# A Semantics-based User Interface Model for Content Annotation, Authoring and Exploration

Der Fakultät für Mathematik und Informatik
der Universität Leipzig
eingereichte

# DISSERTATION

zur Erlangung des akademischen Grades

Doktor-Ingenieur
(Dr. Ing.)

im Fachgebiet Informatik

vorgelegt

von **M.Sc. Ali Khalili**

geboren am 26. Juni 1984 in Karaj, Iran

Die Annahme der Dissertation wurde empfohlen von:

1. Prof. Dr. Klaus-Peter Fähnrich, Universität Leipzig
2. Prof. Dr. Roberto García, Universitat de Lleida

Die Verleihung des akademischen Grades erfolgt mit Bestehen der Verteidigung am 26.1.2015 mit dem Gesamtprädikat
magna cum laude.

# Bibliographic Data

**Title:** A Semantics-based User Interface Model for Content Annotation, Authoring and Exploration
**Author:** Ali Khalili
**Institution:** Universität Leipzig, Fakultät für Mathematik und Informatik
**Statistical Information:** 182 pages, 78 figures, 11 tables, 165 literature references

## Abstract

The Semantic Web and Linked Data movements with the aim of creating, publishing and interconnecting machine readable information have gained traction in the last years. However, the majority of information still is contained in and exchanged using unstructured documents, such as Web pages, text documents, images and videos. This can also not be expected to change, since text, images and videos are the natural way in which humans interact with information. Semantic structuring of content on the other hand provides a wide range of advantages compared to unstructured information. Semantically-enriched documents facilitate information search and retrieval, presentation, integration, reusability, interoperability and personalization. Looking at the life-cycle of semantic content on the Web of Data, we see quite some progress on the backend side in storing structured content or for linking data and schemata. Nevertheless, the currently least developed aspect of the semantic content life-cycle is from our point of view the user-friendly manual and semi-automatic creation of rich semantic content.

In this thesis, we propose a semantics-based user interface model, which aims to reduce the complexity of underlying technologies for semantic enrichment of content by Web users. By surveying existing tools and approaches for semantic content authoring, we extracted a set of guidelines for designing efficient and effective semantic authoring user interfaces. We applied these guidelines to devise a semantics-based user interface model called WYSIWYM (What You See Is What You Mean) which enables integrated authoring, visualization and exploration of unstructured and (semi-)structured content. To assess the applicability of our proposed WYSIWYM model, we incorporated the model into four real-world use cases comprising two general and two domain-specific applications. These use cases address four aspects of the WYSIWYM implementation: 1) Its integration into existing user interfaces, 2) Utilizing it for lightweight text analytics to incentivize users, 3) Dealing with crowdsourcing of semi-structured e-learning content, 4) Incorporating it for authoring of semantic medical prescriptions.

# Publications

This thesis is based on the following conference and journal publications, in which I have been author or contributor. *At the respective chapter and section*, I included the references to the appropriate publications.

## Conference Publications, peer-reviewed

- *conTEXT – Lightweight Text Analytics using Linked Data*,
  In proceedings of the 11th Extended Semantic Web Conference (ESWC2014) [Khalili et al., 2014].
  * 1st Prize of the AI Mashup Challenge 2014.

- *WYSIWYM Authoring of Structured Content Based on Schema.org*,
  In proceedings of the 14th International Conference on Web Information Systems Engineering (WISE 2013) [Khalili and Auer, 2013b].

- *Semantic Medical Prescriptions – Towards Intelligent and Interoperable Medical Prescriptions*, In proceedings of the IEEE 7th International Conference on Semantic Computing (ICSC 2013) [Khalili and Sedaghati, 2013a].
  * Winner of the WoLE2013 challenge (Doing Good by Linking Entities).
  * Best-poster prize at Leipzig Research Festival for Life Sciences 2012.

- *CrowdLearn: Crowd-sourcing the Creation of Highly-structured e-Learning Content*, In proceedings of the 5th International Conference on Computer Supported Education (CSEDU 2013) [Tarasowa et al., 2013].
  * Nominated for the best-paper award.

- *SlideWiki: Elicitation and Sharing of Corporate Knowledge Using Presentations*, In proceedings of the 18th International Conference on Knowledge Engineering and Knowledge Management (EKAW 2012) [Khalili et al., 2012b].
  * Nominated for the best-paper award.

- *The RDFa Content Editor - From WYSIWYG to WYSIWYM*,
  In proceedings of the 36th International Conference on Computer Software and Applications (COMPSAC 2012) [Khalili et al., 2012a].
  * Best-paper award.

## Journal Publications, peer-reviewed

- *WYSIWYM – Integrated Visualization, Exploration and Authoring of Semantically Enriched Un-structured Content*, Semantic Web Journal [Khalili and Auer, 2014].
- *User Interfaces for Semantic Authoring of Textual Content: A Systematic Literature Review*, Journal of Web Semantics [Khalili and Auer, 2013a].
- *Crowd-sourcing (semantically) Structured Multilingual Educational Content (CoSMEC)*, Open Praxis Journal [Tarasowa et al., 2014].
  * Creative Innovation Project Award 2014 for OpenCourseWare Excellence.
- *A WYSIWYM Interface for Semantic Enrichment of E-Prescriptions using Linked Open Drug Data*, International Journal On Advances in Life Sciences [Khalili and Sedaghati, 2013b].

*For my parents,*
*Simin Ozar & Mohammad Ebrahim*
*who are the smile of my life...*

*And for my wife,*
*Bita*
*who is the best gift of my life...*

# Acknowledgments

PhD research is not a lonely journey. There are people who support, guide and inspire you to pave the way as you go on this journey.

First and foremost I would like to thank my supervisors Prof. Klaus-Peter Fähnrich and Prof. Sören Auer, without their expertise and guidance this thesis would not have been possible. I have been amazingly fortunate to work with Sören who was not only my mentor but also a good friend. He gave me the freedom to explore on my own, and at the same time the guidance to recover when my steps faltered. I would like to appreciate all his contributions of time, ideas and funding to make my PhD experience fruitful and stimulating. I hope that one day I would become as good an advisor to my students as Sören has been to me.

I am also thankful to all my colleagues at AKSW research group for their support and constructive comments during the development of this thesis. Special thanks go to Dr. Axel-C. Ngonga Ngomo, Dr. Jens Lehman, Timofey Ermilov, Ivan Ermilov, Sebastian Hellman, Saeedeh Shekarpour and Amrapali Zaveri.

I would like to thank the German Academic Exchange Service (DAAD) for awarding me the scholarship to pursue my PhD in Germany.

"He who teaches me a word makes me his servant". I am very grateful to all my former teachers and advisors in Iran who taught me new things and helped me to build up my knowledge. I am especially thankful to Dr. Shahriar Mohammadi, Dr. M. Jafar Tarokh, Dr. Eslam Nazemi and Dr. Fereidoon Shams.

I also would like to express my deepest gratitude to my family who never failed to support me in all my endeavors. I am grateful to my father who has not only been a successful manager outside the family but also a wise leader for the whole family. The same level of gratitude goes to my mother as my first principal at school who taught me patience and persistence when coping with problems. I am also grateful to my brothers HamidReza, Hamed and Mohammad Amin who have been supportive in all my steps. I really appreciate their love, inspiration, emotional support, understanding, prayers and endless encouragement to pursue my higher education.

And finally, I would like to thank my beloved wife Bita for her unwavering support and encouragement, which helped me to tolerate and flatten all the difficulties of living abroad. Bita's contribution even goes beyond the family support since she helped me in writing one chapter of this thesis, which was based on a joint interdisciplinary project we collaborated on.

# Contents

Chapter 1

# Introduction

> "I begin with an idea and then it becomes something else." — *Pablo Picasso*

The Semantic Web and Linked Data movements with the aim of creating, publishing and interconnecting machine readable information have gained traction in the last years (cf. *LODStats*[1] [Auer et al., 2012b]). However, the majority of information still is contained in and exchanged using unstructured documents, such as Web pages, text documents, images and videos. This can also not be expected to change, since text, images and videos are the natural way in which humans interact with information. Semantic structuring of content on the other hand provides a wide range of advantages compared to unstructured information:

- For *search and retrieval* enriching documents with semantic representations helps to create more efficient and effective search interfaces, such as faceted search [Tunkelang, 2009] or question answering [Lopez et al., 2011].
- In *information presentation* semantically enriched documents can be used to create more sophisticated ways of flexibly visualizing information, such as by means of semantic overlays as described in [Burel et al., 2009].
- For *information integration* semantically enriched documents can be used to provide unified views on heterogeneous data stored in different applications by creating composite applications such as semantic mashups [Ankolekar et al., 2007].
- To realize *personalization*, semantic documents provide customized and context-specific information which better fits user needs and will result in delivering customized applications such as personalized semantic portals [Sah et al., 2007].
- For *reusability* and *interoperability* enriching documents with semantic representations (e.g. using the Simple Knowledge Organization System (SKOS)[2] and Dublin Core vocabularies[3]) facilitates exchanging content between disparate systems and enables building applications such as executable papers [Muller et al., 2011].

[1] http://stats.lod2.eu
[2] http://www.w3.org/2004/02/skos/
[3] http://dublincore.org/

Natural Language Processing (NLP) technologies (e.g. named entity recognition and relationship extraction) as well as formalisms for the integrated representation of unstructured and semantic content (such as *RDFa* and *Microdata*) aim at bridging the semantic gap between unstructured and semantic representation formalisms. However, in order for humans to truly benefit from this integration, we still need user-friendly interfaces which enable integrated visualization, exploration and authoring of unstructured and structured content. Looking at the life-cycle of semantic content on the Web of Data [Auer et al., 2012a], we see quite some progress on the backend side. For storing structured content, a variety of triple stores have been developed and their performance and maturity improves steadily (cf. recent triple store benchmarking efforts such as the *DBpedia benchmark* [Morsey et al., 2011]). Similarly tools and algorithms for linking data and schemata are progressing and approaches are deployed for the use on the emerging Web of Data [Ngomo et al., 2013]. Nevertheless, the currently *least* developed aspect of the semantic content life-cycle is from our point of view the user-friendly *manual and semi-automatic* creation of rich semantic content.

In this thesis, we propose a semantics-based user interface model which aims to reduce the complexity of underlying technologies for semantic enrichment of content by Web users. This will facilitate the realization of the so called *Social Semantic Web* [Breslin et al., 2009, Siorpaes and Simperl, 2010] – bringing a social novelty, rather than a technical one to the current Semantic Web. By surveying existing tools and approaches for semantic content authoring, we extracted a set of guidelines for designing efficient and effective semantic authoring user interfaces. We applied these guidelines to devise a semantics-based user interface model called What You See Is What You Mean (WYSIWYM) which enables integrated authoring, visualization and exploration of unstructured and (semi-)structured content. To assess the applicability of our proposed WYSIWYM model, we incorporated the model into four real-world use cases comprising two general and two domain-specific applications. These use cases address four aspects of the WYSIWYM implementation: 1) Its integration into existing user interfaces, 2) Utilizing it for lightweight text analytics to incentivize users, 3) Dealing with crowdsourcing of semi-structured e-learning content, 4) Incorporating it for authoring of semantic medical prescriptions.

The specific requirements as well as the benefits of utilizing WYSIWYM user interface model in each use case are also discussed in this thesis.

## 1.1. User Scenario

In this section we describe a user scenario (depicted in Figure 1.1) to motivate our work:

Alice is a European journalist who works for an online magazine called DataWeb magazine. She wants to write an article about the most demanded Data Science[4]

[4]Data Science involves using automated methods to analyze massive amounts of data and to

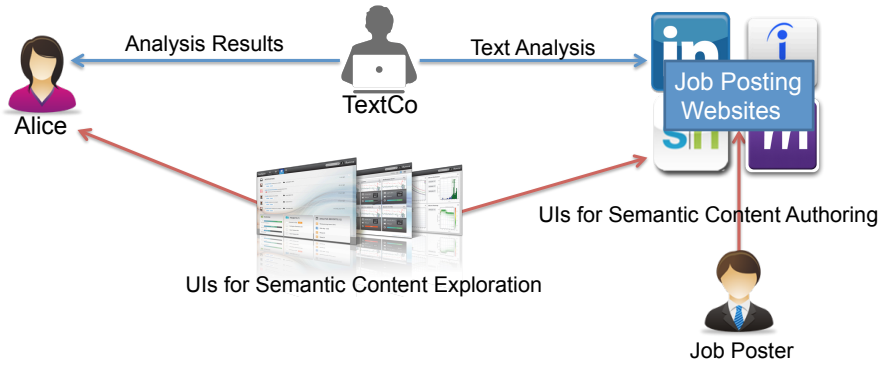


Figure 1.1.: A simple user scenario to exploit semantically-enriched content.

skills for IT jobs on the various job markets in Europe. In the traditional way, Alice should make interviews with companies to collect data about the required skills. Going one step further, Alice can use the information published on the job posting Websites to extract the skills-related data for IT jobs in Europe. Alice needs to read through the full descriptions of each job and manually extract the required Data Science skills. Taking merely LinkedIn into consideration, there are more than 8.000 IT-related jobs posted by the European companies in 2012-2014.[5] Manual analysis of these amount of data is a cumbersome and time-consuming task. Also, Alice wants to prepare different diagrams illustrating the skills demand in different national or regional markets as well as their evolution over time for inclusion into her article. She also plans to regularly review the results in the next 2-3 years and update her readers on recent developments. To facilitate this task, Alice can contact the text analysis company TextCo. She needs to give TextCo all the collected data and wait until they process these data and return her the analysis result which might still be expensive and time consuming. TextCo has much expertise in such NLP tasks and a number of tools at hand. However, these tools have to be configured, adapted and integrated into a specific workflow, by highly-skilled but expensive experts. TextCo estimates 2-3 person months effort to perform this task and costs well above 10.000 Euro. Alice is not sure, whether her article justifies such an investment.

What if Alice (with no or limited knowledge of programming) can perform this sophisticated NLP task with just some clicks herself? The intuitive semantics-based User Interfaces (UIs) for content annotation, authoring and exploration as discussed in this thesis are one way to realize this vision.

extract knowledge from them. (http://datascience.nyu.edu/)

[5] http://context.aksw.org/linkedin/

## 1.2. Motivation

In the following we outline the rational and motivation underlying the research presented in this thesis:

**M1. Semantic content authoring is cumbersome (difficult, time-consuming, error-prone, requires knowledge representation expertise).**
Structure makes data more useful, but at the same time makes data entry more cumbersome [Chang et al., 2013]. Regardless of what additional knowledge is needed for users to semantically annotate or enrich content, the mere requirement to specify information precisely and unambiguously can be burdensome to users. [Ross and Nisbett, 1991] identified this problem as *channel factors* – "small but critical" facilitators that could dramatically impact a person's actions. Channel factor analysis has identified how even simple intervening steps to a task such as needing to explicitly assign a name to a file at time of its creation will impact a user's decision whether to begin the task.

Another barrier is that creating structured content requires more cognitive load from the users because they have to learn extra user interface elements [Van Kleek et al., 2007] while being familiar with Knowledge Representation (KR) techniques – means by which users express and communicate their meanings to machine [Davis et al., 1993]. Even if users grasp the required knowledge for semantic content authoring, the manual process of structured content authoring will be time-consuming and error-prone. People naturally use partial, incomplete, or often vague descriptions of content, however, for semantic authoring of content, they are required to be precise and explicitly state their meaning using the available user interfaces.

**M2. There is a lack of approaches, technologies and tools to facilitate collaboration when authoring (semi-)structured content.**
There is a huge amount of amateur and expert users who collaborate on and contribute to the Social Web. Harnessing the power of such crowds can significantly enhance and widen the practice of semantic content authoring. However, one of the least developed aspects in the current semantic authoring systems is the support of collaboration and crowdsourcing [Khalili and Auer, 2013a]. As discussed by [d'Aquin et al., 2008], the collaboration-centric perspective on the process of semantic content authoring introduces both challenges and interesting research directions. Semantic authoring can take place transparently as a consequence of the activity of a community of users and opportunistic ways of creating (semi-)structured content can emerge from tasks other than the ones explicitly targeting the creation of new content – that is, as a side effect of user activity.

**M3. There is a lack of incentives and instant gratification for users to adopt semantic content authoring.**
One of the main obstacles hampering the adoption of semantic content authoring,

is the lack of incentives and clear benefits to create semantic content [Łazaruk et al., 2012]. The benefits of using semantic content are often de-coupled from the effort of creating the semantic content thus demotivating users to author semantic content. As discussed by [Simperl, 2012], understanding, stimulating and rewarding user's participation in semantic data management would result in massive production of useful semantic content.

**M4. There is a lack of standardization of UI technologies in the domain of semantic content authoring.**
Currently, semantic authoring user interfaces are not standardized and often do not function consistent with user expectations. UI elements serve as proxies through which users can manipulate information objects [Huynh et al., 2003]. Inconsistent UIs are hard for users to master and will impel a cognitive load for the users. For example, if the same control is used to perform different actions for semantic content authoring, it will be impossible for users to apply what they have learned and predict the outcome.

On the other hand, without standard shared user interface elements and techniques, the development time and cost for semantic content authoring applications will be higher. In this situation, more time is spent in re-designing existing UIs rather than focusing on innovating. This will result in a high-barrier in creating sophisticated semantic content authoring applications. Results of surveys reported by [Heitmann et al., 2009], [Paulheim and Probst, 2010] and [Hachey, 2011] support this fact as well.

Standardization of UI technologies in the domain of semantic content authoring will ease user experience by providing consistency across different systems. Standardization will help developers with best practices for exposing functionality to the end user when building Semantic Content Authoring (SCA) applications.

## 1.3. Research Questions and Contributions

Figure 1.2 shows on overview of the research questions addressed in this thesis together with our key contributions to answer them. These research questions were derived from the motivations discussed in Section 1.2.

**RQ1. What are existing approaches for user-friendly semantic content authoring?**

By answering this research question, we aim to create a holistic view on existing approaches and tools for user-friendly semantic content authoring. This is crucial for conceiving guidelines for developing more effective and intuitive semantic authoring interfaces. We divide this general research question into the following more concrete sub-questions:

- RQ1.1. How to classify existing approaches for semantic content authoring?

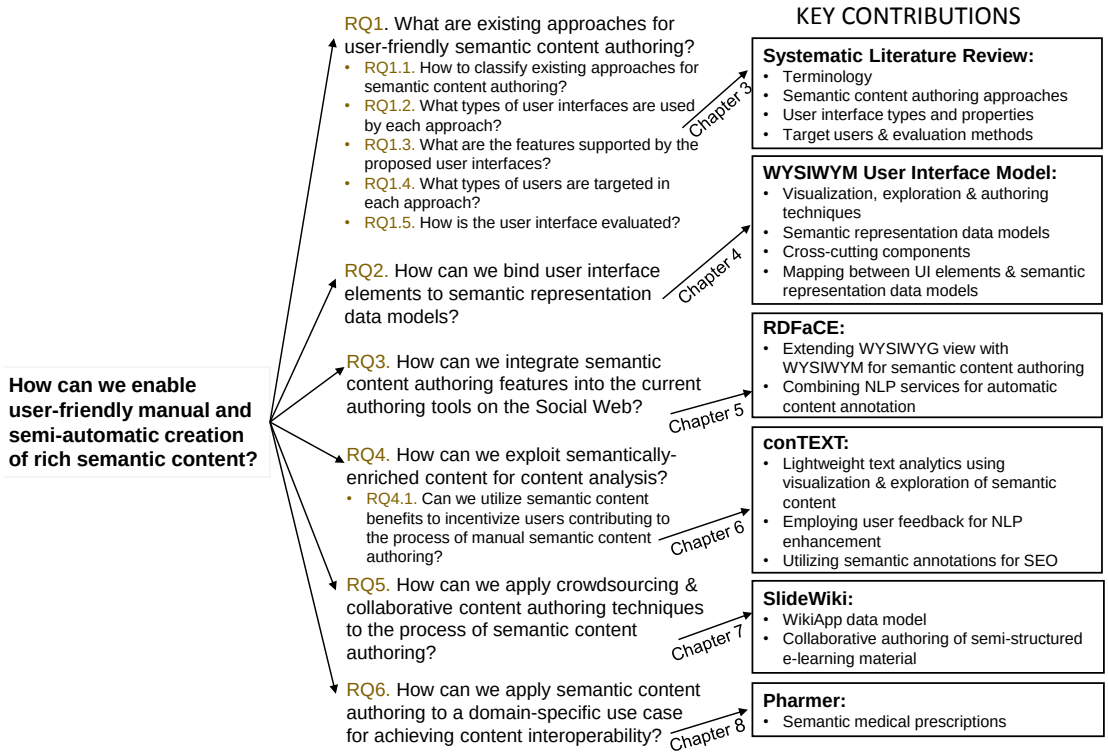


Figure 1.2.: Summary of research questions and key contributions.

- RQ1.2. What types of user interfaces are used by each approach?
- RQ1.3. What are the features supported by the proposed user interfaces?
- RQ1.4. What types of users are targeted in each approach?
- RQ1.5. How is the user interface evaluated?

To answer these research questions, we followed a formal *systematic literature review* process which is discussed in Chapter 3. As a result, we first specify a terminology to define and unify the basic concepts used in the literature. To answer the RQ1.1, we classified existing approaches for SCA into two categories namely Top-Down and Bottom-Up. Furthermore, our data analysis revealed a set of quality attributes for SCA systems together with the corresponding user interface features which are suggested for their realization. These quality attributes plus the UI features are used to answer the RQ1.2 and RQ1.3. In order to answer RQ1.4 and RQ1.5, we extracted the types of users as well as user evaluation methods discussed in the primary studies.

### RQ2. How can we bind user interface elements to semantic representation data models?

By answering this research question, we aim to provide a standardized semantics-based user interface model which is derived by means of bindings between existing semantic representation data model elements and configurations of particular UI elements. The model serves the purpose of providing a terminology for software engineers and UI designers to communicate efficiently and effectively thereby designing and implementing novel applications for authoring, visualization, and exploration of semantic content.

To answer this research question, we devised the *WYSIWYM* (What-You-See-Is-What-You-Mean) concept and formalized key components of this concept in Chapter 4. The key components included a list of existing UI elements and techniques for content visualization, exploration and authoring, an analysis of existing semantic representation data models, a set of bindings between UI elements and semantic models as well as cross-cutting components to facilitate authoring of semantic content.

### RQ3. How can we integrate semantic content authoring features into the current authoring tools on the Social Web?

By answering this research question, we aim to promote the practice of semantic content authoring among Social Web users with minimum additional efforts. Instead of inventing new authoring tools, we plan to enrich existing content editors with facilities for authoring of semantic content.

To answer this research question, we proposed an approach called *RDFaCE* in Chapter 5. RDFaCE extends the current What You See Is What You Get (WYSIWYG) editors with standard WYSIWYM UI elements therefore enabling semantic content authoring in current Social Web platforms such as Content Management Systems (CMSs), Weblogs and Wikis. In order to facilitate semantic content authoring, we propose to utilize a combination of existing NLP services for automatic content annotation.

### RQ4. How can we exploit semantically-enriched content for content analysis?

By answering this research question, we aim to create a lightweight text analytics architecture of participation based on the semantically enriched content. This allows end-users to exploit Linked Data for analyzing and visualizing their content, be it a Weblog, *Twitter* or *Facebook* feed, Website or any article collection.

To answer this research question in Chapter 6, we created a lightweight text analytics platform called *conTEXT* which provides different analytics views on semantic content by utilizing our WYSIWYM user interface model. Within this research question, we also examined the following sub-question:

- RQ4.1. Can we utilize semantic content benefits to incentivize users contributing to the process of manual semantic content authoring?

Coupling rich views for text analytics to the UIs for semantic content authoring would encourage users to participate in the process of semantic content authoring. As an instant gratification, in conTEXT, we give users direct feedback on what information can be extracted from their works. At the same time we want to incorporate their feedback and revisions of the semantic annotations back in the NLP processing loop. As another user incentive, in conTEXT, we provided mechanisms to exploit semantically-enriched content for Search Engine Optimization (SEO).

**RQ5. How can we apply crowdsourcing & collaborative content authoring techniques to the process of semantic content authoring?**

By answering this research question, we aim to create a mechanism to enable collaborative authoring of (semi-)structured content. The approach is needed to deal with the increased complexity of the (semi-)structured content as well as the operations on this content.

To answer this research question, we proposed a data model called *WikiApp* together with its implementation *SlideWiki* in Chapter 7. WikiApp addresses the collaborative aspects of semantic content authoring and deals with users as its first class citizen. With SlideWiki, we present a domain-specific use case for creating highly-structured content for e-learning.

**RQ6. How can we apply semantic content authoring to a domain-specific use case for achieving content interoperability?**

By answering this research question, we aim to investigate the adoption of semantic content authoring in a specific domain having content interoperability as its area of focus.

To answer this research question, in Chapter 8, we created a customized application called *Pharmer* which enables authoring of semantic medical prescriptions using *Linked Open Drug Data (LODD)*. Semantic prescriptions provide interoperability between patients, physicians, pharmacists, pharma companies and insurance companies.

## 1.4. Thesis in a Glance

As depicted in Figure 1.3, the chapters and sections of this thesis are arranged as follows:

Chapter 2 constitutes a basic scientific background on Semantic Web which is required for the reader to understand the thesis. The chapter starts by defining the Semantic Web followed by discussing the Resource Description Frameworka (RDF) as the basic building block of the Semantic Web. Then, it moves to describing

Ontology concept, different types of ontologies and Schema.org as an example of lightweight ontology. The SPARQL Protocol and RDF Query Language (SPARQL) is then explained as a means to query the web of data. The chapter also includes a short overview on Triplestore as a technology to store RDF data as well as NLP Interchange Format (NIF) to deal with interoperability between natural language processing tools and services.

Chapter 3 provides an overview on the field of SCA. We followed a systematic literature review approach to survey the existing UIs for SCA. The research protocol containing the research questions, search strategies, study selection criteria, data extraction and analysis methods as well as an overview of the included studies are described in this chapter. We define a terminology for the domain of semantic content authoring and elaborate on the results of our systematic review. The results consist of a set of quality attributes together with their corresponding UI types and features required for SCA systems. The quality attributes include aspects such as usability, automation, generalizability, collaboration, customizability and evolvability. Three existing semantic authoring tools are discussed and compared based on the defined quality attributes. Finally, the chapter concludes by an overview on open research and technology challenges in the field of semantic content authoring.

Chapter 4 defines the WYSIWYM UI model for integrated authoring, visualization and exploration of unstructured and (semi-)structured content. The chapter reviews the former use of WYSIWYM term and discusses the existing approaches for binding UIs to semantic models. In this chapter, we apply the results achieved in Chapter 3 to present a formal definition of the WYSIWYM concept including elements such as semantic representation data models, authoring, visualization and exploration techniques as well as bindings and helper components. The main contribution of this chapter lies in providing a comprehensive mapping between existing user interface elements and the elements of semantic representation data models.

Chapter 5 discusses how WYSIWYM can be integrated into the existing rich text editors. As a general use case, we present RDFaCE approach and its implementation which is built on top of the existing WYSIWYG editors to enable semantic content authoring. RDFaCE provides different views on the semantic content suitable for different personas involved in the process of semantic content authoring. We present the automatic content annotation feature which is realized by consuming NLP services. The chapter contains an extensive evaluation of five NLP services and proposes an approach for the combination of these NLP services to achieve superior performance compared to each individual approach. Furthermore, the chapter covers three use cases of RDFaCE including semantic blogging, data journalism and search engine optimization. Finally an evaluation of RDFaCE content authoring environment using a sizable user group and measuring subjective as well as objective usage characteristics are presented.

Chapter 6 demonstrates how we can employ semantic annotations for providing lightweight text analytics. This chapter presents another general use case of the

WYSIWYM model which allows ordinary web users to use sophisticated text analytics – democratizing the NLP usage. In this chapter, we present conTEXT as a platform for lightweight text analytics. conTEXT provides a flexible text analytics architecture of participation by innovative combination of different pieces of web services such as Named Entity Recognition (NER), relation extraction, sentiment analysis, visualization and exploration. One of the main contributions of conTEXT is to incentivize users to perform manual content annotation. In this direction, we discuss how WYSIWYM model in conTEXT can be used to collect user feedback for refining the automatically generated annotations. The chapter also includes a usability and usefulness study of conTEXT using a sizable user group. Finally, the chapter concludes by discussing the main benefits of conTEXT for users.

Chapter 7 presents a domain-specific use case of WYSIWYM model for collaborative authoring of semi-structured e-learning content. In order to deal with the collaboration and crowdsourcing aspects of content authoring, we propose a data model called WikiApp as a refinement of traditional Entity-Relation (ER) data model. Definition of the WikiApp data model together with an approach for model-driven generation of WikiApp implementations are discussed in this chapter. In this chapter, we also present an application called SlideWiki. SlideWiki employs the WikiApp data model together with the WYSIWYM interface for collaborative authoring of educational material – crowdlearning. Two use cases of SlideWiki as a platform for OpenCourseWare authoring and as a platform for elicitation and sharing of corporate knowledge are presented in this chapter. The chapter also elaborates on different features of SlideWiki and provides a usability evaluation of the platform.

Chapter 8 demonstrates another domain-specific use case of the WYSIWYM model. In this chapter, we define the concept of Semantic Medical Prescriptions as intelligent and interoperable medical prescriptions, which are achieved by utilizing semantic annotations. Pharmer as a WYSIWYM implementation to create semantic prescriptions is then presented. Pharmer follows the RDFaCE approach discussed in Chapter 5 with minor customization for the e-health domain (e.g. real-time drug tagging and adverse drug interaction finder). We discuss the Pharmer architecture as well two possible use cases of Pharmer in this chapter. The chapter also includes an example scenario together with the results of our usability evaluation.

Finally, Chapter 9 concludes with a discussion of the contributions of the thesis and proposes future work for each of them.

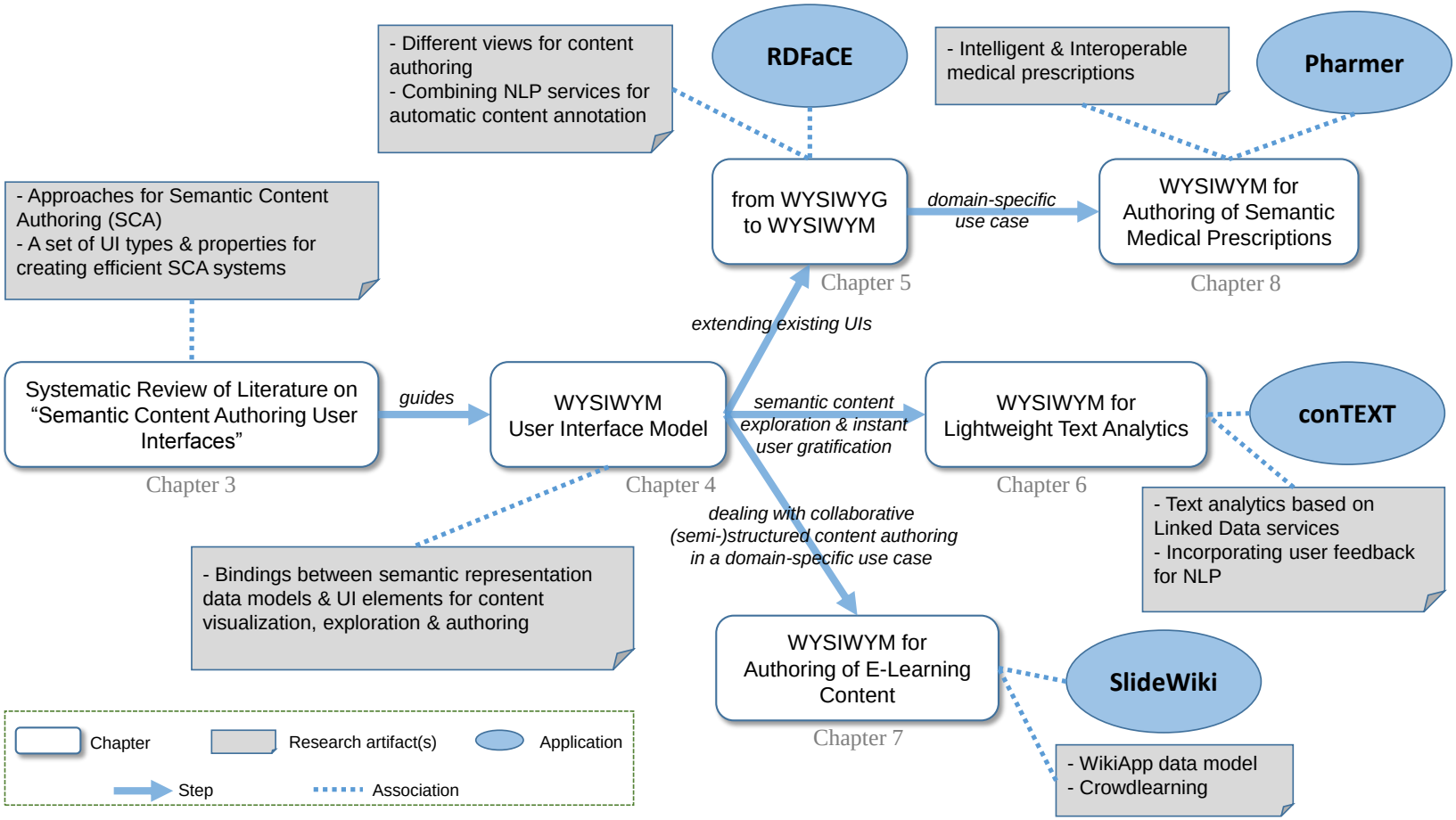


Figure 1.3.: Overview of the chapters together with their corresponding research & application artifacts.

Chapter 2

# Semantic Web Technologies

> "The meaning of things lies not in the things themselves, but in our attitude towards them."
> — *Antoine de Saint-Exupery*

This chapter provides the background knowledge required for understanding the contribution of this work. It gives a general overview of the Semantic Web by describing basic concepts such as the RDF serialization formats, the ontology and its languages in detail. This chapter is mainly based on [Yu, 2007][1].

The rest of the chapter is organized as follows: In Section 2.1, we define Semantic Web. In Section 2.2, we describe RDF data model as the basic building block for the Semantic Web. In Section 2.2.1, Section 2.2.2 and Section 2.2.3 we describe the basic elements of RDF in more detail. In Section 2.2.4, we introduce some RDF serialization formats. In Section 2.3, we define the term Ontology, its classification as well as Schema.org as an example of lightweight ontology. In Section 2.4, we describe the SPARQL query language. Section 2.6 explains the use of natural language processing on the Semantic Web. Finally, in Section 2.5, we shortly explain triplestores.

## 2.1. The Definition of Semantic Web

There are many different definitions of the Semantic Web. Tim Berners-Lee, the inventor of the World Wide Web, defined it as "not a separate Web but an extension of the current one, in which information is given well-defined meaning, better enabling computers and people to work in cooperation." [Berners-Lee et al., 2001] In other words, Semantic Web allows the machines not only to present data but also to process it. The Semantic Web is a "Web of data". Pieces of well-defined data are interlinked to form a global Web, as an extension to the current Web of documents, using the same basic technologies and infrastructure.

There is a dedicated team of people at the World Wide Web consortium (W3C) working to improve, extend and standardize the Semantic Web, and many languages, publications, tools have already been developed (e.g. [Tramp et al., 2010, Heino et al., 2009]). W3C have defined Semantic Web as "the idea of having data on the Web defined and linked in a way that it can be used by machines not just for display purposes, but for automation, integration, and reuse of data across various

[1]Standard components of the Semantic Web and definitions for them were taken from the book as they are following the defined standards and are widely used. Examples for each component are provided by the author.

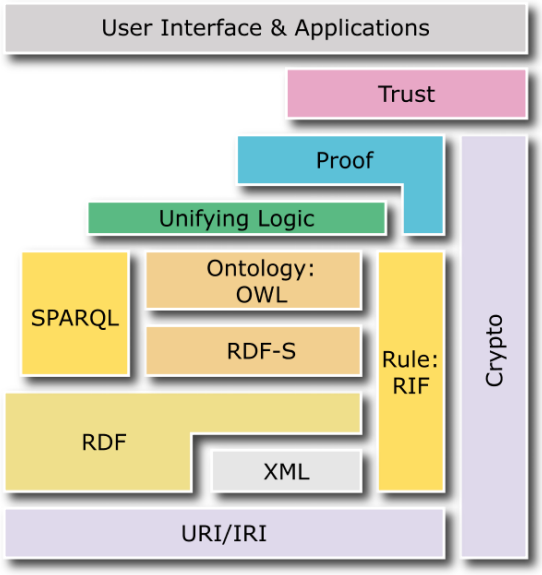


Figure 2.1.: Semantic Web technology stack.

applications." [W3C, 2009] In other words, Semantic Web is the machine-readable Web. Semantic Web can be thought of as an efficient way of representing the data on the World Wide Web, or as a globally linked database.

As depicted in Figure 2.1, Semantic Web depends on several technologies including RDF and URIs. In the following sections we describe each of these technologies in details.

## 2.2. Resource Description Framework (RDF)

RDF is an XML-based language for describing information contained in a Web resource. This Web resource can be anything, for example a Web page or a Website. RDF is the basic building block for supporting the Semantic Web, and is same as HTML is for the conventional Web.

The properties of RDF are:

- RDF is a language recommended by W3C [W3C, 2004] and it is all about metadata,
- RDF is capable of describing any fact (resource) independent of any domain,
- RDF provides a basis for *coding*, *exchanging*, and *reusing* structured metadata,
- RDF is structured; i.e. it is machine-understandable. Machines can do useful operations with the knowledge expressed in RDF,

- RDF allows *interoperability* among applications exchanging *machine understandable* information on the Web.

RDF has several basic elements, namely *Resource*, *Property* and *Statement*, which are discussed in the following subsections.

### 2.2.1. Resource

A resource is any thing that is described by an RDF expressions. The resource can be a Website, a person, an ubiquitous device or anything else. Resource is identified by a **URI**. The rationale of using URIs is that the name of a resource must be globally unique.

In fact, Uniform Resource Locators, commonly used for accessing Web sites, are simply a subset of URIs. URIs take the same format as URLs, e.g. `http://aksw.org/AliKhalili`. The main reason behind this is that the domain name used in the URL is guaranteed to be unique, therefore the uniqueness of the resource is ensured. In that case the domain name is used as a namespace. Unlike URLs, URIs may or may not refer to an actual Website or a Web page.

### 2.2.2. Property

Property is a resource that has a name and can also be used to describe some specific aspect, characteristic, attribute or relation of the given resource. For instance, `http://xmlns.com/foaf/0.1/name`, denotes the name of some thing. In other words, this property relates a resource representing a thing to its name.

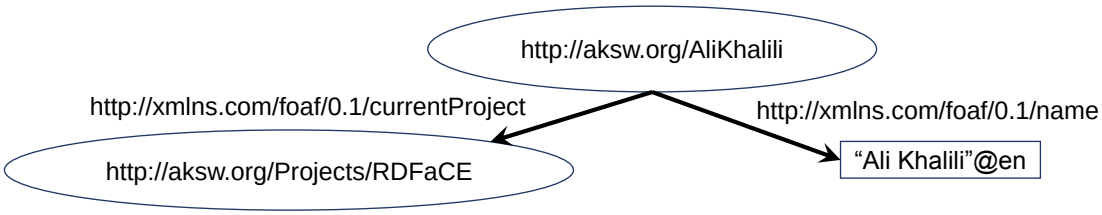


Figure 2.2.: RDF statement represented as a directed graph.

### 2.2.3. Statement

An RDF Statement is used to describe properties of resources. It is also called a triple and has the following format

<resource (subject)> <property (predicate)> <property value (object)>.

The property value (object) can be a string, literal or another resource referenced by the URI. For example:

<`http://aksw.org/AliKhalili`>

<http://xmlns.com/foaf/0.1/currentProject>
<http://aksw.org/Projects/RDFaCE>.

This RDF statement simply states that "The subject identified by http://aksw.org/AliKhalili has a property identified by http://xmlns.com/foaf/0.1/currentProject, whose value is equal to http://aksw.org/Projects/RDFaCE". This means that person "Ali Khalili" has a "currentProject" which is "RDFaCE".

In fact, RDF statements can also be expressed as directed graphs, as shown in Figure 2.2.

It is worth pointing out here that the subject or the object or both can be an anonymous resource, called a "blank node". Blank nodes are used basically when the key purpose of a specific resource is to provide a context for some other properties to appear. In order to distinguish a blank node from the others, the RDF parser generates an *internal* unique identifier for each blank node. In other words, this identifier given to the blank node helps in identifying the node in a certain RDF document, whereas the URI given to a resource is guaranteed to be globally unique.

Since URIs can be large, there is a short format for writing them i.e. by using a prefix. For instance, if we use http://aksw.org/ as a prefix and give it a label e.g. `aksw`, then resource http://aksw.org/AliKhalili can be written as `aksw:AliKhalili`. Similarly, if http://xmlns.com/foaf/0.1/ is used as a prefix with label `foaf`, then the properties http://xmlns.com/foaf/0.1/name and http://xmlns.com/foaf/0.1/currentProject, can be written as `foaf:name` and `foaf:currentProject` in short form. This format is very useful in writing human-readable RDF statements.

Whenever more triples describing a specific resource are added, the machine gets more knowledge about that resource. Table 2.1 shows more RDF statements about Ali Khalili. This means that the resource of Ali Khalili is the subject of other statements, which give more details about that resource. Note that the object of a particular statement can be in turn the subject of other statement(s), e.g. Ali Khalili has a current project identified by URI `akswProject:RDFaCE` and the knowledge base contains more information about that project as well. Also, note that the object of the second and fifth statement (a number and a date) has a trailing datatype. This small knowledge base can also be viewed as a directed graph as shown in Figure 2.3.

Using these simple RDF statements you can pose complex queries to the machine, e.g. "What is the homepage of Ali Khalili's current project?".

### 2.2.4. RDF Serialization Formats

Serializing RDF data is a very crucial issue since different platforms and environments work better with different data formats. There are already several formats such as *RDF/XML*, *Turtle*, *N-Triples*, *Notation3*, *N-Quads* and *JSON-LD* for serializing RDF data. There are also formats such as *RDFa* and *Microdata* to

| Subject | Predicate | Object |
|---|---|---|
| aksw:AliKhalili | rdf:type | foaf:Person |
| aksw:AliKhalili | foaf:age | "30"^^xsd:int |
| aksw:AliKhalili | foaf:skypeID | "alii.khalilii" |
| aksw:AliKhalili | foaf:birthday | "1984-06-26"^^xsd:date |
| aksw:AliKhalili | foaf:name | "Ali Khalili"@en |
| aksw:AliKhalili | foaf:currentProject | akswProject:RDFaCE |
| akswProject:RDFaCE | foaf:homepage | <http://rdface.aksw.org> |

Table 2.1.: Sample RDF statements.

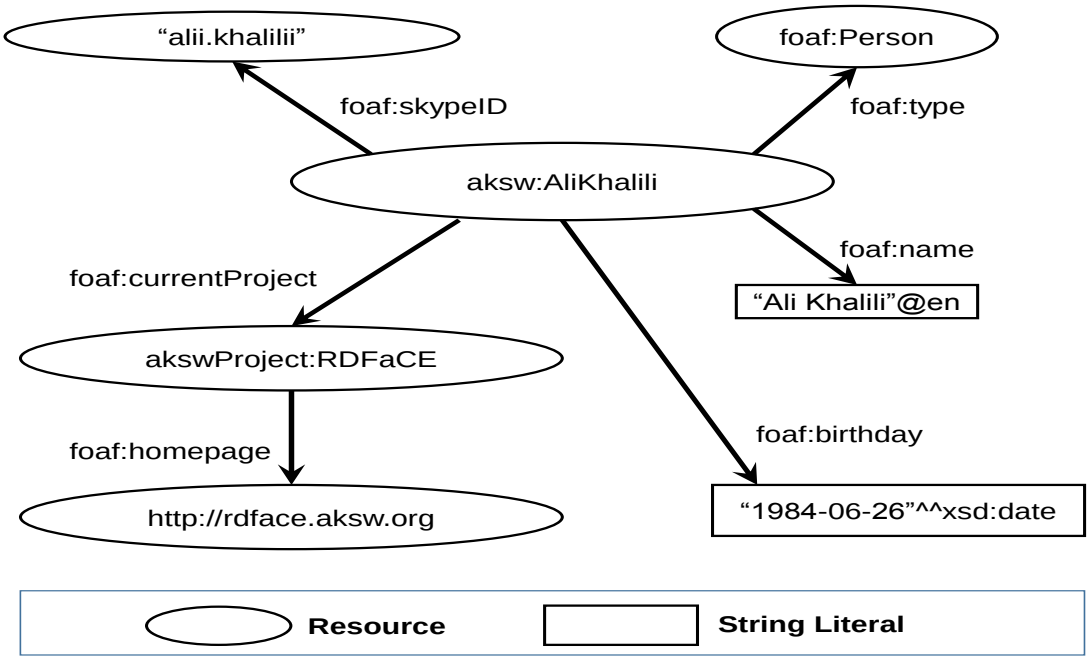


Figure 2.3.: Small knowledge base about Ali Khalili represented as a graph.

embed RDF data within HTML documents by offering additional HTML attributes. In the sequel, we explain the main serialization formats used in this thesis:

### RDF/XML

RDF/XML represents RDF triples in XML format [Beckett, 2004]. The RDF/XML format is convenient for machines since the traditional XML format is commonly adopted and there are a variety of libraries available that simplify interaction with this format. Figure 2.4 shows our RDF example in RDF/XML format.

```
<rdf:RDF xmlns:log="http://www.w3.org/2000/10/swap/log#"
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
    <rdf:Description rdf:about="http://aksw.org/Projects/RDFaCE">
        <homepage xmlns="http://xmlns.com/foaf/0.1/" rdf:resource="http://rdface.aksw.org"/>
    </rdf:Description>

    <Person xmlns="http://xmlns.com/foaf/0.1/" rdf:about="http://aksw.org/AliKhalili">
        <currentProject rdf:resource="http://aksw.org/Projects/RDFaCE"/>
        <birthday rdf:datatype="http://www.w3.org/2001/XMLSchema#date">1984-06-26</deathDate>
        <age rdf:datatype="http://www.w3.org/2001/XMLSchema#int">30</age>
        <skypeID xmlns="http://dbpedia.org/property/" xml:lang="en">alii.khalilii</skypeID>
        <name xmlns="http://dbpedia.org/property/" xml:lang="en">Ali Khalili</name>
    </Person>
</rdf:RDF>
```

Figure 2.4.: Sample RDF/XML format.

```
@prefix aksw: <http://aksw.org/> .
@prefix akswProject: <http://aksw.org/Projects/> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

aksw:AliKhalili a foaf:Person;
foaf:age "30"^^xsd:int;
foaf:currentProject akswProject:RDFaCE;
foaf:birthday "1984-06-26"^^xsd:date;
foaf:skypeID "alii.khalilii";
foaf:name "Ali Khalili"@en .

akswProject:RDFaCE foaf:homepage <http://rdface.aksw.org> .
```

Figure 2.5.: Sample N3 format.

### N3

N3 stands for Notation3 and is a shorthand notation for representing RDF graphs. N3 was designed to be easily read by humans and it is not an XML-compliant language [Berners-Lee and Connolly, 2011]. Figure 2.5 shows our RDF example in N3 format.

### JSON-LD

JSON-LD[2] is a JSON-based format to serialize RDF and Linked Data (LD). It is designed to easily integrate into deployed systems that already use JavaScript Object Notation (JSON), and provides a smooth upgrade path from JSON to JSON-LD. JSON-LD specifies a number of syntax tokens and keywords. The main keywords include:

- **@context** - Used to define the short-hand names that are used throughout a JSON-LD document. These shorthand names are called terms and help developers to express specific identifiers in a compact manner.

[2] http://www.w3.org/TR/json-ld/

```
{
    "@graph": [
        {
            "@id": "http://aksw.org/Projects/RDFaCE",
            "http://xmlns.com/foaf/0.1/homepage": {
                "@id": "http://rdface.aksw.org"
            }
        },
        {
            "@id": "http://aksw.org/AliKhalili",
            "@type": "http://xmlns.com/foaf/0.1/Person",
            "http://xmlns.com/foaf/0.1/age": {
                "@type": "http://www.w3.org/2001/XMLSchema#int",
                "@value": "30"
            },
            "http://xmlns.com/foaf/0.1/birthday": {
                "@type": "http://www.w3.org/2001/XMLSchema#date",
                "@value": "1984-06-26"
            },
            "http://xmlns.com/foaf/0.1/currentProject": {
                "@id": "http://aksw.org/Projects/RDFaCE"
            },
            "http://xmlns.com/foaf/0.1/name": {
                "@language": "en",
                "@value": "Ali Khalili"
            },
            "http://xmlns.com/foaf/0.1/skypeID": "alii.khalilii"
        }
    ]
}
```

Figure 2.6.: Sample JSON-LD format.

- **@id** - Used to uniquely identify things that are being described in the document with Internationalized Resource Identifiers (IRIs) or blank node identifiers.
- **@value** - Used to specify the data that is associated with a particular property in the graph.
- **@language** - Used to specify the language for a particular string value or the default language of a JSON-LD document.
- **@type** - Used to set the data type of a node or typed value.

Figure 2.6 shows our RDF example in JSON-LD format

### RDFa

Resource Description Framework in Attributes (RDFa)[3] is a W3C Recommendation that adds a set of attribute level extensions to XHTML for embedding RDF metadata within Web documents. RDFa provides the following set of attributes:

- **about** – a URI or Compact URI (CURIE) specifying the resource the

[3] http://www.w3.org/TR/rdfa-syntax/

```
<div vocab="http://schema.org/" typeof="Person">
    <b property="name">Ali Khalili</b>
    <img src="akhalili.jpg" property="image" />
    <span property="jobTitle">PhD Student</span>
    <div property="address" typeof="PostalAddress">
        <span property="streetAddress">
            Augustusplatz 10
        </span>
        <span property="addressLocality">Leipzig</span>
        <span property="addressCountry">Germany</span>
        <span property="postalCode">04109</span>
    </div>
    <span property="telephone">(0049)341-97-32329</span>
    Ali's home page:
    <a href="http://www.ali1k.com" property="url">
        ali1k.com
    </a>
</div>
```

Figure 2.7.: Sample RDFa format.

metadata is about.

