TELESUP
Textual Self-Learning Support Systems

Sebastian Furth[1] and Joachim Baumeister[1,2]
[1] denkbares GmbH, Friedrich-Bergius-Ring 15, 97076 Würzburg, Germany
[2] University of Würzburg, Institute of Computer Science,
Am Hubland, 97074 Würzburg, Germany
{firstname.lastname}@denkbares.com

Abstract. The regular improvement and adaptation of an ontology is a key factor for the success of an ontology-based system. In this paper, we report on an ongoing project that aims for a methodology and tool for ontology development in a self-improving manner. The approach makes heavy use of methods known in natural language processing and information extraction.

1 Introduction

Today, intelligent systems successfully provide support in many complex (production) processes. Typical application areas of such systems are processes in mechanical engineering and in the medical domain. The core of an intelligent system is the knowledge base, that monitors the requirements and derives support actions.

The development of the knowledge base is usually complex and time-consuming, since complex correlations need to be considered for the derivation knowledge. In domains with frequent changes of the knowledge, for instance, new experiences in processes, it is necessary to frequently modify/adapt the knowledge base. This continuous improvement/adaptation of the knowledge base is a key factor for the long-term success of the system. As the original creation of the knowledge the continuous adaptation is also a complex and time-consuming task. The goal of the presented project is the implementation of a development tool for support systems that includes self-learning capabilities to regularly adapt the included knowledge base. In the general context of the project SELESUP (Self-Learning Support Systems) various types of sources for learning can be connected. The project SELESUP comprises the sub-projects STRUSUP (Structural Self-Learning Support Systems) and TELESUP (Textual Self-Learning Support Systems) that exploit structured and textual data respectively.

In the TELESUP sub-project we

1. define an ontology as the primary knowledge representation for the knowledge base, and
2. use unstructured data, especially text, as the primary resource for the learning method.

The use of such a tool allows for a significant increase of efficiency concerning the development and maintenance of intelligent support-systems.

2 The TELESUP Process

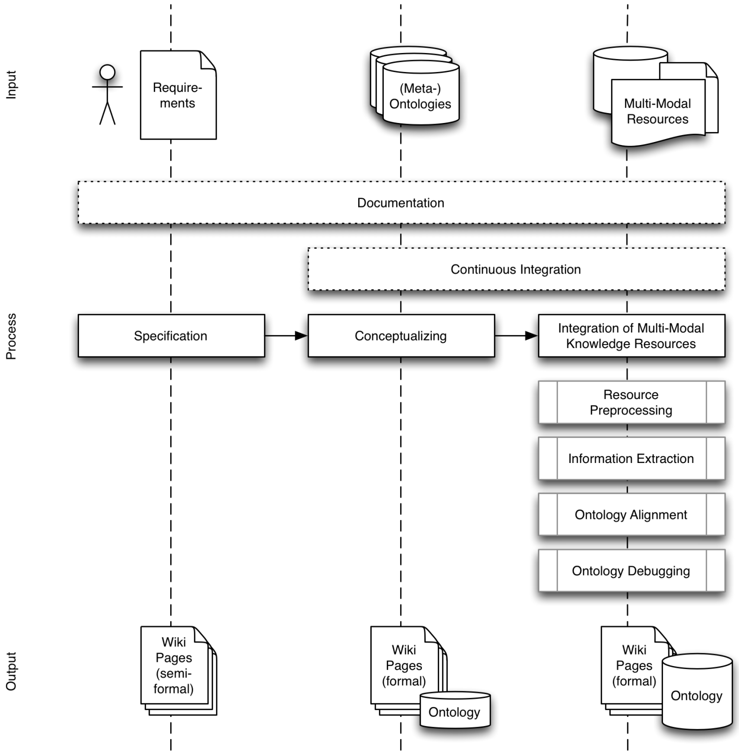


Fig. 1. The input, output and steps of the TELESUP process.

2.1 Problem Description (Distinction)

The TELESUP process, depicted in Figure 1, aims to support the effective development of ontologies for support systems. The application scenario for the developed ontologies is usually located in the context of technical support systems. In this domain, ontologies are often comprehensive and complex. Therefore we consider the development of the core ontology structure as a manual task that requires intensive coordination between multiple domain experts. In order to ensure scalability we usually follow the model “less semantics, more data”, i.e., we focus on the integration of data instead of the usage of heavy semantics. Technical support systems usually need to consider a lot of domain specific multi-modal resources. The pool of these knowledge resources usually increases constantly over time, e.g., due to the introduction of new machines. As the information contained in these resources should also be represented in the ontology, the population must be considered highly volatile.

