cases and the data view implied by the 'things' occurring there."*

Just like natural language signifiers appear in a variety of contexts, the same may apply to signifiers, terms and expressions, in programming languages. An often-used software architecture relies on one or several databases that facilitate the integration of data and serve a multitude of applications (cf. Figure 3).

Each particular data object is put into a variety of application contexts. Each application context assumes its own set of concepts. These are often formed by queries.

Consider the following SPARQL query [1] which selects all researchers ?X who are members of some research group that is part of one of the institutes of the department of computer science, "uniko:informatik".

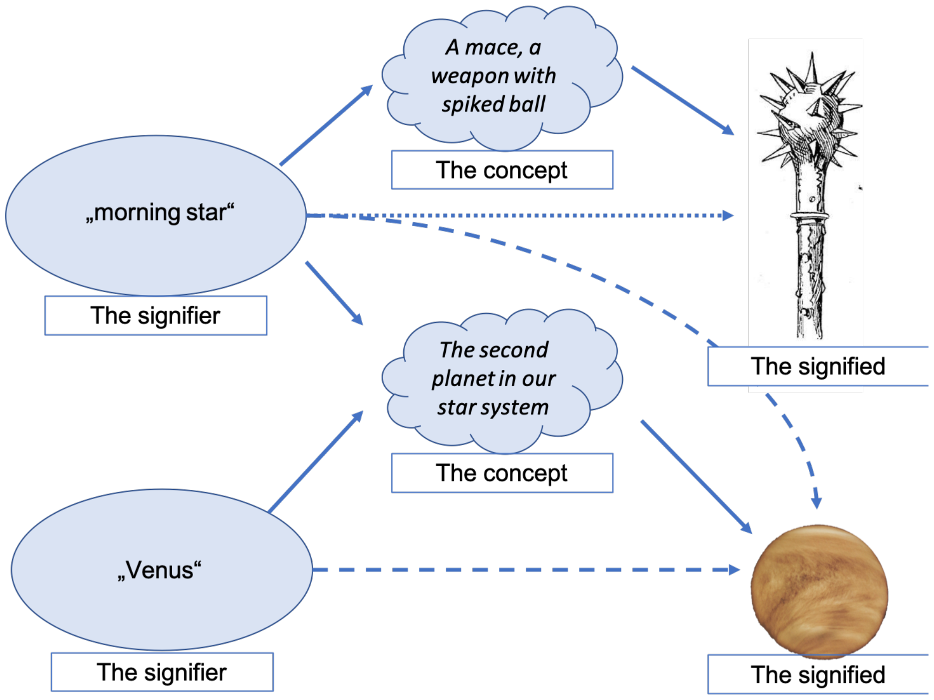

Fig. 2. A “morning star” may allude to different concepts and objects.

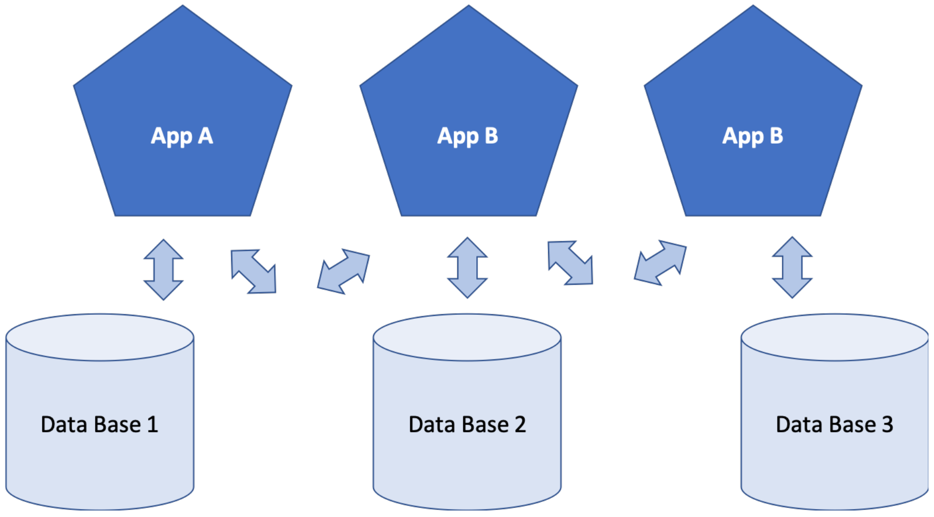

Fig. 3. Integration data bases serve a multitude of applications, which put objects in a variety of contexts.

```
SELECT ?X
WHERE  {
         ?X rdf:type :Researcher.
         ?X :memberOf ?Y.
         ?Y rdf:type :ResearchGroup.
         ?Y :memberOf ?Z.
         ?Z rdf:type :Institute.
         ?Z :memberOf     uniko:Informatik.
}
```

The extension of this query is the set of objects, i.e. the extension of a concept, which we might call “uniko:Informatik-researcher”. If the different institutes operate different databases (“data base 1” to “data base 3”), their extension will vary, but the intension might stay the same. If the different data bases come with different schemata/ontologies there might even be the need to describe such concepts using various intensional descriptions.