- **rel** and **rev** – specifying a relationship and reverse-relationship with another resource, respectively.
- **src**, **href** and **resource** – specifying the partner resource.
- **property** – specifying a property for the content of an element or the partner resource.
- **content** – optional attribute that overrides the content of the element when using the property attribute.
- **datatype** – optional attribute that specifies the datatype of text specified for use with the property attribute.
- **typeof** – optional attribute that specifies the RDF type(s) of the subject or the partner resource (the resource that the metadata is about).

Figure 2.7 shows an example of RDFa annotation.

### Microdata

Microdata[4] is an HTML5 specification used to nest semantics within existing content on Web pages. It provide the following set of attributes:

- **itemscope** – Creates the Item and indicates that descendants of this element contain information about it.
- **itemtype** – A valid URL of a vocabulary that describes the item and its properties context.
- **itemid** – Indicates a unique identifier of the item.
- **itemprop** – Indicates that its containing tag holds the value of the specified item property. The properties name and value context are described by the

[4] http://www.w3.org/TR/microdata/

```
<div itemscope itemtype="http://schema.org/Person">
    <b itemprop="name">Ali Khalili</b>
    <img src="akhalili.jpg" itemprop="image" />
    <span itemprop="jobTitle">PhD Student</span>
    <div itemprop="address" itemscope itemtype="http://schema.org/PostalAddress">
        <span itemprop="streetAddress">
            Augustusplatz 10
        </span>
        <span itemprop="addressLocality">Leipzig</span>
        <span itemprop="addressCountry">Germany</span>
        <span itemprop="postalCode">04109</span>
    </div>
    <span itemprop="telephone">(0049)341-97-32329</span>
    Ali's home page:
    <a href="http://www.ali1k.com" itemprop="url">
        ali1k.com
    </a>
</div>
```

Figure 2.8.: Sample Microdata format.

items vocabulary. Properties values usually consist of string values, but can also use URLs using the `a` element and its `href` attribute, the `img` element and its `src` attribute, or other elements that link to or embed external resources.

- **itemref** – Properties that are not descendants of the element with the `itemscope` attribute can be associated with the item using this attribute. It provides a list of element ids (not `itemids`) with additional properties elsewhere in the document.

Figure 2.8 shows an example of Microdata annotation.

## 2.3. Ontology

W3C defines an ontology as "the terms used to describe and represent an area of knowledge." [Heflin, 2004].

This definition has several aspects that should be discussed. First, the definition states that an ontology is used to describe and represent an area of knowledge. In other words, an ontology is domain specific; it does not represents all knowledge areas, but one specific area of knowledge. A domain is simply a specific subject area or sphere of knowledge, such as literature, medicine, education, etc.

Second, the ontology contains terms and relationships among those terms. Terms are also called classes, or concepts; these words are interchangeable. The relationships between these classes can be expressed by using a hierarchy, i.e. superclasses represent higher-level concepts and subclasses represent finer concepts. The finer concepts have all the attributes and features that the higher concepts have.

Third, in addition to the aforementioned relationships among classes, there is

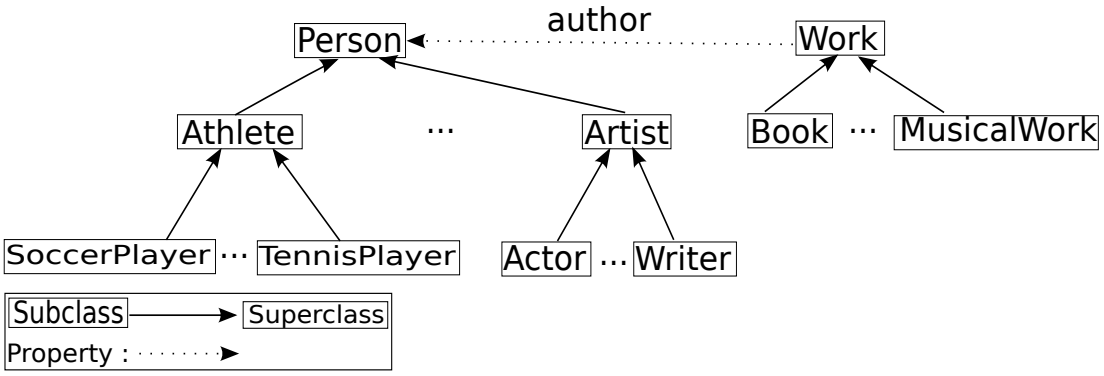


Figure 2.9.: Excerpt of the DBpedia ontology.

another level of relationship expressed by using a special group of terms called properties. These property terms describe various features and attributes of the concepts and they can also be used to associate different classes together. Thus, the relationships among classes are not only superclass or subclass relationships, but relationships expressed in terms of properties as well.

In other words, an ontology defines a set of classes (e.g. "Person", "Book", "Writer"), and their hierarchy, i.e. which class is a subclass of another one (e.g. "Writer" is a subclass of "Person"). The ontology also defines how these classes interact with each other, i.e. how different classes are connected to each other via properties (e.g. a "Book" has an author of type "Writer" ).

Figure 2.9 shows an excerpt of the ontology representing DBpedia[5]. This ontology shows that there is a class called "Writer" which is a subclass of the class "Artist", which in turn a subclass of "Person". *William Shakespeare, Johann Wolfgang von Goethe*, and *Dan Brown* are candidate instances of the class "Writer". The same applies to the class "Work" and its subclasses. Note that there is a property called "author" relating an instance of class "Work" to an instance of the class "Person" i.e. it relates a work to its author. For instance, the book titled "First Folio" is an instance of classes "Work" and "Book", and related via property "author" to its author "William Shakespeare", which is an instance of the classes "Person", "Artist" and "Writer".

So, why do we need ontologies? The main benefits of an ontology are:

- it provides a common and shared understanding/definition about certain key concepts in the domain,
- it provides a way for reuse of domain knowledge,
- it makes the domain assumptions explicit,

[5]http://dbpedia.org/

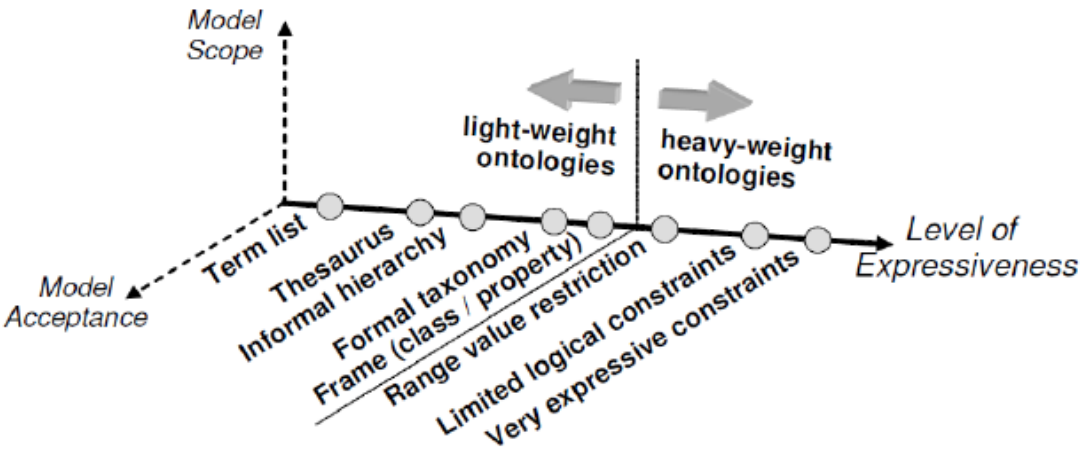


Figure 2.10.: Level of expressiveness of ontologies (source:[Schaffert, 2006]).

- it provides a way to encode knowledge and semantics such that machines can understand it.

### 2.3.1. Ontology Classification

There are various types of ontologies differing in multiple aspects. [Schaffert, 2006] has classified ontologies along three dimensions: model scope, level of expressiveness and model acceptance. The model scope refers to the area or coverage that is of interest. The acceptance dimension deals with the target communities of the application and its knowledge model and various methods of building consensus within a specific community. The level of expressiveness is particularly significant and is briefly described below.

**Level of expressiveness (Light-weight and Heavy-weight ontologies).** The spectrum of expressiveness of ontologies is illustrated in Figure 2.10. There are two main groups – lightweight ontologies and heavyweight ontologies. Based on the their level of expressiveness, eight sub categories are defined:

1. A *term list* or controlled vocabulary contains a list of keywords. Such lists are typically used to restrict possible values for properties of some kind of instance data in the domain.

2. A *thesaurus* also defines relations between terms, e.g. proximity of terms.

3. An *informal taxonomy* defines an explicit hierarchy of generalization and specialization, but there is no strict inheritance, i.e. an instance of a sub-class is not necessarily also an instance of the super-class.

4. A *formal taxonomy* defines a strict inheritance hierarchy.

5. A *frame* or class/property based ontology is similar to object-oriented models. A class is defined by its position in the subclass hierarchy and its properties. Properties are inherited by sub-classes and realized in instances.

6. A *range value restriction* defines, in addition, restrictions for the defined properties. The restrictions may be data type or domain restrictions.

7. By using *limited logic constraints*, property values may be further restricted.

8. A *very expressive* ontology often use first-order logic constraints. These constraints may include disjoint classes, disjoint coverings, inverse relationships, part-whole relationships, etc.

### 2.3.2. Schema.org

With heavy semantics, powerful reasoning can be done, however such systems cannot tolerate any inconsistency. On the other hand, with lightweight ontologies not much reasoning can be done. However, there is far less risk of inconsistencies because only little ontological agreements are in place. With little semantics, applications can scale very well. This is a significant aspect when we consider the huge scale of the web, which is important for the practical realization of the Semantic Web vision. Therefore, lightweight ontologies have become more popular and widespread. `Schema.org` as an example of such lightweight ontologies has gained attention in recent years.

`Schema.org` is an effort initiated by the popular search engines *Bing*, *Google* and *Yahoo!* on June 2011 to define a broad, Web-scale and shared vocabulary focusing on popular concepts. It stakes a position as a lightweight *middle* ontology that does not attempt to have the scope of an *ontology of everything* or go into depth in any one area. A central goal of having such a broad schema all in one place is to simplify things for mass adoption and cover the most common use cases [Ronallo, 2012]. `Schema.org` vocabulary can be used along with the Microdata, RDFa, or JSON-LD formats to add information to HTML Web pages (cf. Figure 2.7 and Figure 2.8).

The broadest item type in `Schema.org` is `Thing`, which has four properties: `name`, `description`, `url`, and `image`. More specific types share properties with broader types. For example, a `Place` is a more specific type of `Thing`, and a `LocalBusiness` is a more specific type of `Place`. More specific items inherit the properties of their parent. For example, a `LocalBusiness` is a more specific type of `Place` and a more specific type of `Organization`, so it inherits properties from both parent types. Figure 2.11 shows `LocalBusiness` schema and its properties on `Schema.org`.

**Thing > Place > LocalBusiness**

A particular physical business or branch of an organization. Examples of LocalBusiness include a restaurant, a particular branch of a restaurant chain, a branch of a bank, a medical practice, a club, a bowling alley, etc.

| Property | Expected Type | Description |
|---|---|---|
| Properties from LocalBusiness | | |
| **branchOf** | Organization | The larger organization that this local business is a branch of, if any. |
| **currenciesAccepted** | Text | The currency accepted (in ISO 4217 currency format). |
| **openingHours** | Duration | The opening hours for a business. Opening hours can be specified as a weekly time range, starting with days, then times per day. Multiple days can be listed with commas ',' separating each day. Day or time ranges are specified using a hyphen '-'.<br>- Days are specified using the following two-letter combinations: `Mo`, `Tu`, `We`, `Th`, `Fr`, `Sa`, `Su`.<br>- Times are specified using 24:00 time. For example, 3pm is specified as `15:00`.<br>- Here is an example: `<time itemprop="openingHours" datetime="Tu,Th 16:00-20:00">Tuesdays and Thursdays 4-8pm</time>`.<br>- If a business is open 7 days a week, then it can be specified as `<time itemprop="openingHours" datetime="Mo-Su">Monday through Sunday, all day</time>`. |
| **paymentAccepted** | Text | Cash, credit card, etc. |
| **priceRange** | Text | The price range of the business, for example $$$. |
| Properties from Place | | |
| **address** | PostalAddress | Physical address of the item. |
| **aggregateRating** | AggregateRating | The overall rating, based on a collection of reviews or ratings, of the item. |
| **containedIn** | Place | The basic containment relation between places. |
| **event** | Event | Upcoming or past event associated with this place or organization. Supercedes events. |
| **faxNumber** | Text | The fax number. |
| **geo** | GeoCoordinates or | The geo coordinates of the place. |

Figure 2.11.: An example schema (LocalBusiness) from Schema.org.

## 2.4. SPARQL Query Language

"The SPARQL Protocol and RDF Query Language (SPARQL) is a query language and protocol for RDF." [Clark et al., 2008]. SPARQL is a W3C standard and it is used to ask queries against RDF graphs. SPARQL allows the user to write queries that consist of triple patterns, conjunctions (logical "and"), disjunctions (logical "or") and/or a set of optional patterns [Wikipedia, 2013]. Examples of these optional patterns are: `FILTER`, `REGEX` and `LANG`.

The SPARQL query specifies the pattern(s) that the resulting data should satisfy. The results of SPARQL queries can be result sets or RDF graphs. SPARQL has four query forms, specifically `SELECT`, `CONSTRUCT`, `ASK` and `DESCRIBE` [Prud'hommeaux and Seaborne, 2008].

Let us take an example to clarify the usage of SPARQL. Assume that we want to ask the query "What is the homepage of Ali Khalili's current project?" to our small knowledge base. Figure 2.12 shows a SPARQL query to get information about the homepage of Ali Khalili's current project.

In Figure 2.12, lines 1 and 2 define prefixes in order to write URIs in their short forms. Line 3 declares the variables that should be rendered to the output of that query, which is only one variable `?homepage`. Note that SPARQL variables start either with a question mark "`?`", or with a dollar sign "`$`". Line 4 states that for the

```
1 PREFIX aksw: <http://aksw.org/>
2 PREFIX foaf: <http://xmlns.com/foaf/0.1/>
3 SELECT ?homepage
4 WHERE {aksw:AliKhalili foaf:currentProject ?project.
5 ?project foaf:homepage ?homepage. }
```

Figure 2.12.: SPARQL query to get the homepage of Ali Khalili's current project.

statement with subject `aksw:AliKhalili` and property `foaf:currentProject`, we want the value of its object to be assigned to a variable called `?project`. Upon execution, this variable will take the value of `akswProject:RDFaCE`. In line 5, we want variable `?project` which now has the value `akswProject:RDFaCE`, to be the subject of the next statement. In other words, the statement will be `akswProject:RDFaCE foaf:homepage ?homepage`. Now, variable `?homepage` is the only unknown variable of the statement, and it will take the value `http://rdface.aksw.org`. Eventually, its value will be rendered to the output.

## 2.5. Triplestore

The crucial question here is "How do we store RDF data for efficient and quick access?". Basically, RDF data is stored in triplestores. A triplestore is a software program capable of storing and indexing RDF data efficiently, in order to enable querying this data easily and effectively. A triplestore for RDF data is like Relational Database Management System (RDBMS) for relational databases.

Most triplestores support SPARQL query language for querying RDF data. As there are several RDBMSs in the wild, such as Oracle[6], MySQL[7] and SQL Server[8], similarly there are several triplestores. Virtuoso [Erling and Mikhailov, 2007], Sesame [Broekstra et al., 2002] and BigOWLIM [Bishop et al., 2011] are typical examples of triplestores for desktop and server computers. DBpedia, for example, uses Virtuoso as the underlying triplestore.

## 2.6. Natural Language Processing on the Semantic Web

Natural Language Processing (NLP) and information extraction services usually deal with tasks such as Named Entity Recognition (NER), Keyword Extraction (KE), Automatic Term Recognition (ATR), Wikification (WKF), Entity Linking (EL), language detection, Part-Of-Speech (POS) tagging, text classification, morphological analysis, relation extraction, sentiment analysis, etc.

[6] http://www.oracle.com/us/products/database/overview/index.html
[7] http://www.mysql.com
[8] http://www.microsoft.com/en-us/sqlserver/default.aspx

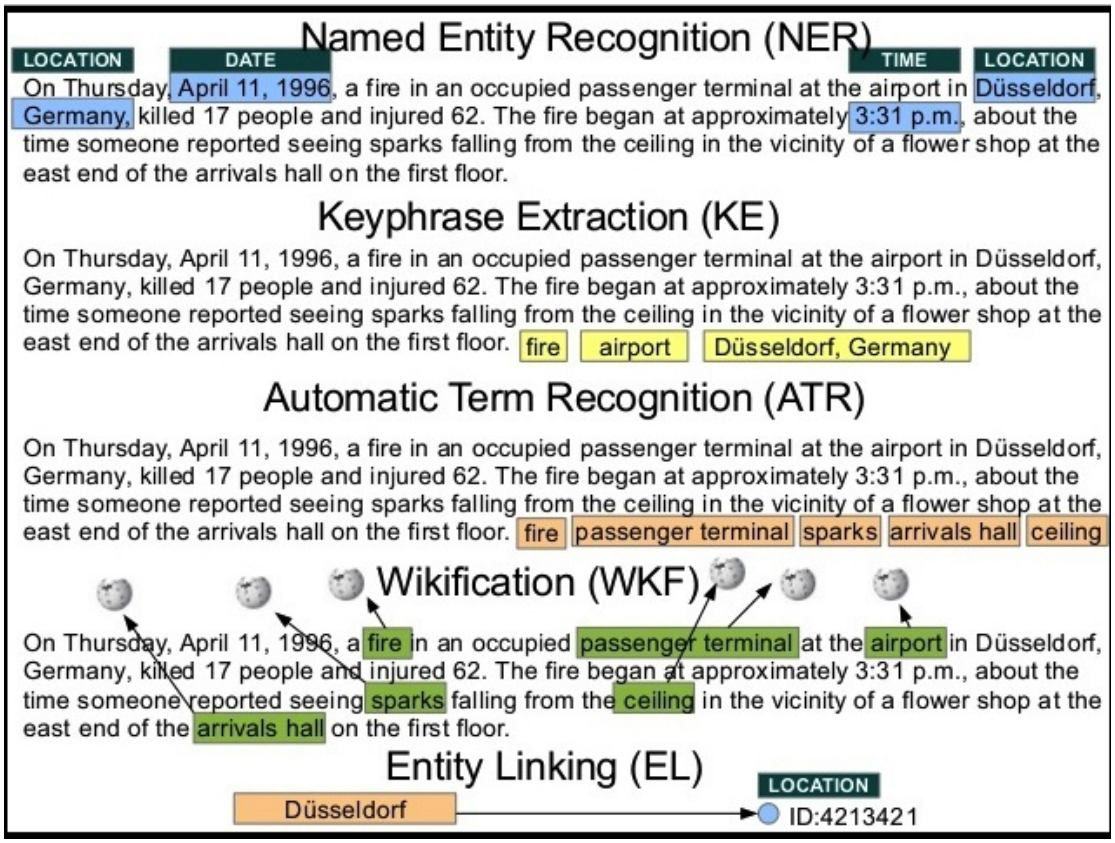


Figure 2.13.: Examples of information extraction subtasks (source:[Mendes, 2013]).

As shown in Figure 2.13, the main information extraction subtasks include[Mendes, 2013]:

- *NER.* Inserting tags into the text to mark each string that represents a person, organization, or location name, or a date or time stamp, or a currency or percentage figure.
- *KE.* Extracting the significant phrases from a document or collection of documents.
- *ATR.* Discovering phrases which are significant for a given domain (e.g. in technical documents, or documents with focus on a particular domain).
- *WKF.* Automatically marking up phrases with links to Wikipedia pages that describe those phrases.
- *EL.* Associating entity mentions recognized in text with their corresponding identifiers in a Knowledge Base (KB). The input of the task is an entity name and an example document where that name was used. The output is

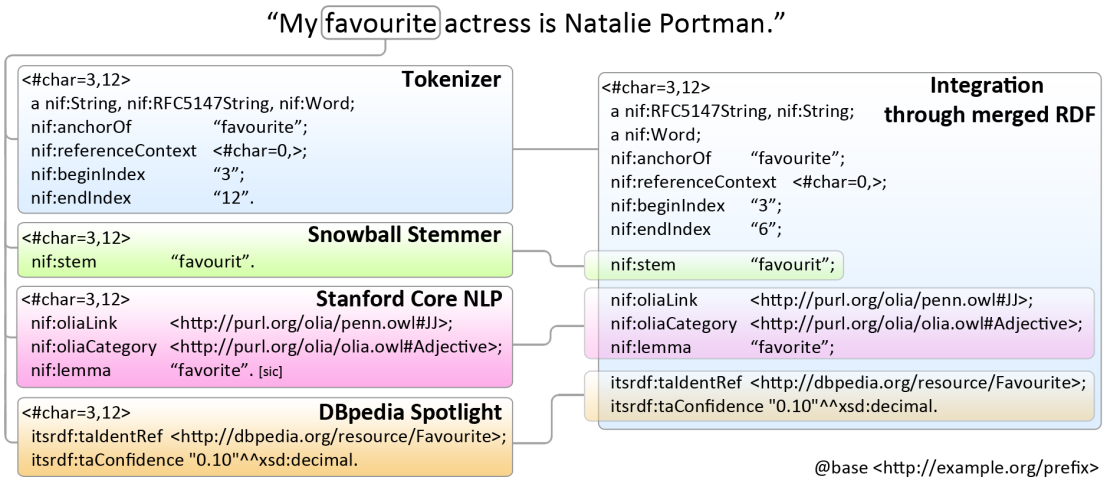


Figure 2.14.: An example of NIF integration (source:[Hellmann et al., 2013]).

an entity identifier for the referent of that mention in the KB.

Many NLP tasks can greatly benefit from making use of the Semantic Web technologies and the wealth of knowledge being available in structured form. *NLP Interchange Format (NIF)* [Hellmann et al., 2013][9] is an RDF/Web Ontology Language (OWL)-based format that aims to achieve interoperability between NLP tools, language resources and annotations. NIF addresses the interoperability problem on three layers: the *structural*, *conceptual* and *access* layer. NIF is based on a LD enabled URI scheme for identifying elements in (hyper-)texts (structural layer) and a comprehensive ontology for describing common NLP terms and concepts (conceptual layer). NIF-aware applications produce output (and possibly also consume input) adhering to the NIF Core ontology as REST services (access layer). NIF can be used for import and export of data from and to NLP tools and services. As shown in Figure 2.14, based on the normalization and tokenization, the combined RDF of several NLP tools merges naturally based on the subject URIs.

[9] http://persistence.uni-leipzig.org/nlp2rdf/

Chapter 3

# Concepts and State of the Art

> "A successful man is one who can lay a firm foundation with the bricks others have thrown at him." — *David Brinkley*

In this chapter, we provide an overview on the rapidly emerging field of Semantic Content Authoring (SCA). We conducted a systematic literature review comprising a thorough analysis of 31 primary studies out of 175 initially retrieved papers addressing the semantic authoring of textual content. We obtained a comprehensive set of quality attributes for SCA systems together with corresponding UI features suggested for their realization. The quality attributes include aspects such as usability, automation, generalizability, collaboration, customizability and evolvability. The primary studies were surveyed in the light of these quality attributes and we performed a thorough analysis of three SCA systems. The proposed quality attributes and UI features facilitate the evaluation of existing approaches and the development of novel more effective and intuitive semantic authoring interfaces.

The rest of this chapter is organized as follows: In Section 3.1 we describe the research method and the review protocol used for conducting the systematic review. In Section 3.2 we define the terminology of the investigated domain then we elaborate on the results of the review by surveying the extracted quality attributes in Section 3.4. In Section 3.8 we discuss three existing semantic authoring tools and describe them in the light of the quality attributes. In Section 3.9 we report on the gaps and open research issues revealed from the results of our systematic literature review. Finally in Section 3.10 we conclude the chapter.[1]

## 3.1. Research Method

We followed a formal systematic literature review process for this study based on the guidelines proposed in [Dyba et al., 2007, Kitchenham, 2004]. A systematic literature review is an evidence-based approach to thoroughly search studies relevant to some pre-defined research questions and critically select, appraise, and synthesize findings for answering the research questions at hand. Systematic reviews maximize the chance to retrieve complete data sets and minimize the chance of bias. As part of the review process, we developed a protocol (described in the sequel) that provides a plan for the review in terms of the method to be followed, including the research questions and the data to be extracted.

[1] The contents of this chapter have been published as [Khalili and Auer, 2013a].

### 3.1.1. Research Questions

The goal of our survey is analyzing existing user interfaces for semantic content authoring and thereby providing a set of quality attributes, which can serve as guidelines for designing suitable and effective user interfaces for semantic content authoring. To achieve this goal we aim to answer the research question RQ1 (cf. Section 1.3):

*What are existing approaches for user-friendly semantic content authoring?*

We divide this general research question into the following more concrete sub-questions:

- RQ1.1. *How to classify existing approaches for semantic content authoring?*
- RQ1.2. *What types of user interfaces are used by each approach?*
- RQ1.3. *What are the features supported by the proposed user interfaces?*
- RQ1.4. *What types of users are targeted in each approach?*
- RQ1.5. *How is the user interface evaluated?*

After doing some pilot searches and consulting experts in the field, we obtained a list of pilot studies that served as a basis for the systematic review.

### 3.1.2. Search Strategy

To cover all the relevant publications, we used the following electronic libraries:

- ACM Digital Library
- IEEE Xplore Digital Library
- ScienceDirect
- SpringerLink
- ISI Web of Sciences

Based on the research question and pilot studies, we found the following basic terms to be most appropriate for the systematic review:

1. *semantic* OR *linked data* OR *web of data* OR *data web*
2. *content* OR *web page* OR *document*
3. *authoring* OR *annotating* OR *annotation* OR *annotate* OR *enrich* OR *edit*

To construct the search string, all these search terms were combined using Boolean "AND" as follows:

1 AND 2 AND 3

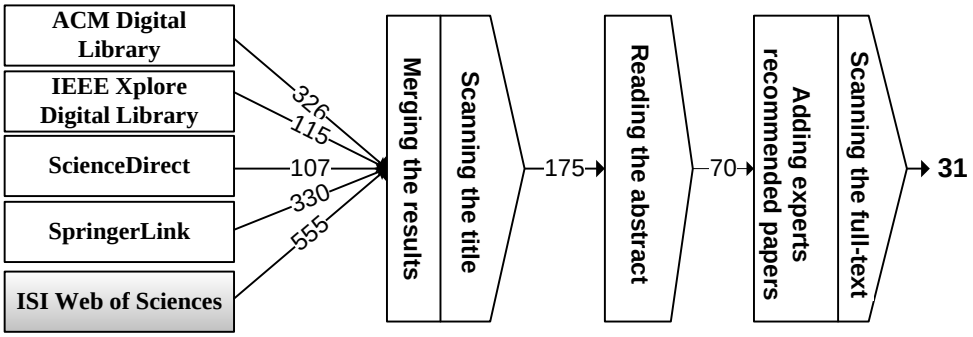


Figure 3.1.: Steps followed to scope the search results.

The next decision was to find the suitable field (i.e. title, abstract and full-text) to apply the search string on. In our experience, searching in the 'title' alone does not always provide us with all relevant publications. Thus, 'abstract' or 'full-text' of publications should potentially be included. On the other hand, since the search on the full-text of studies results in many irrelevant publications, we chose to apply the search query additionally on the 'abstract' of the studies. This means a study is selected as a candidate study if its title or abstract contains the keywords defined in the search string. In addition, we limited our search to the publications that are written in English and are published after 2002 (when the first ISWC conference was held).

### 3.1.3. Study Selection

Some of the studies might contain the keywords used in the search string but might still be irrelevant for our research questions. Therefore, a study selection has to be performed to include only studies that contain useful information for answering the research question.

Peer-reviewed articles that satisfy all the following inclusion criteria are selected as primary studies:

- I1. A study that focuses on semantic content authoring.
- I2. A study that either proposes a user interface or a set of user interface features for the purpose of semantic content authoring.

Studies that met any of the following criteria were excluded from the review:

- E1. A study that does not focus on semantic content authoring but only mentions the term e.g. as an example or use case.
- E2. A study that does not propose any user interface or user interface feature for semantic content authoring but only a generic, non-user interface supported method, approach or algorithm for semantic annotation.

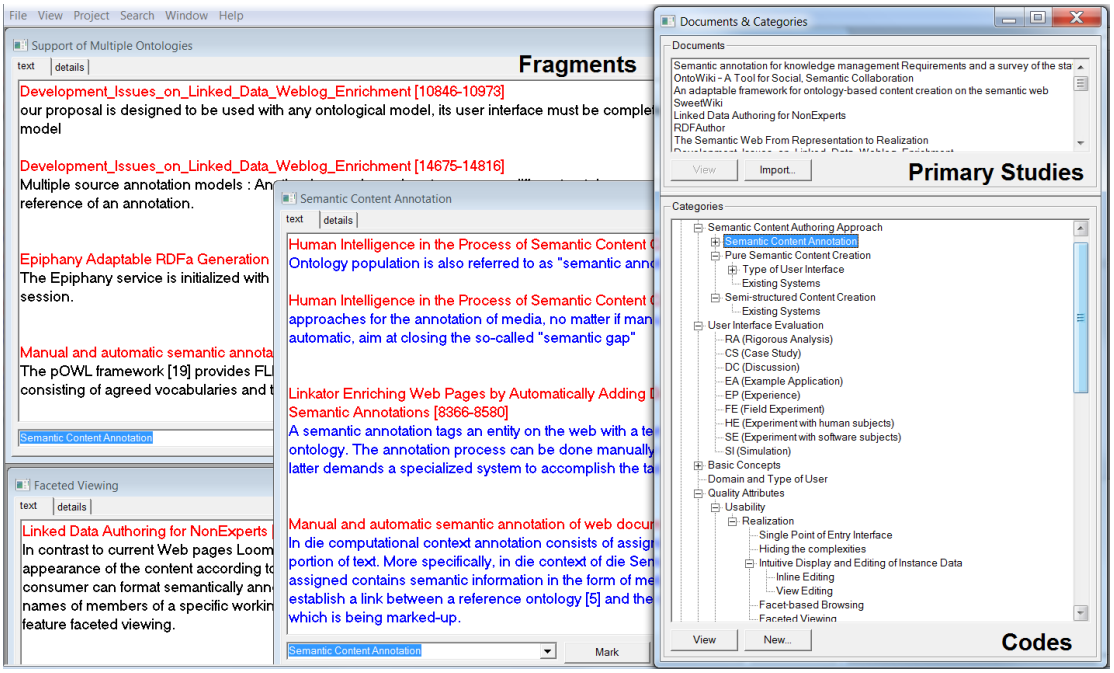


Figure 3.2.: The screenshot of the coding software showing the generated list of codes from the primary studies.

- E3. A study that is not about Web-based content authoring (e.g. studies about semantic authoring in word processors like LaTeX).
- E4. A study that is only about the ontology creation or ontology annotation (e.g. using natural language).
- E5. A study that does not discuss textual Web content authoring but other modalities such as image, audio or video annotation.

We conducted our review in early July 2011. As a consequence, our review included studies that were published and/or indexed before that date. As shown in Figure 3.1, we first applied the search query on each data source separately. Subsequently, to remove duplicate studies, we merged the results obtained from the different data sources. To remove irrelevant studies, we scanned the articles by title and thereby reduced the number of studies to 175. Then, we read the abstract of each publication carefully and further decreased the number of studies to 70. Finally, we added a list of additional papers recommended by experts and then scanned the full-text of the publications. We checked the full-text of studies to see if they fit with our predefined selection criteria. The result comprised 31 publications that represented our final set of *primary studies*.

### 3.1.4. Data Extraction and Analysis

The bibliographic metadata about each primary study were recorded using the bibliography management platform *JabRef*[2]. In addition, we extracted the following information from each paper:

- The used approach for semantic content authoring.
- The type of user interface.
- The features supported by the user interface.
- The domain and type of user.
- The evaluation method used in the paper.

To analyze the information appropriately, we required a suitable qualitative data analysis method applicable to our dataset. A common method that is used for this purpose is the *grounded theory method* because the theories (the SCA approaches and UI features) are "grounded" in the data [Glaser and Strauss, 1967].

The *Constant Comparison* method, one of the grounded theory techniques, has been often used in analyzing data and generating categories of data. Although the Constant Comparison method can be used on any set of data, it is particularly suitable for data that is context sensitive [Seaman, 1999] (i.e. data can be interpreted differently in different contexts). To interpret SCA approaches and UI features correctly, one often needs to understand in which context the approach and feature is proposed and how it is addressed. For instance, consider one study that mentions "evolvability" as a feature for UI. Without understanding the context of this feature, we cannot conclude whether this feature is about designing evolvable UIs or about supporting annotation/ontology evolvability in the UI (which is our aim here).

Miles and Huberman [Miles and Huberman, 1994] described *coding* as a procedure for the constant comparison method. Codes are tags or labels for assigning units of meaning to the descriptive or inferential information compiled during a study. Codes are efficient data-labeling and data-retrieval devices. One method of creating codes which is employed in our review is creating a provisional "start-list" of codes prior to fieldwork. We created this list from our research questions and the pilot studies. To carry out the analysis systematically, we used the following coding procedures proposed by Lincoln Guba[Miles and Huberman, 1994]:

- *Filling-in*: We read each study carefully and added the codes for related fragments and items. As new insights or new ways of looking at the data emerged, we reconstructed our coherent coding schema.
- *Extension*: If needed, we returned to materials coded earlier and interrogated them in a new way, with a new theme, construct, or relationship.

[2] http://jabref.sourceforge.net/

- *Bridging*: If new or previously not understood relationships within units of a given category were found, we recorded that relationship.
- *Surfacing*: We identified new categories which contained the previously created codes.

As shown in Figure 3.2, we used the *Weft QDA* software[3] to record the codes. The final list of codes are available online[4].

### 3.1.5. Overview of Included Studies

For quantitative analysis purposes, we performed some queries on the collected database of primary studies. The distribution of studies per year as shown in Figure 3.3 indicates an increasing intensity of research in the area of semantic content authoring. The remarkable rise after 2008 can be explained with the emergence and adoption of weak semantic techniques (the so-called 'lowercase' Semantic Web), such as the use of Microformats[5], RDFa[6] and Microdata[7]. These techniques facilitate semantic content authoring by embedding semantic annotations into the HTML Web pages.

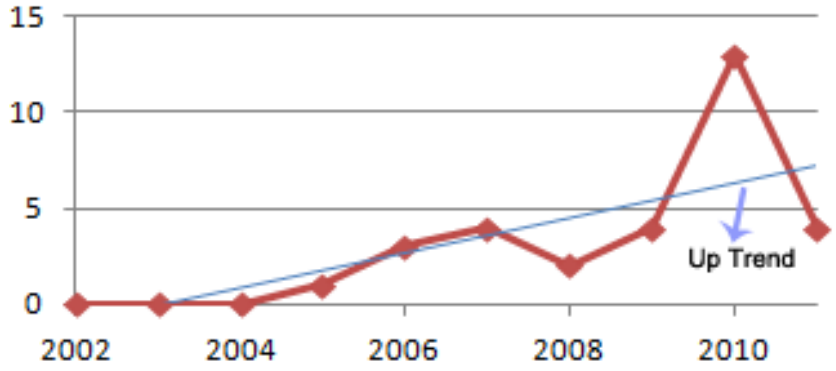


Figure 3.3.: Publications per year.

The primary studies included 14 conference papers, 11 journal articles, 4 workshop papers, one thesis and one technical report. Among them, the following four studies are survey papers. Uren et al. [Uren et al., 2006] have reported a comprehensive review of the studies and applications for semantic annotations which were published before 2006. In [Heitmann et al., 2009], Heitmann et al. conducted an empirical survey of Semantic Web applications and have reviewed the challenges of them. Paulheim and Probst [Paulheim and Probst, 2010] surveyed the

[3] http://www.pressure.to/qda/
[4] http://rdface.aksw.org/SLR/codes.qdp
[5] http://microformats.org/
[6] http://www.w3.org/TR/rdfa-syntax/
[7] http://www.w3.org/TR/microdata/

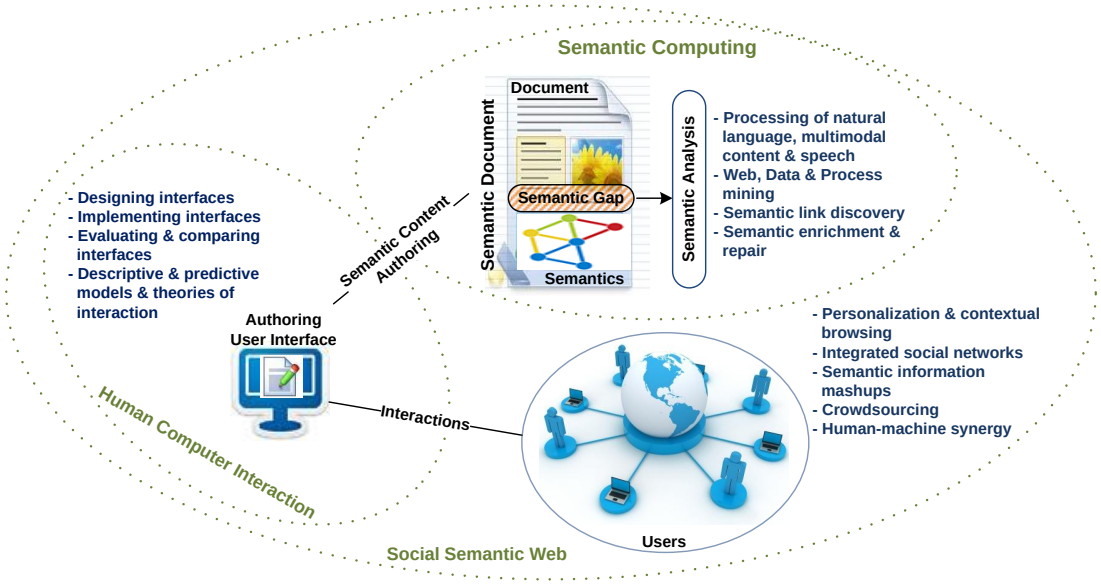


Figure 3.4.: Semantic content authoring ecosystem.

ontology-enhanced user interfaces and have introduced a schema for characterizing the requirements of ontology-enhanced user interfaces. In [Hachey, 2011], Hachey and Gaševic provide an overview of the current progress and gaps in the area of Semantic Web user interfaces in general. Compared to these surveys, the focus and the coverage range of our survey are different. The goal of our survey is to perform a *systematic analysis* of the existing collection of research material addressing the *user interface* aspects of *Web-based* semantic authoring systems. We focus on *semantic authoring of textual content* and cover the literature published between 2002 and 2011.

## 3.2. Terminology

The terminology basis of this chapter is depicted in Figure 3.4. In the sequel we describe the individual concepts in more detail:

**Semantic Gap** is a term coined to describe the discrepancy between low-level technical features of multimedia, which can be automatically processed to a great extent, and the high-level, meaning-bearing features a user is typically interested in [Siorpaes and Simperl, 2010]. As discussed in [Chu et al., 2009], semantic gaps in the process of constructing and managing digital content can be divided into three types namely *human-to-machine*, *machine-to-machine*, and *machine-to-human*. In this thesis we mainly focus on the machine-to-machine semantic gaps that are important when searching or reusing content by machines. In this context, semantics consists of concepts and their logical relationships in an explicit form.

When a machine processes the semantics, the lack of a common vocabulary may lead to alterations in the original semantics thus resulting in semantic gaps.

**Semantic Computing** is a research field that addresses the extraction and processing of the semantics of digital content and naturally expressed user intentions to help retrieve, manage, manipulate, or even create the content. Semantic computing aims to bridge the semantic gap by employing appropriate semantic analysis techniques such as *natural language processing*, *processing of multimodal content*, *speech recognition*, *Web, data and process mining*, *semantic link discovery* as well as *semantic enrichment and repair*. Semantic Web knowledge representation techniques (e.g. *OWL*[8], *RDF*, *RDFa*, *SPARQL*, *SKOS*) help to bridge the semantic gap through a common ground of shared vocabularies and ontologies [Sheu et al., 2010, Hasida, 2007].

**Semantic Document** is an intelligent document (with explicit semantic structure) which "knows about" its own content so that it can be automatically processed in unforeseen ways. These benefits, however, come at the cost of increased authoring effort [Hasida, 2007, Uren et al., 2006].

**Semantic Content Authoring (SCA)** is a tool-supported manual composition process aiming at the creation of semantic documents which are:

- fully semantic in the sense that their original data model uses a semantic knowledge representation formalism (such as RDF, RDF-Schema[9] or OWL) or
- based on a non-semantic representation form (e.g. text or hypertext), which is enriched with semantic representations during the authoring process.

. With an ontology and a user interface appropriate for the type of content, semantic authoring can be easier than traditional composition of content and the resulting content can be of higher quality [Hasida, 2007].

**Semantic Content Authoring User Interface (SCAUI)** is a human accessible interface with capabilities for modifying and writing semantic documents.

**Human Computer Interaction (HCI)** is a research field that aims to improve the interactions between users and computers by making computers more usable and receptive to the user's needs.

[8] http://www.w3.org/OWL/
[9] http://www.w3.org/TR/rdf-schema/

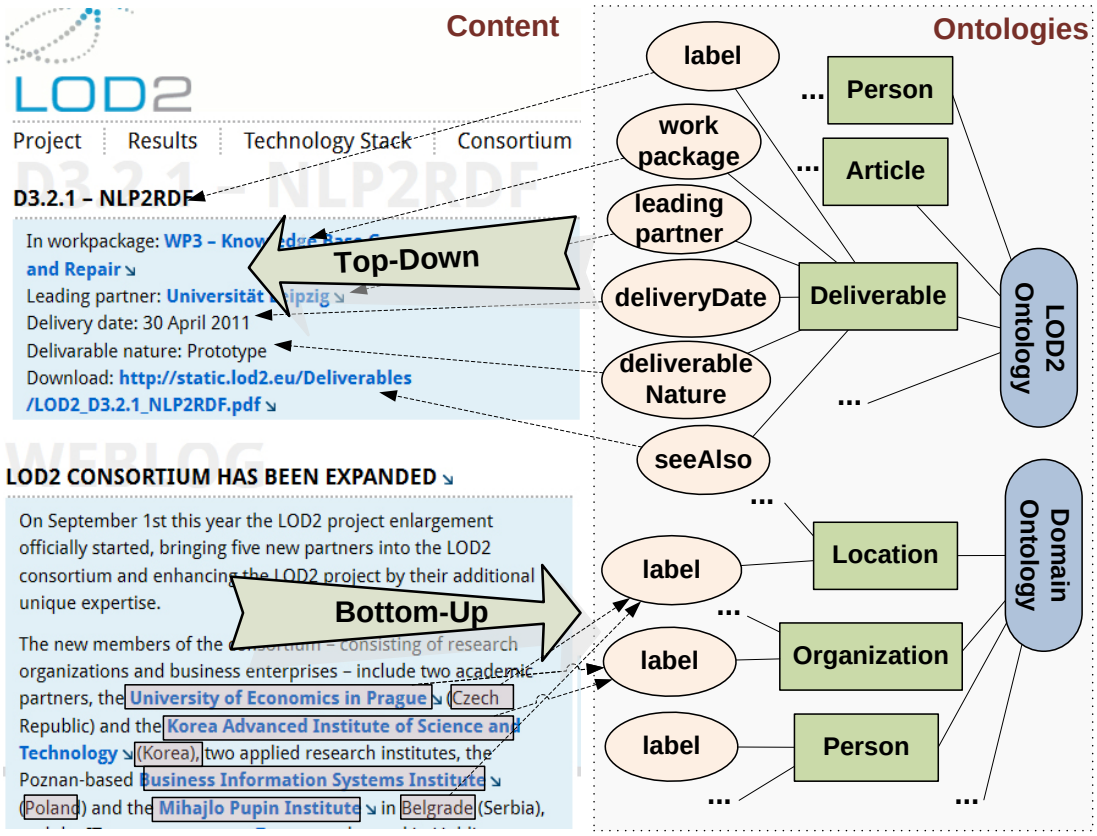


Figure 3.5.: Top-Down and Bottom-Up approaches for semantic content authoring.

**Social Semantic Web** is a very general research field triggered by the advent of Web 2.0. It aims at bringing a social novelty, rather than a technical one by providing user-friendly tools to facilitate broad user participation in the process of creating semantic content [Siorpaes and Simperl, 2010]. The Social Semantic Web vision comprises many of the aforementioned domains and techniques.

## 3.3. Semantic Authoring Approaches

There are already different approaches proposed for semi-structured but non-semantic content authoring (e.g. [Benson et al., 2010]). These approaches aim at immediate user gratification in the form of useful visualizations and interesting data aggregation but do not focus on using shared vocabularies and formal ontologies, which ultimately facilitate portability and reuse. With regard to explicit semantic content authoring recent approaches can be roughly classified into the categories *Top-Down* and *Bottom-Up*. As demonstrated in Figure 3.5, the classification is based on the starting point of the authoring process which can be ontologies (with upper level of expressiveness) or unstructured content (with lower level of

expressiveness). A third category of approaches (called *Middle-Out* or *Hybrid*) that balance between the Top-Down and the Bottom-Up approach can also be considered, but is beyond the scope of this research.

### 3.3.1. Bottom-Up Approaches

These approaches which are usually called semantic annotation techniques (a.k.a. semantic markup [Araujo et al., 2010]) aim to annotate existing documents using a set of predefined ontologies. The basic ingredients of a semantic annotation system are ontologies, the documents and the annotations that link ontologies to documents [Uren et al., 2006]. Here, we need two kinds of ontologies [Valkeapaeae et al., 2007]: *Annotation ontologies* (i.e. metadata schemata) which define what kind of properties and value types should be used for describing a resource. For example, the Dublin Core schema uses elements such as `dc:title`, `dc:creator`, `dc:subject`, etc. *Domain ontologies* which are used to define vocabularies providing possible values for metadata properties. Examples are *eClassOWL*[10] defining products and services, *MeSH*[11] defining medical subjects or the *DBpedia*[12] knowledge base, which is a cross-domain ontology extracted from *Wikipedia*.

The result of the annotation process is a document that is marked-up semantically. For that concern, some markup strategies also known as *Semantic Annotation Techniques* or *Lower Semantic Techniques* are proposed. Semantic markups provide additional information (metadata) about an existing piece of data. It helps to bridge the ambiguity of the natural language by telling computers what data in text means and how data items are related. There are already different mechanisms to annotate Web documents:

**Microformats**[13] is an approach to integrate semantic markup into XHTML and HTML documents. Microformats re-purpose existing markup definitions (particularly the HTML `class` attribute) in a non-standard way to convey (meta-) data. This approach is limited to a set of few published Microformat templates and thus not easily extensible for domain-specific applications. Moreover, it is not possible to validate Microformat annotations since no proper grammar is used for their definition.

**embedded RDF (eRDF)**[14] is similar to Microformats but annotates HTML using RDF. However, it faces the same criticism as Microformats, since it uses the same non-standard compatible annotating strategy[Araujo et al., 2010]. eRDF was invented in 2005, and partly inspired by Microformats. The eRDF specification is now obsolete, superseded by RDFa and Microdata.

[10] http://www.heppnetz.de/projects/eclassowl/
[11] http://www.ncbi.nlm.nih.gov/mesh
[12] http://dbpedia.org/
[13] http://microformats.org
[14] https://github.com/iand/erdf

**Resource Description Framework in Attributes (RDFa)** as explained in Section 2.2.4 is a W3C Recommendation for semantic markup. RDFa fulfills the principles of interoperable metadata such as publisher independence, data reuse, self containment, schema modularity and evolvability to a good extent.

**Microdata** as explained in Section 2.2.4 is an HTML5 specification used to nest semantics within existing content on Web pages. Microdata is already in use by popular search engines for interpreting the information contained in a Web page by exploiting *Schema. org*.
There are normally two types of metadata applied to a document in the process of semantic annotation:

- *Content metadata* describe specific things the author of the document wishes to write about (e.g. people, cities, etc.). These content-related metadata cover a broad domain of information [Möller et al., 2006]. NLP annotation APIs (e.g. *DBpedia Spotlight*[15]) are one approach to automatically add content metadata into a document.

- *Context metadata* refers to the general topic, structure or temporal aspects of a document (e.g. title, theme or creation date of a document). These context-related metadata cover a very specific domain of information. Semantic Tagging (e.g. *Faviki*[16]) and structured templates [Quint and Vatton, 2007] are two approaches to automatically embed context-related metadata in a document.

### 3.3.2. Top-Down Approaches

These approaches which are also called *Ontology Population* [Siorpaes and Simperl, 2010] techniques aim to create semantic content based on a set of initial ontologies which are extended during the population process. When compared to the bottom-up approaches, these approaches deal with semantic representations from the beginning instead of lifting unstructured content to a semantic level. These approaches combine ontological rigour with flexible user interface constructs to create semantic content. Semantic templates as discussed in [Auer et al., 2006, Di Iorio et al., 2010, Thórisson et al., 2010] are one technique to realize this goal. In this approach each class of the ontology has an associated template. A page using that template represents each instance of a class. Data properties are displayed as simple text while object properties are displayed as links to other pages (representing other instances of the ontology). Users can also edit the underlying ontology which will result in changes of the corresponding templates.

[15] http://spotlight.dbpedia.org
[16] http://www.faviki.com

## 3.4. Quality Attributes

In order to evaluate the strengths and weaknesses of different SCA systems, we assess the systems according to predefined criteria which we call *Quality Attributes*. Quality attributes are non-functional requirements used to evaluate the performance of a system. They are widely used in architecture development and assessment as high-level characteristics which systems enclose. In the context of this work, quality attributes represent the areas of concern regarding the development of an SCA system from the viewpoint of its consumers.