The scenario described above leads to a couple of requirements constituting an ontology engineering process for textual self-learning support systems. As multiple domain experts and ontology engineers are involved in the development of the core structure of the ontology the process should support collaboration. The integration of vast amounts of multi-modal knowledge resources is usually a challenging, time- and cost-intensive task during the development of an ontology. Therefore we propose the (semi-)automatic population of the ontology by exploiting these resources with methods adapted from the field of Information Extraction and/or the broader field of Natural Language Processing. The preservation of the ontology’s consistency is a major challenge when incorporating (semi-)automatic ontology population approaches in the ontology engineering process. Additionally the multi-modal resources should be considered as valuable sources for ontology refinement suggestions.

2.2 Specification of the Ontology Structure

The first phase of the ontology engineering process considers the collaborative specification of the ontology’s core structure, i.e. identifying required classes and relations between them. Requirements, scope, and the level of formalization is specified in a semi-structured way. In addition to the core structure of the ontology, tests for the validation and verification are specified. The performance/technical specification known from classical software engineering is an appropriate analogy.

2.3 Conceptualizing the Specification

Baumeister et al. [3] proposed the Knowledge Formalization Continuum, i.e. informal knowledge gets subsequently refined into explicit knowledge. Following this idea the specification of the ontology’s core structure is formalized in this phase. The continuum allows for the stepwise conceptualization of the ontology specification, e.g. by first collecting relevant terms for a domain that subsequently get formalized to concepts, which then are described further by using domain specific properties. The formalization follows the level defined in the specification document and uses a standardized ontology language like RDF(S) [21] or OWL [12]. Additionally the phase allows for the integration of (meta-)ontologies, e.g. SKOS [19] or specific upper ontologies for the application scenario. The conceptualization also covers the test specification, i.e., concrete test cases need to be formulated that are able to validate and verify the ontology. There exist various ontology evaluation methods that can be utilized, e.g. data-driven [5] or task-based [15] ontology evaluation. We consider the conceptualization of a complex ontology specification a rather manual task that needs a lot of coordination between ontology engineers and domain experts. Ontology Learning [6] techniques might facilitate the conceptualization by providing suggestions that can be used as basis for discussions between the experts.

2.4 Integration of Multi-Modal Knowledge Resources

Multi-Modal Knowledge Resources Knowledge usually exists in a variety of forms, ranging from highly structured documents (e.g. XML) to completely unstructured resources (e.g. scanned texts, images, videos etc.). We call these documents multi-modal knowledge resources. In technical support systems relevant examples are all forms of technical documentation, e.g. handbooks, repair manuals, service plans or schematics. Additionally documents created for the production process contain valuable information, e.g. a bill of material can be exploited to suggest a component hierarchy of a product.

Resource Preprocessing The various kinds of multi-modal knowledge resources usually need to be preprocessed to improve accessibility for the subsequent information extraction tasks, e.g., when confronted with PDF documents. In general the goal of this phase is to incrementally add structure to previously unstructured documents. Despite the conversion of the file format (e.g. PDF to XML) typical preprocessing tasks from the field of Natural Language Processing are applied, i.e. segmentation, tokenization, part-of-speech tagging, and the detection of structural elements (e.g. tables, lists, headlines). Another important topic in this phase is data cleaning, i.e., preprocess the data in a way that the results are free from noisy data that might affect the information extraction results.

Extracting Relevant Information One of the main challenges during the integration of multi-modal domain knowledge is the extraction of the relevant information from the different sources. After the resources have been preprocessed they are accessible for information extraction methods, e.g., extraction rules that are typically used in rule-based Information Extraction. In general extraction rules can either be formulated by domain experts or automatically learned using Machine Learning algorithms, e.g. LP2 [7], WHISK [17] or TraBaL [9]. The process presented here allows both the manual formulation as well as the (semi-)automatic learning of rules. For the latter one, terminology created during the specification and/or conceptualization phase might be exploited, i.e., used to annotate documents that then serve as training data for the Machine Learning algorithms. The extraction rules are mainly used to extract candidates for the population of the core ontology structure. Additional information that could potentially serve for refinements of the ontology structure might be considered.