We have extended the programming language Scala [12] such that we can embed these queries into programm code, comparable to LINQ [8], but in addition we derive types statically and dynamically based on how the queries represent programming concepts and include these concepts as part of the static (and, sometimes, dynamic) analysis [6].

Core to this approach is the *ad hoc* definition of the objects G and attributes M that we use to describe the concepts. So far, we have used description logics [2] to do this type inference. Hence, the above query would receive the type:

$$\begin{aligned} &Researcher \sqcap \exists memberOf. \\ &\qquad \exists(ResearchGroup \sqcap \exists.memberOf. \\ &\qquad\qquad (Institute \sqcap \exists memberOf.\{uniko:Informatik\})) \end{aligned}$$

However, we are further exploring closed-world representations of types, where formal concept analysis might come into play.

5 Opportunities and Challenges for Formal Concept Analysis

We gave two example areas of conceptual analysis that require contextualization. While formal concept analysis is based on the notion of *formal context* to our knowledge there is only limited work that can handle the *application context* of signifiers/attributes and objects relating them over multiple formal contexts.[5]

We agree with a conclusion that Snelting [14] made when he surveyed the use of FCA in software analysis. He states that:

[5] See [13] for a survey on the integration of description logics and formal concept analysis. It includes the representation of multiple formal contexts to represent concepts in non-hierarchical relationships.

"Future work in lattice theory must show whether the structure theory of concept lattices can be extended in such a way that typical local situations occuring in software analysis can be handled."

We see this as a tremendous opportunity in software engineering, but also in natural language processing. In the latter case, black box systems building on ELMo or BERT may excel when it comes to tasks such as question answering, but when the system is to be made transparent there is a need to elucidate and discretize structures to a level of generalization understandable for the human user — and FCA may come in handy for this purpose.

References

1. Sparql 1.1 query language. Tech. rep., W3C Recommendation (21 March 2013), http://www.w3.org/TR/sparql11-query/
2. Baader, F., Horrocks, I., Lutz, C., Sattler, U.: Introduction to Description Logic. Cambridge University Press (2017)
3. Cimiano, P., Hotho, A., Staab, S.: Learning concept hierarchies from text corpora using formal concept analysis. J. Artif. Intell. Res. **24**, 305–339 (2005). https://doi.org/10.1613/jair.1648, https://doi.org/10.1613/jair.1648
4. Devlin, J., Chang, M., Lee, K., Toutanova, K.: BERT: pre-training of deep bidirectional transformers for language understanding. CoRR **abs/1810.04805** (2018), http://arxiv.org/abs/1810.04805
5. Hesse, W., Tilley, T.: Formal concept analysis used for software analysis and modelling. In: Formal Concept Analysis, Foundations and Applications. pp. 288–303 (2005). https://doi.org/10.1007/11528784_15, https://doi.org/10.1007/11528784_15
6. Leinberger, M., Lämmel, R., Staab, S.: The essence of functional programming on semantic data. In: Programming Languages and Systems - 26th European Symposium on Programming, ESOP 2017. pp. 750–776 (2017). https://doi.org/10.1007/978-3-662-54434-1_28
7. Lin, D., Pantel, P.: Induction of semantic classes from natural language text. In: Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining, San Francisco, CA, USA, August 26-29, 2001. pp. 317–322 (2001), http://portal.acm.org/citation.cfm?id=502512.502558
8. Meijer, E., Beckman, B., Bierman, G.M.: LINQ: reconciling object, relations and XML in the .net framework. In: Proceedings of the ACM SIGMOD International Conference on Management of Data, Chicago, Illinois, USA, June 27-29, 2006. p. 706 (2006). https://doi.org/10.1145/1142473.1142552, https://doi.org/10.1145/1142473.1142552
9. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: 27th Annual Conference on Neural Information Processing Systems 2013. December 5-8, 2013, Lake Tahoe, Nevada, United States. pp. 3111–3119 (2013), http://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality
10. Peters, M.E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., Zettlemoyer, L.: Deep contextualized word representations. In: Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational

Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers). pp. 2227–2237 (2018), https://aclanthology.info/papers/N18-1202/n18-1202

11. Schmelzeisen, L., Staab, S.: Learning taxonomies of concepts and not words using contextualized word representations: A position paper. CoRR **abs/1902.02169** (2019), http://arxiv.org/abs/1902.02169
12. Seifer, P., Leinberger, M., Lämmel, R., Staab, S.: Semantic query integration with reason. The Art, Science, and Engineering of Programming **3**(3) (2019). https://doi.org/10.22152/programming-journal.org/2019/3/13
13. Sertkaya, B.: A survey on how description logic ontologies benefit from FCA. In: Proceedings of the 7th International Conference on Concept Lattices and Their Applications, Sevilla, Spain, October 19-21, 2010. pp. 2–21 (2010), http://ceur-ws.org/Vol-672/paper2.pdf
14. Snelting, G.: Concept lattices in software analysis. In: Formal Concept Analysis, Foundations and Applications. pp. 272–287 (2005). https://doi.org/10.1007/11528784_14
15. Wille, R.: Restructuring lattice theory: An approach based on hierarchies of concepts. Ordered Sets pp. 445–470 (1982)