Based on the qualitative analysis of our primary studies, we obtained 11 quality attributes. For each quality attribute we extracted one or more UI feature(s). Features describe a specific type or property of UI that can be used to realize an intended quality attribute. The features are directly (e.g. faceted browsing) or indirectly (e.g. UIs for mobile devices) addressing the required UI functionality for an SCA system. Table 3.1 surveys the quality attributes and various UI approaches for their implementation. In the sequel we describe each of the 11 quality attributes in more detail.

### Usability

Usability is a measure of the quality of a user's experience in interacting with a system. In ISO 9241 usability is defined as the *effectiveness*, *efficiency* and *satisfaction* with which specified users achieve specified goals in particular environments. Lauesen [Lauesen, 2005] and Nielsen [Nielsen, 2012] add more factors such as *learnability* and *utility* to usability definition. In the context of this work, we consider the following factors for defining the usability:

(A) *Efficiency*. How efficient is the system for the frequent user to expend appropriate amounts of resources in relation to the effectiveness achieved in a specified context of use?

(B) *Effectiveness*. How effective is the system to achieve specified tasks with accuracy and completeness?

(C) *Satisfaction*. How satisfied is the user with the system?

(D) *Learnability*. How easy is the system to learn for various groups of users?

(E) *Utility* (or *Usefulness*). Assesses whether the system enables users to solve real problems in an acceptable way.

| Quality Attribute | Realization |
|---|---|
| Usability | Single Point of Entry Interface [Karger and Quan, 2005, Auer et al., 2006, Uren et al., 2006], Faceted Browsing [Auer et al., 2006, Frosterus et al., 2011], Faceted Viewing [Luczak-Roesch, 2009, Heese et al., 2010, O'Donoghue et al., 2010], Inline Editing and View Editing [Auer et al., 2006, Tramp et al., 2010] |
| Customizability | Living UIs [Haase et al., 2010], Providing Different Semantic Views [Auer et al., 2006, Di Iorio et al., 2010, Muller et al., 2011, Berners-Lee et al., 2007] |
| Generalizability | Supporting Multiple Ontologies [Uren et al., 2006, Buffa et al., 2008, Heese et al., 2010, Navarro-Galindo and Samos, 2010, Adrian et al., 2010, Ruiz-Rube et al., 2010], Supporting Ontology Modification [Uren et al., 2006, Valkeapaeae et al., 2007, Di Iorio et al., 2010], Supporting Heterogeneous Document/Content Formats [Uren et al., 2006, Heese et al., 2010] |
| Collaboration | Access Control [Uren et al., 2006, Ruiz-Rube et al., 2010, Tramp et al., 2010], Support of Standard Formats [Uren et al., 2006, Valkeapaeae et al., 2007, Buffa et al., 2008, Heese et al., 2010, Luczak-Roesch, 2009, Ruiz-Rube et al., 2010, Tramp et al., 2010, Klebeck et al., 2011], UIs for Social Collaboration [Auer et al., 2006, Berners-Lee et al., 2007] |
| Portability | Cross-browser Compatibility [Valkeapaeae et al., 2007, Navarro-Galindo and Samos, 2010], UIs for Mobile Devices [Ermilov et al., 2011] |
| Accessibility | Accessible UIs [Hachey, 2011] |
| Proactivity | Resource Suggestion [Luczak-Roesch, 2009, Buffa et al., 2008], Real-time Semantic Tagging [Heese et al., 2010, O'Donoghue et al., 2010], Concept Reuse [Heese et al., 2010, O'Donoghue et al., 2010, Auer et al., 2006, Buffa et al., 2008], Real-time Validation [Valkeapaeae et al., 2007, Di Iorio et al., 2010] |
| Automation | Automatic Annotation [Siorpaes and Simperl, 2010, Araujo et al., 2010, Klebeck et al., 2011, Kiyavitskaya et al., 2009, Valkeapaeae et al., 2007, Buffa et al., 2008, Luczak-Roesch, 2009] |
| Evolvability | Resource Consistency [Heese et al., 2010], Document and Annotation Consistency [Uren et al., 2006], Versioning and Change Tracking [Auer et al., 2006] |
| Interoperability | Support of Standard Formats [Uren et al., 2006, Valkeapaeae et al., 2007, Buffa et al., 2008, Heese et al., 2010, Luczak-Roesch, 2009, Ruiz-Rube et al., 2010, Tramp et al., 2010, Klebeck et al., 2011], Semantic Syndication [Auer et al., 2006] |
| Scalability | Support of Caching [Auer et al., 2006, Herzig and Ell, 2010], Suitable Storage Strategies [Auer et al., 2006, Uren et al., 2006, Navarro-Galindo and Samos, 2010, Luczak-Roesch, 2009] |

Table 3.1.: List of quality attributes together with their corresponding UI features suggested for SCA systems.

Simplicity is the main prerequisite of usability. An SCA system should, as a rule, hide technical concepts related to markup languages and ontologies from the non-expert end-users [Valkeapaeae et al., 2007, Tramp et al., 2010]. It is crucial to provide end-users with easy to use interfaces that simplify the annotation process and place it in the context of their everyday work. More attention needs to be paid to decrease or blur the gap between the normal authoring process and the semantic authoring process. SCA systems should focus on the user's main task [Heese et al., 2010]. Usually, a user wants to perform the task of writing some text and not to annotate content. Integrating semantic authoring process into the commonly used packages is one approach to encourage users to view semantic authoring as part of the authoring process not as an afterthought process [Uren et al., 2006].

The following features of UIs are proposed for improving the usability of SCA systems:

- *Single Point of Entry Interface.*
  It means the environment in which users annotate documents should be integrated with the one in which they create, read, share and edit them. So, there is no added user effort involved in creating a semantic content versus a conventional approach, because the real work is done by the software through capturing semantics that is already being provided by the user [Karger and Quan, 2005, Auer et al., 2006, Uren et al., 2006]. This will minimize user actions as well as memory load thereby increasing the efficiency, user satisfaction, learnability and utility of the system.

- *Faceted Browsing.*
  Faceted browsing is a technique for accessing a collection of information represented using a faceted classification, allowing users to explore by filtering the available information. In the UI which implements this technique, all property values (i.e. facets) of a set of selected instances are analyzed. If for a certain property the instances have only a limited set of values, those values are offered to further restrict the instance selection. Hence, this way of navigation through data will never lead to empty results [Auer et al., 2006, Frosterus et al., 2011]. This feature is useful when searching for available resources or vocabularies. Faceted browsing increases the efficiency and effectiveness of the system by improving the navigability.

- *Faceted Viewing.*
  Faceted viewing [Luczak-Roesch, 2009, Heese et al., 2010] also known as augmented browsing [O'Donoghue et al., 2010] is very similar to faceted browsing but is used to distinguish the semantically annotated content from the normal content based on the different facets selected by user. For example, highlighting the names of members of a specific working group with a yellow background in the text. Similar to the faceted browsing, faceted viewing will increase the efficiency and effectiveness of the system by improving the navigability.

| Criteria<br>- UI feature | Usability Factors<br>Efficiency | Effectiveness | Satisfaction | Learnability | Utility |
|---|---|---|---|---|---|
| **Minimal Action**<br>- Single point of entry UI<br>- Inline editing & view editing | + | | + | | |
| **Minimal Memory Load**<br>- Single point of entry UI<br>- Integration into the commonly used packages | + | | + | + | + |
| **Navigability**<br>- Faceted browsing<br>- Faceted viewing | + | + | | | |
| **Simplicity & Familiarity**<br>- Simple UIs focusing on user's main task<br>- Integration into the commonly used packages | | | | + | |

Table 3.2.: Relation between usability factors and criteria ('+' indicates the positive effect of a criteria on usability factors).

- *Inline Editing and View Editing.*
  An SCA system should provide different editing modes for editing single and batch items. Inline editing allows editing items by clicking on them. View editing supports the editing of a combination of items in a specific view in one single step [Auer et al., 2006, Tramp et al., 2010]. This feature helps users to edit items in a minimum number of steps (minimal action) thereby increasing the efficiency and user satisfaction.

Table 3.2 shows how the aforementioned UI type and properties affect our previously defined usability factors. It is based on the *QUIM* model defined by Seffah et al. [Seffah et al., 2006]. Quality in Use Integrated Measurement (QUIM) model brings together usability factors, criteria, metrics, and data mentioned in various standards or models for software quality and defines them and their relations with one another in a consistent way.

### Customizability

Customizability is the ability of a system to be configured according to users' needs and preferences. Instead of being a static form strictly dependent on a given schema, an SCA system should provide mechanism to tailor its functionalities based on the user needs [Di Iorio et al., 2010]. In [O'Donoghue et al., 2010] the concept of "semantics in the eyes of the end-user" is introduced which means an

SCA system should provide different views for different personas using the system.

The following features of UIs are proposed for improving the customizability of SCA systems:

- *Living UIs.*
  A Living UI is a user interface that configures itself to automatically display the information most relevant to the user, dynamically adjusts to changing data, and still allows single users to customize according to their preferences [Haase et al., 2010]. End-user development techniques like *Programming by Example (PbE)* allow inferring user intents in real interactions and according to that providing customized outputs [Perdrix et al., 2009].

- *Providing Different Semantic Views.*
  Semantic views allow the generation of different views on the same metadata schema and aggregations of the knowledge base based on the roles, personal preferences, and local policies of the intended users [Auer et al., 2006, Di Iorio et al., 2010, Muller et al., 2011]. Such views can be either generic or domain specific. Generic views provide visual representations of instance data according to certain property values (e.g. map view or calendar view). Domain specific views address the requirements of a particular domain user (e.g. chemists need specific views for visualizing the atomic structure of chemical compounds).

### Generalizability

Generalizability is the ability of a system to adapt to different situations or use cases. An SCA system should support a wide range of metadata schemata in a flexible way. In fact, the more flexible and adaptable a system is, the more valuable it is for different contexts and users. A generic SCA system reduces the costs of supporting new schemata considerably, by following the evolution of existing standards and integrating heterogeneous resources [Di Iorio et al., 2010]. *Adaptivity* is an important capability of a generic system. An SCA system should be adaptable to different annotation and authoring uses with different kinds of contents to be processed [Valkeapaeae et al., 2007, Adrian et al., 2010]. In most of the cases generalizability is in opposition to *Usability* of a system. For instance, adding more and more syntactic possibilities counteracts ease of use for SCA systems [Auer et al., 2006]. The following features of UIs are proposed for improving the generalizability of SCA systems:

- *Supporting Multiple Ontologies.*
  A domain is usually described by several ontologies. For example, in a medical context there may be one ontology for general metadata about a patient and other technical ontologies that deal with diagnosis and treatment. SCA systems need to be able to support multiple ontologies [Uren et al., 2006, Buffa et al., 2008, Heese et al., 2010, Navarro-Galindo and Samos, 2010, Adrian

et al., 2010, Ruiz-Rube et al., 2010, Frosterus et al., 2011, Luczak-Roesch, 2009, Auer et al., 2006]. In a generic SCA system, the user interface must be completely decoupled from the ontological models. Models should be able to be added at runtime and become immediately accessible to the users [Ruiz-Rube et al., 2010, Buffa et al., 2008].

- *Supporting Ontology Modification.*
  A generic SCA system should provide users with user-friendly interfaces to modify the structure (classes and properties) of ontologies [Uren et al., 2006, Valkeapaeae et al., 2007, Di Iorio et al., 2010, Auer et al., 2006]. In this case, the system also needs to deal with consistency issues which might arise between ontologies and annotations with respect to ontology changes (a.k.a. *Ontology Maintenance* [Uren et al., 2006]).

- *Supporting Heterogeneous Document and Content Formats.*
  Supporting heterogeneous document and content formats is a prerequisite for integrating semantic authoring and annotation into the existing work practices [Uren et al., 2006, Heese et al., 2010]. A generic SCA system should be able to import documents in different formats such as word processor files, spreadsheets, graphics files and complex mixtures of them. It also needs to provide appropriate semantic annotations for different content types. For example, during the content annotation, a data table should be treated differently then raw text, because a table implicitly expresses relationships between the entries of a row (or column).

### Collaboration

Collaboration refers to the ability of a system to support cooperation between different users of the system. An SCA system should support collaborative semantic authoring, where the authoring process can be shared among different authors at different locations. This is a key requirement of knowledge sharing between users from different fields who are contributing to and reusing intelligent documents [Uren et al., 2006, Valkeapaeae et al., 2007, d'Aquin et al., 2008]. Web 2.0 applications and related technologies provide incentives to their users for collaboration and lead to rapidly growing amounts of content. Triggered by the success of the Web 2.0 phenomenon the *Social Semantic Web* idea has gained momentum yielding tools that allow collaboration and participation incorporating semantics by lay users. As a result, many collaborative and community-driven approaches to semantic content creation have been proposed. Examples are *Semantic Wikis* and *Semantic Tagging Systems* (e.g. Faviki[17]) which exploit Web 2.0 principles and technologies to facilitate broad user participation and collaboration in the process of creating semantically enriched or annotated content [Siorpaes and Simperl, 2010, Herzig and Ell, 2010]. [Di Iorio et al., 2010] divides semantic wikis into two main

[17] http://www.faviki.com/

categories according to their connections with the ontologies: *wikis for ontologies* and *ontologies for wikis*. The classification is very similar to our proposed top-down (cf. Section 3.3.2) and bottom-up approaches (cf. Section 3.3.1). *Access control* and *supporting standard formats* are two additional independent prerequisites of collaboration in an SCA system [Uren et al., 2006, Ruiz-Rube et al., 2010, Tramp et al., 2010]. The SCA system should allow to distinguish between writeable and non-writeable content based on the users permission level. It also needs to support standard formats that promote the collaboration and make it possible to share and re-use the generated content. To realize collaboration, an SCA system should provide appropriate UI elements for *meta-level interactions* around different types of semantically created content such as rating, tagging and discussing. Supporting social networking features such as following other authors, subscribing to changes for watching the evolution of content [Berners-Lee et al., 2007] as well as reusing and repurposing of content are also important to increase the collaboration in an SCA system.

#### Portability

Portability is the ability of a system to run under different environments. The user of an SCA system should be able to use the system at any location without installing any special software [Valkeapaeae et al., 2007, Navarro-Galindo and Samos, 2010]. When focusing on Web-based UIs, compatibility between different existing web browsers and access technologies becomes an important issue. As a requirement for UI, cross-browser compatibility should ideally be ensured in an SCA system. Designing suitable UIs for mobile and ubiquitous devices is another aspect which needs to be taken into the account as powerful mobile computing devices are becoming common among the users [Ermilov et al., 2011].

#### Accessibility

Accessibility describes the degree to which a software system is available to as many people as possible. It can be viewed as the ability to access and benefit from some system. Accessibility is often used to focus on people with disabilities or special needs and their right of access to system. As mentioned in [Hachey, 2011], papers discussing accessibility are clearly lacking in the context of Semantic Web UIs.

#### Proactivity

Proactivity is the ability of a system to act in advance of a future situation, rather than just reacting. It means taking control and making things happen rather than just adjusting to a situation or waiting for something to happen. An SCA system should provide users with pre-filled form fields, suggestions, default values etc. These facilities simplify the authoring process, as they reduce the number of

actions users have to perform. Moreover, they reduce the possibility that users provide incomplete or empty metadata [Di Iorio et al., 2010].

The following features of UIs are proposed for improving the proactivity of SCA systems:

- *Real-time Semantic Tagging.*
  Real-time tagging means creating annotations while the user is typing [Heese et al., 2010]. This will significantly increase the annotation speed [O'Donoghue et al., 2010]. Users are not distracted since they do not have to interrupt their current authoring task. This type of UI needs a client-side component which interacts with the server asynchronously.

- *Resource Suggestion.*
  An SCA system should provide users with a set of entity (i.e. URI) suggestions to facilitate the annotation process for non-expert users [Luczak-Roesch, 2009, Buffa et al., 2008].

- *Concept Reuse.*
  An SCA system becomes increasingly advantageous, if once defined concepts (e.g. classes, properties, or instances) are as much reused and interlinked as possible [Auer et al., 2006]. Suggesting already defined concepts to users (particularly new and inexperienced users) will facilitate their contribution to the system.

- *Real-time Validation.*
  When the annotation is completed by user, the SCA system should apply validation mechanisms to check the correctness of the values. Validating metadata while they are being created improves the overall quality of the documents and does not require further consistency checks, which might be difficult or even impossible once the provider of metadata has completed the job [Valkeapaeae et al., 2007, Di Iorio et al., 2010].

### Automation

Automation is the ability of a system to automatically perform its intended tasks thereby reducing the need for human work. In the context of semantic authoring it means the provision of facilities for automatic mark-up of documents to facilitate the economical annotation of large document collections [Uren et al., 2006]. The automatic process of annotating is composed basically of finding terms in documents, mapping them against an ontology, and disambiguating common terms. There are wide ranges of approaches that carry out automatic annotation of texts. Most of them employ natural language processing and information extraction techniques. These approaches differ in architecture, information extraction tools and methods, initial ontology, amount of manual work required to perform annotation, as well as performance [Siorpaes and Simperl, 2010, Araujo et al.,

2010]. Existing automated SCA systems can be divided into two categories: semi-automatic and fully-automatic systems. In semi-automatic systems [Valkeapaeae et al., 2007, Luczak-Roesch, 2009], the user is provided with a set of suggestions to select from. So, disambiguation is performed with the help of user (i.e. incorporating user feedback to enhance the automation results). In fully-automatic systems [Klebeck et al., 2011, Kiyavitskaya et al., 2009], annotations are generated without any intervention by users. Fully-automatic systems can generally be regarded as falling into three categories [Uren et al., 2006]. The most basic kind use rules or wrappers written by hand those try to capture known patterns for the annotations. Then there are two kinds of systems that learn how to annotate. Supervised systems learn from sample annotations marked up by the user. A problem with these methods is that picking enough good examples is a non-trivial and error-prone task. In order to tackle this problem unsupervised systems employ a variety of strategies to learn how to annotate without user supervision, but their accuracy is still limited.

Automated SCA systems should take into account user interface design issues related to minimizing intrusiveness while maximizing accuracy. Completely automated systems, which do not involve any user interaction in the process of semantic content creation, are out of scope of this work. User interaction is required to supervise, assess or evaluate the automated annotation thereby creating accurate semantic content.

### Evolvability

Evolvability is defined as the capacity of a system for adaptive evolution. An SCA system should support evolution of the annotated document [Uren et al., 2006, Heese et al., 2010, Navarro-Galindo and Samos, 2010, Ruiz-Rube et al., 2010, Di Iorio et al., 2010]. To achieve this goal, it should take into account the following consistency constraints:

- *Resource Consistency.*
  If users annotate the same resource in different texts, it is important to reference the same resource in the generated RDF statements. Otherwise, we obtain many resources that are not interlinked and the statements in the repository are not very useful and meaningful [Heese et al., 2010].

- *Document and Annotation Consistency.*
  *Supporting Ontology Modification* as discussed in Section 3.4 is an important feature for the generalizability of an SCA system. In this case, the system also needs to deal with consistency issues that might arise between ontologies and annotations. One of the important issues for the design of a semantic authoring environment is to determine how changes should be reflected in the knowledge base of annotated documents and whether changes to ontologies create conflicts with existing annotations [Uren et al., 2006]. Ontologies

change sometimes but some documents change many times. So, it is crucial for an SCA system to track the annotation evolution.

An SCA system should provide appropriate UIs for versioning and change tracking to deal with document and annotation evolution.

### Interoperability

Interoperability is the ability of a system to work and interact with other systems. An SCA system should provide mechanisms to interoperate together with other systems which generate or consume the semantic content created. The following features of UIs are proposed for improving the interoperability of SCA systems:

- *Support of Standard Formats.*
  To minimize the problems of interoperability the SCA system should be built on standards. There are already many standards for semantic content serialization (e.g. typical RDF serializations and particular RDFa), representation (e.g. RDF/RDF-S/OWL/RIF and established vocabularies such as *Semantically-Interlinked Online Communities (SIOC)*, *SKOS*, *Friend Of A Friend (FOAF)*, *rNews*, etc.) and exchange (e.g. Linked Data, Web Services, REST). Supporting standard formats and avoiding proprietary formats are essential for compatibility of data with other systems [Uren et al., 2006, Valkeapaeae et al., 2007, Buffa et al., 2008, Heese et al., 2010, Luczak-Roesch, 2009, Ruiz-Rube et al., 2010, Tramp et al., 2010, Klebeck et al., 2011].

- *Semantic Syndication.*
  Semantic syndication supports the distribution of information and their integration into other applications by providing mechanisms such as *Semantic Atom* [Patel and Khuba, 2009] and *Semantic Pingback*[18] [Auer et al., 2006].

### Scalability

Scalability refers to the capability of a system to maintain performance under an increased work load. An SCA system should support scalability as for example, the number of users, data or annotations increase. Support of *caching* and implementing a suitable *storage strategy* play an important role in achieving an scalable SCA system [Auer et al., 2006, Uren et al., 2006, Navarro-Galindo and Samos, 2010, Herzig and Ell, 2010, Luczak-Roesch, 2009]. Annotations can be directly stored in the document or stored separately in a triple store. Most of the current SCA systems adopt a dual storage strategy of semantic annotations. In this case, annotations are stored in a server-side triple store and also embedded in the same document where annotations are undertaken in a way that is completely transparent to the user. A dual storage approach poses a redundancy but allows

[18] http://aksw.org/Projects/SemanticPingBack

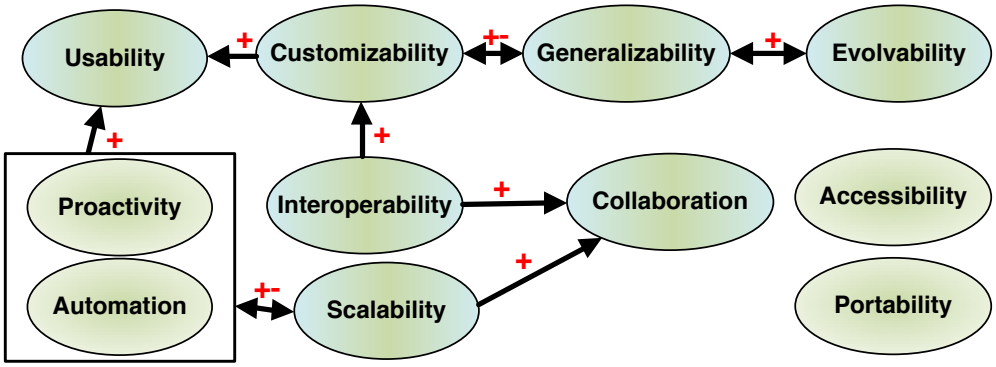


Figure 3.6.: Quality attributes dependencies ('+': positive effect, '+-': reciprocal effect).

information from heterogeneous resources to be queried centrally and in real-time as a knowledge base [Auer et al., 2006, Uren et al., 2006].

## 3.5. Quality Attributes Dependencies

The aforementioned quality attributes are not completely isolated and independent from each other but have overlaps and relations with each other. Figure 3.6 shows an overview of these quality attributes with their interrelations. Proactivity, automation and customizability will improve the usability of an SCA system. Proactive and automatic systems provide users with helpers, which facilitate the usage of the system. Customized systems are configured based on the user needs thereby increase the overall usability of the system.

Scalability will enhance the level of system collaboration. A scalable system will support more users and annotations thereby more collaboration in the system. Interoperability will also enhance the collaboration support of a system, since an interoperable system supports users of different systems. It can also support importing user data from other systems that will play a positive role in enhancing the customizability.

Evolvability and generalizability are directly related. The more evolvable to change a system is, the more generic it will be and vice versa. Customizability and generalizability share a reciprocal relation. A generic system will decrease its customization and a customizable system needs to focus on specific user needs and thus lacks generalizability. Scalability also has a reciprocal relation to proactivity and automation. Having scalability with larger data, computing proactivity and automation actions may become too heavy and complex to handle.

## 3.6. User Types

Table 3.3 shows the list of tools discussed in our primary studies. The following tools were described in the primary studies: OntoWiki [Auer et al., 2006], SAHA [Frosterus et al., 2011], OWiki [Di Iorio et al., 2010], SemCards [Thórisson et al., 2010], DataPress [Benson et al., 2010], Loomp [Luczak-Roesch, 2009], Semantic MediaWiki [Herzig and Ell, 2010], SweetWiki [Buffa et al., 2008], Information Workbench [Muller et al., 2011], RDFAuthor [Tramp et al., 2010], FLERSA [Navarro-Galindo and Samos, 2010], LinkedBlog [Ruiz-Rube et al., 2010], SemiBlog [Möller et al., 2006], HayStack semantic blogging [Karger and Quan, 2005], Reflect [O'Donoghue et al., 2010], Ontos-feeder [Klebeck et al., 2011], Epiphany [Adrian et al., 2010], Linkator [Araujo et al., 2010], Tabulator [Berners-Lee et al., 2007].

For each tool, we extracted the type of user, domain of the tool and the authoring approach employed in the tool. There are two general types of users mainly discussed in the studies:

- End user or normal users which have no or limited knowledge of the domain on which the annotations or semantic structures are applied. They constitute the majority of the population using the Internet to browse for information and communicate with others.
- Domain experts which have a broad knowledge of the domain on which the annotations or semantic structures are applied. They are usually consisted of the researchers or engineers with a top-down view of problems.

As our results revealed, the majority of studies (i.e. all the tools which employed the bottom-approach) were addressing tools that are appropriate for end users. Tools that were adopting the top-down approach needed users to have knowledge of the corresponding domain as well as ontology concepts. It is worth mentioning that by domain-independent, we mean that the tool is not limited to any predefined domains and is flexible enough to be applied in arbitrary domains. For instance, *OntoWiki* is domain-independent while it requires a domain-expert as user. This is due to the fact, that when a user wants to create semantic content using OntoWiki, he should have a broad view on the selected domain to define the required ontologies (i.e. the available classes, their possible relationships, constraints, class properties, data types, etc.) and to populate the data accordingly. Otherwise he cannot create semantic content with the tool.

| Tool | Ref. | User Type | Domain | Authoring Approach |
|---|---|---|---|---|
| OntoWiki | [Auer et al., 2006] | Domain expert | Domain-independent | Top-Down |
| OWiki | [Di Iorio et al., 2010] | Domain expert | Domain-independent | |
| SAHA | [Frosterus et al., 2011] | Domain expert | Governmental data | |
| SemCards | [Thórisson et al., 2010] | End-user (non-expert) | Domain-independent | |
| RDFAuthor | [Tramp et al., 2010] | End-user & Domain-expert | Domain-independent | |
| Tabulator | [Berners-Lee et al., 2007] | End-user & Domain-expert | Domain-independent | |
| Reflect | [O'Donoghue et al., 2010] | End-user (researchers) | Chemistry | Bottom-Up |
| Epiphany | [Adrian et al., 2010] | End-user | CMS | |
| Ontos-feeder | [Klebeck et al., 2011] | End-user (journalist) | Journalism | |
| DataPress | [Benson et al., 2010] | End-user (blogger) | Blogs & Wikis | |
| Loomp | [Luczak-Roesch, 2009] | End-user (journalist) | Journalism | |
| Semantic MediaWiki | [Herzig and Ell, 2010] | End-user (Wiki users) | Domain-independent | |
| LinkedBlog | [Ruiz-Rube et al., 2010] | End-user (blogger) | Blogs | |
| SweetWiki | [Buffa et al., 2008] | End-user (Wiki users) | Domain-independent | |
| Linkator | [Araujo et al., 2010] | End-user | CMS | |
| FLERSA | [Navarro-Galindo and Samos, 2010] | End-user | CMS | |
| Information Workbench | [Muller et al., 2011] | End-user (researchers) | Paper review & publishing | |
| HayStack semantic blogging | [Karger and Quan, 2005] | End-user (blogger) | Blogs | |
| SemiBlog | [Möller et al., 2006] | End-user (blogger) | Blogs | |

Table 3.3.: User types, domain and authoring approach of the surveyed SCA systems.

## 3.7. User Interface Evaluation

In this section we briefly outline various methods for user interface evaluation and report about their usage in the surveyed papers. Table 3.4 lists existing methods for user interface evaluation adopted from [Fitzpatrick, 1998, Chen and Babar, 2011].

Among the primary studies, the majority of studies ([Auer et al., 2006, Di Iorio et al., 2010, Luczak-Roesch, 2009, Ruiz-Rube et al., 2010, Thórisson et al., 2010, Tramp et al., 2010, Muller et al., 2011, Araujo et al., 2010, Navarro-Galindo and Samos, 2010, Möller et al., 2006, Klebeck et al., 2011, Berners-Lee et al., 2007, Herzig and Ell, 2010]) were only using an *Example Application* as their evaluation method. A few studies ([Frosterus et al., 2011, Buffa et al., 2008, O'Donoghue et al., 2010, Adrian et al., 2010, Karger and Quan, 2005]) were also using the *Discussion* method together with an example application. One study ([Benson et al., 2010]) used *Interview* and *Questionnaire* methods for UI evaluation. Other papers were either survey papers or the papers, which only mentioned some user interface features and did not provide any UI evaluation method. The results distinctly exposes the lack of formal and systematic UI evaluation methods in the context of SCA systems.

Analyzing the suitability of each evaluation method for measuring the quality attributes introduced in Section 3.4 is beyond the scope of this work but to bring some examples: Among the evaluation methods, *Simulation* is suitable to measure the *Scalability* of an SCA system. Most of the evaluation methods (e.g. *Empirical Methods*, *Questionnaire*, *Interview*, *Observation* and *Modeling Methods*) can be used to measure the *Usability* of an SCA system. *Observation* method seems to be suitable to measure the level of *Collaboration* in an SCA system and so on.

| Evaluation Method | Definition |
|---|---|
| Empirical Methods (Case Study) | An empirical inquiry that investigates a contemporary phenomenon within its real-life context; A usability evaluation specialist tests a well-defined hypothesis by measuring subject (user) behavior while he manipulates variables. |
| Discussion | Provided some qualitative, textual, opinion-oriented evaluation. E.g., compare and contrast, oral discussion of advantages and disadvantages. |
| Example Application | Authors describe an application and provide an example to assist in the description, but the example is "used to validate" or "evaluate" as far as the authors suggest. |
| Observation (Experience) | The result has been used on real examples, but not in the form of case studies or controlled experiments, the evidence of its use is collected informally or formally. A usability evaluation specialist acts as the observer of users as they interact with computers, noting user successes, difficulties, likes, dislikes, preferences and attitudes. |
| Questionnaire | The use of a set of items (questions or statements) to capture statistical data relating to user profiles, skills, experience, requirements, opinions, preferences and attitudes. |
| Interview | A formal consultation or meeting between a usability evaluation specialist and user(s) to obtain information about work practices, requirements, opinions, preferences and attitudes. |
| User Groups | Availing of the wealth of knowledge and experience of organized (user forum) and selected (beta site) end users. |
| Cognitive Walkthroughs | A step-by-step evaluation of a design by a cognitive psychologist in order to identify potential user psychological difficulties with the system. |
| Heuristic Methods | The use of a team of usability evaluation specialists to review a product or prototype in order to confirm its compliance with recognized usability principles and practice. |
| Review Methods | The review and reuse of the wealth of experimental and empirical evidence in the research literature and in the de-facto standards established by the software industry. |
| Simulation | Execution of a system within artificial data, using a model of the real world. |
| Modeling Methods | Using models like GOMS (Goals, Operations, Methods and Selection) and KLM (Keystroke Level Modeling) to predict and provide feedback on user interactions and difficulties. |

Table 3.4.: User interface evaluation methods.

## 3.8. Example Tools

In this section we look at three available SCA systems and compare them according to the quality attributes defined in Section 3.4. We will investigate their strengths and weaknesses based on our proposed taxonomy of quality attributes and UI features which are required for SCA systems. We have selected these three example tools so that top-down and bottom-up tools, domain expert and end user tools as well as domain-independent and domain-specific tools are covered. Among the tools two (i.e. Ontowiki and SAHA 3) follow the top-down approach (cf. Section 3.3.2) and one (i.e. Loomp) follow a bottom-up approach (cf. Section 3.3.1) for semantic content authoring.

We used the criteria *availability of an online demo*, *availability of technical implementation details*, having *up-to-date support* and *number of quality attributes addressed in the tool* to select these 3 tools out of the 20 tools discovered in the literature (cf. Table 3.3). Table 3.5 summarizes the assessment of the tools according to the defined quality attributes.

### 3.8.1. OntoWiki

*OntoWiki*[19] [Auer et al., 2006] is a tool that provides support for agile, distributed knowledge engineering scenarios. Ontowiki facilitates the visual presentation of a knowledge base as an information map, with different views on instance data.

Regarding the technical realization, the system is implemented in *PHP* using the *Zend framework*[20]. It supports the *MySQL* database and the *Virtuoso* triple store as storage backends and the authoring interface is built using jQuery UI[21].

Figure 3.7 shows a screenshot of OntoWiki. OntoWiki was applied in a number of use cases. Examples include: semantic content management [Heino et al., 2011], collaborative requirements engineering with *SoftWiki* [Lohmann et al., 2008] and historical, prosopographical knowledge engineering with the *Professor's Catalogue* [Riechert et al., 2010].

Ontowiki as a single point of entry UI adopts the top-down approach for semantic authoring. It provides a semantic search feature with support for faceted browsing. It also supports two complementary knowledge base authoring strategies: a) *Inline editing*, which enables users to edit small information chunks (i.e. statements). b) *View editing*, which enables users to edit common combinations of information (such as an instance of a distinct class) in one single step. In order to do so, Ontowiki uses *RDFAuthor* [Tramp et al., 2010] to make generated RDFa views editable. Regarding the customizability, OntoWiki supports different semantic views of the knowledge base, which can be generic or domain-specific. It also supports editing multiple ontologies including both the instances and structures of

[19] http://ontowiki.net
[20] http://framework.zend.com/
[21] http://jqueryui.com/

| | OntoWiki | SAHA 3 | Loomp |
|---|---|---|---|
| **Usability** | • Single point of entry UI<br>• Faceted browsing<br>• Inline editing/view editing | • Single point of entry UI<br>• Faceted browsing<br>• Inline editing | • Single point of entry UI<br>• Faceted viewing |
| **Customizability** | • Semantic views: domain specific & generic (e.g. map, calendar) | - | - |
| **Generalizability** | • Multiple ontology support<br>• Ontology modification support | • Multiple ontology support | • Multiple ontology support |
| **Collaboration** | • Access control<br>• Standard formats: RDF, RDFa<br>• Social collaboration UIs: rating and commenting UIs | • Access control<br>• Standard formats: RDF<br>• Social collaboration UIs: online chat | • Standard formats: RDF, RDFa |
| **Portability** | • Cross-browser compatibility<br>• UI for mobile devices | • Cross-browser compatibility | • Cross-browser compatibility |
| **Accessibility** | - | - | - |
| **Proactivity** | • Resource suggestion<br>• Concept reuse | • Resource suggestion<br>• Concept reuse<br>• Real-time validation | • Resource suggestion<br>• Concept reuse |
| **Automation** | - | - | - |
| **Evolvability** | • Resource consistency<br>• Document & annotation consistency<br>• Versioning & change tracking | • Resource consistency<br>• Document and annotation consistency | • Document and annotation consistency |
| **Interoperability** | • Standard formats: RDF, RDFa<br>• Semantic syndication: semantic pingback | • Standard formats: RDF | • Standard formats: RDF, RDFa |
| **Scalability** | • Caching support<br>• Storage strategy: backend independent (Mysql, Virtuoso) | • Storage strategy: using a server-side triple store | • Storage strategy: using a server-side triple store |

Table 3.5.: Comparison of OntoWiki, SAHA 3, Loomp according to the quality attributes.

the ontologies. As a Web-based system, it provides cross-browser compatibility and has a specific UI for mobile devices. To provide proactivity, OntoWiki uses the AJAX technology to interactively propose already defined concepts while the user types in new information to be added to the knowledge base (i.e. Concept Reuse).

OntoWiki also provides versioning and evolution features to track, review and selectively roll-back changes and supports semantic syndication (employing *Semantic Pingback* and Linked Data interfaces) to interoperate with other systems. OntoWiki is backend independent to some extend and supports two different types of storage engines. It also provides a caching component to optimize the performance of the system.

As a drawback, OntoWiki does not provide any UI elements to facilitate accessibility and automation. It supports only the editing of structured content thus lacking UIs for the annotation of unstructured or semi-structured content. Furthermore, it does not provide real-time tagging and validation for increasing the overall proactivity.

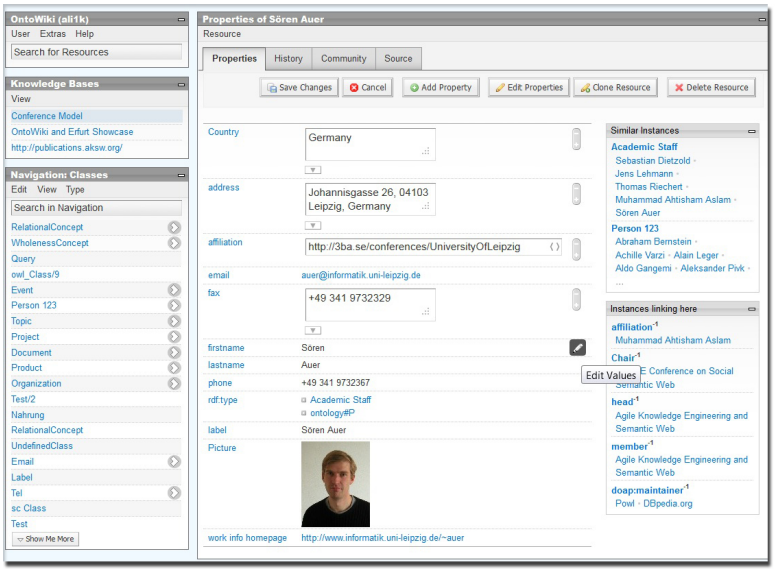


Figure 3.7.: Screenshot of the OntoWiki instance view with inline editing.

### 3.8.2. SAHA 3 Metadata Editor

*SAHA 3*[22] [Frosterus et al., 2011] is an RDF metadata editor for collaborative content creation and instant semantic content publishing on the Semantic Web. Regarding the technical realization, the system is implemented in Java on top of the *Spring framework*[23]. The data model is based on the *Jena TDB*[24] RDF database and the editor interface is built using *DWR*[25] and the *Dojo*[26] AJAX components.

Figure 3.8 shows a screenshot of SAHA 3. *DataFinland*[27] as a semantic portal for open and linked datasets is one use case of SAHA 3.

Like OntoWiki, SAHA 3 uses the top-down approach for semantic authoring and a single point of entry UI with inline editing features. It supports faceted browsing when integrated into the faceted portal engine HAKO[28] for content publishing. SAHA 3 supports multiple ontologies as well as collaborative simultaneous editing. Resources that are being edited by one user are locked for editing by other users. A chat facility has been integrated into the editor to facilitate instant discussions between peer editors.

Regarding proactivity, SAHA 3 also provides real-time semantic validation and concept reuse. As shown in [Kurki and Hyvönen, 2010], SAHA 3 has proven a

[22] http://demo.seco.tkk.fi/saha/saha3/
[23] http://www.springsource.com/
[24] http://openjena.org/TDB/
[25] http://directwebremoting.org/
[26] http://www.dojotoolkit.org/
[27] http://www.seco.tkk.fi/linkeddata/datasuomi/
[28] http://www.seco.tkk.fi/tools/hako/

good level of scalability to support large projects.

As a drawback, SAHA 3 does not provide any UI elements to provide customizability, accessibility and automation. Like OntoWiki, it only supports structured content authoring thus lacking the appropriate UIs for unstructured or semi-structured content annotation. Although it provides some simple UIs for supporting collaboration but lacks sophisticated features regarding social interactions. Since it does not provide any UI elements for versioning and change tracking, evolvability is not well addressed. Scalability could be further improved by adding support for caching and alternative storage backends (i.e. client-side RDF processing).

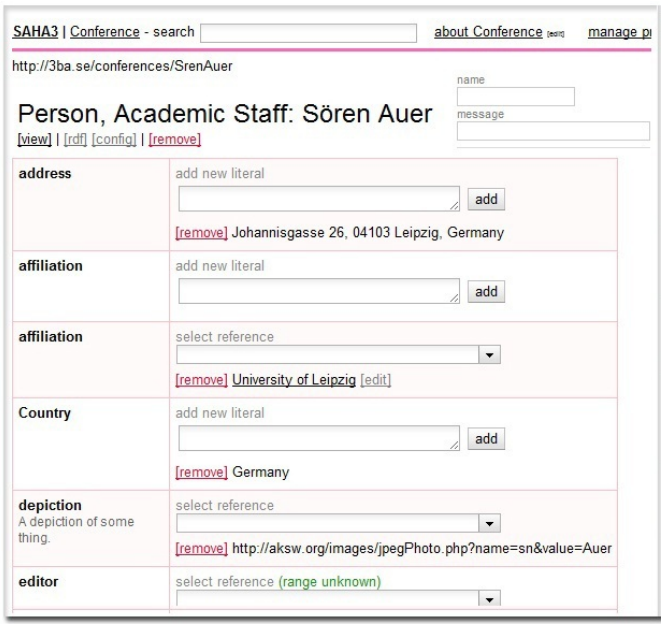


Figure 3.8.: Screenshot of the SAHA 3 inline editing.

### 3.8.3. Loomp

*Loomp*[29] [Luczak-Roesch, 2009] is a tool representing a prove-of-concept for the *One Click Annotation (OCA)* (heese2010) strategy. The Web-based OCA editor allows for annotating words and phrases with references to ontology concepts and for creating relationships between annotated phrases [Heese et al., 2010].

Regarding the technical realization, Loomp is a typical Web application built on the LAMP stack[30]. It serves content either in RDF (e.g. for linked data clients) or in XHTML/RDFa (e.g. for Web browsers).

[29] http://loomp.org

[30] LAMP: Linux operating system, Apache web server, MySQL database, PHP scripting language.

Figure 3.9 shows a screenshot of Loomp. *Data-driven journalism* is mentioned as one of the primary uses cases of Loomp.

Loomp provides a WYSIWYG editor as a single point of entry UI which adopts the bottom-up approach for semantic content authoring. It supports a faceted viewing feature, which highlights user-selected annotations in the Web browser. Loomp facilitates concept reuse and suggestion in order to reduce non-expert user efforts during the annotation process. It also employs RDF and RDFa standard formats which make it interoperable with other systems.

As a drawback, Loomp lacks appropriate UI elements to support customizability, accessibility and automation. It does not provide any UI features for faceted browsing and inline editing of annotations. It also does not allow to directly edit the underlying ontologies thereby extending the annotation domain. Furthermore, Loomp lacks appropriate UI elements for real-time tagging and validation as well as versioning and change tracking. Regarding scalability, no information could be found on how Loomp supports large amounts of users and annotations.

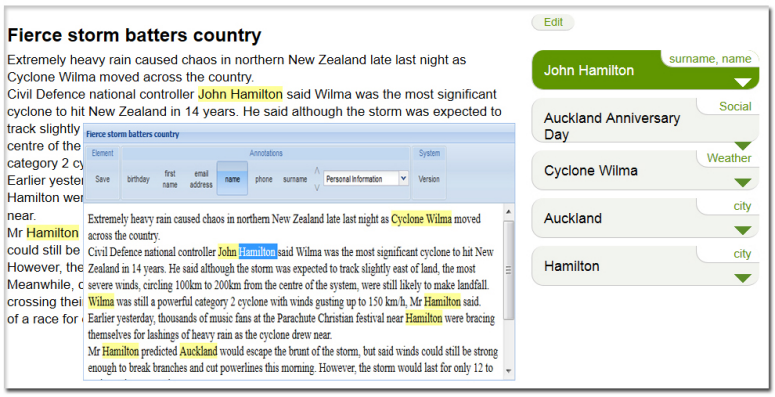


Figure 3.9.: Screenshot of the Loomp faceted viewing UI.

## 3.9. Research and Technology Challenges

The results of our systematic review revealed some research and technology gaps and corresponding challenges with regard to the development of SCAUIs.

**Accessibility.** There is a clear research gap in addressing accessibility issues during the design of SCAUIs. Many semantic authoring tools remain inaccessible to people with disabilities. Providing people with disabilities and special needs with appropriate SCAUIs can facilitate their tasks of information seeking. Semantically annotated content allows alternatives or conditional content in different modalities to be selected based on the type of the user disability and information need. For example, visually impaired people need significantly more time to grasp the structure and gist of a Web site, since visual navigation and structuring elements

are not accessible to them. Once content is semantically annotated, visually impaired people can use this semantic annotation as a means to access and explore the content faster.

The W3C's Web Content Accessibility Guidelines (WCAG)[31] explain how to make Web content more accessible to people with disabilities. As part of WCAG, *Authoring Tool Accessibility Guidelines (ATAG)*[32] and more specifically *Accessible Rich Internet Applications* (WAI-ARIA)[33] suite, define how authoring tools should support accessibility requirements. Consequently, a challenge is to apply and extend the series of accessibility guidelines proposed in ATAG for the purpose of designing accessible SCAUIs.

**Handling complexity in UIs.** One important concern when designing SCAUIs is how to make complex functionality available to the user in a simple way. There are two issues in this context, which need to be addressed. The first one is how to properly map complex functions and algorithms (e.g. entity disambiguation, recommendation and other machine learning algorithms) to corresponding user interface elements. The second issue is how to flatten the user's learning curve by providing adaptive and intelligent UIs which take user knowledge into account. Many current SCA systems bear a bewildering amount of functions and algorithms which confuses both the novice and expert users. This problem causes a significant impediment for a broader use of SCA systems.

Addressing the complexity problem requires the creation of abstract models for complex tasks as well as modeling the user characteristics and behavior. Ideally, the SCAUI should present the users with concepts that are consistent with both designer and users' mental models of that phenomenon in the real world. The above mentioned issues are well addressed in designing Geographic Information System (GIS) UIs [Yu, 2006, Camara et al., 1999]. Now it is a challenge to rethink these issues for the purpose of designing adaptive and flexible SCAUIs.

**Formal and systematic methods for user interface evaluation.** The results of our survey clearly reflects the lack of formal and systematic UI evaluation methods in evaluating SCA systems. As described in Table 3.4, there are several UI evaluation methods which can be used in this context. Nielsen and Molich [Nielsen and Molich, 1990] enumerate four general categories of systematic user interface evaluation methods: formally by employing an analysis technique; automatically by a computerized procedure; empirically by testing users performing experiments; and heuristically.

In *heuristic evaluation*, evaluators inspect a user interface against a guideline to identify usability problems that violate any items on the guideline [Kock et al., 2009]. Our list of quality attributes and UI features (cf. Section 3.4) can be used

[31] http://www.w3.org/WAI/intro/wcag.php
[32] http://www.w3.org/WAI/intro/atag.php
[33] http://www.w3.org/TR/wai-aria/

as a guideline for heuristic evaluation of SCA system UIs. This will require fewer resources than testing with users and can be applied to the system during the design phase.

**Support of crowdsourcing.** One of the missing aspects of developing collaborative SCA systems is the support of crowdsourcing. There is a huge amount of amateur and expert users who collaborate on and contribute to the Social Web. Harnessing the power of such crowds can significantly enhance and widen the results of semantic content authoring and annotation. Crowdsourcing as a distributed problem-solving and production model is defined to address this aspect of collective intelligence [Howe, 2006].

In order to support crowdsourcing, an SCA system needs to provide appropriate UIs. In [Geiger et al., 2011], Geiger et al. analyze the respective characteristics and requirements related to the design of crowdsourcing systems. Providing small contributions with instant gratification, altruism and a way to establish a reputation are some of these requirements. As a new challenge, it is worth to consider these characteristics when designing SCAUIs.

**UIs for ubiquitous devices.** As discussed in Section 3.4, creating UIs for mobile and ubiquitous devices is an issue which is not well addressed in the literature yet. Mobile and ubiquitous devices are rapidly becoming the central computing and communication devices in people's lives. Ubiquitous computing (a.k.a *everyware* [Greenfield, 2006]) presents new challenges in user interface design. Emerging ubiquitous devices are programmable and come with a growing set of facilities including multi-touch screens and cheap powerful embedded sensors, such as an accelerometer, digital compass, gyroscope, GPS, microphone, and camera [Lane et al., 2010]. Utilizing these rich set of UI facilities when developing SCA systems can improve the user experience in the process of semantic content authoring and annotation. For example, users can easily share their real-time activities with SCA system using mobile sensors or can use some gestures for annotating the content.

Another challenge here is the ability to provide offline functionality with local updates for later synchronization with a web server. SCA systems should take into account the reconciliation problem – the problem of potentially conflicting updates from disconnected clients.

## 3.10. Conclusions

In this chapter we reported on the results of a systematic literature review on user interfaces for semantic content authoring comprising initially 175 papers. The review aimed to answer the RQ1 research questions defined in Section 1.3 by thorough analysis of the 31 most relevant papers. Before addressing the defined research questions, we drew a terminology, which defines the basic concepts used in the literature as well our survey. To answer the RQ1.1, we classified

existing approaches for SCA into two categories namely Top-Down and Bottom-Up discussed in Section 3.3. Furthermore, Our data analysis revealed a set of quality attributes for SCA systems together with the corresponding user interface features which are suggested for their realization. These quality attributes plus the UI features are used to answer the RQ1.2 and RQ1.3. In order to answer RQ1.4 and RQ1.5 we extracted the types of users as well as user evaluation methods discussed in the primary studies and reflected the results in Section 3.7. Open research and technological challenges in the domain of SCA systems were discussed as well. Additionally, to show the applicability of the results, we performed an in-depth comparison of three existing SCA systems according to the defined quality attributes and described their strengths as well as their weaknesses.