Ontology Alignment An important question when handling the extracted candidates for ontology population is whether they are really new concepts or just synonyms for existing concepts. There exist a variety of metrics that can be utilized to measure the similarity between concepts. Besides the well-established string similarity metrics, e.g., Levenshtein distance [13], more elaborated methods exist that use statistics or even consider the semantic relatedness. A combination of several methods can also be used in an ensemble method, as proposed by Curran [8].

Ontology Debugging When using automatic information extraction methods on large sets of resources usually a huge amount of candidates is generated. A major challenge when automatically deploying these candidates to the existing ontology is keeping the ontology in a consistent state. We define a consistent ontology as an ontology that is not only valid in terms of special semantics (e.g. OWL's consistency check) but also pass predefined test cases representing knowledge about the domain (e.g. the tests specified and conceptualized in the preceding phases). When deploying a set of candidates leads to an inconsistent ontology, then abandoning the complete change set is as unrealistic as tracing down the failure cause manually. Consequently, a method for isolating the fault automatically is necessary. We propose the usage of an ontology debugging mechanism, that is able to find the failure-inducing parts in a change set. The faulty parts should be isolated and manually reviewed by a domain expert and/or ontology engineer.

2.5 Continuous Integration

In addition to the debugging mechanisms applied during the integration of multimodal knowledge resources we propose the use of continuous integration (CI) for the development of knowledge systems, enabling the application of automated tests to avoid breaking an ontology. While the main purpose of the ontology debugging mechanism is tracing down the failure-inducing parts in a large change, CI ensures that the ontology is always in a consistent state. Again the test cases formulated on basis of the test specification can be used in CI.

2.6 Documentation

As in Software Engineering the documentation of the developed ontology is a critical success factor as it is the basis for the deployment of the final ontology. When following the phases described so far one can yield not only an ontology but also huge parts of the documentation. Starting with the specification of the ontology the described phases propose the continuous formalization of the ontology. As this specification is the basis for the development of the ontology's structure it can also serve as a basis for the documentation. Additionally we proposed exploiting multi-modal knowledge resource for the population of the ontology. As the employed information extraction techniques are usually able to hold references to the relevant text occurrences huge parts of the ontology population can be documented by providing links to the original text source. The tool used for the development of the ontology should provide an export feature for the documentation in order to ensure convenient distribution/delivery.

3 Tool Support

3.1 KnowWE

Most of the steps in the ontology engineering methodology described above require tool support. We envision an integrated tool that supports the entire process. KnowWE [4] is a semantic wiki that has recently encountered a significant extension of it's ontology engineering capabilities. Besides the ontology engineering features KnowWE also offers an elaborated plugin mechanism that allows for the convenient extension of KnowWE. Thus KnowWE provides a reasonable platform for the implementation of the tool support.

3.2 Ontology Engineering

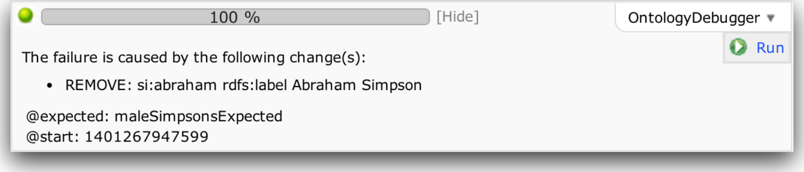
KnowWE provides the possibility to define and maintain ontologies together with strong problem-solving knowledge. As outlined in the following it provides the typical features of an ontology management component [6]. Ontologies can be formulated using the RDF(S) or OWL languages. KnowWE provides different markups for including RDF(S) and OWL: proprietary markups and standardized turtle syntax [20]. In addition, KnowWE already offers possibilities to import (meta-)ontologies, e.g., SKOS. The ontologies are attached to wiki pages, referenced in a special import markup, and can then be used for the development. Besides these ontology management and editing features KnowWE offers a variety of ontology browsing and explanation features. For each concept an info page gives information about the usage of the concept in focus, e.g. which statements or SPARQL queries reference the concept. Additionally arbitrary SPARQL queries can be formulated and even visualized. Besides this a variety of other ontology visualizations are available which are usually used to support the manual ontology engineering, e.g., to explain the existing structure. The ontology engineering process is already supported by the use of continuous integration (CI) as described in [2], enabling the application of automated tests to detect the regression of an ontology.