Essential, foundational quality attributes for an SCA system are, in particular, usability, generalizability, customizability and evolvability. A basic SCA system should fulfill a reasonable level of user-friendliness and adopt to different situations or use cases while providing mechanisms to tailor its functionality based on specific user needs. It also should take into account issues such as consistency constraints and content evolution, which are required for change management. Support of collaboration, interoperability and scalability are quality attributes required when an SCA system is employed in a community-driven environment with large amount of users, systems and interactions. An SCA system should support standard formats and provide appropriate UI elements for social networking including both human-to-human as well as system-to-system interactions. Additionally, it should maintain performance under an increased workload by supplying appropriate storage and caching mechanisms. Automation and proactivity are quality attributes which facilitate usability of SCA systems especially for non-skilled users. Portability and accessibility are, as our survey indicates, not well addressed by the literature so far. The demands for suitable UIs for mobile and ubiquitous devices are increasing as powerful mobile computing devices are becoming more common. Furthermore, providing accessible UIs for people with disabilities or special needs is another requirement which should be taken into account when designing SCA systems.

Chapter 4

# WYSIWYM User Interface Model

> "Design is not just what it looks like and feels like. Design is how it works." — *Steve Jobs*

In this chapter, we present an approach inspired by the WYSIWYM metaphor (What You See Is What You Mean), which addresses the issue of an integrated visualization, exploration and authoring of semantically enriched un-structured content. Our WYSIWYM concept formalizes the binding between semantic representation models and UI elements for authoring, visualizing and exploration. We analyze popular tree, graph and hyper-graph based semantic representation models and elicit a list of semantic representation elements, such as entities, various relationships and attributes. We provide a comprehensive survey of common UI elements for authoring, visualizing and exploration, which can be configured and bound to individual semantic representation elements. Our WYSIWYM concept also comprises cross-cutting helper components based on the results achieved in Chapter 3, which can be employed within a concrete WYSIWYM interface for the purpose of automation, annotation, recommendation, personalization, etc.

The remainder of this chapter is structured as follows: In Section 4.1, we describe the existing approaches for binding UIs to semantic models. Section 4.2 describes the fundamental WYSIWYM concept proposed in this chapter. Subsections of Section 4.2 present the different components of the WYSIWYM model. Finally, Section 4.3 concludes with an outlook on next steps.[1]

## 4.1. Approaches for Semantic UI Models

In this section, we briefly describe the existing approaches which provide a binding between UI components and semantic models.

### 4.1.1. Visual Mapping Techniques

Visual mapping techniques are knowledge representation techniques that graphically represent knowledge structures. Most of them have been developed as paper-based techniques for brainstorming, learning facilitation, outlining or to elicit knowledge structures. According to their basic topology, most of them can be related to the following fundamentally different primary approaches [Haller and Abecker, 2010, Shneiderman, 2000]:

[1] The contents of this chapter have been published as [Khalili and Auer, 2014].

- *Mind-Maps.* Mind-maps are created by drawing one central topic in the middle together with labeled branches and sub-branches emerging from it. Instead of distinct nodes and links, mind-maps only have labeled branches. A mind-map is a connected directed acyclic graph with hierarchy as its only type of relation. *Outlines* are a similar technique to show hierarchical relationships using tree structure. Mind-maps and outlines are not suitable for relational structures because they are constrained to the hierarchical model.

- *Concept Maps.* Concept maps consist of labeled nodes and labeled edges linking all nodes to a connected directed graph. The basic node and link structure of a connected directed labeled graph also forms the basis of many other modeling approaches like *ER* diagrams and *Semantic Networks.* These forms have the same basic structure as concept maps but with more formal types of nodes and links.

- *Spatial Hypertext.* A spatial hypertext is a set of text nodes that are not explicitly connected but implicitly related through their spatial layout, e.g., through closeness and adjacency — similar to a pin-board. Spatial hypertext can show fuzzily related items. To fuzzily relate two items in a spatial hypertext schema, they are simply placed near to each other, but possibly not quite as near as to a third object. This allows for so-called "constructive ambiguity" and is an intuitive way to deal with vague relations and orders. Spatial Hypertext abandons the concept of explicitly interrelating objects. Instead, it uses spatial positioning as the basic structure.

### 4.1.2. Structured Content Visualization

There are already many approaches and tools which address the binding between data and UI elements for visualizing and exploring structured content. Dadzie and Rowe [Dadzie and Rowe, 2011] present the most exhaustive and comprehensive survey to date of these approaches. For example, *Fresnel* [Pietriga et al., 2006] is a display vocabulary for core RDF concepts. Fresnel's two foundational concepts are lenses and formats. Lenses define which properties of an RDF resource, or group of related resources, are displayed and how those properties are ordered. Formats determine how resources and properties are rendered and provide hooks to existing styling languages such as Cascading Style Sheets (CSS).
*Parallax*, *Tabulator*, *Explorator*, *Rhizomer*, *Sgvizler*, *Fenfire*, *RDF-Gravity*, *IsaViz* and *i-Disc for Topic Maps* are examples of tools available for visualizing and exploring structured data. In these tools the binding between semantics and UI elements is mostly performed implicitly, which limits their versatility. However, an explicit binding as advocated by our WYSIWYM model can be potentially added to some of these tools.

In contrast to the structured content, there are few approaches and tools which

allow binding semantic data to UI elements within semantically enriched unstructured content. As an example, *Dido* [Karger et al., 2009] is a data-interactive document which lets end users author semantic content mixed with unstructured content in a Web page. Dido inherits data exploration capabilities from the underlying *Exhibit*[2] framework. *Loomp* as a prove-of-concept for the *One Click Annotation (OCA)* [Heese et al., 2010] strategy is another example in this context (cf. Section 3.8.3). It employs a partial mapping between UI elements and data to hide the complexity of creating semantic data.

### 4.1.3. WYSIWYM

The first use of the WYSIWYM term occurred in 1995 aiming to capture the separation of presentation and content when writing a document. The *LyX editor*[3] was the first WYSIWYM word processor for structure-based content authoring. Instead of focusing on the format or presentation of the document, a WYSIWYM editor preserves the intended meaning of each element. For example, page headers, sections, paragraphs, etc. are labeled as such in the editing program, and displayed appropriately in the browser. Another usage of the WYSIWYM term was by Power et al. [**?**] in 1998 as a solution for *Symbolic Authoring*. In symbolic authoring the author generates language-neutral "symbolic" representations of the content of a document, from which documents in each target language are generated automatically, using *Natural Language Generation* technology. In this What-You-See-Is-What-You-Meant approach, the language generator was used to drive the UI with support of localization and multilinguality. Using the WYSIWYM natural language generation approach, the system generates a feed-back text for the user that is based on a semantic representation. This representation can be edited directly by the user by manipulating the feedback text.

The WYSIWYM term as defined and used in this thesis targets the novel aspect of integrated visualization, exploration and authoring of unstructured and semantic content. Two "You"s in our WYSIWYM concept refer to the end user (with no or limited knowledge of Semantic Web) who is viewing an unstructured content which is semantically enriched by himself. The "Mean" refers to the metadata or semantics, which is encoded in the unstructured content viewed by user.

## 4.2. WYSIWYM Concept

In this section we introduce the fundamental WYSIWYM concept and formalize key elements of the concept. Formalizing the WYSIWYM concept has a number of advantages: First, the formalization can be used as a basis for design and implementation of novel applications for authoring, visualization, and exploration of semantic content. The formalization serves the purpose of providing a terminology

[2] http://simile-widgets.org/exhibit/
[3] http://www.lyx.org/

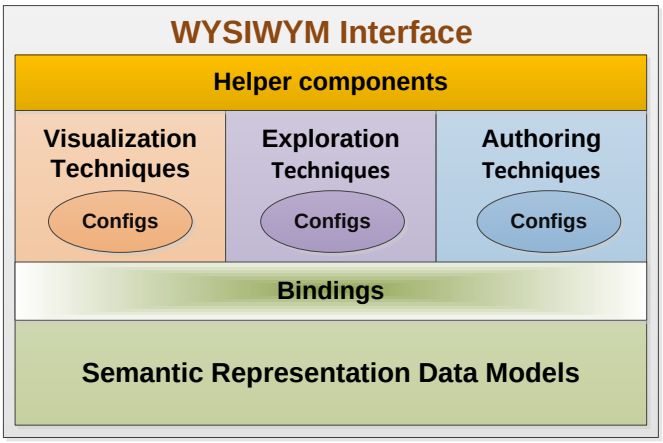


Figure 4.1.: Schematic view of the WYSIWYM model.

for software engineers and UI designers to communicate efficiently and effectively. It provides insights into and an understanding of the requirements as well as corresponding UI solutions for proper design and implementation of semantic content management applications. Secondly, it allows evaluating and classifying existing user interfaces according to the conceptual model in a defined way. This will highlight the gaps in existing applications dealing with semantically enriched documents and will help to optimize them based on the defined requirements.

Figure 4.1 provides a schematic overview of the WYSIWYM concept. The rationale is that elements of a knowledge representation formalism (or data model) are connected to suitable UI elements for visualization, exploration and authoring. Formalizing this conceptual model results in three core definitions (1) for the abstract *WYSIWYM model*, (2) *bindings* between UI and representation elements as well as (3) a concrete instantiation of the abstract WYSIWYM model, which we call a *WYSIWYM interface*.

**Definition 1** (WYSIWYM model)**.** *The WYSIWYM model can be formally defined as a quintuple* $(D, V, X, T, H)$ *where:*

- $D$ *is a set of semantic representation data models, where each* $D_i \in D$ *has an associated set of data model elements* $E_{D_i}$*;*
- $V$ *is a set of tuples* $(v, C_v)$*, where* $v$ *is a visualization technique and* $C_v$ *a set of possible configurations for the visualization technique* $v$*;*
- $X$ *is a set of tuples* $(x, C_x)$*, where* $x$ *is an exploration technique and* $C_x$ *a set of possible configurations for the exploration technique* $x$*;*
- $T$ *is a set of tuples* $(t, C_t)$*, where* $t$ *is an authoring technique and* $C_t$ *a set of possible configurations for the authoring technique* $t$*;*
- $H$ *is a set of helper components.*

*Semantic representation data models* are techniques to define the meaning of data within the context of its interrelationships with other data (cf. Section 4.2.1). Tree, Graph and Hypergraph are examples of commonly used data models. *Visualization techniques* include UI techniques for highlighting, associating and detail viewing of semantic entities (cf. Section 4.2.2). *Exploration techniques* include UI techniques for efficient browsing and navigating semantic data (cf. Section 4.2.3). *Authoring techniques* include UI techniques for adding and editing semantic entities and their relations (cf. Section 4.2.4). *Helper components* are cross-cutting aspects to enhance and customize the user/application requirements of a WYSIWYM interface (cf. Section 4.2.6).

The WYSIWYM model represents an abstract concept from which concrete interfaces can be derived by means of bindings between semantic representation model elements and configurations of particular UI elements.

**Definition 2** (Binding)**.** *A binding b is a function which maps each element of a semantic representation model e ($e \in E_{D_i}$) to a set of tuples $(ui, c)$, where ui is a user interface technique ui ($ui \in V \cup X \cup T$) and c is a configuration $c \in C_{ui}$.*

Figure 4.4 gives an overview on all data model (columns) and UI elements (rows) and how they can be bound together using a certain configuration (cells). The shades of gray in a certain cell indicate the suitability of a certain binding between a particular UI and data model element.

For example, having *tree-based* semantic representation model, *framing and segmentation* UI techniques can be used as external augmentation to *visualize* the *items* in the *text*. It is also possible to use *text formatting* techniques as inline augmentation for highlighting the items but since they might interfere with the current text format, we assume a partial binding for them. A possible configuration for this example binding is to set different border and text colors to distinguish different item types.

Once a selection of data models and UI elements was made and both are bound to each other encoding a certain configuration in a binding, we attain a concrete instantiation of our WYSIWYM model called WYSIWYM interface.

**Definition 3** (WYSIWYM interface)**.** *An instantiation of the WYSIWYM model I called WYSIWYM interface now is a hextuple $(D_I, V_I, X_I, T_I, H_I, b_I)$, where:*

- *$D_I$ is a selection of semantic representation data models ($D_I \subset D$);*
- *$V_I$ is a selection of visualization techniques ($V_I \subset V$);*
- *$X_I$ is a selection of exploration techniques ($X_I \subset X$);*
- *$T_I$ is a selection of authoring techniques ($T_I \subset T$);*

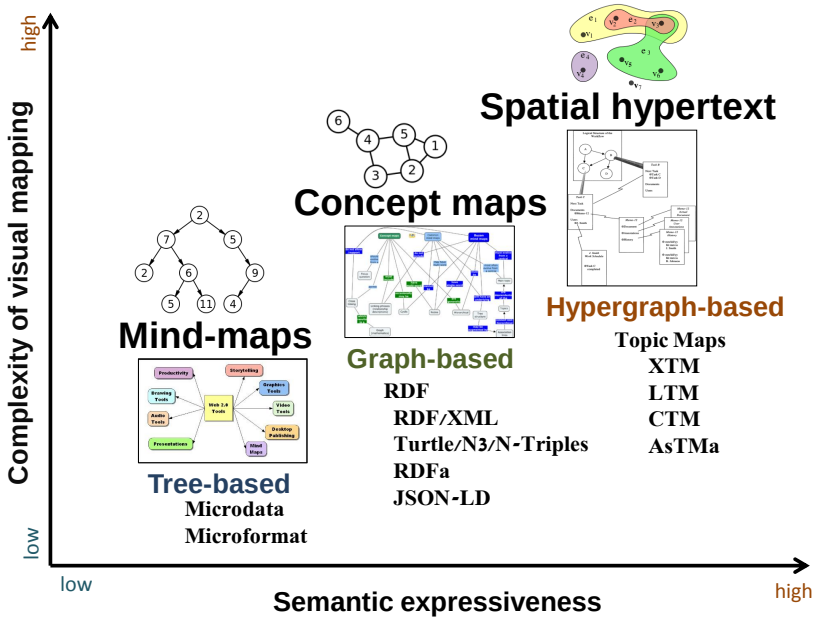


Figure 4.2.: Comparison of existing visual mapping techniques in terms of semantic expressiveness and complexity of visual mapping.

- *$H_I$ is a selection of helper components ($H_I \subset H$);*
- *$b_I$ is a binding which binds a particular occurrence of a data model element to a visualization, exploration and/or authoring technique.*

Note, that we limit the definition to one binding, which means that only one semantic representation model is supported in a particular WYSIWYM interface at a time. It could be also possible to support several semantic representation models (e.g. RDFa and Microdata) at the same time. However, this can be confusing to the user, which is why we deliberately excluded this case in our definition. In the remainder of this sections we discuss the different parts of the WYSIWYM concept in more detail.

### 4.2.1. Semantic Representation Models

Semantic representation models are conceptual data models to express the meaning of information thereby enabling representation and interchange of knowledge. Based on their expressiveness, we can roughly divide popular semantic representation models into the three categories *tree-based*, *graph-based* and *hypergraph-based* (cf. Figure 4.2). Each semantic representation model comprises a number of representation elements, such as various types of entities and relationships. For visualization, exploration and authoring it is of paramount importance to bind the

most suitable UI elements to respective representation elements. In the sequel we briefly discuss the three different types of representation models.

**Tree-based.** This is the simplest semantic representation model, where semantics is encoded in a tree-like structure. It is suited for representing taxonomic knowledge, such as thesauri, classification schemes, subject heading lists, concept hierarchies or mind-maps. It is used extensively in biology and life sciences, for example, in the *APG III system* (Angiosperm Phylogeny Group III system) of flowering plant classification, as part of the Dimensions of the *XBRL* (eXtensible Business Reporting Language)[4] or generically in the *SKOS*[5]. Elements of tree-based semantic representation usually include:

- $E_1$: Item – e.g. `Magnoliidae`, the item representing all flowering plants.
- $E_2$: Item type – e.g. `biological term` for `Magnoliidae`.
- $E_3$: Item-subitem relationships – e.g. `Magnoliidae` referring to subitem `magnolias`.
- $E_4$: Item property value – e.g. the synonym `flowering plant` for the item `Magnoliidae`.
- $E_5$: Related items – e.g. the sibling item `Eudicots` to Magnoliidae.

Tree-based data can be serialized as Microdata or Microformats.

**Graph-based.** This semantic representation model adds more expressiveness compared to simple tree-based formalisms. The most prominent representative is the *RDF* data model (cf. Section 2.2), which can be seen as a set of triples consisting of subject, predicate, object, where each component can be a URI, the object can be a literal and subject as well as object can be a blank node. The most distinguishing features of RDF from a simple tree-based model are: the distinction of entities in classes and instances as well as the possibility to express arbitrary relationships between entities. The graph-based model is suited for representing combinatorial schemes such as concept maps. Graph-based models are used in a very broad range of domains, for example, in the *FOAF*[6] for describing people, their interests and interconnections in a social network, in *MusicBrainz*[7] to publish information about music albums, in the medical domain (e.g. DrugBank, Diseasome, ChEMBL, SIDER) to describe the relations between diseases, drugs and genes, or generically in the *SIOC* vocabulary[8]. Elements of RDF as a typical graph-based data model are:

[4] www.xbrl.org/
[5] http://www.w3.org/2004/02/skos/
[6] http://www.foaf-project.org
[7] https://musicbrainz.org/
[8] http://rdfs.org/sioc/spec/

- $E_1$: Instances – e.g. `Warfarin` as a drug.
- $E_2$: Classes – e.g. `anticoagulants drug` for Warfarin.
- $E_3$: Relationships between entities (instances or classes) – e.g. the `interaction` between `Aspirin` as an antiplatelet drug and `Warfarin` which will increase the risk of bleeding.
- $E_4$: Literal property values – e.g. the halflife for the Amoxicillin.
  - $E_{4_1}$: Value – e.g. `61.3` minutes.
  - $E_{4_2}$: Language tag – e.g. `en`.
  - $E_{4_3}$: Datatype – e.g. `xsd:float`.

RDF-based data can be serialized in various formats, such as RDFa, RDF/XML, JSON-LD or Turtle/N3/N-Triples.

**Hypergraph-based.** A hypergraph is a generalization of a graph in which an edge can connect any number of vertices. Since hypergraph-based models allow n-ary relationships between arbitrary number of nodes, they provide a higher level of expressiveness compared to tree-based and graph-based models. The most prominent representative is the *Topic Maps* data model developed as an ISO/IEC standard which consists of topics, associations and occurrences. The semantic expressivity of Topic Maps is, in many ways, equivalent to that of RDF, but the major differences are that Topic Maps (i) provide a higher level of semantic abstraction (providing a template of topics, associations and occurrences, while RDF only provides a template of two arguments linked by one relationship) and (hence) (ii) allow n-ary relationships (hypergraphs) between any number of nodes, while RDF is limited to triplets. The hypergraph-based model is suited for representing complex schemes such as spatial hypertext. Hypergraph-based models are used for a variety of applications. Amongst them are *musicDNA*[9] as an index of musicians, composers, performers, bands, artists, producers, their music, and the events that link them together, *TM4L* (Topic Maps for e-Learning), clinical decision support systems and enterprise information integration. Elements of Topic Maps as a typical hypergraph-based data model are:

- $E_1$: Topic name – e.g. `University of Leipzig`.
- $E_2$: Topic type – e.g. `organization` for University of Leipzig.
- $E_3$: Topic associations – e.g. `member of a project` which has other organization partners.
- $E_4$: Topic role in association e.g. `coordinator`.

[9] http://www.musicdna.info/


Figure 4.3.: Screenshots of user interface techniques for visualization and exploration: 1-framing using borders, 2-framing using backgrounds, 3-video subtitle, 4-line connectors and arrow connectors, 5-bar layouts, 6-text formatting, 7-image color effects, framing and line connectors, 8-expandable callout, 9-marking with icons, 10-tooltip callout, 11-faceting

- $E_5$: Topic occurrences – e.g. `address`.
  - $E_{5_1}$: value – e.g. `Augustusplatz 10, 04109 Leipzig`.
  - $E_{5_2}$: datatype – e.g. `text`.

Topic Maps-based data can be serialized as an XML-based syntax called XTM (XML Topic Map)[10], LTM (Linear Topic Map Notation)[11], CTM (Compact Topic Maps Notation)[12] and AsTMa (Asymptotic Topic Map Notation)[13].

### 4.2.2. Visualization

The primary objectives of visualization are to present, transform, and convert semantic data into a visual representation, so that, humans can read, query and

[10] www.topicmaps.org/xtm/1.0/
[11] http://www.ontopia.net/download/ltm.html
[12] http://www.isotopicmaps.org/ctm/
[13] http://astma.it.bond.edu.au/

edit them efficiently. We divide existing techniques for visualization of knowledge encoded in text, images and videos into the three categories *Highlighting*, *Associating* and *Detail view*. Highlighting includes UI techniques which are used to distinguish or highlight a part of an object (i.e. text, image or video) from the whole object. Associating deals with techniques that visualize the relation between some parts of an object. Detail view includes techniques which reveal detailed information about a part of an object. For each of the above categories, the related UI techniques are as follows:

**- Highlighting.**

- $V_1$: *Framing and Segmentation* (borders, overlays and backgrounds). This technique can be applied to text, images and videos, we enclose a semantic entity in a colored border, background or overlay. Different border styles (colors, width, types), background styles (colors, patterns) or overlay styles (when applied to images and videos) can be used to distinguish different types of semantic entities (cf. Figure 4.3 no. 1, 2). The technique is already employed in social networking websites such as *Google Plus* and *Facebook* to tag people within images.

- $V_2$: *Text formatting* (color, font, size, margin, etc.). In this technique different text styles such as font family, style, weight, size, color, shadows, margin and other text decoration techniques are used to distinguish semantic entities within a text (cf. Figure 4.3 no. 6). The problem with this technique is that in an HTML document, the applied semantic styles might overlap with existing styles in the document and thereby add ambiguity to recognizing semantic entities.

- $V_3$: *Image color effects*. This technique is similar to text formatting but applied to images and videos. Different image color effects such as brightness/contrast, shadows, glows, bevel/emboss are used to highlight semantic entities within an image (cf. Figure 4.3 no. 7). This technique suffers from the problem that the applied effects might overlap with the existing effects in the image thereby making it hard to distinguish the semantic entities.

- $V_4$: *Marking* (icons appended to text or image). In this technique, which can be applied to text, images and videos, we append an icon as a marker to the part of object which includes the semantic entity (cf. Figure 4.3 no. 9). The most popular use of this technique is currently within maps to indicate specific points of interest. Different types of icons can be used to distinguish different types of semantic or correlated entities.

- $V_5$: *Bleeping*. A bleep is a single short high-pitched signal in videos. Bleeping can be used to highlight semantic entities within a video. Different type of bleep signals can be defined to distinguish different types of semantic entities.

- $V_6$: *Speech* (in videos). In this technique a video is augmented by some speech indicating the semantic entities and their types within the video.

**- Associating.**

- $V_7$: *Line connectors*. Using line connectors is the simplest way to visualize the relation between semantic entities in text, images and videos (cf. Figure 4.3 no. 4). If the value of a property is available in the text, line connectors can also reflect the item property values. Problematic is that normal line connectors cannot express the direction of a relation.

- $V_8$: *Arrow connectors*. Arrow connectors are extended line connectors with arrows to express the direction of a relation in a directed graph.

Besides the line and arrow connectors techniques which explicitly visualize the association between entities, implicit techniques defined as *Gestalt principles* [Johnson, 2014] can be used for modeling association. These techniques are psychological assumptions that impose structure for human visual perception. Principles such as proximity, similarity, continuity, closure, symmetry, figure/ground and common fate can be used to affect our perception of whether and how the objects are organized into groups. Discussing these principles are out of the scope of this work.

**- Detail view.**

- $V_9$: *Callouts*. A callout is a string of text connected by a line, arrow, or similar graphic to a part of text, image or video giving information about that part. It is used in conjunction with a cursor, usually a pointer. The user hovers the pointer over an item, without clicking it, and a callout appears (cf. Figure 4.3 no. 10). Callouts come in different styles and templates such as infotips, tooltips, hint and popups. Different sort of metadata can be embedded in a callout to indicate the type of semantic entities, property values and relationships. Another variant of callouts is the *status bar* which displays metadata in a bar appended to the text, image or video container. A problem with dynamic callouts is that they do not appear on mobile devices (by hover), since there is no cursor.

- $V_{10}$: *Video subtitles*. Subtitles are textual versions of the dialog or commentary in videos. They are usually displayed at the bottom of the screen and are employed for written translation of a dialog in a foreign language. Video subtitles can be used to reflect detailed semantics embedded in a video scene when watching the video. A problem with subtitles is efficiently scaling the text size and relating text to semantic entities when several semantic entities exist in a scene.

### 4.2.3. Exploration

To increase the effectiveness of visualizations, users need to be capable to dynamically navigate and explore the visual representation of the semantic data. The dynamic exploration of semantic data will result in faster and easier comprehension of the targeted content. Techniques for exploration of semantics encoded in text, images and videos include:

- $X_1$: *Zooming*. In a zoomable UI, users can change the scale of the viewed area in order to see more detail or less. The zooming elements and techniques vary on different applications. Zooming in a semantic entity can reveal further details such as property value or entity type. Zooming out can be employed to reveal the relations between semantic entities in a text, image or video. Supporting rich dynamics by configuring different visual representations for semantic objects at different sizes is a requirement for a zoomable UI. The *iMapping* approach[Haller and Abecker, 2010] which is implemented in the semantic desktop is an example of the zooming technique.

- $X_2$: *Faceting*. Faceted browsing is a technique for accessing information organized according to a faceted classification system, allowing users to explore a collection of information by applying multiple filters (cf. Figure 4.3 no. 11). Defining facets for each component of the predefined semantic models enable users to browse the underlying knowledge space by iteratively narrowing the scope of their quest in a predetermined order. One of the main problems with faceted browsers is the increased number of choices presented to the user at each step of the exploration [Deligiannidis et al., 2007].

- $X_3$: *On-demand highlighting*. Unlike the highlighting approach discussed in the visualization methods, on-demand highlighting is used to navigate the semantic entities encoded in text in a dynamic manner. One technique to realize on-demand highlighting is *Bar layout*. In the bar layout, each semantic entity within the text is indicated by a vertical bar in the left or right margin (cf. Figure 4.3 no. 5). The colour of the bar reflects the type of the entity. The bars are ordered by length and order in the text. Nested bars can be used to show the hierarchies of entities. Semantic entities in the text are highlighted by a mouse-over the corresponding bar. This approach is employed in Loomp [Luczak-Roesch, 2009].

- $X_4$: *Expanding & Drilling down*. Expandable callouts are interactive and dynamic callouts which enable users to explore the semantic data associated to a predefined semantic entity (cf. Figure 4.3 no. 8). Drilling down in a callout enables users to move from summary information to detailed data by focusing in on entities. This technique is employed in *OntosFeeder* [Klebeck et al., 2011].

### 4.2.4. Authoring

Semantic authoring aims to add more meaning to digitally published documents. If users do not only publish the content, but at the same time describe what it is they are publishing, then they have to adopt a structured approach to authoring. A semantic authoring UI is a human accessible interface with capabilities for writing and modifying semantically enriched documents. The following techniques can be used for authoring of semantics encoded in text, images and videos:

- $T_1$: *Form editing.* In form editing, a user employs existing form elements such as input/check/radio boxes, drop-down menu, slider, spinner, buttons, and date/color picker, etc. for content authoring.
- $T_2$: *Inline edit.* Inline editing is the process of editing items directly in the view by performing simple clicks, rather than selecting items and then navigating to an edit form and submitting changes from there.
- $T_3$: *Drawing.* Drawing as part of informal user interfaces [Lin et al., ], provides a natural human input to annotate an object by augmenting the object with human-understandable sketches. For instance, users can draw a frame around semantic entities; draw a line between related entities etc. Special shapes can be drawn to indicate different entity types or entity roles in a relation.
- $T_4$: *Drag and drop.* Drag and drop is a pointing device gesture in which the user selects a virtual object by grabbing it and dragging it to a different location or onto another virtual object. In general, it can be used to invoke many kinds of actions, or create various types of associations between two abstract objects.
- $T_5$: *Context menu.* A context menu (also called contextual, shortcut, or pop-up menu) is a menu that appears upon user interaction, such as a right button mouse click. A context menu offers a limited set of choices that are available in the current state, or context.
- $T_6$: *(Floating) Ribbon editing.* A ribbon is a command bar that organizes functions into a series of tabs or toolbars at the top of the editable content. Ribbon tabs/toolbars are composed of groups, which are a labeled set of closely related commands. A floating ribbon is a ribbon that appears when user rolls the mouse over a target area. A floating ribbon increases usability by bringing edit functions as close as possible to the user's point of focus. The *Aloha WYSIWYG editor*[14] is an example of floating ribbon based content authoring.

[14] http://aloha-editor.org

- $T_7$: *Voice commands.* Voice commands permit the user's hands and eyes to be busy with another task, which is particularly valuable when users are in motion or outside. Users tend to prefer speech for functions like describing objects, sets and subsets of objects [Oviatt et al., 2000]. By adding special signals to input voice, users can author semantic content from the scratch.

- $T_8$: *(Multi-touch) gestures.* A gesture is a form of non-verbal communication in which visible bodily actions communicate particular messages. Technically, different methods can be used for detecting and identifying gestures. Movement-sensor-based and camera-based approaches are two commonly used methods for the recognition of in-air gestures [Loecken et al., 2012]. Multi-touch gestures are another type of gestures, which are defined, to interact with multi-touch devices such as modern smartphones and tablets. Users can use gestures to determine semantic entities, their types and relationship among them. The main problem with gestures is their high level of abstraction, which makes it hard to assert concrete property values. Special gestures can be defined to author semantic entities in text, images and videos.

### 4.2.5. Bindings

Figure 4.4 surveys possible bindings between the user interface and semantic representation elements.

The bindings were derived based on the following methodology:

1. We first analyzed existing semantic representation models and extracted the corresponding elements for each semantic model.

2. We performed an extensive literature study regarding existing approaches for visual mapping as well as approaches addressing the binding between data and UI elements. If the approach was explicitly mentioning the binding composed of UI elements and semantic model elements, we added the binding to our mapping table.

3. We analyzed existing tools and applications which were implicitly addressing the binding between data and UI elements.

4. Finally, we followed a predictive approach. We investigated additional UI elements which are listed in existing HCI glossaries and carefully analyzed their potential to be connected to a semantic model element.

Although we deem the bindings to be fairly complete, new UI elements might be developed or additional data models (or variations of the ones considered) might appear, in this case the bindings can be easily extended.

*Partial* binding indicates the situation when a UI technique does not completely cover a semantic model element but still can be used in particular cases. For example, different text colors can be used to highlight predefined item types in

text but since the colors might interfere with the current colors in the text (in case of HTML document), we assign this binding as partial binding. Another example is the line connectors used to represent the relation between items in a tree or graph-based model. In this case, on the contrary to arrow connectors, since we cannot determine the source and destination of the line, we are unable to model directional relations completely, thereby, a partial binding is assigned.

The asterisks in Figure 4.4, indicate the cases when the metadata value is explicitly available in the text and the user just needs to provide the connection (e.g. imagine that we have `Berlin` and `Germany` mentioned in the text and we want to assign the relation `isCapitalOf`).

The following binding configurations (extracted from the literature and current tools) are available and referred to from the cells of Figure 4.4:

- Defining a special border or background style ($C_1$), text style ($C_2$), image color effect ($C_4$), beep sound ($C_5$), bar style ($C_6$), sketch ($C_7$), draggable or droppable shape ($C_8$), voice command ($C_9$), gesture ($C_{10}$) or a related icon ($C_3$) for each type.
- Progressive shading ($C_{11}$) by defining continuous shades within a specific color scheme to distinguish items in different levels of the hierarchy.
- Hierarchical bars ($C_{12}$) by defining special styles for nested bars.
- Grouping by similar border or background style ($C_{13}$), text style ($C_{14}$), icons ($C_{15}$) or image color effects ($C_{16}$).

For example, a user can define a set of preferred border colors to distinguish different item types (e.g. Persons, Organizations or Locations) or to group related items (e.g. all the cities in Germany).

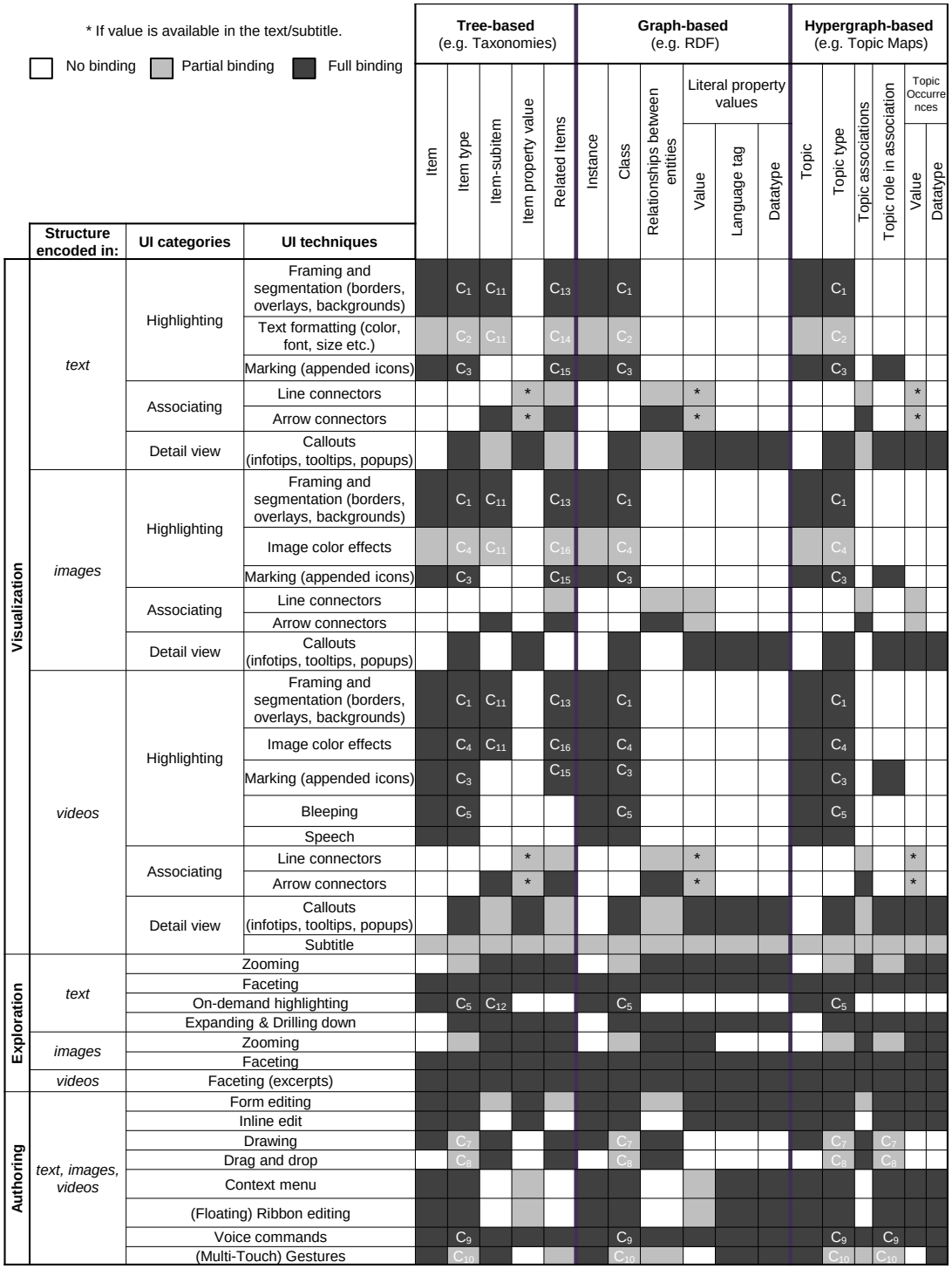


Figure 4.4.: Possible bindings between user interface and semantic representation model elements.

### 4.2.6. Helper Components

In order to facilitate, enhance and customize the WYSIWYM model, we utilize a set of helper components, which implement cross-cutting aspects. These helper components are mainly derived from the quality attributes discussed in Section 3.4. A helper component acts as an extension on top of the core functionality of the WYSIWYM model. The following components can be used to improve the quality of a WYSIWYM UI depending on the requirements defined for a specific application domain:

- $H_1$: *Automation* means the provision of facilities for automatic annotation of text, images and videos to reduce the need for human work and thereby facilitating the efficient annotation of large item collections. For example, users can employ existing NLP services (e.g. named entity recognition, relationship extraction) for automatic text annotation.

- $H_2$: *Real-time tagging* is an extension of automation, which allows to create annotations proactively while the user is authoring a text, image or video. This will significantly increase the annotation speed and users are not distracted since they do not have to interrupt their current authoring task.

- $H_3$: *Recommendation* means providing users with pre-filled form fields, suggestions (e.g. for URIs, namespaces, properties), default values etc. These facilities simplify the authoring process, as they reduce the number of required user interactions. Moreover, they help preventing incomplete or empty metadata. In order to leverage other user's annotations as recommendations, approaches like *Paragraph Fingerprinting* [Hong and Chi, ] can be implemented.

- $H_4$: *Personalization and context-awareness* describes the ability of the UI to be configured according to users' contexts, background knowledge and preferences. Instead of being static, a personalized UI dynamically tailors its visualization, exploration and authoring functionalities based on the user profile and context.

- $H_5$: *Collaboration and Crowdsourcing* enable collaborative semantic authoring, where the authoring process can be shared among different authors at different locations. There are a vast amounts of amateur and expert users which are collaborating and contributing on the Social Web. Crowdsourcing harnesses the power of such crowds to significantly enhance and widen the results of semantic content authoring and annotation. Generic approaches for exploiting single-user Web applications for shared editing [Heinrich et al., 2012] can be employed in this context.

- $H_6$: *Accessibility* means providing people with disabilities and special needs with appropriate UIs. The underlying semantic model in a WYSIWYM UI

can allow alternatives or conditional content in different modalities to be selected based on the type of the user disability and information need.

- $H_7$: *Multilinguality* means supporting multiple languages in a WYSIWYM UI when visualizing, exploring or authoring the content.

## 4.3. Conclusions

In this chapter, we discussed the WYSIWYM concept to answer the research question RQ2 (cf. Section 1.3) for bridging the gap between unstructured and semantic content. With the WYSIWYM concept we presented in this chapter an approach for integrated visualization, exploration and authoring of unstructured *and* semantic content. The WYSIWYM model binds elements of a knowledge representation formalism (or data model) to a set of suitable UI elements for visualization, exploration and authoring. Based on such a declarative binding mechanism, we aim to increase the flexibility, reusability and development efficiency of semantics-rich user interfaces.

Chapter 5

# From WYSIWYG to WYSIWYM

> "Progress is impossible without change, and those who cannot change their minds cannot change anything." — *George Bernard Shaw*

In this chapter we present the RDFaCE approach for enriching WYSIWYG text authoring with WYSIWYM approach discussed in Chapter 4 for the creation of rich semantic content. The contributions of this chapter are in particular:

1. An architecture and implementation of a semantic annotation and authoring environment called RDFaCE (RDFa Content Editor) based on different views on the semantic content including a WYSIWYM view.
2. An extensive evaluation of five automatic text annotation APIs wrt. precision and recall in the domains wiki, blog and news articles.
3. An approach for the combination of different text annotation services, that yields superior performance compared to each individual approach.
4. Three use cases of RDFaCE including semantic blogging, data journalism and search engine optimization.
5. An evaluation of the RDFaCE content authoring environment using a sizable user group and measuring subjective as well as objective usage characteristics.

The rest of this chapter is structured as follows: Section 5.1 provides an overview on the WYSIWYG concept. Section 5.2 introduces the RDFaCE as an implementation of the WYSIWYM concept. Section 5.3 presents the different views for semantic text authoring. In Section 5.4, the combination of different NLP APIs for bootstrapping the semantic annotation process is discussed. Section 5.5 introduces three use cases of RDFaCE. Section 5.6 reports on RDFaCE usability evaluation results. A comparison to the related work is presented in Section 5.7 and finally Section 5.8 concludes the chapter.[1]

## 5.1. WYSIWYG

The term *WYSIWYG* as an acronym for What-You-See-Is-What-You-Get is used in computing to describe a system in which content (text and graphics)

[1] The contents of this chapter have been published as [Khalili et al., 2012a, Khalili and Auer, 2013b].

displayed on-screen during editing appears in a form closely corresponding to its appearance when printed or displayed as a finished product. The first usage of the term goes back to 1974 in the print industry to express the idea that what the user sees on the screen is what the user gets on the printer. *Xerox PARC's Bravo* was the first WYSIWYG editor-formatter [Myers, 1998]. It was designed by Butler Lampson and Charles Simonyi who had started working on these concepts around 1970 while at Berkeley. Later on by the emergence of Web and HTML technology, the WYSIWYG concept was also utilized in Web-based text editors. The aim was to reduce the effort required by users to express the formatting directly as valid HTML markup. In a WYSIWYG editor users can edit content in a view which matches the final appearance of published content with respect to fonts, headings, layout, lists, tables, images and structure. Because using a WYSIWYG editor may not require any HTML knowledge, they are often easier for an average computer user to get started with. The first programs for building Web pages with a WYSIWYG interface were *Netscape Gold*, *Claris HomePage*, and *Adobe PageMill*.

WYSIWYG text authoring is meanwhile ubiquitous on the Web and part of most content creation and management workflows. It is part of CMSs, Weblogs, Wikis, product data management systems and online shops, just to mention a few. However, the WYSIWYG model has been criticized, primarily for the verbosity, poor support of semantics and low quality of the generated code and there have been voices advocating a change towards a *WYSIWYM* (What-You-See-Is-What-You-Mean) model [Spiesser and Kitchen, , Sauer, 2006].

## 5.2. RDFaCE (RDFa Content Editor)

RDFaCE is an implementation of the WYSIWYM concept to integrate the semantic annotation directly into the content creation process and to make the annotation as easy and non-intrusive as possible. This is achieved by accompanying the classical WYSIWYG and source views with views facilitating the semantic annotation. The rationale behind our transition from WYSIWYG to WYSIWYM is to provide an environment to the user, which she is sufficiently familiar with, but at the same time enables her to understand, access and work with semantic annotations.

The RDFaCE system architecture is depicted in Figure 5.1 and consists of three layers. The foundation layer on which we ground the RDFaCE plugin includes the *TinyMCE Rich Text Editor*[2]. This open source HTML editor was chosen because it is very flexible to extend and is used in many popular CMSs, Weblogs, Wikis, discussion forums, etc. Therefore, by focusing efforts on this one particular editor, it is possible to quickly propagate accessible semantic authoring practices to a number of other tools [Treviranus, 2008]. The RDFaCE implementation

[2] http://tinymce.moxiecode.com

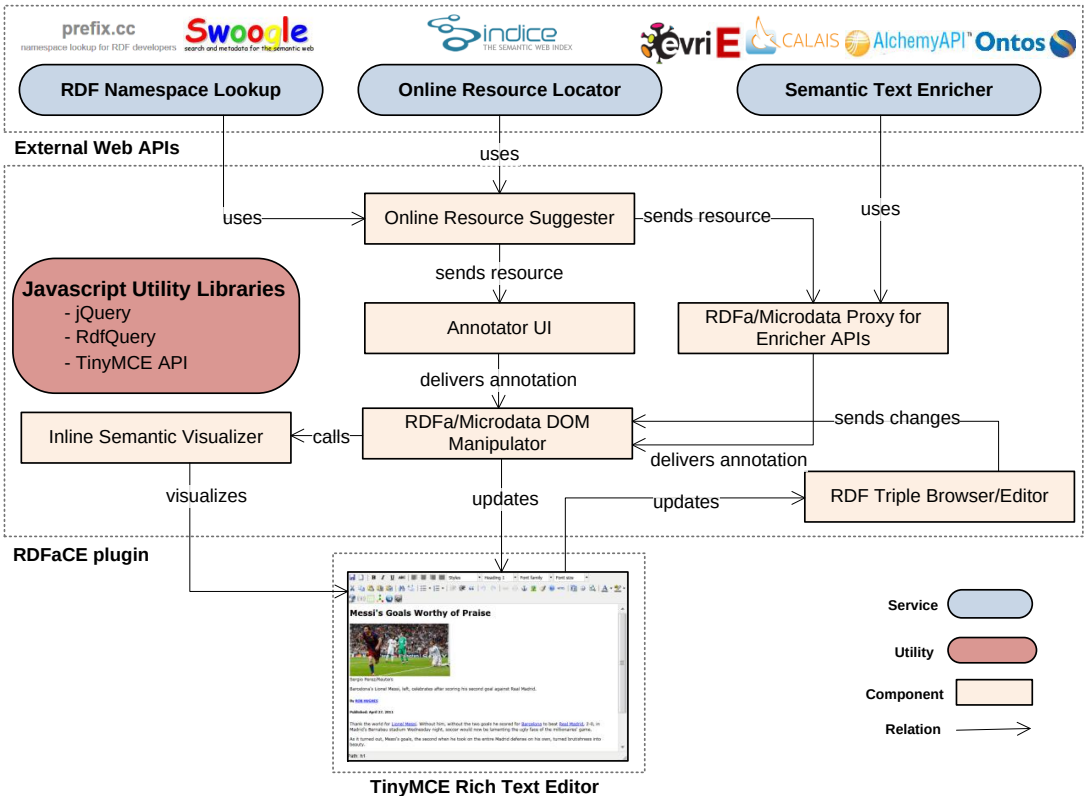


Figure 5.1.: RDFaCE system architecture.

is open-source and available for download together with an explanatory video and online demo at http://rdface.aksw.org. RDFaCE includes the following components:

**Annotator UI.** This component uses the TinyMCE API as well as jQuery UI to provide user friendly interfaces for semantic content editing based on RDFa and Microdata (cf. Section 3.3.1). As shown in Figure 5.2, the normal annotation procedure consists of four steps: 1) Defining appropriate namespaces. 2) Selecting a fragment of the text. 3) Assigning the subject (and type) to be used for the selected fragment. 4) Inserting triples by assigning properties. Besides these steps, RDFaCE provides some shortcuts to expedite the creation of new triples. For instance users can use the context menu and select from a list of predefined properties to instantly add a triple. After annotations are received by users, they are delivered to RDFa/Microdata Document Object Models (DOMs) manipulator.

**RDFa/Microdata DOM Manipulator.** This component is responsible for manipulating the Document Object Model (DOM) according to the desired RDFa or Microdata annotation. The simplest solution to add RDFa/Microdata attributes

to content is using `<span>` tags. For each new annotation, a new `<span>` can be created containing related RDFa/Microdata attributes. Although this approach is simple to implement it generates a lot of *redundant* `<span>` tags. It might also result in *invalid* HTML code when annotating a block of content which already has a `<div>` tag. This is due to the fact, that `div` is a block-level element whereas `span` is an inline element according to the HTML standard. To cope with these issues, the RDFa DOM Manipulator component tries to find the valid and optimized annotation which manipulates original content as minimally as possible. Before adding a new tag for annotation, it tries to see whether it is possible to add the annotation to an existing tag. If this is possible, it will update the current tag rather than adding a new HTML tag. In case a new tag is required, it also employs either `<span>` or `<div>` tags depending on whether the content is a block or inline element to prevent invalidness of HTML code.

**Inline Semantic Visualizer.** The main goal of the inline semantic visualizer is to provide a kind of on-demand visualization, which can be included/excluded on the fly within the WYSIWYG content editing. This component uses a set of predefined CSS styles to distinguish the semantically annotated content from the normal content. To visualize semantic annotations without modifying the content, dynamic style sheets are used. Different types of borders with different colors are used to present RDFa/Microdata annotated content which might be overlapping. To show the value of RDFa/Microdata attributes which are not visible in normal text, CSS tooltips are used. To prevent altering content, tooltips are created on the fly each time the user moves the mouse pointer over annotated content. Each time a new annotation is added by RDFa DOM manipulator, this component is called to visualize the editor.

**RDF Triple Browser and Editor.** This component extracts the RDF triples embedded in the text and provides the edit and delete functionality for these triples. This component is in a mutual relation to rich text editor and is dynamically updated when a new annotation is added to the text (also the text editor is updated when a triple is modified here). When user edits or deletes a triple, these changes are delivered to RDFa/Microdata DOM manipulator to update the content correspondingly.

**Online Resource Suggester.** This component provides the user with a set of accessible online resources. In order to perform this task, it accesses a number of external Web APIs. The Online Resource Suggester works in a close relation to Annotator UI. It facilitates the task of annotating content by searching the terms which are selected by user and suggesting corresponding URIs.

**RDFa/Microdata Proxy for Enricher APIs.** This component acts as a proxy to make the output of enricher APIs (i.e. NLP text annotation services) consumable as

RDFa. Most of the current text enricher APIs do not provide any RDFa/Microdata output. Therefore, we need to convert their generated output into RDFa/Microdata. To do this, the RDFa/Microdata proxy first sends the content to an external semantic text enrichment service. The output of the service is then converted to an standard format which includes label, URI, type, positions and properties related to the extracted entities. Then a mapping to a desired vocabulary is performed in order to make appropriate annotations. These annotations are delivered to the RDFa/Microdata DOM manipulator to update the content correspondingly. In case an URI is needed for an entity, the online resource suggester is used to assign an URI to the entity.

All the former components use Javascript utility libraries like jQuery and RDFQuery[3] to implement their functions. To facilitate semantic annotation of content, RDFaCE also uses a number of external Web APIs. Online APIs are invoked to carry out the following functions:

- *RDF Namespace Lookup*: In order to avoid that users have to type complete URIs, common namespace prefixes can be used everywhere in RDFaCE. These are looked-up using the `prefix.cc` service. Furthermore, in case users want to add a new property for which they do not even know an appropriate vocabulary, RDFaCE can look-up an appropriate vocabulary and property resource using the Swoogle[4] service.

- *Online Resource Locating*: Finding an appropriate URI for the resources which are selected by users can facilitate annotation process to a good extend. When users select a part of the text and want to create a statement about the respective entity, the online resource locator will do a *Sindice*[5] search to find suitable resources that match with the users selected item.

- *Semantic Text Enrichment*: Starting to annotate a document from the scratch is very tedious and time consuming. There are already some Natural Language Processing (NLP) APIs available on the Web which extract specific entities and relations from the text. By using these APIs, we can provide a good starting point for further user annotations. Users then can modify and extend these automatically pre-annotated content. RDFaCE currently uses the *OpenCalais*, *Ontos*, *Alchemy*, *Extractiv* and *Evri* NLP APIs[6] to enrich the text.

[3] http://code.google.com/p/rdfquery/

[4] http://swoogle.umbc.edu/

[5] http://sindice.com/

[6] These Web APIs are available at: OpenCalais - http://www.opencalais.com, Ontos - http://www.ontos.com, Alchemy - http://www.alchemyapi.com, Extractiv - http://extractiv.com and Evri - http://www.evri.com

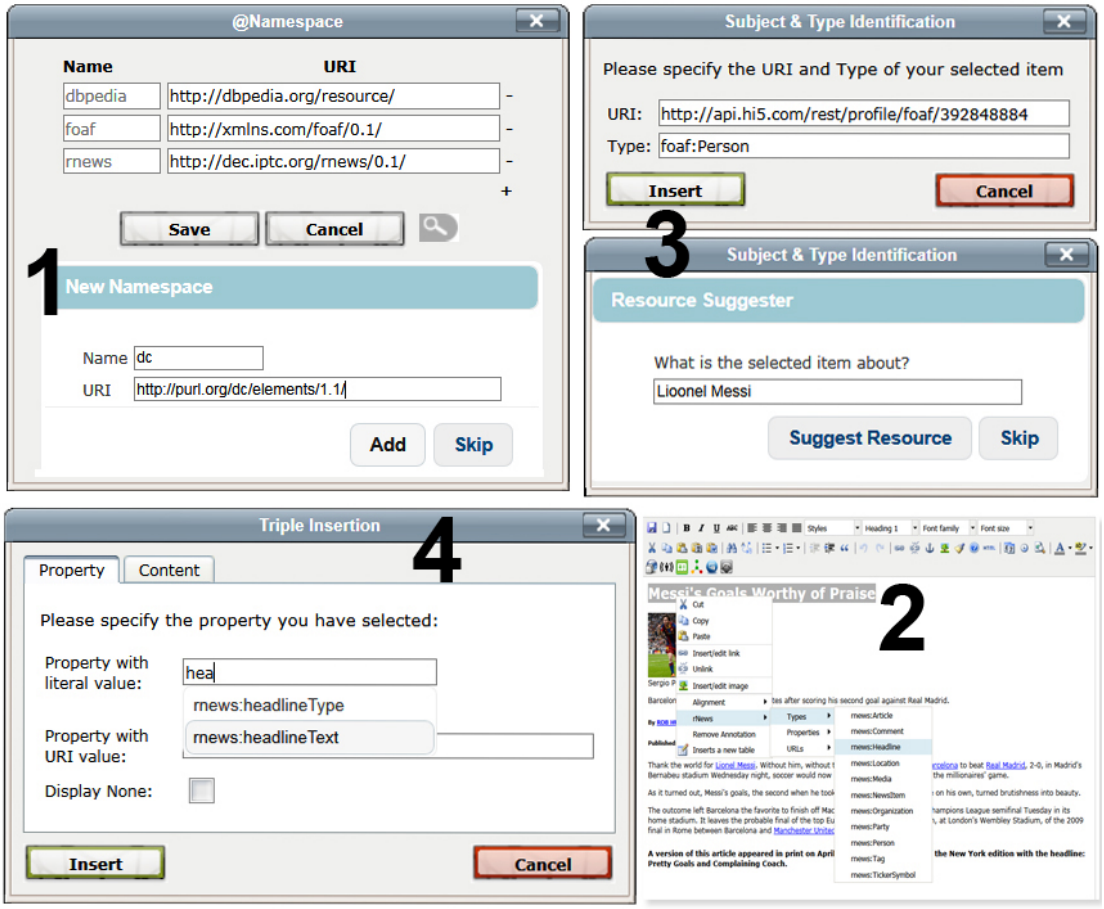


Figure 5.2.: Annotation user interface.

## 5.3. Views for Semantic Text Authoring

The main innovation of RDFaCE is the support of different views on the semantically annotated content. RDFaCE supports four different views for semantic text authoring, which are shown in Figure 5.3 and explained in more detail in the sequel. The user can easily switch between these views and even use them in parallel. The views are synchronized so that applying changes in one of the views automatically updates other views.

**WYSIWYG View** The What-You-See-Is-What-You-Get view as discussed in Section 5.1 is the classical interface for rich-text authoring and used by authors, journalists etc. WYSIWYG text authoring is meanwhile ubiquitous on the Web and part of most content creation and management workflows. Users authoring content are used to interact with a WYSIWYG views and there exists a wide variety of WYSIWYG editors and editing components, which can be used on the

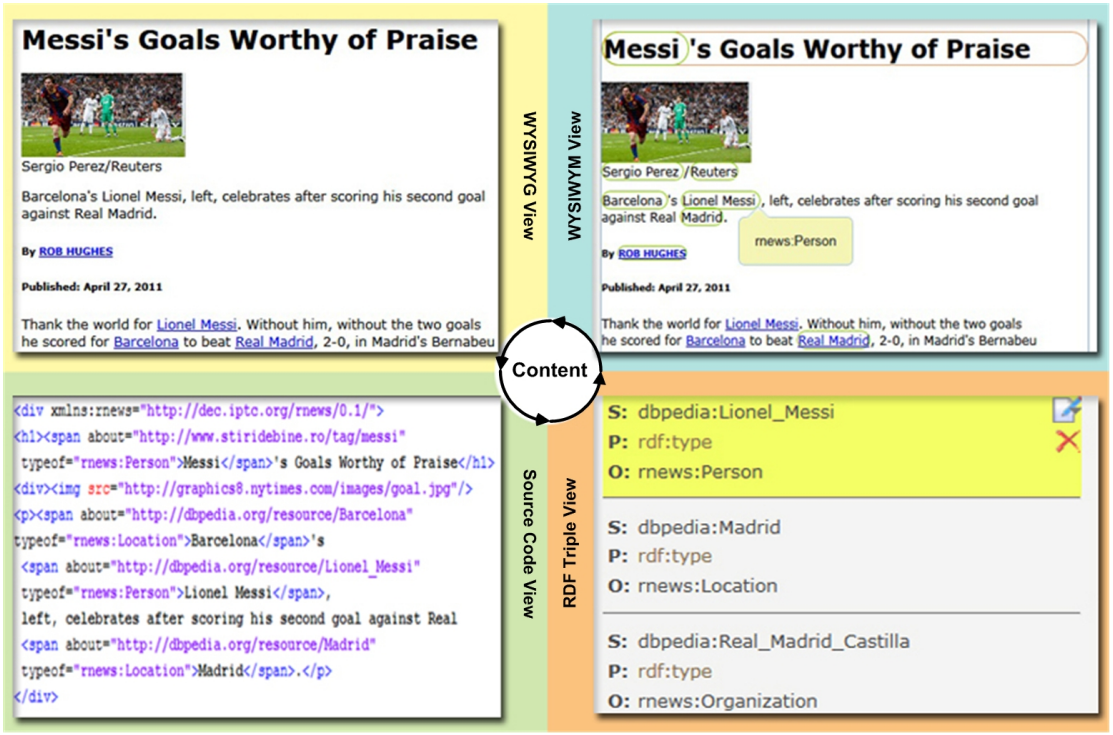


Figure 5.3.: The four views for semantic text authoring.

Web or offline.

**WYSIWYM View** RDFaCE employs the What-You-See-Is-What-You-Mean model described in Chapter 4 as an extension of the WYSIWYG view, which highlights named entities and other semantic information. Figure 5.4 shows a screenshot of different UI bindings employed in RDFaCE. The highlighting is realized with special CSS3 selectors for the RDFa/Microdata annotations. They are thus easily configurable in terms of color borders, backgrounds etc. When pointing with the mouse on a highlighted annotation RDFaCE shows additional information concerning the particular annotation as a tooltip. RDFaCE also supports editing in the WYSIWYM view by letting a user select entities she wants to annotate and provisioning of respective annotation functionality either via the context menu or a specific form, which opens as an overlay.

**RDF Triple View** This view summarizes all the facts, which can be extracted from the annotated text. It provides a deeper semantic view when compared to WYSIWYM view. There might be some triples not visible in the WYSIWYM view (e.g. annotations hidden using the CSS `display:none` style) but visible in this view. Since the triple view reveals all the triples embedded in the text, it can be called as WYMIWYS (What-You-Mean-Is-What-You-See) view. The triple

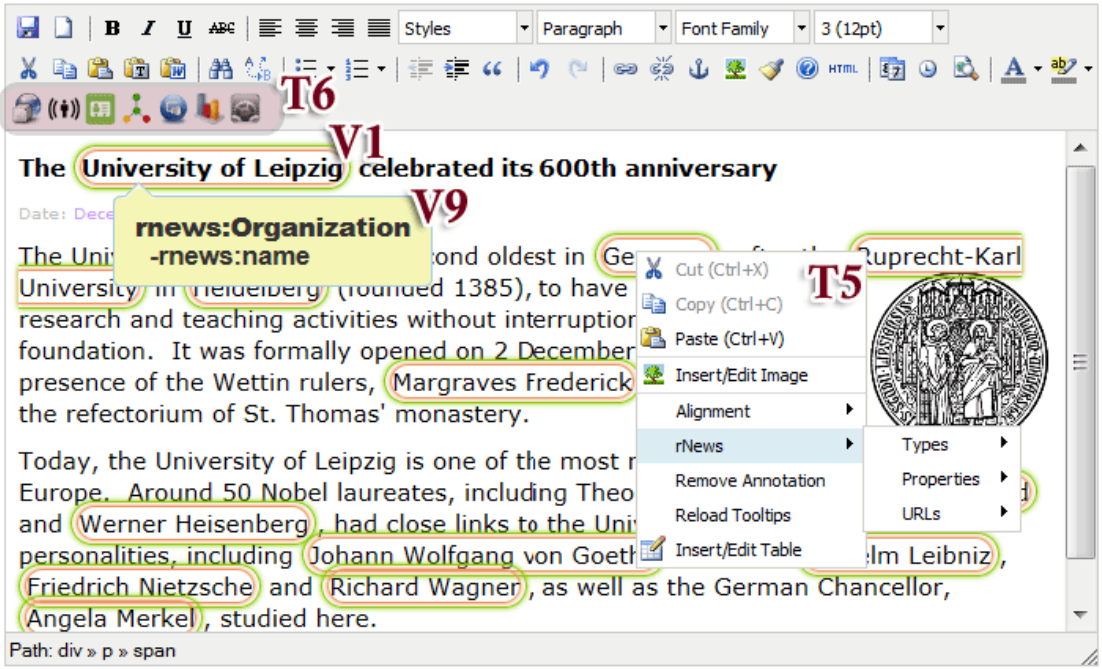


Figure 5.4.: RDFaCE WYSIWYM implementation (T6 indicates the RDFaCE menu bar, V1 – the framing of named entities in the text, V9 – a callout showing additional type information, T5 – a context menu for revising annotations).

view is (as all other views) updateable, i.e. facts can be directly deleted, which results in the removal of the corresponding RDFa/Microdata annotations. The triple view is useful for curators and to a lesser extend for the authors for quickly verifying that entities and facts were correctly annotated.

**Source Code View** Finally, the source code view shows the HTML source of the article including the RDFa/Microdata annotations. This view is primarily intended for software engineers supervising the publication workflow as well as knowledge engineers. Since all formating and interactive functionality (e.g. tooltips) is integrated via dynamic linking of CSS3 stylesheets with special selectors for the RDFa annotations, the source code view is not spoiled with any markup related to the WYSIWYM visualization.

## 5.4. Combining NLP-API results

One of the main features supported by RDFaCE is combining the results of multiple NLP APIs. Using this approach, we can harness synergies arising from the combination of different approaches for automatic text annotation. Users can

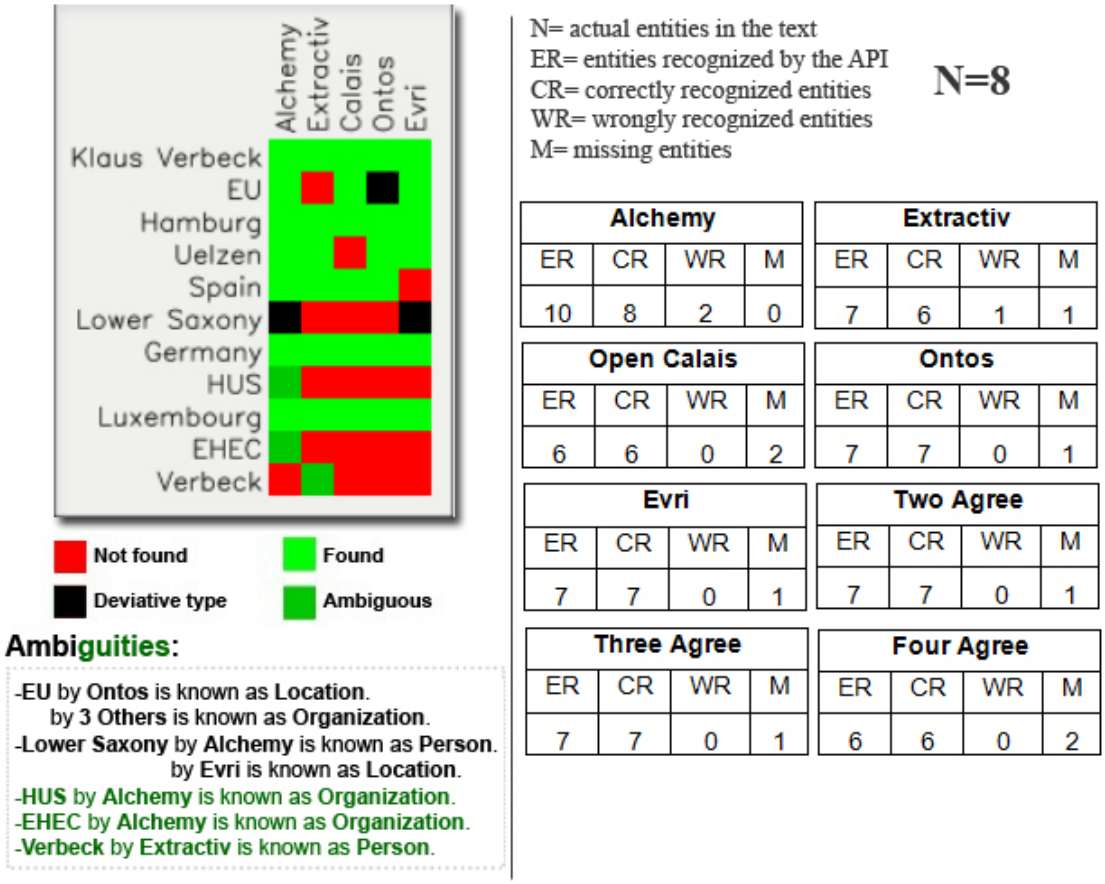


Figure 5.5.: Generated results of different NLP APIs for article #1.

select a set of NLP APIs and determine how they want to combine the results. The combination can be performed based on the agreement between two or more of the involved APIs.

Figure 5.5 shows the annotation results of the 5 different NLP APIs Alchemy, Extractive, OpenCalais, Ontos and Evri for a sample text. On the left, a heatmap visualization reflects the list of items recognized by each API. Black and dark green cells indicate cases that need disambiguation. Black cells indicate that there is a conflict between two or more APIs when recognizing the type of a common entity. In this case we have to investigate what the correct type is. Dark green cells indicate that an entity is recognized only by one API. In this case, the error probability is high and further investigation is required.

We use *Precision*,*Recall* and *F-measure* [Makhoul et al., 1999] as metrics for evaluating the correctness of the recognized entities found by each API as well as combined APIs:

$$Recall = \frac{Correctly\ Recognized\ Entities}{Actual\ Entities\ in\ the\ Text} \tag{5.1}$$

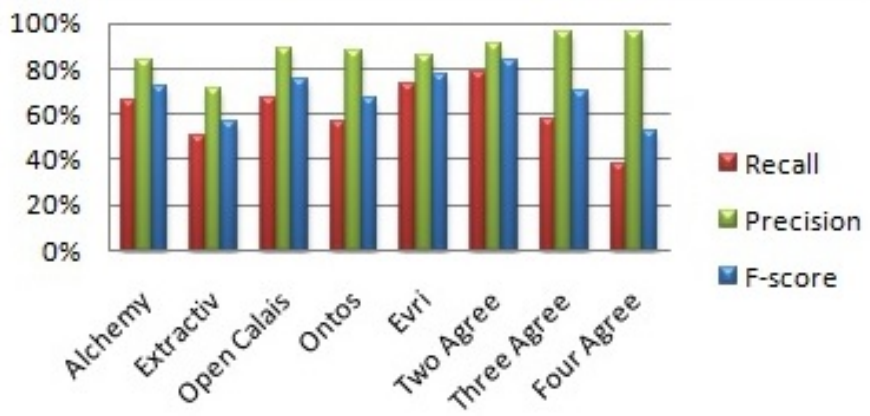


Figure 5.6.: Avg. Precision, Recall and F-score for each API & their combination.

$$Precision = \frac{Correctly\ Recognized\ Entities}{Entities\ Recognized\ by\ the\ API} \tag{5.2}$$

$$F = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{5.3}$$

To compare the results of the different APIs, 31 articles[7] were collected in the three categories news articles, Weblog posts and Wikipedia articles. For each article, the following analysis was performed:

We carefully analyzed the text and manually annotated it by recognizing references to location, person and organization entities. As a result we obtained a list of actual entities together with their types. Then we used the RDFaCE enrichment feature to automatically annotate the text by employing the external NLP APIs. By analyzing the RDFaCE generated heatmaps (cf. Figure 5.5) and the disambiguation of recognized entities, we extracted the number of recognized entities, correctly recognized, wrongly recognized and missing entities. Based on these values, Recall (5.1), Precision (5.2) and F-Score (5.3) were calculated for each API as well as for various combinations of APIs. Results of calculating these metrics for each API as well as for the situation when 2, 3 or 4 of the APIs agree on a recognized entity are shown in Table 5.1.

[7]Available together with the gold standard at http://rdface.aksw.org/samples/

Table 5.1.: Recall, Precision and F-score for each API and combined APIs.

| | **Alchemy** | | | **Extractive** | | | **OpenCalais** | | | **Ontos** | | | **Evri** | | | **2 Agree** | | | **3 Agree** | | | **4 Agree** | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Article | **R** | **P** | **F** | **R** | **P** | **F** | **R** | **P** | **F** | **R** | **P** | **F** | **R** | **P** | **F** | **R** | **P** | **F** | **R** | **P** | **F** | **R** | **P** | **F** |
| News#1 | 100 | 80 | 89 | 75 | 86 | 80 | 75 | 100 | 86 | 88 | 100 | 93 | 88 | 100 | 93 | 88 | 100 | 93 | 88 | 100 | 93 | 75 | 100 | 86 |
| News#2 | 88 | 70 | 78 | 88 | 58 | 70 | 100 | 89 | 94 | 88 | 88 | 88 | 100 | 73 | 84 | 100 | 89 | 94 | 88 | 100 | 93 | 50 | 100 | 67 |
| News#3 | 82 | 90 | 86 | 91 | 83 | 87 | 100 | 92 | 96 | 73 | 100 | 84 | 100 | 69 | 81 | 100 | 92 | 96 | 82 | 100 | 90 | 73 | 100 | 84 |
| News#4 | 78 | 95 | 86 | 57 | 72 | 63 | 83 | 100 | 90 | 65 | 100 | 79 | 74 | 85 | 79 | 78 | 95 | 86 | 74 | 100 | 85 | 57 | 100 | 72 |
| News#5 | 67 | 71 | 69 | 22 | 57 | 32 | 67 | 75 | 71 | 61 | 100 | 76 | 94 | 89 | 92 | 83 | 88 | 86 | 56 | 91 | 69 | 33 | 100 | 50 |
| News#6 | 76 | 87 | 81 | 53 | 64 | 58 | 76 | 93 | 84 | 47 | 89 | 62 | 76 | 93 | 84 | 88 | 88 | 88 | 71 | 100 | 83 | 35 | 100 | 52 |
| News#7 | 50 | 80 | 62 | 63 | 77 | 69 | 44 | 100 | 61 | 50 | 89 | 64 | 56 | 82 | 67 | 63 | 100 | 77 | 38 | 100 | 55 | 25 | 100 | 40 |
| News#8 | 79 | 92 | 85 | 71 | 91 | 80 | 86 | 100 | 92 | 86 | 100 | 92 | 93 | 100 | 96 | 100 | 100 | 100 | 79 | 100 | 88 | 64 | 100 | 78 |
| News#9 | 80 | 80 | 80 | 60 | 75 | 67 | 60 | 67 | 63 | 50 | 100 | 67 | 80 | 89 | 84 | 80 | 89 | 84 | 60 | 86 | 71 | 30 | 75 | 43 |
| News#10 | 73 | 100 | 84 | 18 | 67 | 29 | 73 | 73 | 73 | 100 | 92 | 96 | 73 | 89 | 80 | 91 | 100 | 95 | 64 | 100 | 78 | 45 | 100 | 63 |
| News#11 | 46 | 86 | 60 | 38 | 83 | 53 | 54 | 70 | 61 | 31 | 100 | 47 | 46 | 67 | 55 | 62 | 80 | 70 | 38 | 100 | 56 | 15 | 100 | 27 |
| Avg. | 74 | 84 | 78 | 58 | 74 | 62 | 74 | 87 | 79 | 67 | 96 | 77 | 80 | 85 | 81 | **85** | 93 | **88** | 67 | 98 | 78 | 46 | **98** | 60 |
| Blog#1 | 55 | 75 | 63 | 36 | 67 | 47 | 82 | 100 | 90 | 55 | 100 | 71 | 64 | 88 | 74 | 82 | 100 | 90 | 45 | 100 | 63 | 18 | 100 | 31 |
| Blog#2 | 53 | 75 | 62 | 65 | 85 | 73 | 35 | 100 | 52 | 12 | 67 | 20 | 59 | 91 | 71 | 71 | 92 | 80 | 35 | 86 | 50 | 6 | 100 | 11 |
| Blog#3 | 40 | 50 | 44 | 60 | 55 | 57 | 40 | 80 | 53 | 40 | 80 | 53 | 50 | 83 | 63 | 60 | 67 | 63 | 20 | 100 | 33 | 20 | 100 | 33 |
| Blog#4 | 14 | 20 | 17 | 14 | 14 | 14 | 29 | 50 | 36 | 57 | 80 | 67 | 29 | 33 | 31 | 29 | 40 | 33 | 14 | 0 | 0 | 0 | 0 | 0 |
| Blog#5 | 43 | 100 | 60 | 57 | 100 | 73 | 57 | 100 | 73 | 43 | 75 | 55 | 43 | 75 | 55 | 57 | 100 | 73 | 29 | 100 | 44 | 14 | 100 | 25 |
| Blog#6 | 75 | 92 | 83 | 38 | 67 | 48 | 69 | 100 | 81 | 75 | 92 | 83 | 81 | 100 | 90 | 100 | 100 | 100 | 81 | 100 | 90 | 63 | 100 | 77 |
| Blog#7 | 67 | 75 | 71 | 67 | 55 | 60 | 33 | 100 | 50 | 33 | 60 | 43 | 89 | 80 | 84 | 67 | 100 | 80 | 56 | 100 | 71 | 33 | 100 | 50 |
| Blog#8 | 86 | 100 | 92 | 57 | 89 | 70 | 79 | 100 | 88 | 50 | 78 | 61 | 57 | 89 | 70 | 86 | 92 | 89 | 71 | 100 | 83 | 57 | 100 | 73 |
| Blog#9 | 45 | 100 | 63 | 45 | 71 | 56 | 64 | 64 | 64 | 73 | 89 | 80 | 64 | 100 | 78 | 73 | 80 | 76 | 45 | 100 | 63 | 27 | 100 | 43 |
| Blog#10 | 70 | 78 | 74 | 40 | 57 | 47 | 80 | 100 | 89 | 50 | 100 | 67 | 90 | 100 | 95 | 100 | 100 | 100 | 80 | 100 | 89 | 40 | 100 | 57 |
| Avg. | 55 | 77 | 63 | 48 | 66 | 54 | 57 | 89 | 68 | 49 | 82 | 60 | 62 | 84 | 71 | **72** | 87 | **78** | 48 | 89 | 59 | 28 | **90** | 40 |
| Wiki#1 | 53 | 100 | 69 | 42 | 53 | 47 | 74 | 100 | 85 | 47 | 75 | 58 | 74 | 82 | 78 | 63 | 92 | 75 | 63 | 100 | 77 | 32 | 100 | 48 |
| Wiki#2 | 50 | 57 | 53 | 31 | 100 | 48 | 50 | 89 | 64 | 44 | 70 | 54 | 56 | 100 | 72 | 56 | 82 | 67 | 25 | 100 | 40 | 19 | 100 | 32 |
| Wiki#3 | 57 | 92 | 71 | 19 | 67 | 30 | 43 | 82 | 56 | 33 | 78 | 47 | 71 | 100 | 83 | 57 | 92 | 71 | 33 | 100 | 50 | 19 | 100 | 32 |
| Wiki#4 | 78 | 100 | 88 | 78 | 95 | 86 | 78 | 95 | 86 | 83 | 100 | 90 | 91 | 100 | 95 | 91 | 100 | 95 | 78 | 100 | 88 | 74 | 100 | 85 |
| Wiki#5 | 100 | 100 | 100 | 25 | 50 | 33 | 100 | 80 | 89 | 75 | 100 | 86 | 75 | 75 | 75 | 100 | 80 | 89 | 75 | 100 | 86 | 50 | 100 | 67 |
| Wiki#6 | 74 | 78 | 76 | 58 | 61 | 59 | 89 | 81 | 85 | 32 | 86 | 46 | 89 | 85 | 87 | 84 | 94 | 89 | 74 | 100 | 85 | 37 | 100 | 54 |
| Wiki#7 | 80 | 92 | 86 | 33 | 83 | 48 | 73 | 92 | 81 | 87 | 100 | 93 | 80 | 86 | 83 | 87 | 100 | 93 | 80 | 100 | 89 | 67 | 100 | 80 |
| Wiki#8 | 63 | 77 | 69 | 69 | 85 | 76 | 63 | 83 | 71 | 38 | 86 | 52 | 69 | 85 | 76 | 63 | 77 | 69 | 56 | 100 | 72 | 38 | 100 | 55 |
| Wiki#9 | 56 | 90 | 69 | 50 | 67 | 57 | 69 | 92 | 79 | 38 | 67 | 48 | 75 | 100 | 86 | 75 | 100 | 86 | 56 | 100 | 72 | 38 | 100 | 55 |
| Wiki#10 | 61 | 100 | 76 | 39 | 78 | 52 | 67 | 92 | 77 | 50 | 75 | 60 | 78 | 58 | 67 | 83 | 100 | 91 | 44 | 100 | 62 | 33 | 100 | 50 |
| Avg. | 67 | 89 | 76 | 44 | 74 | 54 | 71 | 89 | 77 | 53 | 84 | 63 | 76 | 87 | 80 | **76** | 92 | **82** | 59 | 100 | 72 | 41 | **100** | 56 |
| All Avg. | 66 | 83 | 72 | 51 | 71 | 57 | 67 | 88 | 75 | 57 | 88 | 67 | 73 | 86 | 78 | **78** | 90 | **83** | 58 | 95 | 70 | 38 | **96** | 52 |

The results (cf. Figure 5.6) show that Alchemy, OpenCalais, Ontos and Evri deliver comparable results, while Extractive is a little behind. The ranking with regard to F-Score for all individual categories as well as for the average over all categories is: 1. Evri. 2. OpenCalais, 3. Alchemy, 4. Ontos, 5. Extractive. That the ranking is the same for all categories as well as the overall average indicates that all services perform homogeneously across the different categories. Another interesting observation is that all services deliver the best F-Score for news articles followed by Wiki articles and blog posts. A plausible reason for this is the degree of formality and quality checks, which are more likely with news articles than with blog posts.

As we consider more agreement on an entity to be recognized (i.e. two, three or four APIs have to agree), we obtain a higher precision but lower recall. The interesting result of our analysis is that we have the highest F-score when *two or more APIs agree* on an entity. In this case, we also get the highest recall and the result is independent from the type of text (i.e. News, Weblog or Wiki article). Further increasing the requirement of agreement, however, dramatically decreases recall.

## 5.5. Use Cases and Variations of RDFaCE

The RDFaCE approach is very versatile and can be applied in a vast number of use cases. Also, our implementation based on the widely used TinyMCE editor makes RDFaCE directly applicable in many usage scenarios. In this section we introduce three use cases of RDFaCE, which exemplify the versatility of the approach.

### 5.5.1. Semantic Blogging in WordPress

*WordPress*[8] is an open source Weblog tool and publishing platform. WordPress is often customized into a Content Management System (CMS) and is used by over 14% of the 1,000,000 biggest websites (54.4% of CMS market share) [W3Techs, 2011]. WordPress uses TinyMCE as its content editor. That makes it extremely easy to add the RDFaCE plugin[9] for semantic content authoring within this CMS. With the integration of RDFaCE into the WordPress, the availability of semantically annotated content on the Web can be substantially increased. Figure 5.7 shows an screenshot of RDFaCE integrated into WordPress for semantic blogging. Since releasing the RDFaCE WordPress plugin, the tool has been downloaded over 2000 times.[10]

[8] http://wordpress.org
[9] Available at: http://wordpress.org/plugins/rdface/
[10] http://wordpress.org/plugins/rdface/stats/

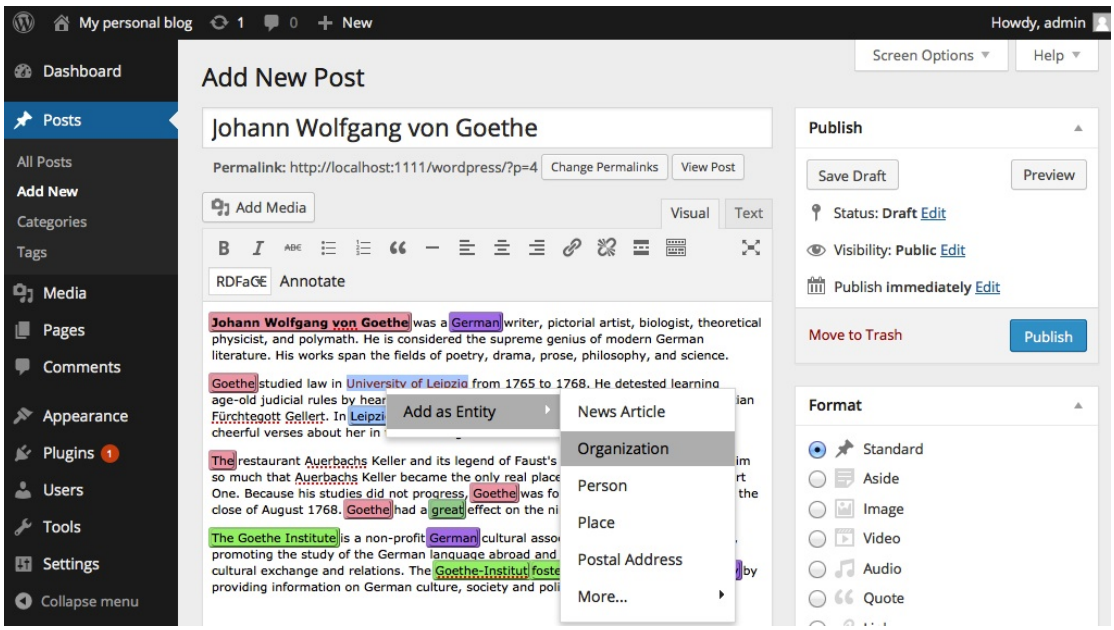


Figure 5.7.: Screenshot of RDFaCE integrated into WordPress.

## 5.5.2. Data Journalism using rNews

*rNews*[11] is a proposed standard for using RDFa to annotate HTML documents with news-specific metadata. rNews is proposed by International Press Telecommunications Council (IPTC), which is a consortium of the world's major news agencies, publishers and industry vendors. rNews defines a small set of core concepts for annotating news articles and a few properties for each concept. Concepts include `NewsItem`, `Tag`, `Person`, `Article`, `Media`, `Headline`, `Location`, `Organization`, `Party`, `TickerSymbol` and `Comment`. These annotations are derived from the best practices found in the news industry. We developed a specific version of RDFaCE called *RDFaCE-Lite* which is well suited for the rNews vocabulary. It provides an autosuggestion feature for the classes and properties defined in the rNews vocabulary. RDFaCE-Lite also provides a context menu for the rNews vocabulary so that users can easily annotate their news articles using the rNews vocabulary. Furthermore, RDFaCE-Lite maps the output of different annotation APIs to the rNews vocabulary thereby providing a base set of automatically annotated content for journalists and content managers. Supporting rNews will enable Data Journalism[12] – ability to tell a compelling story, with the sheer scale and range of digital information. Figure 5.8 and Figure 5.9 present the architecture as well as an screenshot of RDFaCE-Lite.

[11] http://dev.iptc.org/rNews
[12] http://datajournalismhandbook.org/

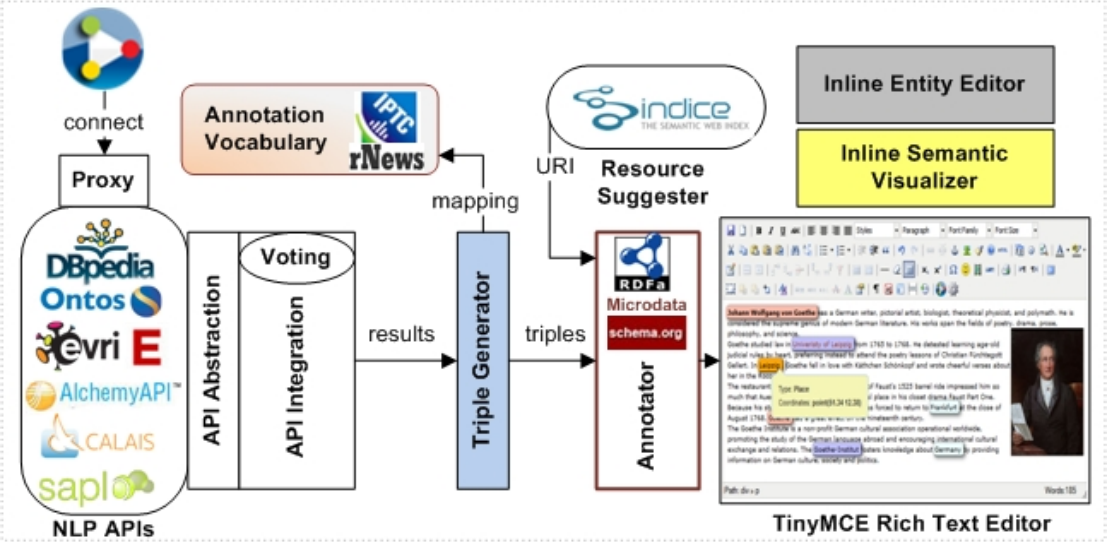


Figure 5.8.: Architecture of RDFaCE-Lite.

### 5.5.3. Search Engine Optimization (SEO) using Schema.org

Schema.org as discussed in Section 2.3.2 provides a collection of schemas, i.e., HTML tags, that Webmasters can use to markup their pages in ways recognized by major search providers. This would help search engines to improve the display of search results, making it easier for people to find the right Web pages. From the users side, having schema markup on users Websites makes it easier for search engines to interpret their content and, therefore, be more likely to be included in the search results for a related query. This will lead to more traffic which attracts many web users to adopt this new technology. Although, there are already a few tools like *Schema Creator*[13], *Microdata Generator*[14] and *Structured Data Markup Helper*[15] which have tried to enable semantic markup based on Schema.org, none of the existing tools provide a user-friendly, flexible and comprehensive solution for semantic markup. The available tools have the following drawbacks, which make them out of reach for normal web users:

- They do not provide any mechanism for automatic content markup.
- They do not support the whole Schema.org vocabulary and are limited to a limited set of schemas. This doesn't allow them to be adapted to new use cases.
- They are not integrated into the user's authoring progress and need additional efforts (e.g. installing and learning new tools) for users to be employed.
- They usually need users to have some knowledge of the markup and Schema.org

[13] http://schema-creator.org/
[14] http://www.microdatagenerator.com/
[15] https://www.google.com/webmasters/markup-helper/

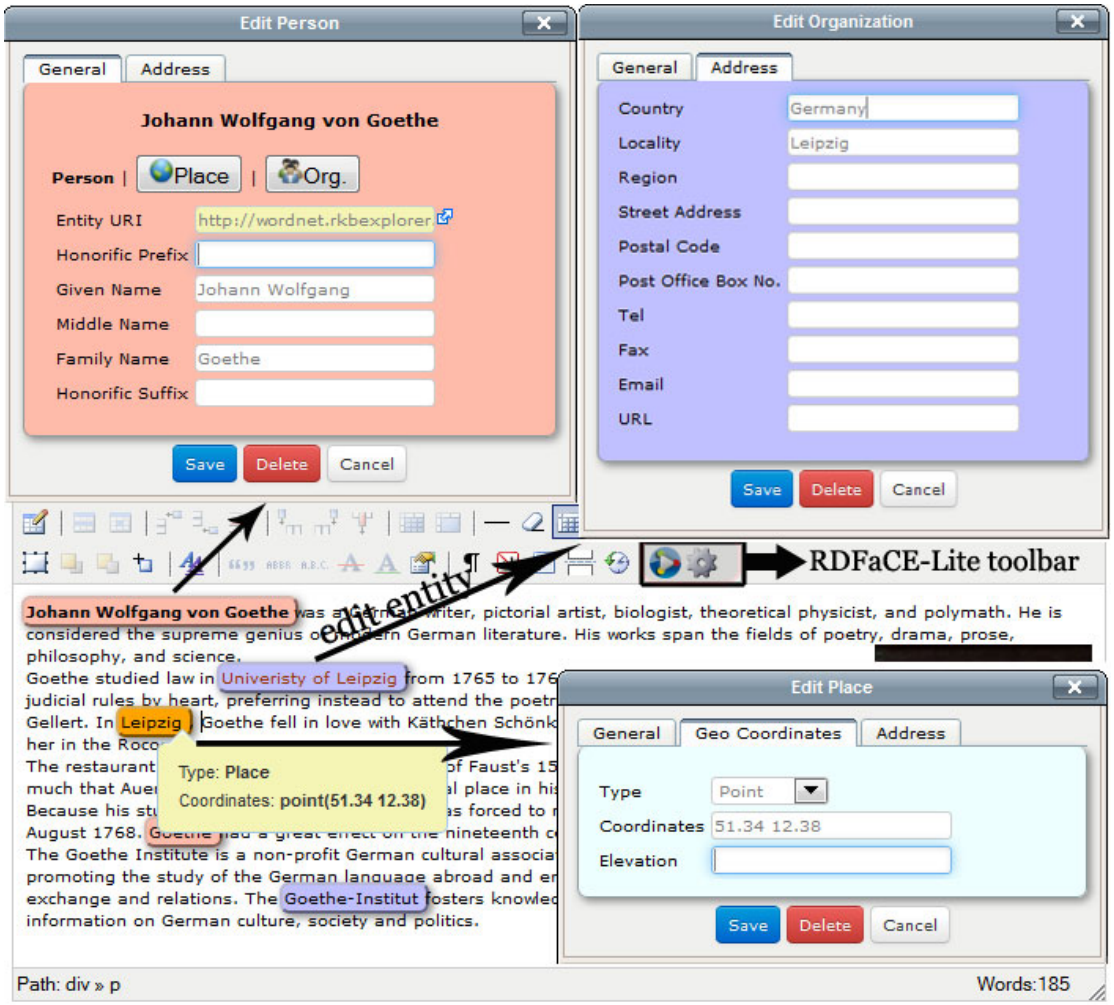


Figure 5.9.: Screenshot of RDFaCE-Lite with support for rNews.

vocabulary thereby cannot be used by non-programmer end users.

RDFaCE comes with a special edition customized for Schema.org vocabulary. In this version, different color schemes are assigned to different schemas defined in Schema.org. Users are able to create a subset of Schema.org schemas for their intended domain and customize the colors for this subset. In this version, nested forms are dynamically generated from the selected schemas for authoring and editing of the annotations.

Figure 5.10 presents the configuration steps in RDFaCE Schema.org edition. The first step is to model the user's domain of interest by selecting a subset of Schema.org schemas. For example user might select NewsArticle, Person, Organization and Place schemas as his desirable schemas. For each schema, the range of properties will be checked in order to include derivative schemas as well (e.g. PostalAddress and Country for the Place schema). The results of this step

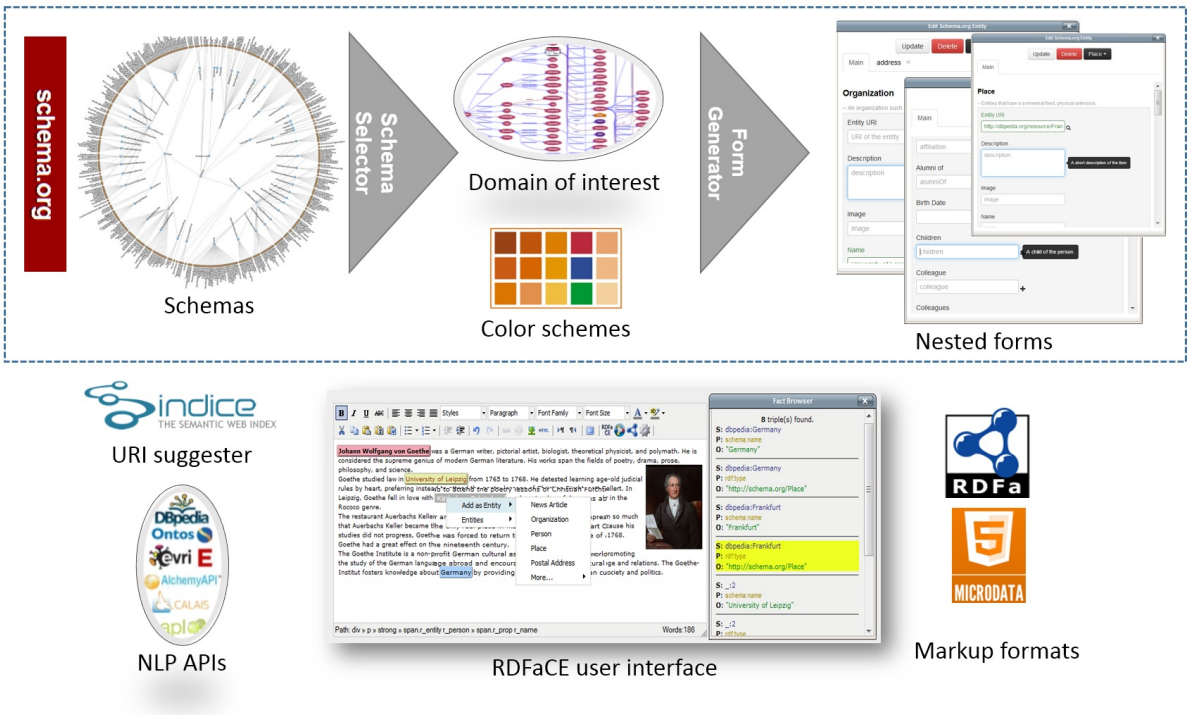


Figure 5.10.: Configuration steps in RDFaCE Schema.org edition.

is an input JSON file which describes the selected schemas together with their corresponding properties. For visualization of schemas, we need to assign unique colors to the selected schemas. We use an algorithm to automatically generate a light color scheme for the schemas. The color scheme is available as CSS styles and is easily configurable by users.

The next step is to generate appropriate forms based on the selected schemas. Form inputs are created based on the corresponding data type defined as range of the schema properties. For example we add Datepicker UI for properties with Date as their range. These forms are then used to add metadata into the text.

The final step in configuration is to select the desired markup format (e.g. RDFa or Microdata) as well as desired NLP APIs (e.g. DBpedia Spotlight[16]) for automatic annotation of content. Users can select multiple NLP APIs and determine how they want to combine the results. The combination can be performed based on the agreement between two or more of the involved APIs. Users are also able to set a confident level for automatic annotation and can limit the type of recognized entities to only annotate specific entities like Persons and Places.

**Example Scenario.** On-page markup based on Schema.org enables search engines to increasingly understand the information on Web pages and provide richer search

[16] http://spotlight.dbpedia.org/

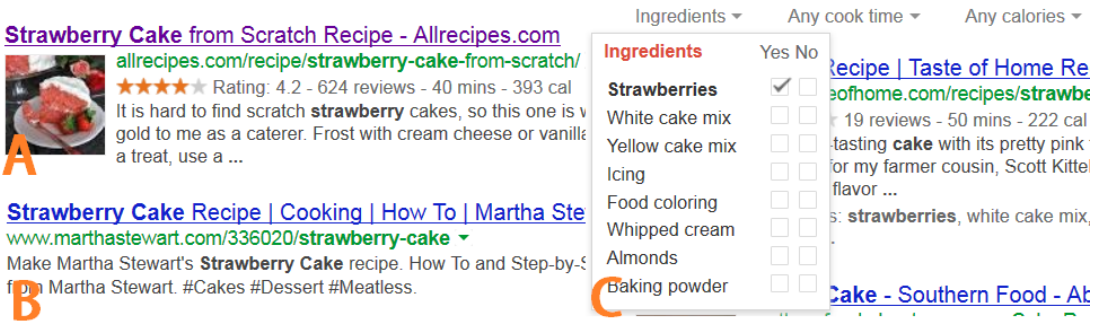


Figure 5.11.: Search results improved by rich snippets. A: enhanced recipe, B: normal recipe, C: browsing recipes by ingredients, cook time and calories.

results. *Rich Text Snippets* as an example of on-page markup provides an immediate advantage and motivation for Web users to embed structured content into their documents. Rich snippets comprise a wide range of schemas such as *Breadcrumbs*, *Events*, *Music*, *Organizations*, *People*, *Products*, *Recipes*, *Review ratings*, *Reviews*, *Software Applications* and *Videos*. Web documents which are annotated based on these schemas will attract more attention by people searching the Web due to the richness of presented information. Figure 5.11 shows as example of enhanced search results for recipes powered by rich snippets.

As an example scenario, Figure 5.13 presents the RDFaCE WYSIWYM interface employed to annotate a sample recipe rich snippet. The user simply selects the parts of the text and annotates them using the corresponding schema from Schema.org. On the background, RDFaCE generates the corresponding RDFa or Microdata markup based on the following rules:

- if the selected text already has an HTML tag, metadata will be added as new attributes for the current tag (e.g. Figure 5.12 line 2 or 6).
- if the selected text does not have an HTML tag, a new `<SPAN>` or `<DIV>` tag with corresponding attributes will be created (e.g. Figure 5.12 line 1 or 3).
- if no text is selected, a new `<META>` tag with corresponding attributes will be created (e.g. Figure 5.12 line 17 or 18).

## 5.6. Usability Evaluation

Since releasing RDFaCE, the tool (WordPress plugin and independent TinyMCE plugins) has been downloaded over 3000 times and the online demo page has received more than 5000 unique visits. we also could collect considerable feedback

```
<div itemscope itemtype="http://schema.org/Recipe">
    <h2 itemprop="name">Strawberry Cake</h2>
    <p>By <span itemprop="author">Ali Khalili</span>, July 8, 2013
       </p>
    <h4>Ingredients</h4>
    <ul>
        <li itemprop="ingredients">2 cups white sugar</li>
        <li>...</li>
    </ul>
    <p>Preparation time: 10 mins</p>
    <p>Cooking time: 30 min</p>
    <p>Ready in 40 min</p>
    <p>
        <span itemscope
           itemtype="http://schema.org/NutritionInformation"
           itemprop="nutrition">
           Calories: <span itemprop="calories">393 kcal</span>
        </span>
    </p>
    <meta itemprop="dateCreated" content="2013-07-08">
    <meta itemprop="prepTime" content="PT15M">
</div>
```

Figure 5.12.: Example of Microdata annotations generated by RDFaCE.

from RDFaCE end-users on the Social Web. For a concrete evaluation of RDFaCE usability, we conducted an experiment with 16 participants of ISSLOD 2011 summer school[17]. For the experiment, we developed a usability test platform[18]. The experiment consisted of the following steps:
First, some basic information about semantic content authoring along with a demo showcasing different RDFaCE features was presented to the participants as a video. Then, participants were asked to use RDFaCE to annotate three text snippets – a wiki article, a blog post and a news article (News#2, Blog#4, Wiki#3 from our sample articles). For each text snippet, a timeslot of five minutes was available to use different features of RDFaCE for annotating occurrences of persons, locations and organizations with suitable entity references (i.e. Linked Data URIs). Subsequently, a survey was presented to the participants were they were asked some questions about their experience while working with RDFaCE. Questions were targeting six factors of usability [Lauesen, 2005] namely *Fit for use*, *Ease of learning*, *Task efficiency*, *Ease of remembering*, *Subjective satisfaction*

[17]Summer school on Linked Data: http://lod2.eu/Article/ISSLOD2011
[18]Available online at: http://rdface.aksw.org/usability

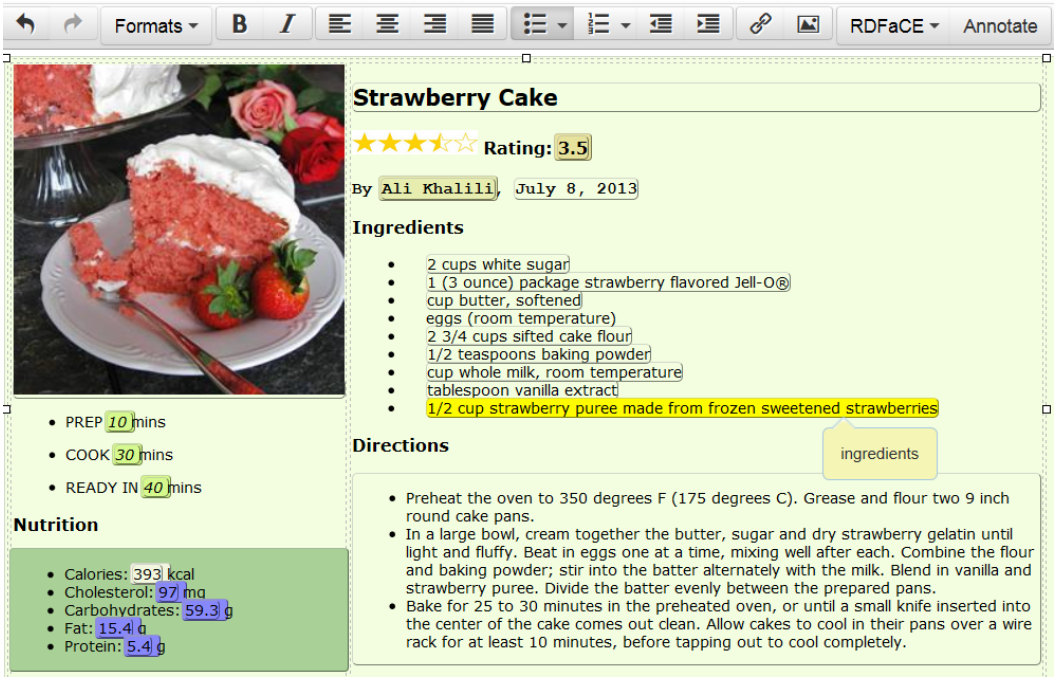


Figure 5.13.: Using RDFaCE to annotate recipes based on Schema.org.

| Skill/ Level | heard of it | basic | advanced | expert |
|---|---|---|---|---|
| Skill in Semantic Web | 6.25% | 37.50% | 37.50% | 18.75% |
| Skill in RDFa | 18.75% | 37.50% | 37.50% | 6.25% |

Table 5.2.: Participants level of knowledge.

and *Understandability*. Results of individual user annotations as well as the results of the survey were carefully analyzed for extracting subjective and objective usage characteristic of RDFaCE, respectively. In the following we report about the result of this experiment:

*Participants*. Participants included students (85%) and researchers (15%) working on different aspects of computer science and informations systems. As shown in Table 5.2, they bear different level of knowledge in Semantic Web and in particular RDFa, varying from basic to expert knowledge.