3.3 Ontology Population

KnowWE already provides possibilities for the basic knowledge engineering, management, and browsing, it thus covers the tool support necessary to specify and conceptualize the ontology structure. However, it lacks support for automatically populating an ontology by exploiting multi-modal knowledge resources. As described above for the exploitation of these resources, the access to preprocessing and information extraction algorithms is necessary. In the TELESUP project we extend KnowWE with connectors to preprocessing and information extraction algorithms. These connectors allow the configuration and the execution of the specific algorithms and provide potential candidates for the population of the ontology. In order to provide a convenient processing of the resources it will be possible to define pipelines of preprocessing and information extraction algorithms. For each pipeline the resources they shall process will be selectable.

3.4 Ontology Evaluation and Debugging

As described before KnowWE already offers different features for evaluating and debugging an ontology. The continuous integration extension of KnowWE allows to test an ontology continuously against specified test cases. Currently these test cases are mostly based on explicitly defining the expected results of SPARQL queries. We already proposed the usage of these test cases in order to find failure-inducing statements in a change set [11]. Therefore we developed a debugging plugin (see Figure 2) that is based on the Delta Debugging idea for software development proposed by Zeller [22]. Within the scope of the TELESUP project we will extend KnowWE's evaluation and debugging features in order to allow for constraint-based and/or task-based evaluation and debugging. For the latter we will also improve KnowWE's revision handling of formal knowledge, e.g. by introducing a time-machine plugin for different knowledge representations that allows the access of specific snapshots of an ontology.



The failure is caused by the following change(s):

- REMOVE: si:abraham rdfs:label Abraham Simpson

@expected: maleSimpsonsExpected
@start: 1401267947599

Fig. 2. KnowWE's delta debugger presenting a failure-inducing change.

4 Related Work

We presented an ontology engineering methodology that proposes to (1) manually specify and conceptualize the core ontology structure, (2) semi-automatically populates the ontology by exploiting multi-modal knowledge resources and (3) strongly emphasizes the quality management. The idea of guiding the ontology engineering process with a methodology is not new. The presented methodology is loosely related to METHONTOLOGY [10]. METHONTOLOGY is a methodology that starts with a formal specification of the ontology, and then acquires and conceptualizes relevant knowledge. It concludes with explicit implementation, evaluation and documentation steps and also allows for the integration of (meta-)ontologies. The major difference to the presented methodology is that TELESUP is able to analyze multi-modal knowledge resources and then automatically populates the ontology. Additionally TELESUP provides more sophisticated evaluation and debugging approaches. Pinto et al. [14] proposed DILIGENT, a methodology that strongly emphasizes the coordination in a distributed ontology engineering process. The methodology proposed by [18] is a five step approach that starts with a feasibility study, specifies the requirements in a kickoff and then continuously refines, evaluates and evolutes the ontology. While the continuous refinement and evaluation of the ontology is comparable to TELESUP, we do not focus on the continuous evolution of the ontology, as we consider the ontology structure to be rather static.

There is a lot of related work regarding the automatic population of ontologies using information extraction technologies. As the identification and selection of appropriate information extraction techniques is subject of the TELESUP project and we focus on the underlying methodology in this paper we do not give a detailed description of related work in this field, but the BioOntoVerb proposed by Ruiz-Martnez et al. [16] is an example for a framework that transforms un-structured, semi-structured and structured data (i.e. multi-modal knowledge resources) to instance data.

Parts of the presented methodology can also be considered related to approaches known from Case-Based Reasoning (CBR) [1], e.g. the specification of the ontology structure and its subsequent conceptualization correspond to the definition of a vocabulary and similarity measures in CBR, while populating the ontology is similar to creating cases.

5 Conclusion

We proposed an ontology engineering methodology that is based on the Knowledge Formalization Continuum and incorporates information extraction techniques for the automatic population of an ontology structure by exploiting multi-modal knowledge resources. The underlying process emphasizes the quality management using Continuous Integration and the ability to trace down failure-inducing changes automatically. We presented the actual state of KnowWE and it's already available ontology engineering abilities. In order to ensure proper tool support for the proposed methodology, we have also outlined the extensions to KnowWE that will be implemented as part of the TELESUP project. Besides the actual implementation of the presented extensions to KnowWE an extensive case study will be the main subject of our future work. The goal of the case study will be to evaluate whether the methodology and the tool support can significantly increase the efficiency concerning the development and maintenance of intelligent support-systems.