*Usability Factors*. During the experiment we collected considerable qualitative feedback from the end-users. As shown in Table 5.3, the overall feedback of users was positive and they provided constructive feedback to enhance the usability of RDFaCE. They frequently told us that they are impressed with the functionality of RDFaCE to support their desired tasks. They found the UI easy to learn but in some cases had difficulties to distinguish between property and subject suggestions. Some users suggested to change triple insertion to property insertion and some suggested to improve the visualization of URI suggestion results so that they can easily choose the appropriate one. Most of the users found the UI easy to remember

| Usability Factor/Grade | Poor | Fair | Neutral | Good | Excellent |
|---|---|---|---|---|---|
| Fit for use | 0% | 12.50% | 31.25% | **43.75%** | 12.50% |
| Ease of learning | 0% | 12.50% | **50%** | 31.25% | 6.25% |
| Task efficiency | 0% | 0% | **56.25%** | 37.50% | 6.25% |
| Ease of remembering | 0% | 0% | 37.50% | **50%** | 12.50% |
| Subjective satisfaction | 0% | 18.75% | **50%** | 25% | 6.25% |
| Understandability | 6.25% | 18.75% | 31.25% | **37.50%** | 6.25% |

Table 5.3.: Usability factors derived from the survey.

and a few suggested to change some RDFaCE toolbar icons to more descriptive ones.

*Annotations.* Figure 5.14 reflects the number of annotations (triples) as well as the time of annotation per user for each of the text fragments. From the results we can see that almost (except two cases) all users have been able to create semantic text content. The annotation time for the last text snippet has decreased for most of the users which is an indicator for increased familiarity of the users with RDFaCE.

## 5.7. Comparison of RDFaCE to Existing SCA Tools

There are already many Semantic Content Authoring (SCA) systems available. *RADiFy*[19], *WYMeditor*[20], *DataPress* [Benson et al., 2010], *Loomp* [Luczak-Roesch, 2009] and *FLERSA* [Navarro-Galindo and Samos, 2010] are some examples of SCA systems which adopt the bottom-up approach. We can also mention *RDFauthor* [Tramp et al., 2010] and *SAHA* 3 [Frosterus et al., 2011] as two examples which adopt the top-down approach for semantic authoring. *OntosFeeder*[21] and *Epiphany*[22] are also two partially related tools. They do not provide editing functionality for RDFa generated content but can be used as complementary tools to RDFaCE which deliver a set of initial RDFa annotations to be edited and extended later on by RDFaCE. As an another related work we can mention *Named Entity Recognition and Disambiguation (NERD)* [Rizzo and Troncy, 2011][23] which is an evaluation framework which records and analyzes ratings of Named Entity extraction and disambiguation tools. The main difference between RDFaCE and NERD is that RDFaCE employs the voting approach to combine the results of NLP APIs for automatic annotation but NERD expects a human being to manually compare the results of different NLP APIs and choose the right one for annotation.

Figure 5.15 provides a comparison between the three popular SCA systems

[19] http://duncangrant.co.uk/radify/
[20] http://www.wymeditor.org
[21] http://wordpress.org/extend/plugins/ontos-feeder/
[22] http://projects.dfki.uni-kl.de/epiphany/
[23] http://nerd.eurecom.fr/

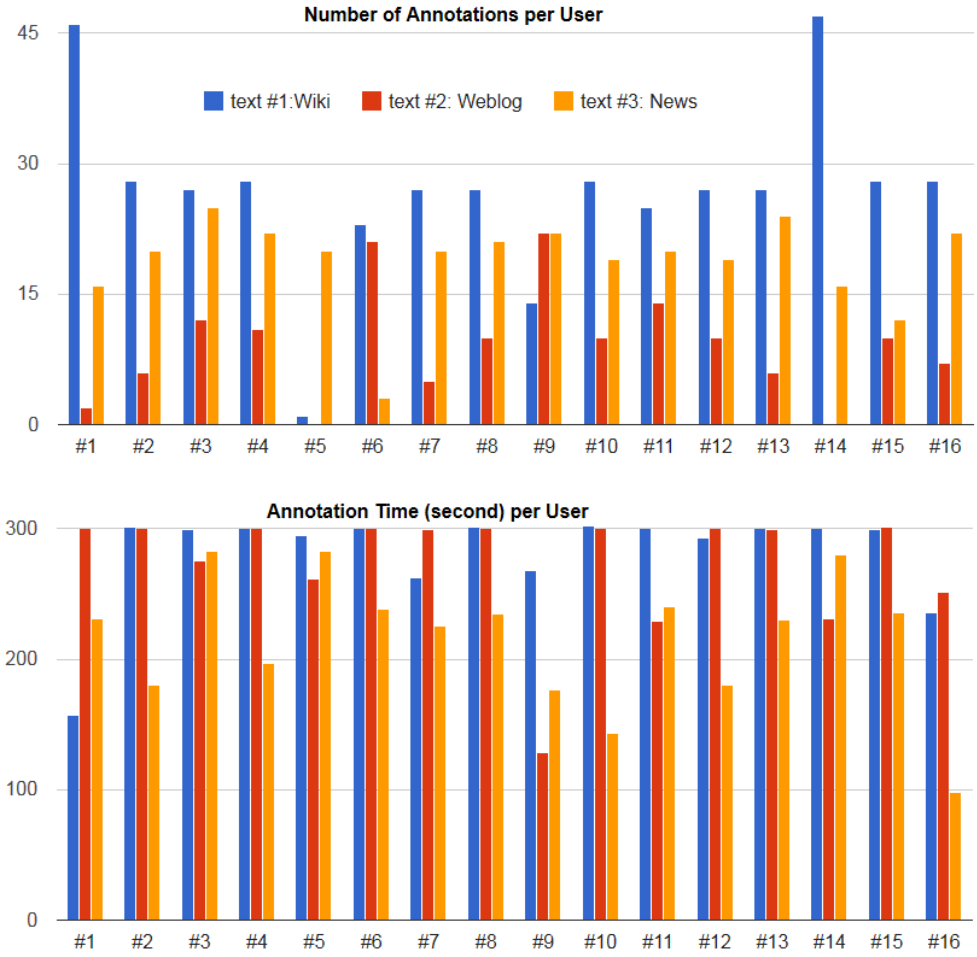


Figure 5.14.: Results of usability test. (top) Number of annotations per user. (bottom) Annotation time per user.

(RDFauthor, SAHA 3 and Loomp) and RDFaCE based on the quality attributes discussed in Chapter 3. Here we have compared the tools based on the quality attributes that were already addressed during the development of RDFaCE. RDFauthor is a tool for editing RDFa contents. The RDFauthor approach is based on the idea of making arbitrary XHTML views with integrated RDFa annotations editable [Tramp et al., 2010]. RDFauthor converts an RDFa-annotated view directly into an editable form thereby hiding the RDF and related ontology data models from novice users. It is backend independent to some extend and supports two different types of storage engines. Although RDFauthor has as RDFaCE the goal to make RDFa editing simple by abstracting the details of RDFa authoring both differ in two crucial aspects: Firstly, RDFauthor assumes that the RDFa content is already existing while RDFaCE provides the feature to creating new RDFa annotations. Secondly, instead of using forms to edit RDFa contents, RDFaCE employs inline editing of contents by providing a rich semantic text editor. Saha

| | RDFauthor | SAHA 3 | Loomp | RDFaCE |
|---|---|---|---|---|
| Usability | -Single point of entry UI<br>-Inline editing | -Single point of entry UI<br>-Inline editing | -Single point of entry UI<br>-Faceted viewing | -Single point of entry UI<br>-Inline editing |
| Customizability | - | - | - | -Semantic views:<br>WYSIWYM, WYSIWYG<br>, triple view,<br>source code view |
| Proactivity | -Resource suggestion<br>-Concept reuse | -Resource suggestion<br>-Concept reuse<br>-Real-time validation | -Resource suggestion<br>-Concept reuse | -Resource suggestion |
| Automation | - | - | - | -Automatic annotation:<br>NLP APIs |
| Scalability | -Storage strategy:<br>backend independent<br>(Mysql, Virtuoso) | -Storage strategy:<br>server-side triple store | -Storage strategy:<br>server-side triple store | -Storage strategy:<br>on-the-fly client-side<br>triple storage |

Figure 5.15.: Comparison of RDFauthor, SAHA 3, Loomp and RDFaCE according to the quality attributes.

3 is another meta data editor which is very similar to RDFauthor but supports real-time validation in addition (see Section 3.8.2).

Loomp is an editor which allows annotating words and phrases with references to ontology concepts (see Section 3.8.3). It supports a faceted viewing feature, which highlights user-selected annotations in the Web browser. The main difference between Loomp and RDFaCE is that Loomp relies on the functionality of a server managing the semantic content while RDFaCE provides client-side annotation for modifying RDFa content directly. Morever, Loomp uses a triple store on the server side but in RDFaCE, triples are created on the fly in the user browser.

The main advantages of RDFaCE comparing to other tools are twofold: Providing different views for authoring semantic documents as well as supporting automatic content annotation, which improve the customizability and automation remarkably. Furthermore, since RDFaCE processes the annotations client-side within the user's browser and does not require any central storage backend, it is highly scalable.

## 5.8. Conclusions

This chapter addressed the research question RQ3 (cf. Section 1.3) to integrate semantic authoring features into the current tools on the Social Web. With RDFaCE we presented an approach and its implementation of a WYSIWYM editor based on complementing the classical WYSIWYG view with three additional views on the semantic representations. We showed that with RDFaCE the semantic annotation and enrichment can be easily integrated into the content authoring pipelines commonly found in many content centric scenarios.

Chapter 6

# WYSIWYM for Lightweight Text Analytics

"Simplicity is the ultimate sophistication."
— *Leonardo da Vinci*

In this chapter we present a text analytics architecture of participation, which employs WYSIWYM UI model to allow ordinary people with no or limited knowledge of programming to use sophisticated NLP techniques for analyzing and visualizing their content, be it a Blog, Twitter feed, Website or article collection. Different exchangeable components can be plugged into this architecture, making it easy to tailor for individual needs. We evaluate the usefulness of our approach by comparing both the effectiveness and efficiency of end users within a task-solving setting. Moreover, we evaluate the usability of our approach using a questionnaire-driven approach.

The chapter is structured as follows: Section 6.1 describes the current analytical information imbalance. In Section 6.2, we introduce conTEXT for democratizing the NLP usage. We show that conTEXT fills a gap in the space of related approaches in Section 6.3. The general workflow and interface design is presented in Section 6.4. The different visualizations and views supported by conTEXT are discussed in Section 6.5 before we present our implementation in Section 6.6. We show the results of a qualitative and quantitative user evaluation in Section 6.7 before we conclude in Section 6.8.[1]

## 6.1. Analytical Information Imbalance

Currently, there seems to be an imbalance on the Web. Hundreds of millions of users continuously share stories about their life on social networking platforms such as *Facebook*, *Twitter* and *Google Plus*. However, the conclusions that can be drawn from analyzing the shared content are rarely shared back with the users of these platforms. The social networking platforms on the other hand exploit the results of analyzing user-generated content for targeted placement of advertisements, promotions, customer studies etc. One basic principle of data privacy is, that every person should be able to know what personal information is stored about herself in a database (cf. OECD privacy principles[2]). We argue, that this principle does *not* suffice anymore and that there is an *analytical information imbalance*. People

[1] The contents of this chapter have been published as [Khalili et al., 2014].

[2] http://oecdprivacy.org/#participation

should be able to find out what patterns can be discovered and what conclusions can be drawn from the information they share.

Let us look at the case of a typical social network user Judy. When Judy updates her social networking page regularly over years, she should be able to discover what the main topics were she shared with her friends, what places, products or organizations are related to her posts and how these things she wrote about are interrelated. Currently, the social network Judy uses analyzes her and other users data in a big data warehouse. Advertisement customers of the social networking platform, can place targeted adds to users being interested in certain topics. Judy, for example, is sneaker aficionado. She likes to wear colorful sports shoes with interesting designs, follows the latest trends and regularly shares her current favorites with her friends on the social network. Increasingly, advertisements for sportswear are placed within her posts. Being able to understand what conclusions can be drawn by analyzing her posts will give Judy at least some of the power back into her hands she lost during the last years to Web giants analyzing big user data.

## 6.2. conTEXT: A Text Analytics Architecture of Participation

In order to mitigate the current analytical information imbalance, we created *conTEXT*[3] – a text analytics architecture of participation, which allows end-users to use sophisticated NLP techniques for analyzing and visualizing their content, be it a Weblog, Twitter, Facebook, G+, LinkedIn feed, Website or article collection. With almost no effort, users can analyze the information they share and obtain similar insights as social networking sites. The conTEXT architecture comprises interfaces for information access, natural language processing (currently mainly NER) and visualization. Different exchangeable components can be plugged into this architecture. Users are empowered to provide manual corrections and feedback on the automatic text processing results, which directly increase the semantic annotation quality and are used as input for attaining further automatic improvements. An online demo of the conTEXT is available at `http://context.aksw.org`.

conTEXT empowers users to answer a number of questions, which were previously impossible or very tedious to answer. Examples include:

- Finding all articles or posts related to a specific person, location or organization.
- Identifying the most frequently mentioned terms, concepts, people, locations or organizations in a corpus.

[3]We choose the name conTEXT, since our approach performs analyzes *with* (Latin 'con') text and provides contextual visualizations for discovered entities in text.

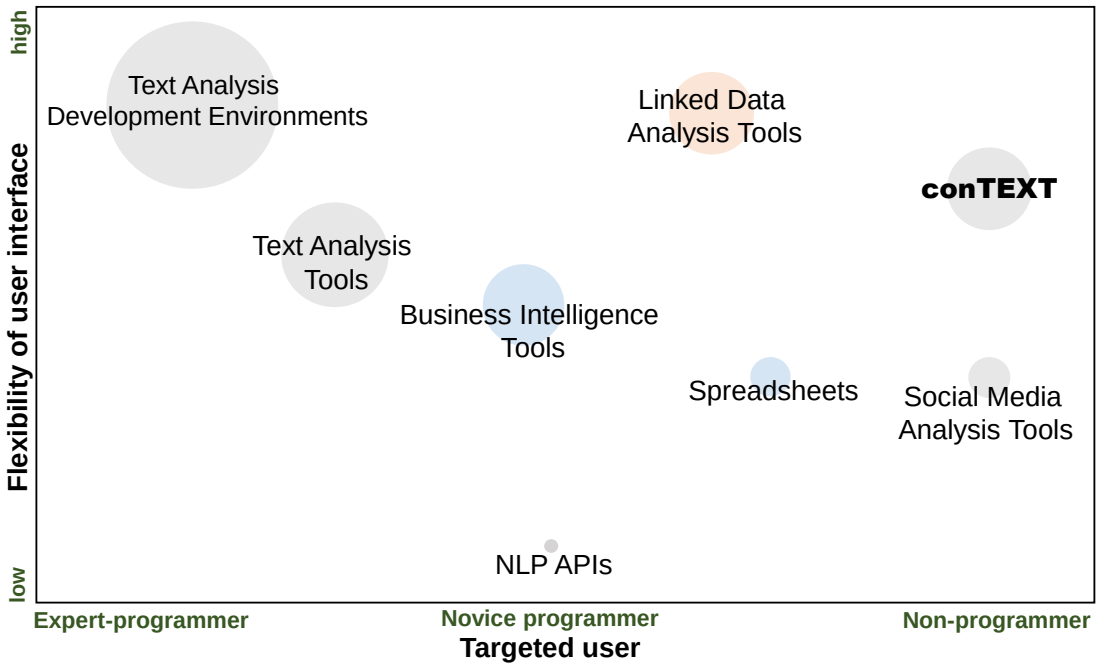


Figure 6.1.: Flexibility of user interfaces and targeted user groups as well as genericity (circle size) and degree of structure (circle color) for various analytics platforms.

- Showing the temporal relations between people or events mentioned in the corpus.
- Discovering typical relationships between entities.
- Identifying trending concepts or entities over time.
- Find posts where certain entities or concepts co-occur.

conTEXT lowers the barrier to text analytics by providing the following key features:

- No installation and configuration required.
- Access content from a variety of sources.
- Instantly show the results of analysis to users in a variety of visualizations.
- Allow refinement of automatic annotations and take feedback into account.
- Provide a generic architecture where different modules for content acquisition, natural language processing and visualization can be plugged together.

## 6.3. Classification of Existing Text Analysis Tools

Analytics (i.e. the discovery and communication of meaningful patterns in data) is a broad area of research and technology. Involving research ranging from *NLP* and *Machine Learning* to *Semantic Web*, this area has been very vibrant in

recent years. Existing tools in the domain of analytics can be roughly categorized according to the following dimensions:

- *Degree of structure.* Typically, an analytics system extracts patterns from a certain type of input data. The type of input data can vary between *unstructured* (e.g. text, audio, videos), *semi-structured* (e.g. text formats, shallow XML, CSV) and *structured* data (e.g. databases, RDF, richly structured XML).

- *Flexibility of user interface.* Analytics systems provide different types of interfaces to communicate the found patterns to users. A flexible UI should support techniques for *exploration*, *visualization* as well as even *feedback and refinemment* of the discovered patterns. This dimension also evaluates the *interactivity* of UIs, *diversity* of analytical views as well as the capability to *mix* results.

- *Targeted user.* An analytics system might be used by different types of users including *non-programmer*, *novice-programmer* and *expert-programmer*.

- *Genericity.* This dimension assesses an analytics system in terms of *genericity of architecture* and *scalability*. These features enable reuse of components as well as adding new functionality and data at minimal effort.

Figure 6.1 provides an abstract view of the state-of-the-art in analytics according to these dimensions.

*Text analysis development environments* usually provide comprehensive support for developing customized text analytics workflows for extracting, transforming and visualizing data. Typically they provide a high degree of genericity and interface flexibility, but require users to be expert-programmers. Examples include the *IBM Content Analytics platform*[4], *GATE* [Cunningham et al., 2011], *Apache UIMA* [Ferrucci and Lally, 2004].

*Text analysis tools* provide a higher level of abstraction (thus catering more novice users) at the cost of genericity. Yang et al. [Yang et al., 2013] recently published an extensive text analytics survey from the viewpoint of the targeted user and introduced a tool called *WizIE* which enables novice programmers to perform different tasks of text analysis. Examples include *Attensity*[5], *Thomson Data Analyzer*[6] *Trendminer* [Preotiuc-Pietro et al., 2012] and *MashMaker* [Ennals et al., 2007].

*Business intelligence (BI) tools* are applications designed to retrieve, analyze and report mainly highly-structured data for facilitating business decision making. BI tools usually require some form of programming or at least proficiency in

[4] http://www-03.ibm.com/software/products/us/en/contentanalyticssearch
[5] http://www.attensity.com
[6] http://thomsonreuters.com/thomson-data-analyzer/

query construction and report designing. Examples include *Zoho Reports*[7], *SAP NetWeaver*[8], *Jackbe*[9], and *RapidMiner* [Jungermann, 2009].

*Spreadsheet-based tools* are interactive applications for organization and analysis of data in tabular form. They can be used without much programming skills, are relatively generically applicable and provide flexible visualizations. However, spreadsheet-based tools are limited to structured tabular, data and can not be applied to semi-structured or text data. Examples include *Excel*, *DataWrangler* [Kandel et al., 2011], *Google Docs Spreadsheets* and *Google Refine*.

*NLP APIs* are web services providing natural language processing (e.g. named entity recognition and relation extraction) for analyzing web pages and documents. The use of these APIs requires some form of programming and flexible interfaces are usually not provided. Examples include *Alchemy*, *OpenCalais*, *Apache OpenNLP*.[10]

*Linked Data analysis tools* support the exploration and visualization of Linked Data (LD). Examples include *Facete*[11] for spatial and *CubeViz*[12] for statistical data. Dadzie and Rowe [Dadzie and Rowe, 2011] present a comprehensive survey of approaches for visualizing and exploring LD. They conclude that most of the tools are designed only for tech-users and do not provide overviews on the data.

*Social Media analysis tools* such as *SRSR*, *TweetDeck*[13], *Topsy*[14], *Flumes*[15], and *Trendsmap*[16] focus in comparison to conTEXT primarily on the content aggregation across large repositories (e.g. Twitter as a whole) and perform popularity and trend analysis. conTEXT on the other hand aims at providing different exploration and visualization means for more specific types of content exploiting the extracted semantics.

When comparing these different analytics tool categories according to the dimensions genericity, UI flexibility, target users and degree of structure we discovered a lack of tools dealing with unstructured content, catering non-expert users and providing flexible analytics interfaces. The aim of developing the text analytics tool conTEXT is to fill this gap.

## 6.4. Workflow and Interface Design

**Workflow.** Figure 6.2 shows the process of text analytics in conTEXT. The process starts by collecting information from the web or social web. conTEXT utilizes standard information access methods and protocols such as RSS/ATOM

[7] http://www.zoho.com/reports/
[8] http://sap.com/netweaver
[9] http://jackbe.com/
[10] A complete list of NLP APIs is available at http://nerd.eurecom.fr/
[11] http://aksw.org/Projects/Facete
[12] http://aksw.org/Projects/CubeViz
[13] http://tweetdeck.com/
[14] http://topsy.com/
[15] http://www.flumes.com/
[16] http://trendsmap.com/

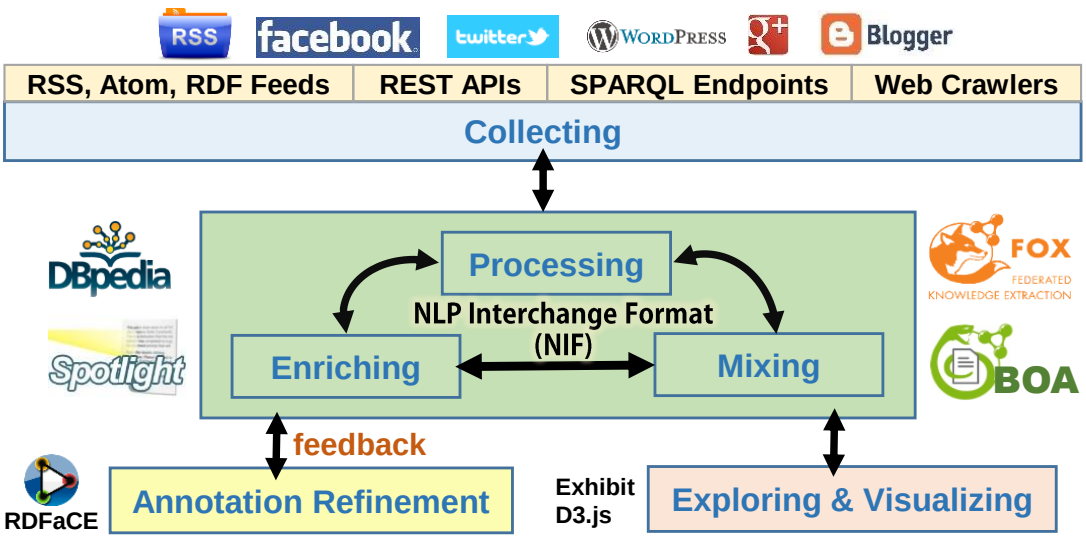


Figure 6.2.: Text analytics workflow in conTEXT.

feeds, SPARQL endpoints and REST APIs as well as customized crawlers for SlideWiki, WordPress, Blogger and Twitter to build a corpus of information relevant for a certain user.

The assembled text corpus is then processed by NLP services. While conTEXT can integrate virtually any NLP services, it currently implements interfaces for *DBpedia Spotlight* [Mendes et al., 2011] and the *Federated knOwledge eXtraction Framework (FOX)* [Ngomo et al., 2011] for discovering and annotating named entities in the text. DBpedia Spotlight annotates mentions of *DBpedia* resources in text thereby links unstructured information sources to the Linked Open Data cloud through DBpedia. FOX is a knowledge extraction framework that utilizes a variety of different NLP algorithms to extract RDF triples of high accuracy from text. Unlike DBpedia Spotlight, which supports all the DBpedia resource types, FOX is limited to `Person`, `Location` and `Organization` types. On the other hand, since FOX uses ensemble learning to merge different NLP algorithms, leads to a higher precision and recall (see [Ngomo et al., 2011] for details).

The processed corpus is then further enriched by two mechanisms:

- DBpedia URIs of the found entities are de-referenced in order to add more specific information to the discovered named entities (e.g. longitude and latitudes for locations, birth and death dates for people etc.).
- Entity co-occurrences are matched with pre-defined natural-language patterns for DBpedia predicates provided by *BOotstrapping linked datA (BOA)* [Gerber and Ngonga Ngomo, 2011] in order to extract possible relationships between the entities.

The processed data can also be joined with other existing corpora in a *text analytics Mashup*. Such a Mashup of different annotated corpora combines information

from more than one corpus in order to provide users an integrated view. Analytics Mashups help to provide more contexts for the text corpus under analysis and also enable users to mix diverse text corpora for performing a comparative analysis. For example, a user's WordPress blog corpus can be integrated with corpora obtained from her Twitter and Facebook accounts. The creation of analytics Mashups requires dealing with the heterogeneity of different corpora as well as the heterogeneity of different NLP services utilized for annotation. conTEXT employs *NIF* [Hellmann et al., 2013] to deal with this heterogeneity. The use of NIF allows us to quickly integrate additional NLP services into conTEXT.

The processed, enriched and possibly mixed results are presented to users using different views for exploration and visualization of the data. *Exhibit* [Huynh et al., 2007][17] (structured data publishing) and *D3.js* [Bostock et al., 2011][18] are employed for realizing a dynamic exploration and visualization experience. Additionally, conTEXT provides an annotation refinement user interface based on the *RDFa Content Editor* (RDFaCE) discussed in Chapter 5 to enable users to revise the annotated results. User-refined annotations are sent back to the NLP services as feedback for the purpose of learning in the system.

**Progressive crawling and annotation.** The process of collecting and annotating a large text corpus can be time-consuming. Therefore it is very important to provide users with immediate results and inform them about the progress of the crawling and annotation task. For this purpose, we have designed special user interface elements to keep users informed until the complete results are available. The first indicator interface is an animated progress bar, which shows the percentage of the collected/annotated results as well as the currently downloaded and processed item (e.g. the title of the blog post). The second indicator interface is a real-time tag cloud, which is updated while the annotation is in progress. We logged all crawling and processing timings during our evaluation period. Based on these records, the processing of a Twitter feed with 300 tweets takes on average 30 seconds and the processing of 100 blog posts approx. 3-4 minutes on standard server with i7 Intel CPU (with parallelization and hardware optimizations further significant acceleration is possible). This shows, that for typical crawling and annotation tasks the conTEXT processing can be performed in almost real-time thus providing instant results to the users.

**Annotation refinement interfaces.** A lightweight text analytics as implemented by conTEXT provides direct incentives to users to adopt and revise semantic text annotations. Users will obtain more precise results as they refine annotations. On the other hand, NLP services can benefit from these manually-revised annotations to learn the right annotations. conTEXT employs the RDFaCE within the faceted browsing view and thus enables users to edit existing annotations while browsing

[17] http://simile-widgets.org/exhibit3/
[18] Data-Driven Document (D3) http://d3js.org/

| Parameter | Description |
|---|---|
| *text* | annotated text. |
| *entityUri* | the identifier of the annotated entity. |
| *surfaceForm* | the name of the annotated entity. |
| *offset* | position of the first letter of the entity. |
| *feedback* | indicates whether the annotation is correct or incorrect. |
| *context* | indicates the context of the annotated corpus. |
| *isManual* | indicates whether the feedback is sent by user or by other NLP services. |
| *senderIDs* | identifier(s) of the feedback sender. |

Table 6.1.: NLP Feedback parameters.

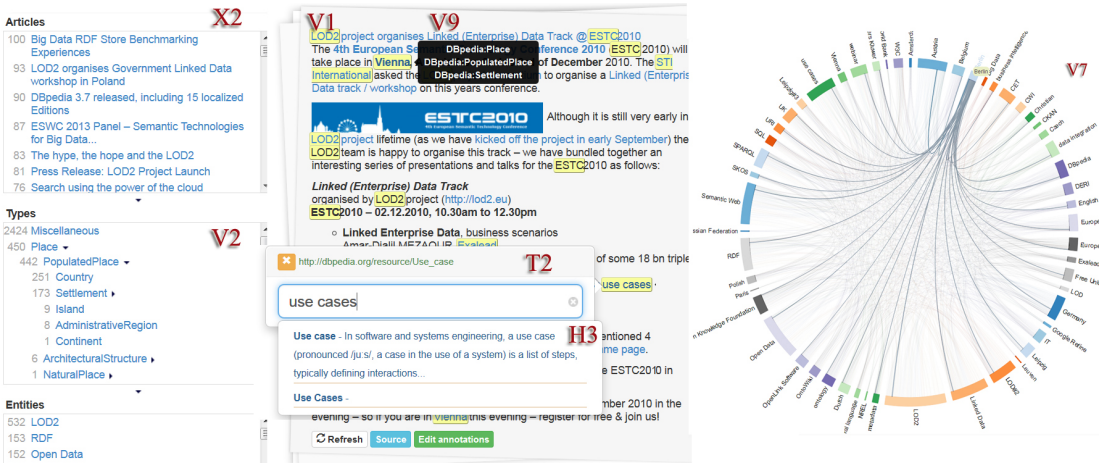


Figure 6.3.: Screenshots of the conTEXT WYSIWYM interface (T2 indicates the inline editing UI, V1 – the framing of named entities in the text, V2 – text margin formatting for visualizing hierarchy, V7 – line connectors to show the relation between entities, V9 – a callout showing additional type information, X2 – faceted browsing, H3 – recommendation for NLP feedback).

the data. The WYSIWYM interface as depicted in Figure 6.3 enables integrated visualization and authoring of unstructured and semantic content (i.e. annotations encoded in RDFa). The manual annotations are collected and sent as feedback to the corresponding NLP service[19]. The feedback encompasses the parameters specified in Table 6.1.

[19]DBpedia Spotlight Feedback API (http://spotlight.dbpedia.org/rest/feedback), FOX Feedback API (http://139.18.2.164:4444/api/ner/feedback)

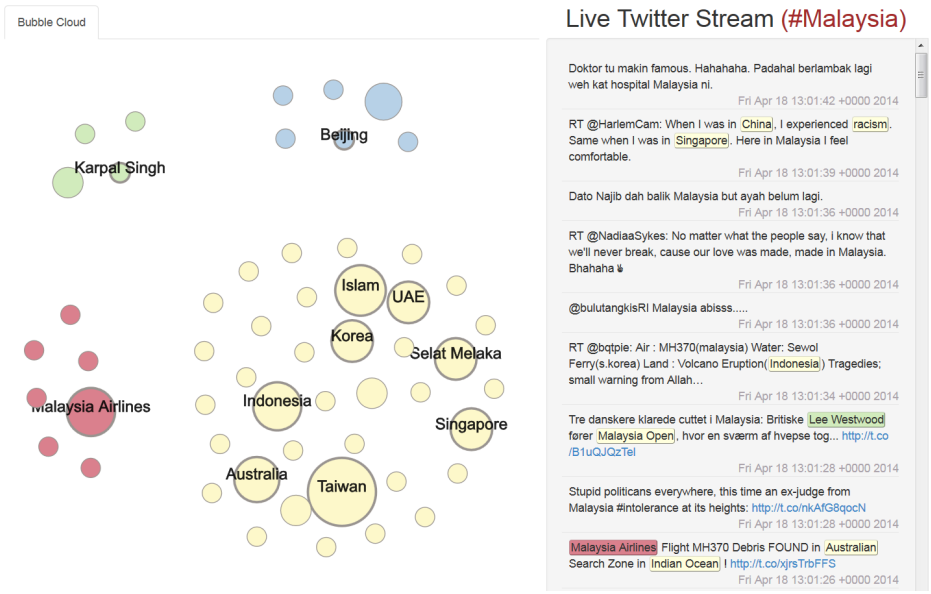


Figure 6.4.: Example of realtime semantic analysis in conTEXT.

**Exploration and visualization interfaces.** The dynamic exploration of content indexed by the annotated entities facilitates faster and easier comprehension of the content and provide new insights. conTEXT creates a novel entity-based search and browsing interface for end-users to review and explore their content. On the other hand, conTEXT provides different visualization interfaces which present, transform, and convert semantically enriched data into a visual representation, so that, users can explore and query the data efficiently. Visualization UIs are supported by noise-removal algorithms which will tune the results for better representation and will highlight the picks and trends in the visualizations. For example, we use a frequency threshold when displaying single resources in interfaces. In addition, a threshold based on the Dice similarity is used in interfaces, which display co-occurrences. By these means, we ensure that the information overload is reduced and that information shown to the user is the most relevant. Note that the user can chose to deactivate or alter any of these thresholds.

**Linked Data interface for Search Engine Optimization (SEO).** As discussed in Section 5.5.3, the `Schema.org` initiative provides a collection of shared schemas that Web authors can use to markup their content in order to enable enhanced search and browsing features offered by major search engines. A direct feature of the Linked Data based text analytics with conTEXT is the provisioning of a *SEO* interface. conTEXT encodes the results of the content annotation (automatic and revisions by the user) in the *JSON-LD* format (cf. Section 3.3.1) which can be directly exposed to schema.org aware search engines. This component employs

the current mapping from the DBpedia ontology to the Schema.org vocabularies[20]. Thus the conTEXT SEO interface enables end-users to benefit from better exposure in search engines (e.g. through Google's *Rich Text Snippets*) with very little effort.

**Real-time semantic analysis.** In addition to its normal functionality, conTEXT also supports real-time content analysis for streaming data like Twitter streams. Figure 6.4 shows an example of real-time semantic anlaysis for Twitter streams on specific hashtags.[21] This way, users can see the live progress of different analytics views on incoming data and thereby can quickly follow the trends that are currently on the social media. Real-time analytics is also useful for the companies and businesses to gain competitive advantage and to improve their customer relationships by monitoring users feedback on social media Websites.

## 6.5. Views for Text Analytics

A key aspect of conTEXT is to provide intuitive exploration and visualization options for the annotated corpora. For that purpose, conTEXT allows to plugin a variety of different exploration and visualization modules, which operate on the conTEXT data model capturing the annotated corpora. By default, conTEXT provides the following views for exploring and visualizing the annotated corpora:

- *Faceted browsing* allows users to quickly and efficiently explore the corpus along multiple dimensions (i.e. articles, entity types, temporal data) using the DBpedia ontology. The faceted view enables users to drill a large set of articles down to a set adhering to certain constraints.
- *Matrix view* shows the entity co-occurrence matrix. Each cell in the matrix reflects the entity co-occurrence by entity types (color of the cell) and by the frequency of co-occurrence (color intensity).
- *Trend view* shows the occurrence frequency of entities in the corpus over the times. The trend view requires a corpus with articles having a timestamp (such as blog posts or tweets).
- *Image view* shows a picture collage created from the entities Wikipedia images. This is an alternative for tag cloud, which reflects the frequent entities in the corpora by using different image sizes.
- *Tag cloud* shows entities found in the corpus in different sizes depending on their prevalence. The tag cloud helps to quickly identify the most prominent entities in the corpora.

[20] http://schema.rdfs.org/mappings.html

[21] An online demo of the real-time semantic analysis for Twitter is available at http://context.aksw.org/resa.

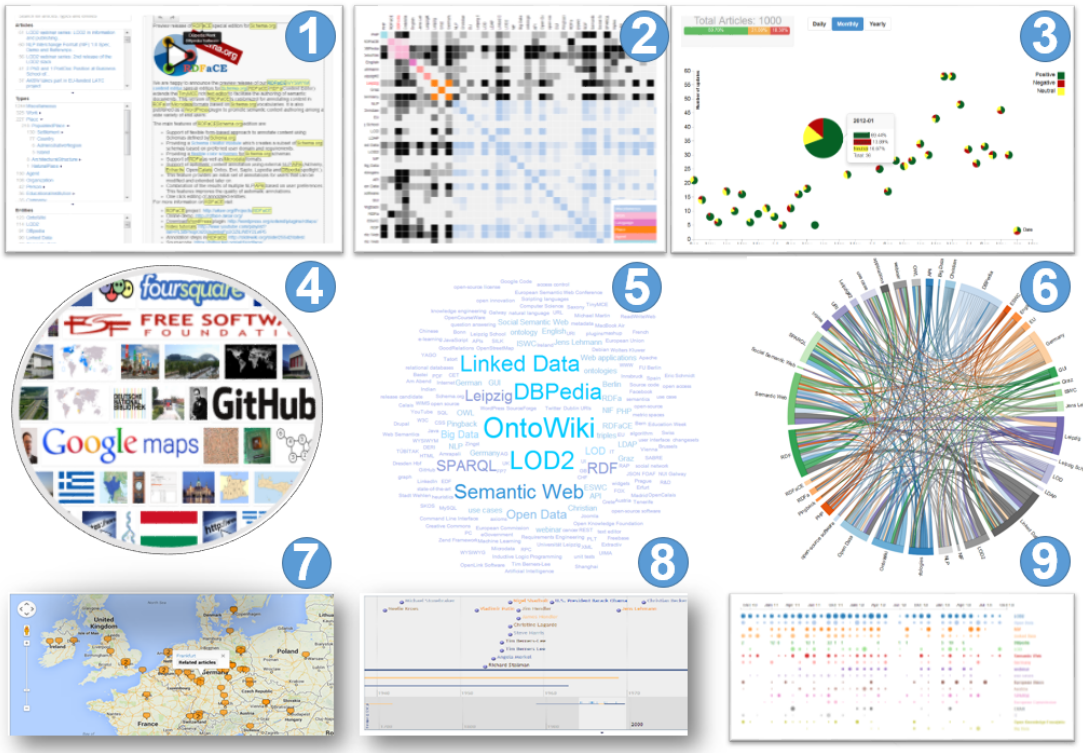


Figure 6.5.: Different views on an analyzed corpus: 1) faceted browser, 2) matrix view, 3) sentiment view 4) image view, 5) tag cloud, 6) chordal graph view, 7) map view, 8) timeline, 9) trend view.

- *Chordal graph view* shows the relationships among the different entities in a corpus. The relationships are extracted based on the co-occurrence of the entities and their matching to a set of predefined natural language patterns.
- *Places map* shows the locations and the corresponding articles in the corpus. This view allows users to quickly identify the spatial distribution of locations refereed to in the corpus.
- *People timeline* shows the temporal relations between people mentioned in the corpus. For that purpose, references to people found in the corpus are enriched with birth and death days found in DBpedia.
- *Sentiment view* shows the overall sentiment of the corpus as well as the sentiment of the individual articles in the corpus.

| Processing stage | Component | Input | Output |
|---|---|---|---|
| *Information access* | RSS/Atom feeds<br>RDF/SPARQL endpoints<br>REST APIs<br>Custom crawlers & scrapers | Textual or semi-structured Web resources | Corpus with metadata (e.g. temporal annotations) |
| *Named Entity Recognition* | DBpedia Spotlight<br>FOX | Corpus | Semantically annotated corpus |
| *Enrichment, authoring & feedback* | BOA<br>RDFaCE | Semantically annotated corpus | Automatically and manually enriched semantic annotations |
| *Visualization & exploration* | Faceted browsing<br>Map view<br>Timeline view<br>Tag cloud<br>Chordal graph view<br>Matrix view<br>Sentiment view<br>Trend view | Semantically annotated and enriched corpus | Exploration and visualization widgets leveraging various semantic annotations |

Table 6.2.: conTEXT's extensible architecture supports a variety of plug-able components for various processing and interaction stages.

## 6.6. Implementation

conTEXT is a Web application implemented in *PHP* and *JavaScript* using a relational database backend (MySQL). The application makes extensive use of the Model-View-Controller (MVC) architecture pattern and relies heavily on *JSON* format as input for the dynamic client-side visualization and exploration functionality.

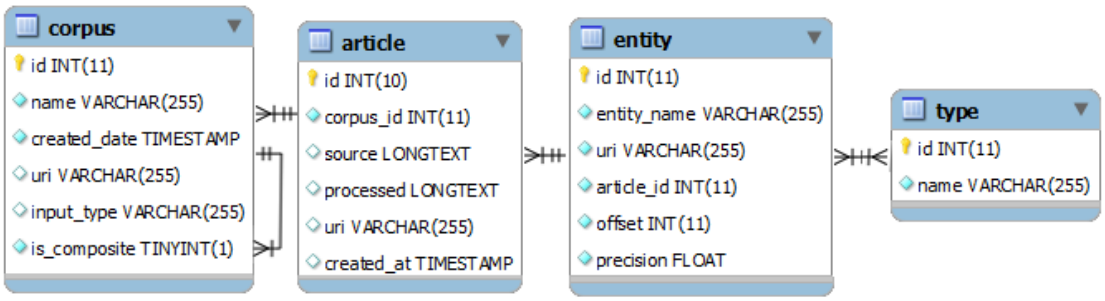


Figure 6.6.: conTEXT data model.

Figure 6.6 shows the conTEXT data model, which comprises `Corpus`, `Article`, `Entity` and `Entity_Type` tables to represent and persist the data for text analytics. A corpus is composed of a set of articles or a set of other corpora (in case of a mixed corpus). Each article includes a set of entities represented by URIs and an annotation score. The `Entity_type` table stores the type(s) for each entity. As described in Section 6.4, conTEXT employs NIF for interoperability between different NLP services as well as different corpora. Figure 6.7 shows a sample NIF annotation stored for an article. In order to create the required input data structures for different visualization views supported by D3.js and Exhibit, we implemented a *data transformer* component. This component processes, merges and converts the stored NIF formats into the appropriate input formats for visualization layouts (e.g. D3 Matrix layout or Exhibit Map layout). After the transformation, the converted visualization input representations are cached on the server-side as JSON files to increase the performance of the system in subsequent runs.

One of the main design goals during the development of conTEXT was modularity and extensibility. Consequently, we realized several points of extensibility for implementation. For example, additional visual analysis views can be easily added. Additional NLP APIs and data collectors can be registered (cf. Table 6.2). The faceted browser based on Exhibit can be extended in order to synchronize it with other graphical views implemented by D3.js and to improve the scalability of the system. Support for localization and internationalization can be added into the user interface as well as to the data processing components.

```
{   "@article": "http://blog.aksw.org/2013/dbpedia-swj",
    "@context": "http://persistence.uni-leipzig.org/nlp2rdf/ontologies/nif-core#",
    "resources": [
        {
            "@id": "http://dbpedia.org/resource/DBpedia",
            "anchorOf": "DBpedia",
            "beginIndex": "1144",
            "endIndex": "1151",
            "@confidence": "0.9",
            "@type": "DBpedia:Software"
        },
        {
            "@id": "http://dbpedia.org/resource/Freebase_(database)",
            "anchorOf": "Freebase",
            "beginIndex": "973",
            "endIndex": "981",
            "@confidence": "0.9",
            "@type": "DBpedia:Misc"
        },
        ...
    ]
}
```

Figure 6.7.: Generated semantic annotations represented in NIF/JSON.

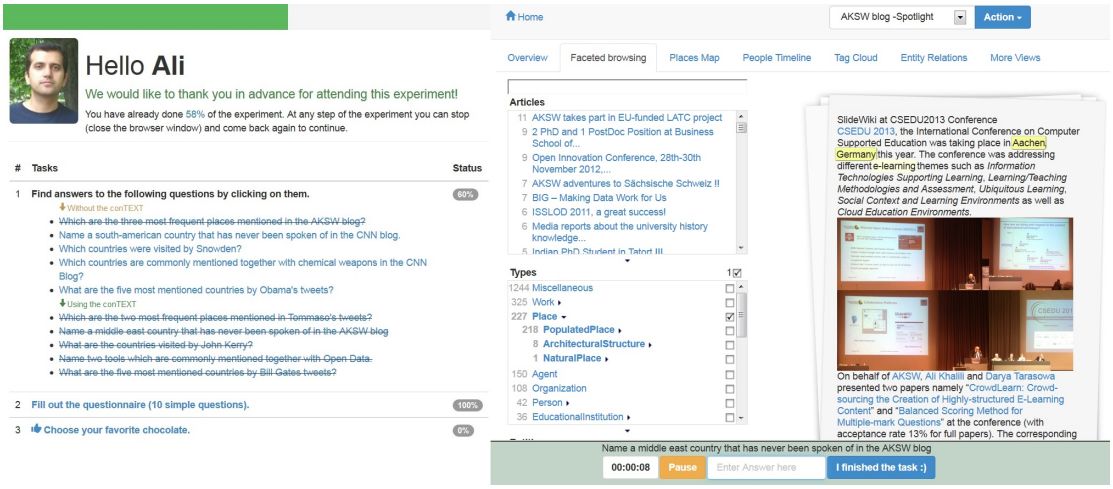


Figure 6.8.: conTEXT task evaluation platform: Left – task view showing the tasks assigned to an evaluation subject, Right – individual task.

## 6.7. Evaluation

The goal of our evaluation was two-fold. First, we wanted to provide quantitative insights in the usefulness of conTEXT. To this end, we carried out a task-driven usefulness study where we measured the improvement in efficiency and effectiveness that results from using conTEXT. Second, we aim to evaluate the usability of our approach.

### 6.7.1. Usefulness study

**Experimental Setup** To achieve the first goal of our evaluation, we carried out controlled experiments with **25** users (20 PhD students having different backgrounds from computer software to life sciences, 2 MSc students and 3 BSc students with good command of English) on a set of 10 questions pertaining to knowledge discovery in corpora of unstructured data. For example, we asked users the following question: "What are the five most mentioned countries by Bill Gates tweets?". The 10 questions were determined as follows: We collected a set of 61 questions from 12 researchers of the University of Leipzig. These questions were regarded as a corpus and analyzed using conTEXT. After removing questions that were quasi-duplicates manually, we chose 10 questions that we subdivided into 2 sets of 5 questions. Each of users involved in the evaluation was then asked to solve one set of questions with conTEXT and the other one without the tool. In all cases, the users were given access to the corpus from which the question was extracted. While answering the questions with conTEXT, the users used the analysis abilities of conTEXT. Else, they were allowed to use all digital search media of their choice except conTEXT. To ensure that we did not introduce any bias in the results due to distribution of hard questions across the two sets, one half of the users was asked to solve the first set of questions with conTEXT while the others did the same with the second set and vice-versa. We evaluated the users' efficiency by measuring the time that they required to answer the questions. Note that the users were asked to terminate any task that required more than 5 minutes to solve. In addition, we measured the users' effectiveness by comparing the answers of each user to a gold standard which was created manually by the authors. Given that the answers to the questions were sets, we measured the similarity of the answers $A$ provided by the each user and the gold standard $G$ by using the *Jaccard* similarity of the two sets, i.e., $\frac{|A \cap G|}{|A \cup G|}$. The platform[22] provided users with a short tutorial on how to perform the tasks using conTEXT and how to add their responses for the questions (cf. Figure 6.8).

**Results** The results of our first series of evaluations are shown in Figures 6.9 and 6.10. On average, the users required 136.4% more time without conTEXT than when using the tool. A fine-grained inspection of the results suggests that our approach clearly enables users to perform tasks akin to the ones provided in the evaluation in less time. Especially complex tasks such as "Name a middle-eastern country that has never been spoken of in the AKSW blog" are carried out more than three times faster using conTEXT. In some cases, conTEXT even enables users to carry out tasks that seemed out of reach before. For example, the question "What are the five most mentioned countries by Bill Gates' tweets?" (Q10) was deemed impossible to answer in reasonable time by using normal search tools by several users. A look at the effectiveness results suggests that those users who tried

[22] available at http://context.aksw.org/app/evaluation

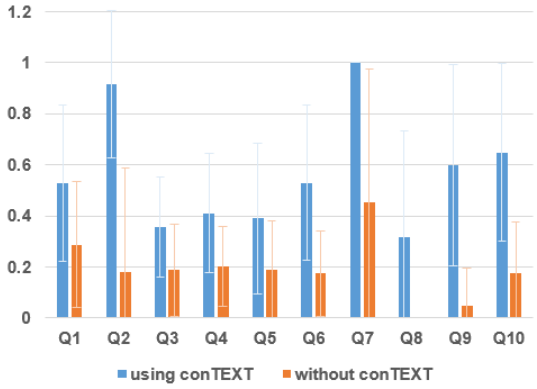


Figure 6.9.: Avg. Jaccard similarity index for answers using & without the conTEXT.

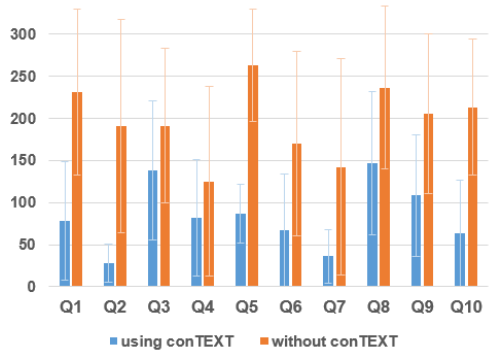


Figure 6.10.: Avg. time spent (in second) for finding answers using & without the conTEXT.

to carry out these task without conTEXT failed as they achieve an average Jaccard score of 0.17 on this particular task while users relying on conTEXT achieve 0.65. The overall Jaccard score with conTEXT lies around 0.57, which suggests that the tasks in our evaluation were non-trivial. This is confirmed by the overall score of 0.19 without conTEXT. Interestingly, the average effectiveness results achieve by users with conTEXT are always superior to those achieved without conTEXT, especially on task Q8, where users without conTEXT never found the right answer. Moreover, in all cases, the users are more time-efficient when using conTEXT than without the tool.

### 6.7.2. Usability study

**Experimental Setup** The goal of the second part of our evaluation was to assess the usability of conTEXT. To achieve this objective, we used the standardized, ten-item Likert scale-based *System Usability Scale (SUS)* [Lewis and Sauro, 2009] questionnaire and asked each person who partook in our usefulness evaluation to partake in the usability evaluation. The questions were part of a Google questionnaire and can be found at `http://goo.gl/JKzgdK`.

**Results** The results of our study (cf. Figure 6.11) showed a mean usability score of **82** indicating a high level of usability according to the SUS score. The responses to question 1 suggests that our system is adequate for frequent use (average score to question 1 = 4.23 ± 0.83) by users all of type (4.29 ± 0.68 average score for question 7). While a small fraction of the functionality is deemed unnecessary by some users (average score of 1.7± 0.92 to question 2, 1.88±1.05 to question 6 and 1.76±1.09 to question 8), the users deem the system easy to use (average score of 4.3± 0.59 to question 3). Only one user suggested that he/she would need a

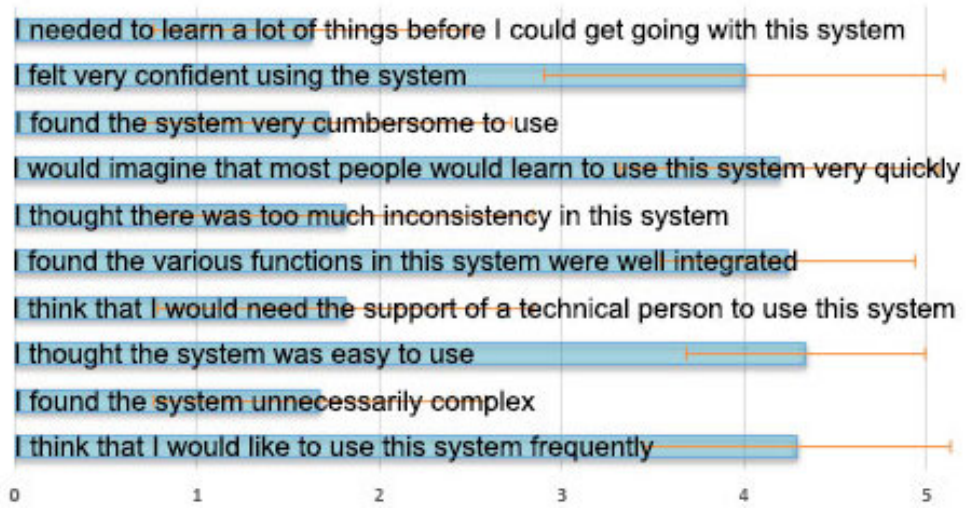


Figure 6.11.: Result of conTEXT usability evaluation using SUS questionnaire.

technical person to use the system, while all other users were fine without one. The modules of the system in itself were deemed to be well integrated (4.23±0.66 average score to question 5). Overall, the output of the system seems to be easy to understand (4.11 ± 1.05 score to question 9) while users even without training assume themselves capable of using the system (1.52± 0.72 to question 10). These results corroborate the results of the first part of our evaluation as they suggest that conTEXT is not only easy to use but provides also useful functionality.

## 6.8. Conclusion

This chapter addressed the research question RQ4 (cf. Section 1.3) to exploit semantically-enriched content for content analysis. With conTEXT, we showcased an innovative text analytics application for end-users, which integrates a number of previously disconnected technologies. In this way, conTEXT is making NLP technologies more accessible, so they can be easily and beneficially used by arbitrary end-users. With regards to RQ4.1, conTEXT provides users with instant benefits for manual content annotation by empowering users to gain novel insights and to complete tasks, which previously required substantial development.

Chapter 7

# WYSIWYM for Authoring of E-Learning Content

> "There are three ingredients in the good life: learning, earning and yearning."
> — *Christopher Morley*

In this chapter we present an application called *SlideWiki* for collaborative authoring of semi-structured educational content. SlideWiki employs the WYSIWYM concept for user-friendly authoring of semi-structured e-learning content – in particular, presentations, slides, diagrams and self-assessment tests. In order to support collaboration and crowdsourcing, SlideWiki utilizes our proposed data model called *WikiApp*. Two use cases of SlideWiki as a platform of authoring of OpenCourseWare and as a tool for elicitation and sharing of corporate knowledge are also described in this chapter.

The rest of the chapter is organized as follows: Section 7.1 describes our proposed data model WikiApp for supporting collaboration and crowdsourcing. In Section 7.3, we introduce SlideWiki as an implementation of WikiApp data model together with two use cases. Section 7.4 elaborates on the architecture and technical implementation details of SlideWiki application. In Section 7.5, we provide a comparison between SlideWiki and existing presentation management systems. Results of our usability evaluation are reflected in Section 7.6. Finally we conclude the chapter in Section 7.7.[1]

## 7.1. WikiApp Data Model

Ward Cunningham's *Wiki* [Leuf and Cunningham, 2001] paradigm is mainly *only* applied to unstructured, textual content thus limiting the content structuring, repurposing and reuse. More recently with the appearance of Semantic Wiki's, the concept was also applied and extended to semantic content[Schaffert et al., 2008]. There are currently two types of Semantic Wikis. *Semantic Text Wikis*, such as Semantic MediaWiki [Krötzsch et al., 2007] or KiWi [Schaffert et al., 2009] are based on semantic annotations of the textual content. *Semantic Data Wikis*, such as OntoWiki [Auer et al., 2006], are based on the RDF data model in the first place. Both types of Semantic Wikis, however, suffer from two disadvantages. Firstly,

[1] The contents of this chapter have been published as [Khalili et al., 2012b, Tarasowa et al., 2013, Auer et al., 2013, Tarasowa et al., 2014]. Some parts of this chapter are written jointly by Darya Tarasowa (`http://aksw.org/DaryaTarasowa`).

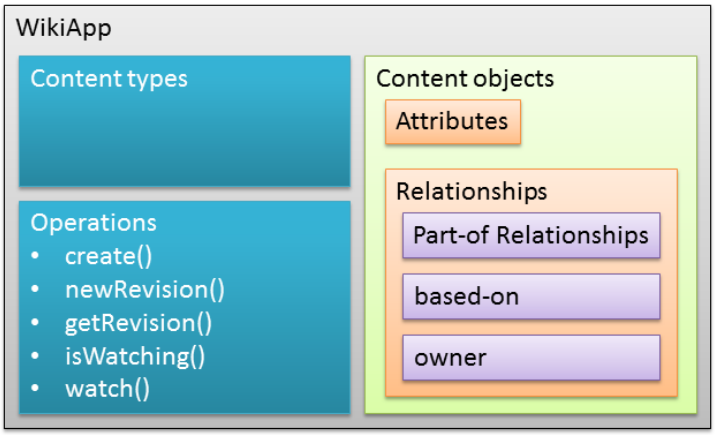


Figure 7.1.: Schematic view of the WikiApp data model.

their *performance and scalability* is restricted by current triplestore technology (cf. Section 2.5), which is still an order of magnitude slower when compared with relational data management, which is regularly confirmed by SPARQL benchmarks such as BSBM [Bizer and Schultz, 2009]. Secondly, Semantic Wikis are generic tools, which are not particularly adapted for certain domains thus substantially increase the usage complexity for users. The latter problem was partially addressed by OntoWiki components such as Erfurt API[2], RDFauthor[3] and Semantic Pingback[4], which evolved OntoWiki into a framework for Web Application development [Heino et al., 2009].

In many potential usage scenarios, the content to be managed by a wiki is neither purely textual *nor* fully semantic. Often (semi-)structured content (e.g. presentations, educational content, laws, skill profiles etc.) should be managed and the collaboration of *large* user communities around such content should be effectively facilitated.

In this section we introduce the fundamental WikiApp concept. The WikiApp concept is based on the following principles:

- *Provenance.* The origin and creation context of all information in a WikiApp implementation should be preserved and well documented.
- *Transparency, openness and peer-review.* Content in a WikiApp implementation should be visible and easily observable for the largest possible audience, thus facilitating review and quality improvements.
- *Simplicity.* WikiApp implementations should be simple to build and use.

[2] https://github.com/AKSW/Erfurt
[3] http://aksw.org/Projects/RDFauthor
[4] http://aksw.org/Projects/SemanticPingback

- *Social collaboration.* Following other users, watching the evolution of content as well as reusing and re-purposing of content in social collaboration networks is at the heart of WikiApp.
- *Scalability.* WikiApp implementations should be scalable and be implementable according to established Web application development practices (such as the MVC pattern).

The aim of the WikiApp concept is to provide a framework for implementing these principles similarly to Ward Cunningham's Wiki concept for traditional text wikis. However, due to the increased complexity of the (semi-)structured content and operations on this content just a high level description of principles is not sufficient to support the creation of domain-specific WikiApp implementations. By devising a formal WikiApp concept we aim to provide a clear and consistent description of the approach, which simplifies the creation of concrete WikiApp instantiations and can be used as a basis for integration WikiApp support into engineering methodologies, development frameworks as well as model-driven code generators. In the sequel, we present a formal description of the WikiApp data model and describe then the base operations on this data model.

### 7.1.1. Data Model

The WikiApp data model is a refinement of traditional Entity-Relation (ER) data model. It adds some additional formalisms in order to make *users* as well as ownership, *part-of and derived-from* relationships first-class citizens of the data model. We illustrate the WikiApp model in Figure 7.1 and formally define it as follows:

**Definition 4** (WikiApp data model)**.** *The WikiApp data model $\mathcal{WA}$ can be formally described by a triple $\mathcal{WA} = (U, T, O)$ with:*

- *$U$ a set of* users.
- *$T$ a set of* content types *with associated property types $P_t$ having this content type as their domain.*
- *$O = \{O_{t \in T}\}$ with $O_t$ being* sets of content objects *for each content type $t \in T$. Each $O_t$ consists of content objects $o_{t,i} = \{P_{t,i}, b_{t,i}, u_{t,i}, c_{t,i}\}$ with:*
  - *$i \in I_T$ being a suitable* identifier *set for the content objects in $O_t$;*
  - *properties $P_{t,i} = Attr_{t,i} \cup Rel_{t,i} \cup Part_{t,i}$ with $Attr_{t,i}$ being a set of literal, possibly typed* attributes*, $Rel_{t,i}$ being a set of* relationships *with other content objects, $Part_{t,i}$ being a set of* part-of and has-part relationships *referring to other content objects;*
  - *$b_{t,i} \in O_t \cup NULL$ referring to* base content object *from which this content object was derived;*

- $u_{t,i} \in U$ *referring to a user being the* owner *of this content object;*
- $c_{t,i}$ *containing the creation timestamp of object* $o_{t,i}$.

The WikiApp data model assumes that all content objects are versioned using the timestamp $c_{t,i}$ and the base content object relation $b_{t,i}$. In practice, however, usually only a subset of the content objects are required to be versioned. For auxiliary content (such as user profiles, preferences etc.) it is usually sufficient to omit a base content object relation. For reasons of simplicity of the presentation and space restrictions we have omitted a separate consideration of such content here. However, this is in fact just a special case of the general WikiApp data model, where the base content object relation $b_{t,i}$ is empty for a subset of the content objects.

The WikiApp data model is compatible with *both* the relational data model as well as the RDF data model. When implemented as relational data, content types correspond to tables and content objects to rows in these tables. Functional attributes and relationships as well as the owner and base-content-object relationships can be modeled as columns (the latter three representing foreign-key relationships) in these tables. For $1-n$ and $m-n$ relationships and non-functional attributes suitable helper tables have to be created. The implementation of the WikiApp data model in RDF is slightly more straightforward: content types resemble classes and content objects instances of these classes. Attributes and relationships can be attached to the classes via `rdfs:domain` and `rdfs:range` definitions and directly used as properties of the respective instances. For reasons of scalability we expect the WikiApp data model to be mainly used with relational backends. However, using techniques such as Triplify [Auer et al., 2009] or other RDB2RDF [Sahoo et al., 2009] mapping techniques a Linked Data interface can be easily added to any WikiApp implementation (cf. Section 7.4).

**Example 7.1** [SlideWiki data model] For our SlideWiki example application (whose implementation is explained in detail in Section 7.4) the data model consists of individual slides (consisting mainly of HTML snippets and some meta-data), decks (being ordered sequences of slides and sub-decks), media assets (which are used within slides) as well as themes (which are associated as default styles with decks and users):

- $T = \{deck, slide, media, theme\}$
- $Attr_{deck} = \{title \to text, abstract \to text,$
  $license \to \{CC-BY, CC-BY-SA\}\},$
  $Rel_{deck} = \{default\_theme \to theme\},$
  $Part_{deck} = \{deck\_content \to deck \cup slide\}$
- $Attr_{slide} = \{content \to text, speaker\_note \to text,$
  $license \to \{CC-BY, CC-BY-SA\}\},$
  $Rel_{slide} = \{uses \to media\},\ Part_{slide} = \{\}$

- $Attr_{media} = \{type \rightarrow \{image, video, audio\}, uri \rightarrow string, license \rightarrow \{CC-BY, CC-BY-SA\}\}$, $Rel_{media} = \{\}$, $Part_{media} = \{\}$
- $Attr_{theme} = \{title \rightarrow string, css\_definition \rightarrow text\}$, $Rel_{theme} = \{\}$, $Part_{theme} = \{\}$

### 7.1.2. Operations

After we introduced the WikiApp data model, we now describe the main operations on the data model. In the spirit of the Wiki paradigm, there is no deletion or updating of existing, versioned content objects. Instead new revisions of content objects are created and linked to their base objects via the $b_{t,i}$ relation.

**Definition 5** (WikiApp operations)**.** *Five base operations are defined on the WikiApp data model:*

- $create(u, t, p) : U \times T \times P_t \rightarrow O_t$ *creates a new content object of type* $t$ *with the owner* $u$ *and properties* $p$*.*
- $newRevision(u, t, i, p) : U \times T \times I_T \times P_t \rightarrow O_t$ *creates a copy of an existing content object* $o_{t,i}$ *of type* $t$ *potentially with a new owner* $u$ *and overriding existing properties with* $p$*.*
- $getRevision(t, i) : T \times I_T \rightarrow O_t \cup false$ *returns the existing content object* $o_{t,i}$ *of type* $t$ *including all its properties or false in case a content object of type* $t$ *with identifier* $i$ *does not exist.*
- $isWatching(u, t, i) : U \times T \times I_T \rightarrow \{true, false\}$ *returns true if the user* $u$ *is watching the content object of type* $t$ *with identifier* $i$ *or false otherwise. Following users is a special case, where the content object type is set to user.*
- $watch(u, t, i) : U \times T \times I_T \rightarrow \{true, false\}$ *toggles user* $u$ *watching the content object of type* $t$ *with identifier* $i$ *and returns the new watch status.*

All operations have to be performed by a specific user and the newly created content objects will have this user being associated as their owner. In addition, when a new revision of an existing content object is created and the original content object is indicated to be part of another content object (by the distinguished part-of relations) the creation of a new revision of the containing content object has to be triggered as well. In our Example 1, this is, for example, triggered when a user creates a new revision of a slide being part of a deck. If the user is not the owner of the containing deck, a new deck revision is automatically created, so as to not implicitly modify other users' decks.

```
Deck:
  attributes:
    title: {type:string}
    abstract: {type:text}
    license: {type:enum,values:['CC-BY','CC-BY-SA']}
  relations:
    Theme:{name:default_theme}
  parts:
    Deck: ~
    Slide: ~
Slide:
  attributes:
    content: {type:text}
    speaker_note: {type:text}
    license: {type:enum,values:['CC-BY','CC-BY-SA']}
  relations:
    Media:{name:uses}
Media:
  attributes:
    type: {type:enum,values:['image','video','audio']}
    uri: {type:string}
    license: {type:enum,values:['CC-BY','CC-BY-SA']}
Theme:
  attributes:
    title: {type:string}
    css_definition: {type:text}
```

Figure 7.2.: Instantiation of the WikiApp DSL representing the SlideWiki model.

## 7.2. Model-driven generation of WikiApp implementations

Using a model-driven Web application engineering approach, developers are able to easily and quickly implement WikiApp applications. We devised a *Domain-Specific Language (DSL)* based on the WikiApp Data Model and a *transformation approach* implemented in a tool called *Wikifier*[5] which receives a WikiApp definition in the DSL and generates the appropriate database (or RDF) schema, classes and methods for interacting with this model as well as the required SQL (or SPARQL) queries. The Wikifier DSL is dedicated to the specific WikiApp *problem representation technique*. In essence its a YAML-formatted[6] file with the definition of content types, their attributes, relations and part-of relations according to the WikiApp data model (cf. Definition 4). Figure 7.2 shows an instantiation of this DSL for our SlideWiki example application.

The Wikifier model transformation is integrated into the code generator of the

[5]Available at: http://slidewiki.aksw.org/wikifier/
[6]YAML Ain't Markup Language: http://yaml.org/

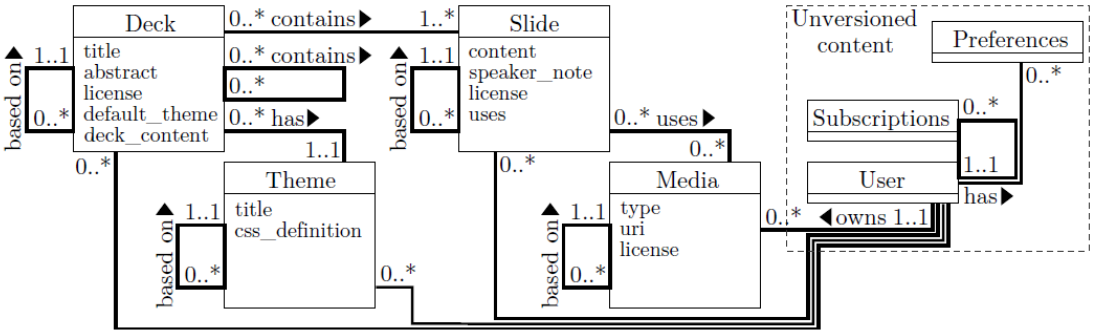


Figure 7.3.: Generated database schema by Wikifier.

*Symfony* framework[7] which is based on the *MVC* design pattern. It transforms the DSL instantiation to the corresponding data models with basic *Create-Retrieve-Update-Delete (CRUD) operations* and the corresponding views and controllers. The generated models will include the following extensions derived from the WikiApp data model:

(a) A `revision` model for each content object with `timestamp` and `based_on` properties.

(b) A `partOf` model which includes the identifier properties of the selected revision models.

(c) A `subscription` model which is used for following revision models.

(d) A `user` model which is referred by each of the generated revision models.

The database schema generated by Wikifier for SlideWiki example is depicted in Figure 7.3. In addition to the generic WikiApp operations (cf. Definition 5) Wikifier creates convenience methods for performing these operations directly from the respective content object classes.

## 7.3. SlideWiki

In this section we describe with SlideWiki a concrete WikiApp implementation, which we created to demonstrate the effectiveness and efficiency of the WikiApp approach. SlideWiki is available publicly at `http://slidewiki.org`. The main idea of SlideWiki is to enable *crowdlearning* – the crowdsourcing of educational content, in particular presentations. The SlideWiki data model was already introduced in Example 1 and shows the relatively complex relationships between decks, slides, media assets and themes. In the sequel, we present two use cases of the SlideWiki application.

[7] `http://symfony.com/`

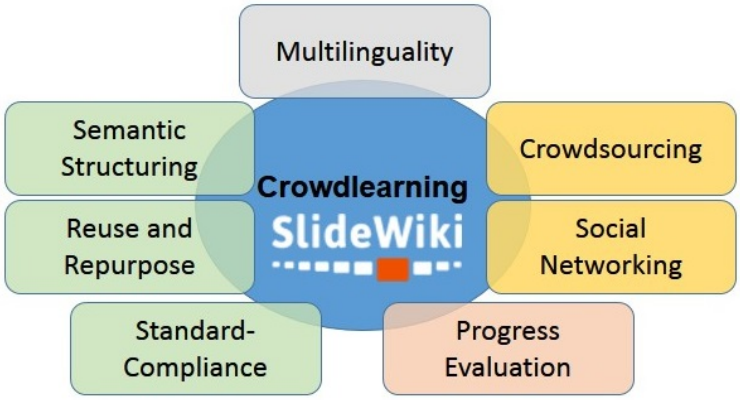


Figure 7.4.: Crowdlearning strategies in SlideWiki.

### 7.3.1. Authoring of OpenCourseWare

While nowadays there is a plethora of Learning Content Management Systems (LCMS), the collaborative, community-based creation of rich e-learning content is still not sufficiently well supported. Few attempts have been made to apply crowdsourcing and Wiki-approaches for the creation of e-learning content. *Wikiversity*[8], for example, is a Wikimedia Foundation project aiming to leverage standard wiki technology for the creation of hypertext e-learning content. *Peer 2 Peer University (P2PU)*[9] and *PlanetMath*[10] are other examples which employ crowdsourcing to create rich e-learning content. P2PU helps users to navigate the wealth of open education materials and supports the design and facilitation of courses. The PlanetMath is a project aiming to become a central repository for mathematical knowledge on the web, with a pedagogical mission. However, we deem, that no real attempt has been made so far to truly apply the concepts behind Wikis and crowdsourcing to develop a specifically tailored technology supporting the creation of (semi-)structured e-learning content.

As defined by Open Education Consortium[11], OpenCourseWare (OCW) is a free and open digital publication of high quality college and university-level educational materials. OCW are free and openly licensed, accessible to anyone, anytime via the internet. As an OCW authoring platform, SlideWiki deals with two types of (semi-)structured learning objects: slide presentations and assessment tests. SlideWiki empowers communities of instructors, teachers, lecturers, academics to create, share and re-use multilingual educational content in a collaborative way.

As depicted in Figure 7.4, to support crowdlearning, SlideWiki realizes the following strategies:

- *Standard-compliance.* SlideWiki adopts the *Sharable Content Object Refer-*

[8] http://wikiversity.org/
[9] http://p2pu.org/
[10] http://planetmath.org/
[11] http://www.openedconsortium.org

*ence Model (SCORM)* standard [ADL, 2011a] and practical recommendations [ADL, 2011b] and expands the standard for the collaborative model. This will decrease the costs associated with building high-quality e-learning content by importing/exchanging content from/between existing LCMSs.

- *Semantic structuring.* Instead of dealing with large learning objects (often whole presentations or tests), SlideWiki employes the WikiApp data model to decompose learning material into fine-grained *learning artifacts*.

- *Reuse and repurpose.* Instead of the full redevelopment, the content can slightly evolve in SlideWiki. This will decrease the cost of content creation, will increase the quality of e-learning content and will support the evolution and adaptation to new requirements.

- *Crowdsourcing.* There are already vast amounts of amateur and expert users, which are collaborating and contributing on the Social Web. Harnessing the power of such crowds in SlideWiki can significantly enhance and widen the distribution of e-learning content.

- *Social networking.* The theoretical foundations for e-Learning 2.0 are drawn from *social constructivism* [Wang et al., 2012]. It is assumed that students learn as they work together to understand their experiences and create meaning. SlideWiki supports social networking activities (e.g. following, discussing, sharing, rating slides and presentations) to enable students to proactively interact with each other to acquire knowledge.

- *Support of multilinguality.* SlideWiki enables the crowd-translation of content to promote open e-learning material among different countries.

- *Progress evaluation.* SlideWiki supports the creation of questions and self-assessment tests based on slide material. This will enable users to evaluate their progress while learning.

SlideWiki brings the following benefits for academic learning:
It enables *educators*, *lecturers* and *teachers* to

- increase the user base by making the content accessible to a world-wide audience.
- get their high-quality e-learning content translated into many different languages.
- engage students in contributing to and discussing the slides.
- create (self-)assessment tests for students.
- involve peer-educators in improving and maintaining the quality and attractiveness of their e-learning content.
- increase the reputation in the community, by sharing qualitative e-learning content.

*Students* can also

- view rich-learning content right in a browser.
- discuss particular content (e.g. a slide or question) with other students and instructors.
- contribute additional content, improvements and feedback.
- assess learning progress using the questionnaires attached to presentations.

### 7.3.2. Elicitation and Sharing of Corporate Knowledge

In medium and large enterprises and organizations presentations are crucial elements of the corporate knowledge exchange. Such organizations are mostly hierarchically organized and communication and knowledge flows usually accompany corporate hierarchies. In addition to sending emails and documents, meetings where presentations are shown to co-workers, subordinates and superiors are one of the most important knowledge exchange functions. Research conducted by the Annenberg School of Communications at UCLA and the University of Minnesota's Training & Development Research Center show that executives on average spend 40-50% of their working hours in meetings.[12] They spend a remarkable amount of time collecting their required materials and creating new presentations. The challenges with current organizational presentations can be roughly divided into the following categories:

- *Sharing and reuse of presentations.* Much of the corporate strategy, direction and accumulated knowledge is encapsulated in presentation files; yet this knowledge is effectively lost because slides are inaccessible and rarely shared. Furthermore offline presentations are hard to locate. Thereby executives usually spend their time creating new slides instead of re-using existing material.

- *Collaborative creation of presentations.* Executives in different departments or countries often unknowingly duplicate their efforts, wasting time and money. To collaboratively create a presentation, the members need to manually download and merge the presentations.

- *Following/discussing presentations.* Finding the most up-to-date presentation is difficult and time-consuming, therefore costly. Furthermore, discussing the content of presentations in face-to-face meetings or email discussions is not efficient within organizations.

- *Tracking/handling changes in presentations.* Tracking and handling changes that occur within different presentations is a time-consuming task which needs opening all offline presentations and manually comparing their content. Additionally there are hundreds of slide copies to change when an original is modified. This cascading change costs a fortune each time.

[12] http://www.shirleyfinelee.com/MgmtStats

- *Handling heterogeneous presentation formats.* Presentations can be created in different formats (e.g. Office Open XML, Flash-based, HTML-based or LaTeX-based presentations) thereby integration and reuse of them will be a cumbersome task for organization members.

- *Ineffective skills management and training.* Medium and large enterprises are obliged by law to provide means for training and qualification to their employees. This is usually performed by seminars, where training material is prepared in the form of presentations. However, it is usually not possible to provide engaging bi-directional and interactive means of knowledge exchange, where employees contribute to the training material.

- *Preserving organization identity.* Having a consistent template and theme including the logo and brand message of organization is of great significance in shaping the organization identity. With offline presentations it is difficult to persistently manage and sustain specific organization templates. Everyone needs to take care of templates and themes individually and managing the changes takes a remarkable amount of time.

SlideWiki as a crowdsourcing platform deals with most of the above-mentioned limitations of current presentation tools within organizations. As a tool for knowledge management within organizations, SlideWiki can be applied to the following areas:

**Developing a shared mental model within organization.** In organizational learning, learning occurs through shared insights and mental models. In this process, organizations obtain the knowledge that is located in the minds of their members or in the epistemological artifacts (maps, memories, policies, strategies and programs) and integrates it with the organizational environment [Valaski et al., 2012]. This shared mental model (a.k.a. *organizational memory*) is the accumulated body of data, information, and knowledge created in the course of an individual organization's existence.

**Combining presentations with social approaches for crowdsourcing.** Presentations when combined with *crowdsourcing* and *collaborative social approaches* can help organizations to cultivate innovation by collecting and expressing the individual's ideas within organizational social structures. As discussed in [Blankenship and Ruona, 2009], there are different types of social structures living in the context of organizations. Work groups, project teams, strategic communities, and learning communities, communities of practice, informal networks, etc. to mention some. These social structures make use of presentations frequently to present and discuss their internal ideas. Therefore, creating an integrated collaborative platform for authoring and sharing presentations will result in exchanging knowledge within and cross these social structures (even supporting inter-organizational knowledge transfer).

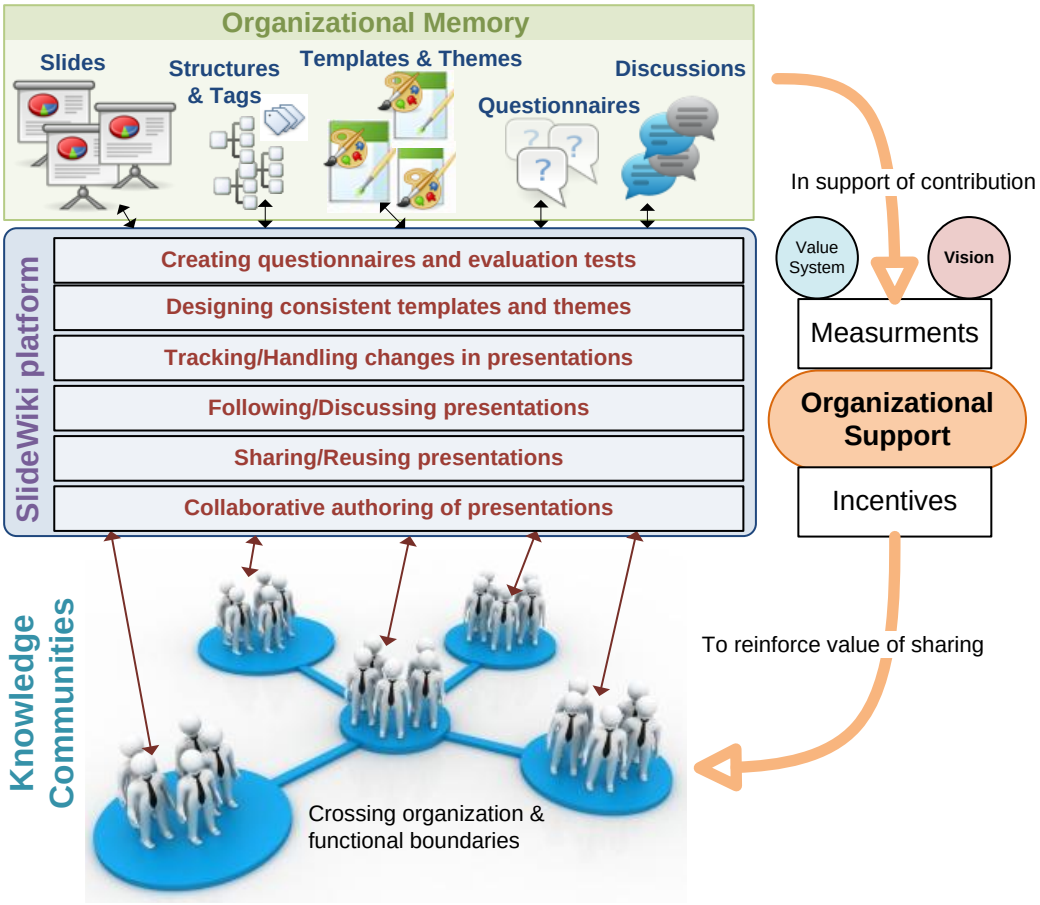


Figure 7.5.: SlideWiki ecosystem for organizational knowledge sharing.

**As a driver for organizational innovation.** Presentations are an important driver of organizational innovation particularly when they are exchanged between social connections that cross functional and organizational boundaries. As discussed in [Fonstad, 2005], improvising is a structured process of innovation that involves responding to changing situations with resources at hand by creating a production and adapting it continuously. Presentation tools enable the creation of so called *Structural Referents* – a representation one develops about a structure. Structural referents support the communities to collaborate on individual's ideas and foster the potential ideas in alignment with the organizational goals. *Ghost Sliding* is a process introduced in [Fonstad, 2005] which utilizes presentation slides as structural referents for collaborative knowledge management. Ghost sliding is an iterative process where consultants draw up quick, rough representations of each slide and discuss them with clients to develop consensus on what statements are going to be included in the final presentation and what data needs to be collected to support those statements. The rationale for ghost-sliding is that by developing explicit

representations of what a consultant is striving for, the consultant could discuss the hypotheses with others and be more efficient about what kind of data to look for.

**As a media for knowledge exchange and training.** As reported in [Cobb and Steele, 2011], PowerPoint presentations are the most used (75.4 %) tool for developing e-learning content within organizations. Presentations contain visualized learning materials, which improve the training of organization members having different levels of knowledge. Enabling users to contribute to these training materials makes it possible to provide engaging bi-directional and interactive means of knowledge exchange.

SlideWiki provides a crowdsourcing platform for elicitation and sharing of corporate knowledge using presentations. It exploits the wisdom, creativity and productivity of the crowd for the collaborative creation of structured presentations. Figure 7.5 shows the SlideWiki ecosystem for supporting organizational knowledge management. SlideWiki provides a collaborative environment, which enables knowledge communities to contribute to dynamic parts of organizational memory, which is encapsulated in presentations. The dynamic view of the structure of organizational memory [Casey and Olivera, 2011] takes into account the social nature of memory. Rather than viewing memory as knowledge stored in a collection of retention bins, the emphasis is on memory as continually constructed and reconstructed by humans interacting with each other and their organizational environment.

In SlideWiki, users from different knowledge communities crossing the organization and functional boundaries can collaboratively create structured online presentations. Users can assign tags and categories for structuring the presentations. The created presentations can be shared and reused to build new synergetic presentations. Users can also track and manage changes occurring within presentations using a revisioning system. Additionally, SlideWiki includes an e-learning component that deals with questionnaires created for each presentation slide. Questionnaires together with the evaluation tests facilitate the training of users within organizations. With regard to preserving the organization identity and branding, SlideWiki supports creating and sharing of templates and themes. Apart from the contribution on authoring of presentation content, SlideWiki also supports social networking activities such as following presentation decks, slides and users as well as discussing the created content.

## 7.4. Implementation

The SlideWiki application makes extensive use of the MVC architecture pattern. The MVC architecture enables the decoupling of the user interface, program logic and database controllers and thus allows developers to maintain each of these components separately. As depicted in Figure 7.6, the implementation comprises

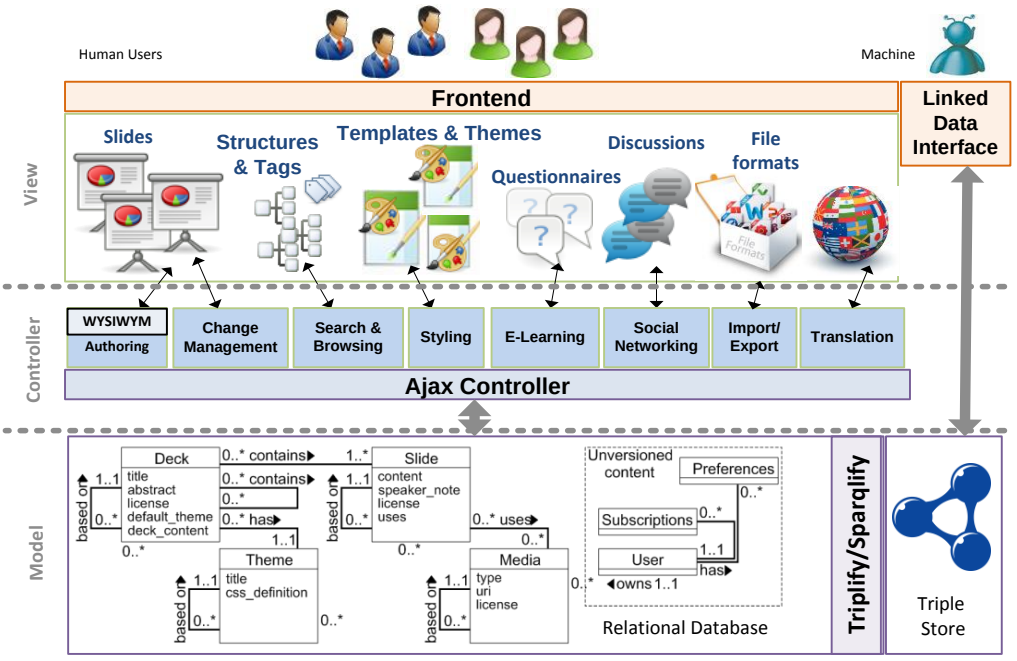


Figure 7.6.: Bird's eye view on the SlideWiki MVC architecture.

the main components: WYSIWYM authoring, change management, search and browsing, styling, e-learning, social networking, import/export, translation as well as linked data interface.. We briefly walk-through these components in the sequel.

**WYSIWYM Authoring.** SlideWiki employs the WYSIWYM interface model together with an inline HTML5 based WYSIWYG text editor for authoring the presentation slides (cf. Figure 7.7). Using this approach, users will see the slideshow output at the same time as they are authoring their slides. The editor is implemented based on ALOHA editor[13] extended with some additional features such as image manager, source manager, equation editor. The inline editor uses Scalable Vector Graphics (SVG) images for drawing shapes on slide canvas. Editing SVG images is supported by SVG-edit[14] with some predefined shapes which are commonly used in presentations. For logical structuring of presentations, SlideWiki utilizes a tree structure together with a context menu by which users can append new or existing slides/decks and drag & drop items for positioning. When creating presentation decks, users can assign appropriate tags as well as footer text, default theme/transition, abstract and additional meta-data to the deck.

[13] http://aloha-editor.org/
[14] http://code.google.com/p/svg-edit/

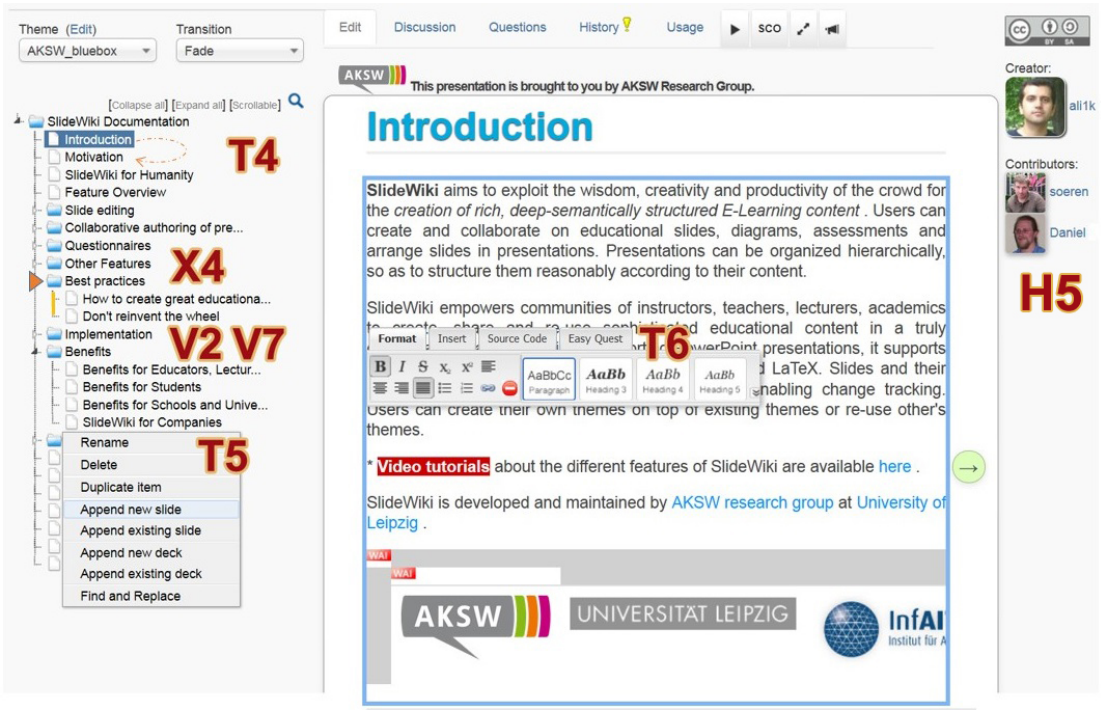


Figure 7.7.: Screenshots of the SlideWiki WYSIWYM interface (V2 – text margin formatting for visualizing content tree, V7 – line connectors to show the relation between slides and decks, X4 – expanding & drilling down to explore content, T4 – drag & drop to change the order of slides and decks, T6 – floating ribbon editing to author slide content, H5 – collaboration and crowdsourcing helper components).

**Change management.** Revision control is natively supported by WikiApp data model. We just define rules and restrictions to increase the performance. There are different circumstances in SlideWiki for which new slide or deck revisions have to be created. For decks, however, the situation is slightly more complicated, since we wanted to avoid an uncontrolled proliferation of deck revisions. This would, however, happen due to the fact, that every change of a slide would also trigger the creation of a new deck revision for all the decks the slide is a part of. Hence, we follow a more retentive strategy. We identified three situations that have to cause the creation of new revisions:

- The user specifically requests to create a new deck revision.
- The content of a deck is modified (e.g. slide order is changed, change in slides content, adding or deleting slides to/from the deck, replacing a deck content with new content, etc.) by a user which is neither the owner of a deck nor a member of the deck's editor group.

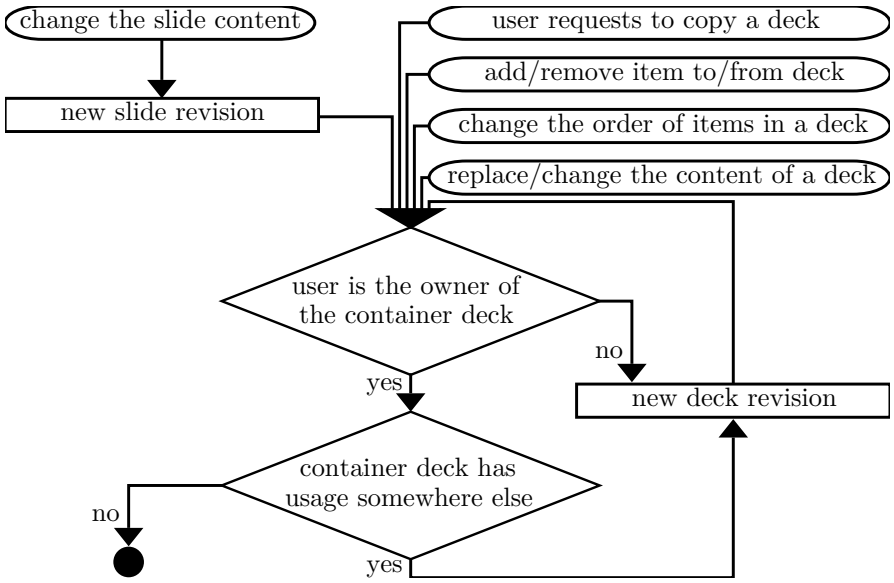


Figure 7.8.: Decision flow during the creation of new slide and deck revisions.

- The content of a deck is modified by the owner of a deck but the deck is used somewhere else.

The decision flow is presented in Figure 7.8. In addition, when creating a new deck revision, we always need to recursively spread the change into the parent decks and create new revisions for them if necessary.

**Search and Browsing.** There are three ways of searching in SlideWiki: by *keywords*, by *metadata* and by *user* (who contributed or follows certain content). We combined keywords and tag search so that users can either 1) search by keywords and then add a tag filter, or 2) show all slides or decks having the tag and then running an additional keyword search on the results. In both cases an ordering a user might have applied is preserved for subsequent searches. In addition to the deck tree user interface for browsing the presentations, a breadcrumb navigation bar is implemented in SlideWiki. Breadcrumb improves the accessibility of system by increasing the user awareness when browsing nested presentations.

**Styling.** In order to create flexible and dynamic templates and styles for presentations, SlideWiki utilizes *Saas* (Syntactically Awesome Stylesheets) language[15]. Sass extends CSS by providing several mechanisms available in programming languages, particularly object-oriented languages, but not available in CSS3 itself. When Sass script is interpreted, it creates blocks of CSS rules for various selectors as defined by the Sass file. Using Saas, SlideWiki users can easily create and reuse presentation themes and transitions.

[15] http://sass-lang.com/

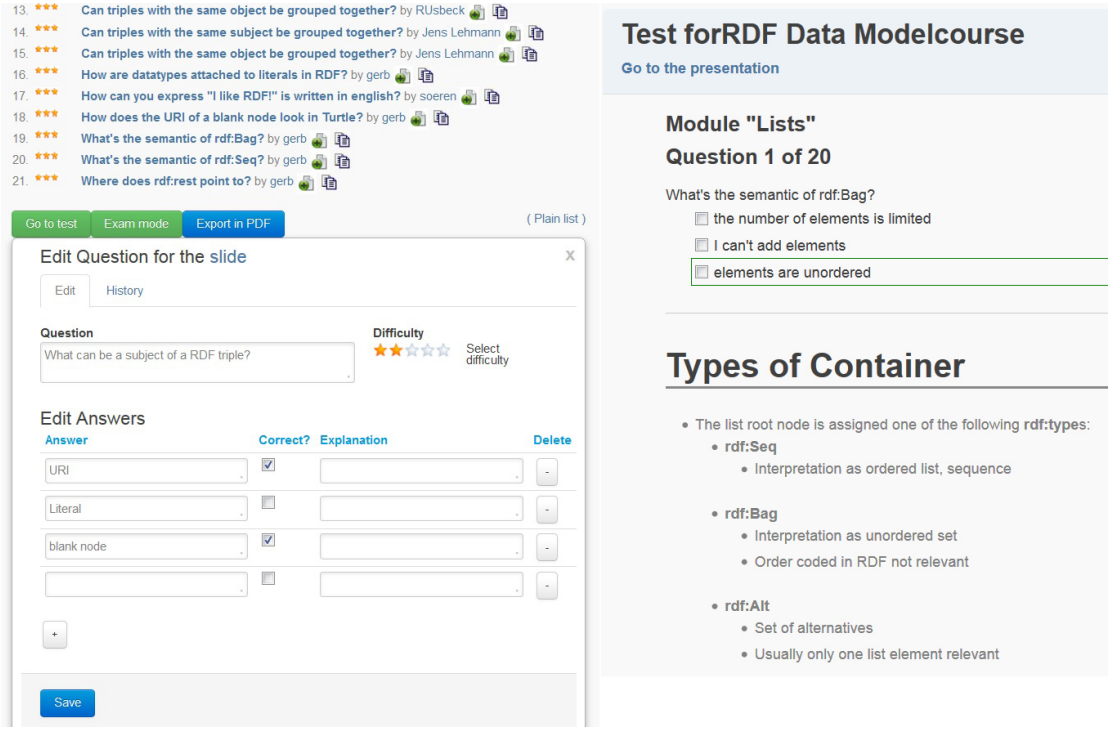


Figure 7.9.: Editing of a question & Test mode in SlideWiki.

**E-learning.** SlideWiki supports the creation of questions and self-assessment tests based on slide material. Each question has to be assigned to at least one slide. Important note here, that the question is assigned not to the slide revision, but to slide itself. Thus, when a new slide revision appears, it continues to include all the list of previously assigned questions. Questions can be combined into tests. The *automatically created* tests include the last question revisions from all the slides within the current deck revision. Manually created tests present a collection of chosen questions (cf. Figure 7.9).

**Social Networking.** As a social software, SlideWiki supports different types of social networking activities. Users can follow items such as decks, slides and other users. They can also rate, tag and discuss decks and slides. Content syndication in multiple formats such as RSS, ATOM, OPML and JSON is provided for created items so that users can subscribe to them. We are currently integrating SlideWiki with popular social networking sites like Twitter, Facebook, GooglePlus and LinkedIn.

**Import/Export.** SlideWiki implementation addresses *interoperability* as its first class citizen. SlideWiki supports import/export of the content from/to existing desktop applications and *Learning Objects RepositoryLearning Objects Repositories*

*(LORs)* thereby allowing users from other LCMSs to access the created content. The main data format used in SlideWiki is HTML. However, there are other popular presentation formats commonly used by desktop application users, such as PowerPoint .pptx presentations, LaTeXand others. We implemented import of the slides from .pptx format and work on the LaTeXformat support is in progress.

**Translation** Our architecture allowed us to implement a translation feature backed by the Google Translate service. After the translation into one of 54 supported languages, the presentation can be edited independently from the original one.

**Linked Data Interface.** While sharing and reusing educational data across institutional and national boundaries is a general goal for both the public and the private education sector, the last decade has seen a large amount of research dedicated to Web-scale interoperability. For example, `LinkedEducation.org` is an open platform which promotes the use of Linked Data for educational purposes. In order to enable the export of SlideWiki content on Data Web as LORs, we employed the RDB2RDF mapping tool Triplify [Auer et al., 2009] to map SlideWiki content to RDF and publish the resulting data on the Data Web. The Triplify configuration for SlideWiki was created manually according to IEEE Learning Objects Metadata (LOM) standard and can be changed to support specific LORs. The SlideWiki Triplify Linked Data interface is available via: `http://slidewikiw.org/triplify`.