Acknowledgments

The work described in this paper is supported by the Bundesministerium für Wirtschaft und Energie (BMWi) under the grant ZIM KF2959902BZ4 "SELESUP – SElf-LEarning SUPport Systems".

References

1. Althoff, K.D.: Case-based reasoning. Handbook on Software Engineering and Knowledge Engineering 1, 549–587 (2001)
2. Baumeister, J., Reutelshoefer, J.: Developing knowledge systems with continuous integration. In: Proceedings of the 11th International Conference on Knowledge Management and Knowledge Technologies. p. 33. ACM (2011)
3. Baumeister, J., Reutelshoefer, J., Puppe, F.: Engineering Intelligent Systems on the Knowledge Formalization Continuum. International Journal of Applied Mathematics and Computer Science (AMCS) 21(1) (2011), http://ki.informatik.uni-wuerzburg.de/papers/baumeister/2011/2011-Baumeister-KFC-AMCS.pdf
4. Baumeister, J., Reutelshoefer, J., Puppe, F.: KnowWE: a Semantic Wiki for knowledge engineering. Applied Intelligence 35(3), 323–344 (2011)
5. Brewster, C., Alani, H., Dasmahapatra, S., Wilks, Y.: Data driven ontology evaluation (2004)
6. Cimiano, P., Mädche, A., Staab, S., Völker, J.: Ontology learning. In: Handbook on ontologies, pp. 245–267. Springer (2009)
7. Ciravegna, F.: $(LP)^2$: Rule Induction for Information Extraction Using Linguistic Constraints (2003)
8. Curran, J.R.: Ensemble methods for automatic thesaurus extraction. In: Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10. pp. 222–229. Association for Computational Linguistics (2002)
9. Eckstein, B., Kluegl, P., Puppe, F.: Towards learning error-driven transformations for information extraction (2011)
10. Fernández-López, M., Gómez-Pérez, A., Juristo, N.: Methontology: from ontological art towards ontological engineering (1997)
11. Furth, S., Baumeister, J.: An Ontology Debugger for the Semantic Wiki KnowWE. In: under review (2014)
12. Hitzler, P., Krötzsch, M., Parsia, B., Patel-Schneider, P.F., Rudolph, S. (eds.): OWL 2 Web Ontology Language: Primer. W3C Recommendation (27 October 2009), available at http://www.w3.org/TR/owl2-primer/
13. Levenshtein, V.: Binary Codes Capable of Correcting Deletions, Insertions, and Reversals. Soviet Physics-Doklady 10(8), 707–710 (1966)
14. Pinto, H.S., Staab, S., Tempich, C.: DILIGENT: Towards a fine-grained methodology for Distributed, Loosely-controlled and evolvInG. In: Ecai 2004: Proceedings of the 16th European Conference on Artificial Intelligence. vol. 110, p. 393. IOS Press (2004)

15. Porzel, R., Malaka, R.: A task-based approach for ontology evaluation. In: ECAI Workshop on Ontology Learning and Population, Valencia, Spain. Citeseer (2004)
16. Ruiz-Martinez, J.M., Valencia-Garcia, R., Martinez-Bejar, R.: BioOntoVerb framework: integrating top level ontologies and semantic roles to populate biomedical ontologies. In: Natural Language Processing and Information Systems, pp. 282–285. Springer (2011)
17. Soderland, S.: Learning information extraction rules for semi-structured and free text. Machine learning 34(1-3), 233–272 (1999)
18. Sure, Y., Staab, S., Studer, R.: Ontology engineering methodology. In: Handbook on ontologies, pp. 135–152. Springer (2009)
19. W3C: SKOS Simple Knowledge Organization System Reference – W3C Recommendation: http://www.w3.org/TR/skos-reference (August 2009)
20. W3C: RDF 1.1 Turtle – W3C Recommendation. http://www.w3.org/TR/turtle/ (February 2014)
21. W3C: RDF Schema 1.1 – W3C Recommendation. http://www.w3.org/TR/rdf-schema/ (February 2014)
22. Zeller, A.: Yesterday, my program worked. Today, it does not. Why? In: Software EngineeringESEC/FSE99. pp. 253–267. Springer (1999)