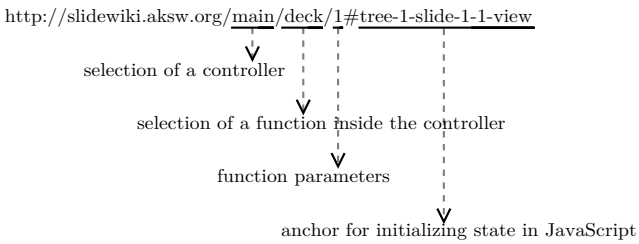


Figure 7.10.: Mapping of URLs to MVC actions in SlideWiki.

**Frontend.** In addition to overall MVC pattern, SlideWiki utilizes a client-side MVC approach (implemented in JavaScript and running inside the users Web browser) to provide users with a rich and interactive user interface. As described in Figure 7.10, there is a hash fragment in the request URL which acts as an input for the client-side MVC handler. This fragment consists of an *identifier* and an *action* name. The identifier itself has four parts which are combined based on the following pattern:
`tree-{container_deck_id}-{content_type}-{content_id}-{content_position}`.
For example, `tree-1-slide-5-2-view` refers to the `view` action which is assigned to the slide with id 5, located at second position of deck with id 1.

The client-side MVC handler as (singleton) controller listens to the hash fragment and once a change has occurred the handler triggers the corresponding actions. Each action has a JavaScript template (implemented using *jQuery templates*) with the corresponding variable place holders. For each action an Ajax call is made and the results are returned to the controller in JSON format. Subsequently, the controller fills the templates with the results and renders them in the browser.

## 7.5. SlideWiki vs. Presentation Management Systems

There are already many Web-based platforms that provide services for online creation, editing and sharing of presentations. *SlideShare.net* is a popular Website for sharing presentations[16]. Comparing to SlideWiki, it does not provide any feature to create and reuse the content of presentations. *SlideRocket.com* and *Prezi.com* are other related works which help people to create fancy and zoomable presentations. In contrast to SlideWiki, they focus more on the visualization aspects rather than the content of the presentations. *Microsoft SharePoint Online* [17] and *SlideBank.com* are two commercial solutions which provide the feature of slide libraries to allow users to work with PowerPoint slide decks stored in the cloud. Despite SlideWiki which is an online platform, these tools adopt the Software-as-a-Service approach to enable a synchronization between desktop applications and Web service providers.

## 7.6. Usability Evaluation

SlideWiki has been already used for teaching *Business Information Systems* and *Semantic Web* lectures in the Chemnitz Technical University, University of Leipzig and University of Bonn. Figure 7.11 shows an screenshot of the Semantic Web lecture series comprising 785 slides collaboratively created by 22 Semantic Web researchers. As checked on May 2014, there were 3100 decks, 11000 deck revisions, 22000 slides, 41000 slide revisions, 2000 questions and 850 active users on SlideWiki. To evaluate the real-life usability of SlideWiki, we performed a usability user study with 22 subjects. Subjects were drawn from the members of AKSW research group at the university of Leipzig and MSc students at the Chemnitz Technical University who took the course Business Information Systems. We used the *SUS* scale to grade the usability of SlideWiki. The results of our survey showed a mean usability score of 69 for SlideWiki which indicates a reasonable level of usability (cf. Figure 7.12). In addition to quantitative results, we also collected a

[16] Other examples include *authorSTREAM* http://www.authorstream.com, *SlideServe* http://www.slideserve.com, *Scribd* http://www.scribd.com and *slideboom* http://www.slideboom.com

[17] http://sharepoint.microsoft.com

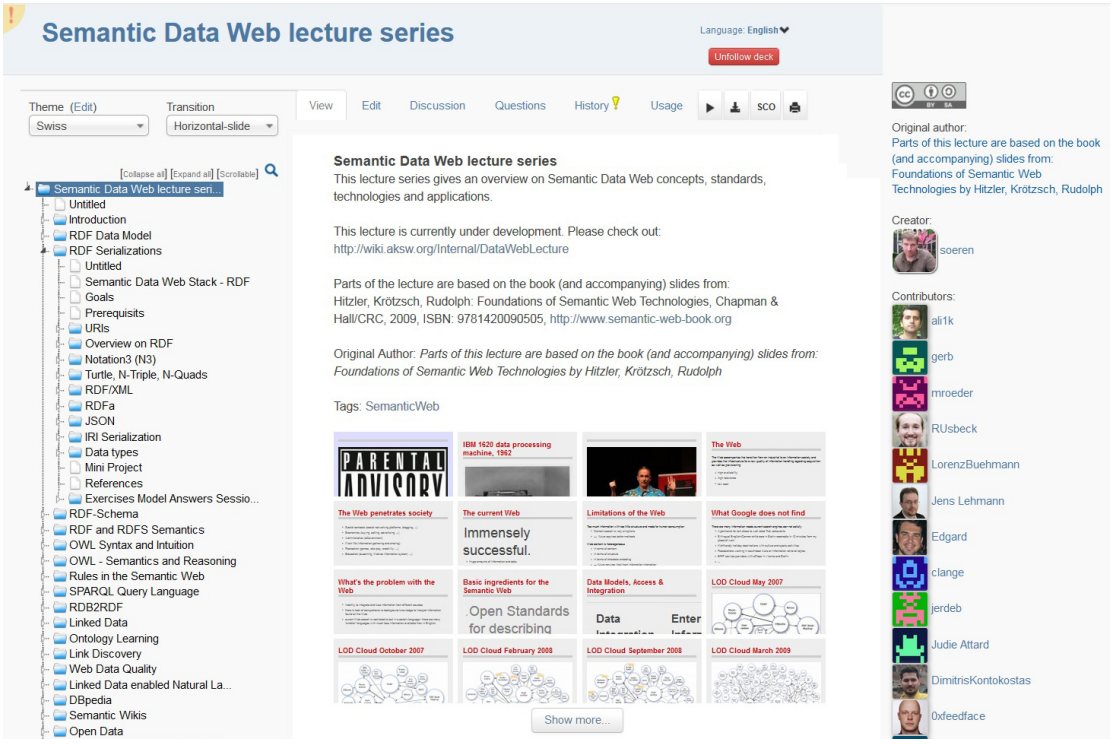


Figure 7.11.: An screenshot of the Semantic Web lecture series created collaboratively on SlideWiki.

number of user suggestions to further improve the SlideWiki platform. For instance some users suggested providing autosave feature, supporting more import/export formats, defining user groups etc.

The students were working with SlideWiki for several weeks, and we collected the statistics for that period. The experiment was not obligatory but students actively contributed by creating additional questions and fixing mistakes. During that period, they created 252 new slide revisions which some of them were totally new slides, others were improved versions of the original lecture slides. Originally the whole course had 130 questions, and students changed 13 of them, fixing the typos or adding additional options to multiple-choice questions. In total, students performed 287 self-assessment tests. The majority of these used the automatically and randomly created tests covering the whole course material. After the experiment, based on the student grades at final exam, we could claim that, more active SlideWiki users received better marks on the real examination.

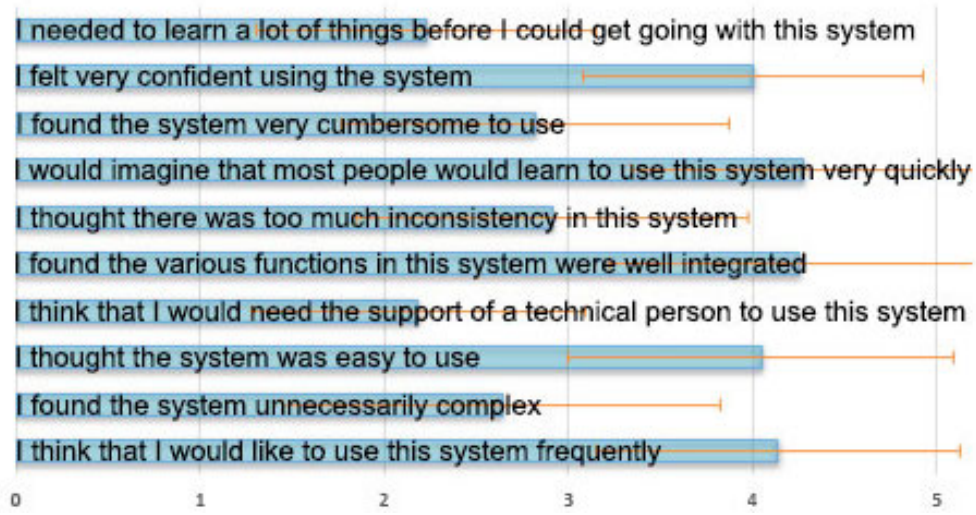


Figure 7.12.: Result of SlideWiki usability evaluation using SUS questionnaire.

## 7.7. Conclusion

This chapter addressed the research question RQ5 (cf. Section 1.3) to apply crowdsourcing and collaborative content authoring techniques to the process of semantic content authoring. We presented the SlideWiki platform for authoring of highly-structured e-learning content. SlideWiki as a crowdlearning platform enables the collaborative authoring of presentations by utilizing the WikiApp data model as well as WYSIWYM user interface model. The created presentations will help to effectively shape rich e-learning materials by utilizing crowd feedback.

Chapter 8

# WYSIWYM for Authoring of Semantic Medical Prescriptions

> "I always wanted to be somebody, but now I realize I should have been more specific."
> — *Lily Tomlin*

In this chapter, we present how WYSIWYM model can be employed and customized in specific domains to provide content interoperability. We will introduce the new concept of *Semantic Medical Prescriptions* as an application of Semantic Web technologies in e-prescription systems. Semantic prescriptions can automatically handle the medication errors occurring in prescriptions and can increase the awareness of the patients about the prescribed drugs and drug consumption in general. We will also showcase Pharmer as our implemented WYSIWYM interface to realize the creation of semantic prescriptions.

The remainder of this article is structured as follows: Section 8.1 and Section 8.2 provide a background on the basic concepts such as e-prescriptions and LODD. In Section 8.3, we describe the Pharmer as a solution to effectively create semantic prescriptions. Then we discuss the possible use cases of Pharmer in Section 8.4. To better demonstrate the possible stakeholders of the Pharmer system, an example scenario is drawn in Section 8.5. Section 8.6 reports the results of our usability evaluation and finally Section 8.7 concludes the chapter.[1]

## 8.1. E-Prescriptions

As reported in MedicineNet [Melissa Conrad Stoppler, 2012], *medication errors* are the most common type of medical errors in health care. Errors such as improper dose of medicine, adverse drug interactions, food interactions, etc. often stem from invalid prescriptions and unawareness of the patients. Medication-oriented errors are usually the result of failures during the medication process [González et al., 2011]. Electronic prescriptions, which are recently gaining attention in the e-health domain, are one of the solutions proposed to solve these types of errors. In an e-prescription system, prescriber electronically sends an accurate, error-free prescription directly to a pharmacy from the point-of-care.

[1] The contents of this chapter have been published as [Khalili and Sedaghati, 2013a]. Some parts of this chapter are written jointly by Bita Sedaghati (bita.sedaghati@uni-leipzig.de) from the institute of pharmacy, university of Leipzig.

During the recent years, the adoption of e-prescriptions has been spreading relatively rapidly. In the US, the so called *Electronic Prescribing Incentive Program* is a reporting program that uses a combination of incentive payments and payment adjustments to encourage electronic prescribing by eligible professionals.[2]. As recently published by [Galanter et al., 2013] hospitals' use of computerized prescriptions prevented 17 million drug errors in a single year in the United States. The *Canadian Medical Association* (CMA) and the *Canadian Pharmacists Association* (CPhA) have approved a joint statement on the future of e-prescribing that aims to have all prescriptions for Canadians created, signed and transmitted electronically by 2015. The Australian government removed commonwealth legislative barriers to electronic prescribing started from 2007[3]. A system called *epSOS* [4] which performs the use of e-prescriptions all around Europe, is currently passing the extensive practical testing phase.

However, one of the main challenges in current e-prescription systems is dealing with the *heterogeneity* of available information sources. There exist already different sources of information addressing different aspects of pharmaceutical research. Information about chemical, pharmacological and pharmaceutical drug data, clinical trials, approved prescription drugs, drugs activity against drug targets such as proteins, gene-disease-drug associations, adverse effects of marketed drugs, etc. are some examples of these diverse information. Managing these dynamic pieces of information within current e-prescription systems without blurring the border of the existing pharmaceutical information islands is a cumbersome task. On the other hand, *Linked Open Data* as an effort to interlink and integrate these isolated sources of information is obtaining more attention in the domain of pharmaceutical, medical and life sciences.

Combining the best practices from Linked Open Data together with e-prescription systems can provide an opportunity for patients, researchers as well as practitioners to collaborate together in a synergetic way. A consequence of introducing LD in health care sector is that it significantly changes the daily duties of the employees of the health care sector. Therefore the most challenging aspect will not be the technology but rather changing the mind-set of the employees and the training of the new technology[Puustjärvi and Puustjärvi, 2006].

## 8.2. Linked Open Drug Data (LODD)

In computing, *Linked Data (LD)* describes a method of publishing structured data so that it can be interlinked and become more useful. It builds upon standard Web technologies such as HTTP and URIs, but rather than using them to serve Web pages for human readers, it extends them to share information in a way that can be read automatically by computers. This enables data from different sources

[2]Electronic Prescribing (eRx) Incentive Program http://www.cms.gov/erxincentive
[3]http://www.medicareaustralia.gov.au/
[4]epSOS : the European eHealth Project http://www.epsos.eu/

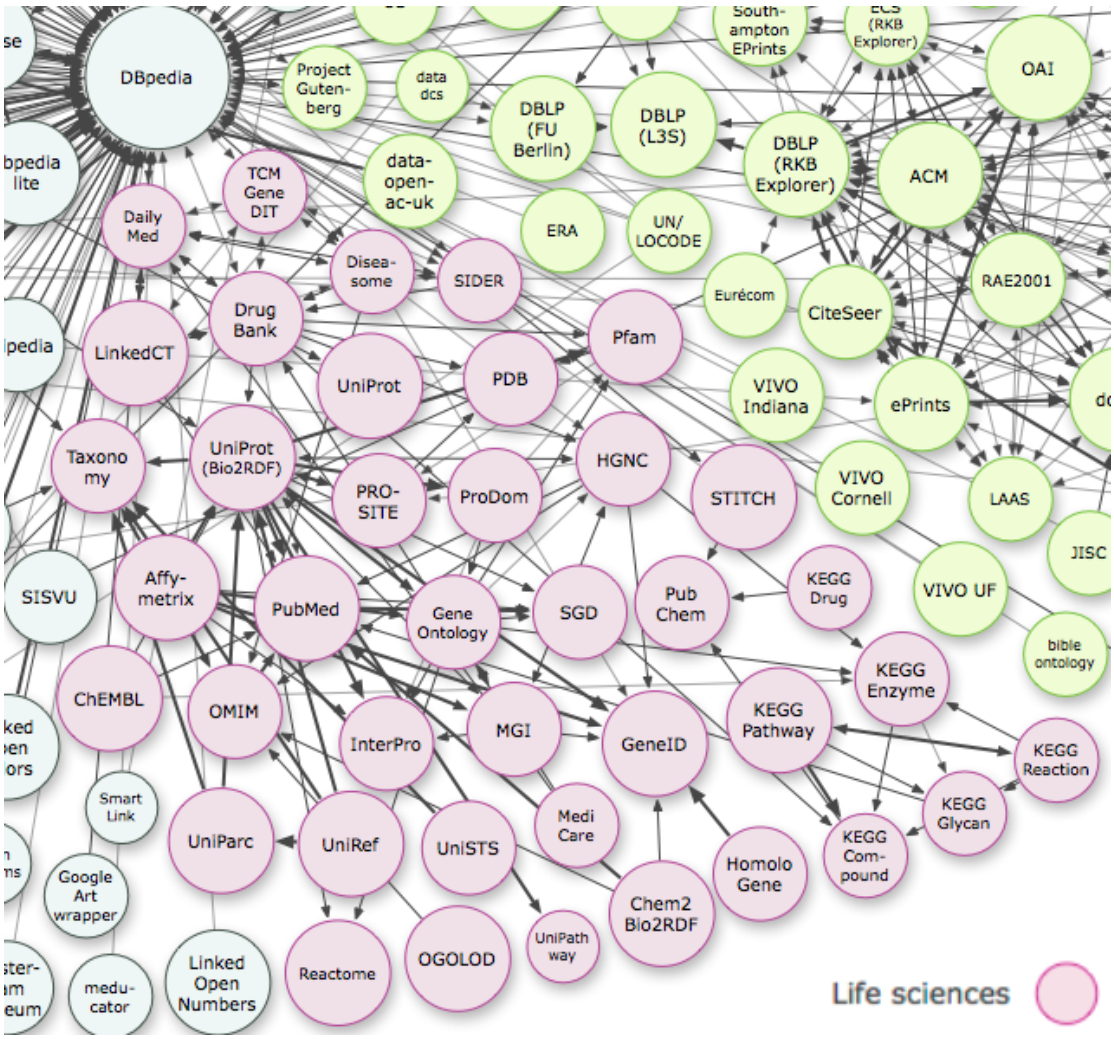


Figure 8.1.: Available datasets related to life sciences and pharmaceutical research.

to be connected and queried [Bizer et al., 2009]. Tim Berners-Lee, the inventor of the Web and LD initiator, suggested a 5 star deployment scheme for *Linked Open Data (LOD)*: 1) make your stuff available on the Web (whatever format) under an open license, 2) make it available as structured data (e.g., Excel instead of image scan of a table), 3) use non-proprietary formats (e.g., CSV instead of Excel), 4) use URIs to identify things, so that people can point at your stuff, 5) link your data to other data to provide context.

Particularly in the areas of health care and life sciences with the wealth of available data, large scale integration projects like Bio2RDF [5], Chem2Bio2RDF [6], and the W3C HCLS's (Health Care and Life Sciences) Linked Open Drug Data

[5] http://bio2rdf.org/

[6] Semantic Web in Systems Chemical Biology http://chem2bio2rdf.wikispaces.com/

(LODD)[Samwald et al., 2011a] have not only significantly contributed to the development of the Linked Open Data effort, but have also made social and technical contributions towards data integration, knowledge management, and knowledge discovery.

There are already many interesting information on pharmaceutical research available on the Web. The sources of data range from drugs general information, interactions and impacts of the drugs on gene expression, through to the results of clinical trials. LODD has surveyed publicly available data about drugs, created LD representations of the data sets, and identified interesting scientific and business questions that can be answered once the data sets are connected (cf. Figure 8.1).

**LODD Applications in Medical Domain.** There exists few approaches that address the medical and pharmaceutical applications using LODD. *TripleMap* (`http://www.triplemap.com`) is a project connecting widespread distribution of journal articles, patents and numerous databases in pharmaceutics research. TripleMap as a Web-based application provides a dynamic visual interface to integrate RDF datasets such as the LODD. Showing an unexpected associations between entities related to researcher's interest is main advantage of TripleMap inspired by the broad interconnected data available in the LODD data sets. The goal of the TripleMap project is to deliver and sustain an 'open pharmacological space' by using and enhancing the state-of-the-art Semantic Web standards and technologies [Samwald et al., 2011b].

Another related project is the Open Pharmacological Space (OPS), *Open PHACTS* (Pharmacological Concept Triple Store `http://www.openphacts.org`) project under the European Innovative Medicines Initiative (IMI `http://www.imi.europa.eu/`). The goal of this project is integration of chemical and biological data using LD standards to support drug discovery [Williams et al., 2012].

*Linked Cancer Genome Atlas Database* [Saleem et al., 2013] as another LD project aims to create an atlas of genetic mutations responsible for cancer. The project provides an infrastructure for making the cancer related data publicly accessible and to enable cancer researchers anywhere around the world to make and validate important discoveries.

Although these projects address the backend side of creating LODD applications, there has been a clear lack of applications with user-friendly, efficient and effective interfaces to make LD resources accessible to end-users outside the biomedical community. One of the use cases of LODD datasets addressed in this chapter is authoring of *Semantic Prescriptions* which are prescriptions enriched by LOD.

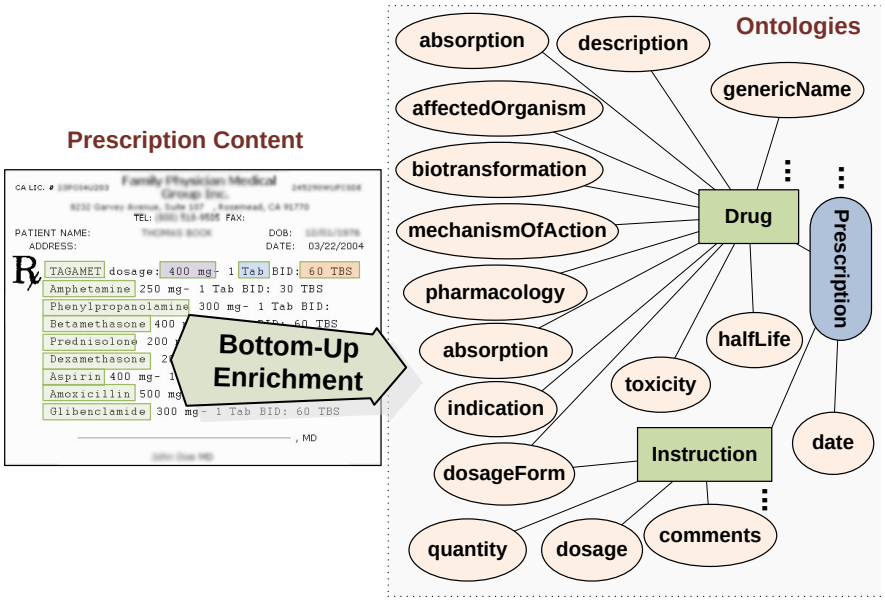


Figure 8.2.: Bottom-up semantic enrichment of prescriptions.

## 8.3. Semantic Authoring of Medical Prescriptions using Pharmer

*Semantic Medical Prescriptions* are intelligent e-prescription documents enriched by dynamic drug-related meta-data thereby know about their content and the possible interactions. As depicted in Figure 8.2, semantic prescriptions are created based on a bottom-up process (cf. Section 3.3.1) in which normal e-prescriptions (unstructured or semi-structured with lower level of expressiveness) are enriched with semantic metadata coming from a set of predefined ontologies (with upper level of expressiveness).

In order to showcase the applicability of semantic prescriptions we implemented an application called *Pharmer*. The Pharmer implementation is open-source and available for download together with an explanatory video and online demo at http://code.google.com/p/pharmer/. Pharmer provides a platform for semantically annotation of conventional e-prescriptions.

We use Schema.org MedicalTherapy and Drug vocabularies as our annotation ontologies and utilize the existing pharmaceutical linked datasets such as DBpedia, DrugBank[7], DailyMed[8] and RxNorm[9] as our domain ontology.

[7]A bioinformatics and cheminformatics resource that combines detailed drug (i.e. chemical, pharmacological and pharmaceutical) data with comprehensive drug target (i.e. sequence, structure, and pathway) available at http://www.drugbank.ca

[8]Information about marketed drugs available at http://dailymed.nlm.nih.gov

[9]A normalized naming system for generic and branded drugs available at http://www.nlm.nih.gov/research/umls/rxnorm/

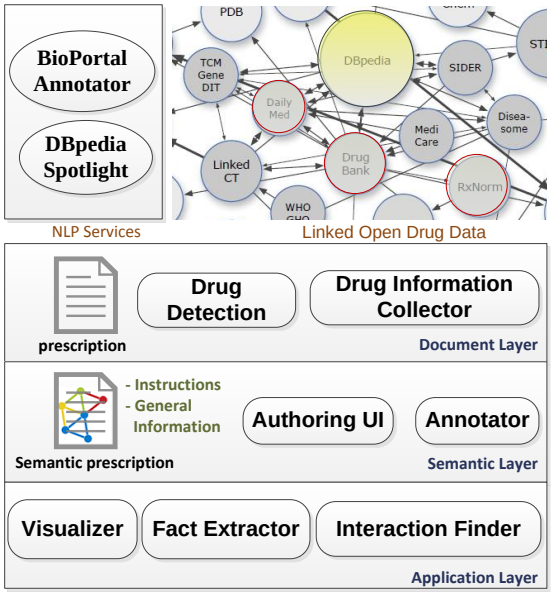


Figure 8.3.: Architecture of the Pharmer system.

## 8.3.1. Architecture

The Pharmer system architecture is depicted in Figure 8.3 and consists of three layers:

**Document Layer** This layer includes the traditional e-prescription document plus two components as *Drug Detection* and *Drug Information Collector*. Drug detection component performs the natural language processing of the e-prescription document to detect the terms referring to a drug in the prescription. The component uses *DBpedia Spotlight* [10] and *BioPortal annotator* [11] NLP services to parse and analyze the text looking for known drugs. BioPortal annotator is an ontology-based Web service that annotates public datasets with biomedical ontology concepts based on their textual metadata.

Automatic drug detection component is configurable so that users can easily add other existing NLP services for drug detection. When user is writing the prescription, this component asynchronously performs the drug recognition and adds the related annotations as real-time semantic tagging.

Another component in this layer is drug information collector which grabs all the information regarding a specific drug from LOD. To pursue this, it utilizes datasets such as DrugBank, DailyMed and RxNorm by sending federated SPARQL queries.

[10] http://spotlight.dbpedia.org/
[11] http://bioportal.bioontology.org/annotator

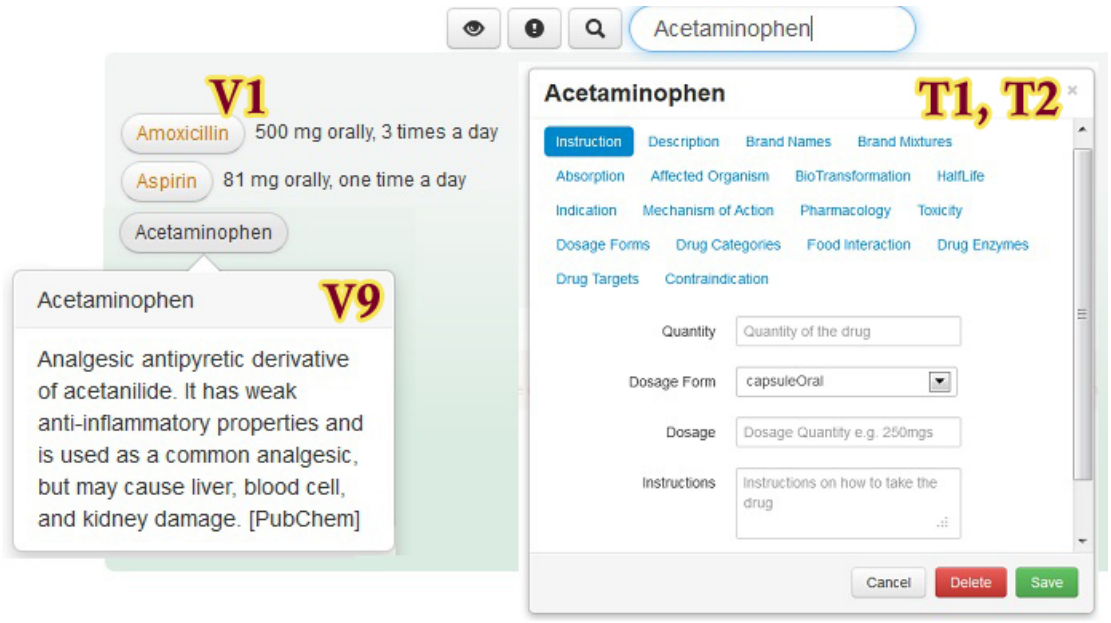


Figure 8.4.: Pharmer WYSIWYM implementation (V1 – highlighting of drugs through framing, V9 – additional information about a drug in a callout, T1/T2 combined form and inline editing of electronic prescriptions).

**Semantic Layer** There are two main components in this layer namely *Annotator* and *Authoring UI*. The annotator component handles the automatic annotation and embeds the general information of the drugs as meta-data into the e-prescription. Annotator adopts the RDFa format. The authoring UI component provides users with a set of input forms to manually embed the meta-data related to prescription instructions into the prescription document.

**Application Layer** This layer provides a set of applications on top of the generated semantic prescriptions. *Interaction Finder* checks the possible interactions between the prescribed drugs and warn the prescriber about them. *Visualizer* is responsible for graphically representing the embedded semantics of a prescription (e.g. as depicted in Figure 8.6). The *Fact Extractor* generates the RDF/Turtle representation of the semantic prescriptions.

## 8.3.2. Features

The main features of Pharmer can be summarized as:

- *WYSIWYM User Interface.* As shown in Figure 8.4, Pharmer employs the WYSIWYM concept described in Chapter 4. In Pharmer, users are able to directly manipulate the conventional e-prescriptions in order to enrich them

with semantics. The generated annotations can be viewed by different sets of user interfaces with are configurable by users. For example, users can select specific border/background colors to distinguish the annotated drugs in a prescription.

- *Providing Different Semantic Views.* Semantic views allow the generation of different views on the same metadata schema and aggregations of the knowledge base based on the roles, personal preferences, and local policies of the intended users. Pharmer suggests two types of views: generic and domain specific views. Generic views provide visual representations of drug information (e.g., as information view depicted in Figure 8.5 or graph view in Figure 8.6). Domain-specific views address the requirements of a particular domain user (e.g., a researcher need specific views for visualizing the atomic structure of chemical compounds).

- *Real-time Drug Tagging.* Real-time tagging means creating drug annotations while the user is typing. This will significantly increase the annotation speed [Heese et al., 2010]. Users are not distracted since they do not have to interrupt their current authoring task. Pharmer has a client-side component which interacts with the server asynchronously to make real-time tagging possible.

- *Drug Suggestion.* When searching for a drug, Pharmer suggests the similar drugs by taking into account the history of search terms and by sending SPARQL queries to the relevant datasets.

- *Automatic Drug Annotation.* Automatic annotation means the provision of facilities for automatic mark-up of prescriptions. The automatic process of annotating in Pharmer is composed basically of finding drug terms in prescription using an NLP service, mapping them against an ontology (i.e., DBpedia), and disambiguating common terms.

## 8.4. Possible Use Cases of Pharmer

### 8.4.1. A Ubiquitous Computing Platform for Semantic E-Prescribing

Mobile and ubiquitous computing devices are increasingly present and prevalent in the health contexts. This trend brings a number of possibilities of *mobile health* (m-health) to address critical aspects of health care and health system needs, by

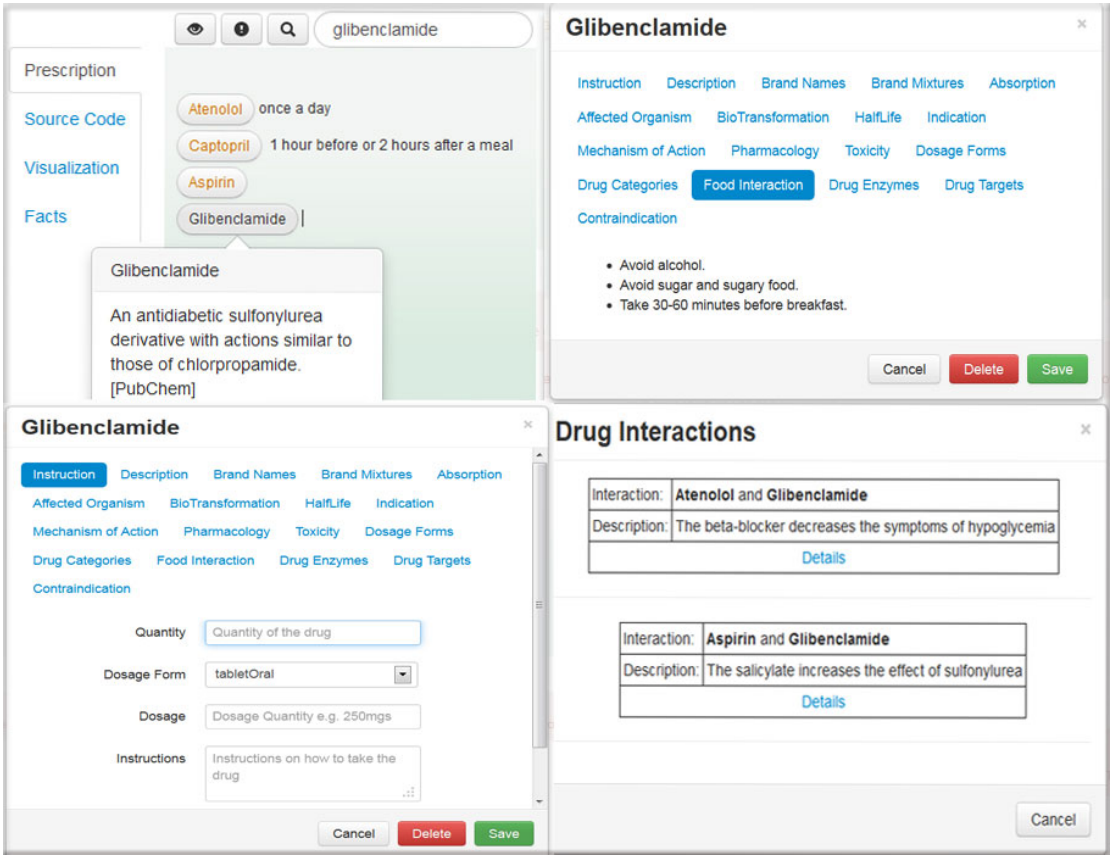


Figure 8.5.: Screenshot of the Pharmer application (top-left: general view, top-right: drug information view, bottom-left: prescription authoring view, bottom-right: drug interaction-finder results).

virtue of these devices' ubiquity, simplicity, and cost-efficiency [Petrucka et al., 2013]. In particular, in the process of semantic e-prescribing, having a mobile application will facilitate the creation of semantic medical prescriptions using any device and in any location.

Pharmer mobile application as shown in Figure 8.7[12] provides a mobile user interface for authoring of semantic prescriptions as well as accessing multi-dimensional data on medical prescriptions. Current ubiquitous devices are programmable and come with a growing set of facilities including multi-touch screens and cheap powerful embedded sensors, such as an accelerometer, digital compass, gyroscope, GPS, microphone, camera and other type of sensors. Utilizing these rich set of facilities in the context of medical prescriptions will enrich the patient medical prescription with sensor data thereby improves the quality of e-health services. For example, the location of user and some indicators like blood pressure or hear rate can be received from sensors by which Pharmer can specify the suitable drugs

[12]Available at http://bitili.com/pharmer/mobile

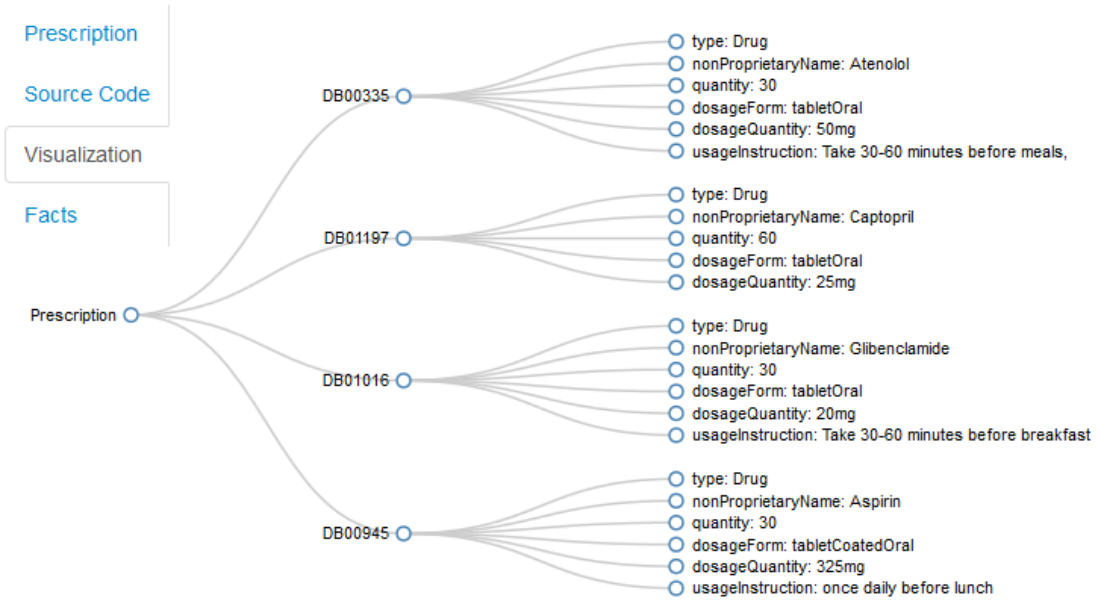


Figure 8.6.: Graph view in Pharmer.

located in pharmacies close to the user.

### 8.4.2. A Professional Social Network for Health-care Service Providers

Pharmer as a prescribing tool can be incorporated in a health care social network. Such a network composed of health care professionals and patients who collaboratively write, correct and modify prescriptions in a semantically enriched environment. This social health care network facilitates relations between patients and health care professionals in order to improve *Shared Decision Making (SDM)*. The traditional model of medical decision-making, in which doctors make decisions on treatment has no longer used in updated health care. The role of the patient, instead, in the consultation has been highlighted, mainly through introducing 'patient-centered' strategies. Therefore, nowadays the models promoting patients active involvement in the decision-making procedure becoming developed.

A model introduced by Charles et al. [Charles et al., 1997] defines shared decision making only under the following four key characteristics. These keys are:

- both the patient and the doctor are involved.
- both parties share information.
- both parties take steps to build a consensus about the preferred treatment.
- an agreement is reached on the treatment to implement.

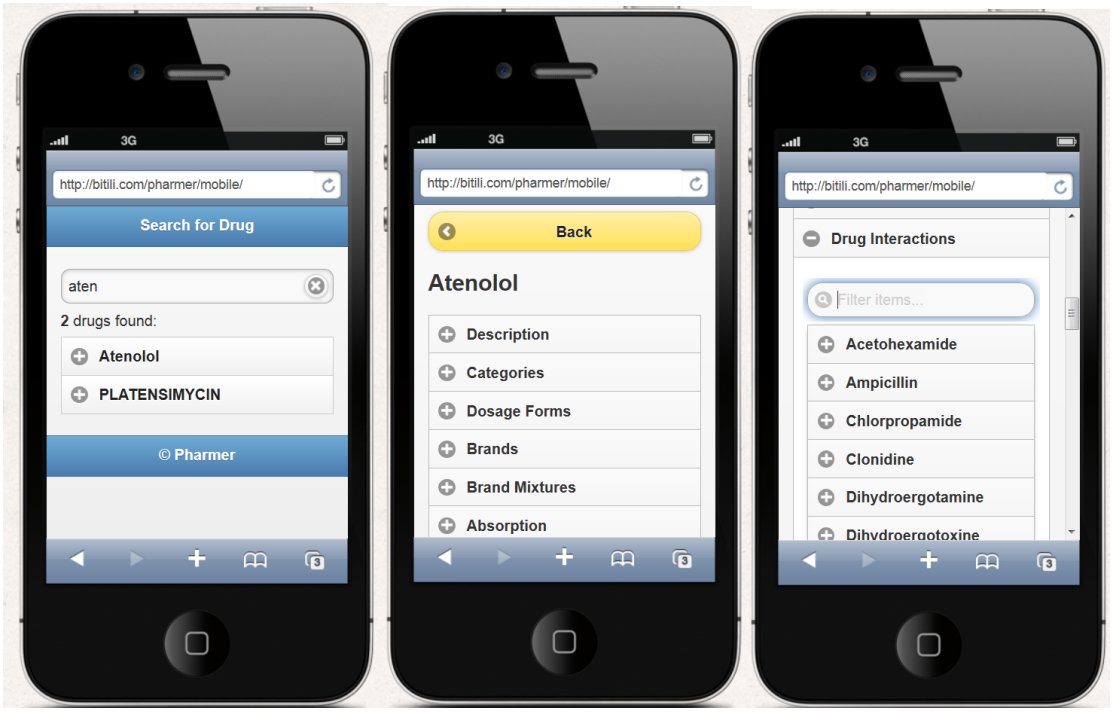


Figure 8.7.: Screenshot of Pharmer mobile application.

Pharmer facilitates the process of SDM through the connection amongst patient and physician on one hand and pharmacist on the other hand.

Access to eLODD enables Pharmer to not only get linked to e-prescribing systems but also to further assist physicians in diagnosis and treatment. Pharmer with direct connection to up-to-date information enables physicians to reconfirm their diagnosis and help them in finding proper treatment approaches. Physician, after general examination can enter the observed symptoms in Pharmer network and there, with the wealth of data available, Pharmer can assist in diagnosis followed by therapies.

## 8.5. Pharmer Stakeholders: Example Scenario

As depicted in Figure 8.8, Pharmer approach is very versatile and can be applied in a vast number of use cases by different stakeholders. The arrows in the figure can be summarized as the following:

1. The physician diagnoses the disease and writes the corresponding semantic prescription using the Pharmer, where patient's medication history is available.

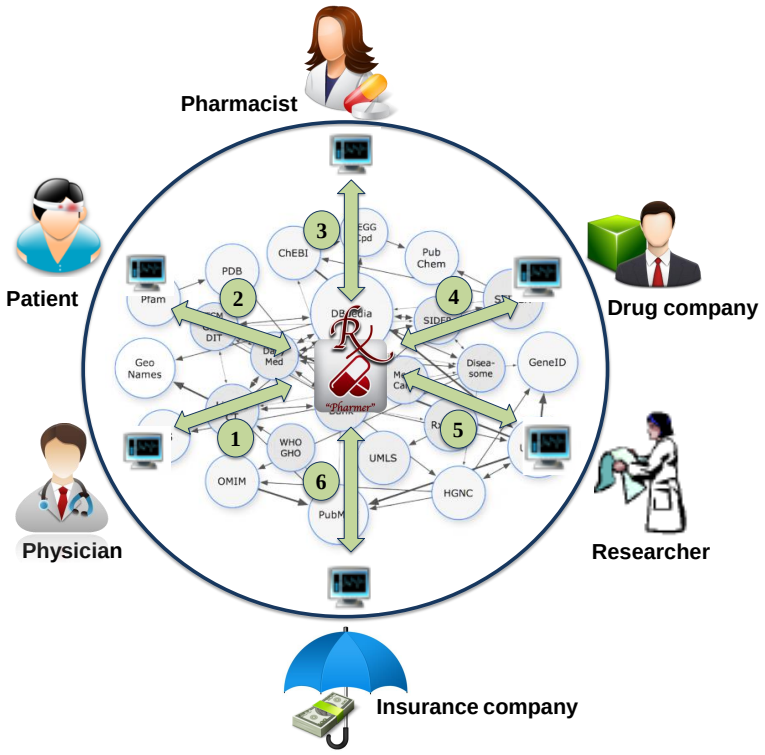


Figure 8.8.: Pharmer ecosystem.

2. The patient accesses to drug information, food interactions and adverse drug reactions via Pharmer.

3. The pharmacist verifies the prescription and considers alternative options suggested by Pharmer.

4. Pharma companies utilize the Pharmer data store in order to balance their production and distribution according to the market taste and demand.

5. The Researchers easily access to the abundant data source and prescription statistical data.

6. Pharmer informs insurance companies to perform fair coverage plans according to covered drugs and patient's medication history.

All the above stakeholders utilize Linked Open Data as their integrated information source.

As a scenario, a 63 year old man with the history of MI (Myocardial Infarction) and type 2 diabetes visits a heart and coronary specialist complaining about frequent headaches and heavy head feeling. The specialist, after general inspection and monitoring vital signs, asks for a blood test. He then considers symptoms including high blood pressure (sys/dias:158/95 mmHg) and high Fasting Blood

Sugar (150 mg/dl). He diagnoses high blood pressure and severe type 2 diabetes. Thereby, The patient profile is defined in Pharmer by patient's information besides diagnosis. "no weight loss" is mentioned as a preference in the patient's profile. Regardless of the patient's preferences, the physician would prescribe `Metformin` as a drug of choice. However, since the major side effect of `Metformin` is weight loss, the physician replaces `Metformin` with `Rosiglitazone`.

Considering the medication that the patient took before (`Glibenclamide` only), The specialist dispenses a new semantic prescription by entering the following drugs:

- **Rosiglitazone** 4 mg Oral Tablet once daily
- **Glibenclamide** 5 mg Oral Tablet bid
- **Atenolol** 50 mg Oral Tablet once daily

He then checks for the possible drug interactions by clicking the attributed button in the Pharmer software. As the Pharmer is connected to LODD, it is capable of recognizing the most recent updated drug interactions (available at Drugbank dataset). He finds out that Sulfunyl Urea class drugs (here `Glibenclamide`) are not compatible to be coadministrated with beta-blockers (here `Atenolol`). So, he needs to replace it with another drug. Using the Pharmer and its connection to Linked Open Data, the physician can find the possible alternatives. Then he decides to choose `Captopril` as replacement.

The semantic prescription is then sent to the patient's pharmacy of choice. There, the pharmacist is able to review the semantic prescription and comments on that directly in the system so that the physician is also aware of the corresponding changes. The pharmacist comments may cause minor or major modifications in the semantic prescription. For instance, using the Pharmer, she is able to check the appropriate dose of each medicine or suggest cheaper alternatives (if possible). In this case, as the `Rosiglitazone` elevates cardiovascular risks, the pharmacist suggests `Rosiglitazone` to be replaced by `Pioglitazone`. This change happens as a realization of the shared decision making between physician, pharmacist and patient. Thereafter, the patient who was referred to the pharmacy takes the prescribed drugs.

Before he starts taking the tablets, he enters in Pharmer system with his ID as patient. There, he is able to observe drug information embedded in the error free semantic prescription besides the preferred time and drug intake instructions. He is also informed about the possible food interactions. The patient's profile completes as he visits physicians or ask for refills. Furthermore, he is followed up by the physician and the pharmacist via the Pharmer. After 2 months, the patient visits another specialist for his recurrent symptoms of diabetes. The specialist via the Pharmer accesses to the patient's medical profile and increases the anti-diabetic drug dose.

A researcher in an academy research institution investigates `Captopril` (as an Angiotansion II antagonist) effect on preventing diabetes recurrence. Having the data from the aforementioned patient follow up along with other similar patients allows investigator to lead her goal. In this case, for example, the `Captopril` along with anti-diabetic drugs led to diabetes recurrence. Observing all the corresponding patient profiles will either confirm or reject the research assumption.

A Pharma company manager requires to determine the compliance rate of `Captopril` in the market in order to balance the production based on market demand. Applying the Pharmer allows him to simply access to these data and decide how to go on with this product. He is also able to collect the evidence which may prevent further dispense of `Captopril` by physicians or consumption among patients. Pharmer allows insurance companies to customize and individualize their services based on patient's medical records. Recruiting Pharmer that contains information on insured drugs, the physician can choose the drugs accordingly. In the scenario, insurance company checks the dispensed medication with the disease and patient's insurance status therefore decides to refund the patient.

## 8.6. Usability Evaluation

In order to determine whether we succeeded to facilitate the creation of semantic prescriptions using Pharmer, we performed a usability user study with 13 subjects. Subjects were drawn from 3 physicians, 4 pharmacist, 3 pharmaceutical researchers and 3 students. We first showed them a tutorial video of using different features of Pharmer[13] then asked each one to create a prescription with Pharmer. After finishing the task, we asked the participants to fill out a questionnaire, which consisted of two parts: feature usage questions and usability experience questions.

We used the *SUS* scale [Lewis and Sauro, 2009] to grade the usability of Pharmer. SUS is a standardized, simple, ten-item Likert scale-based questionnaire giving a global view The results of our survey (cf. Figure 8.9) showed a mean usability score of **75** for Pharmer which indicates a good level of usability. Participants particularly liked the integration of functionality and the ease of learning and use. The confidence in using the system was slightly lower, which we attribute to the short learning phase and diverse functionality. Of course, this is a very simplified view on usability and we expect even better results could be achieved by putting more effort into the Pharmer development. However, our goal was to demonstrate that Pharmer implementations with good usability characteristics could be created with relatively limited effort. In addition to quantitative results, we also collected a number of user suggestions. For instance some users suggested providing a print-friendly document with all the patient's desired information.

[13] Available at `http://youtu.be/eNbbqO-zLQk`

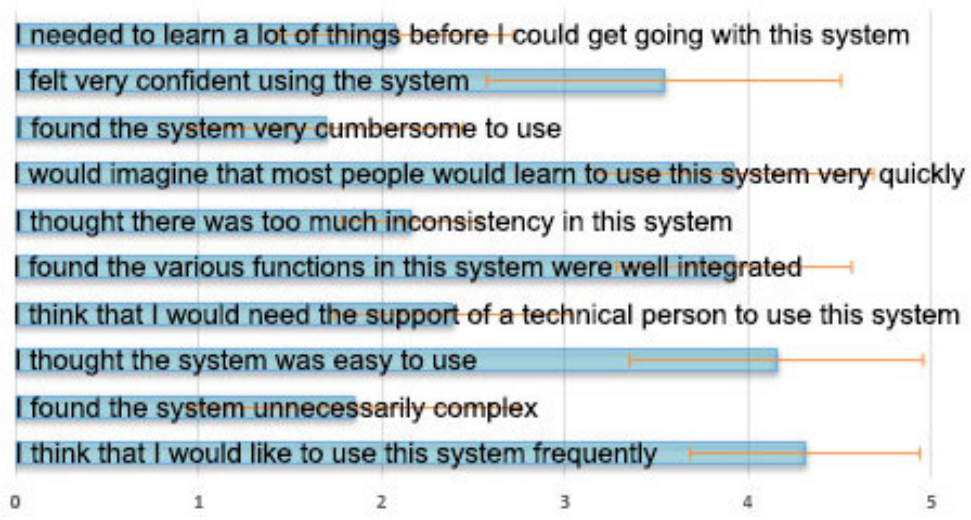


Figure 8.9.: Usability evaluation results for Pharmer.

## 8.7. Conclusion

This chapter addressed the research question RQ6 (cf. Section 1.3) to apply semantic content authoring to a domain-specific use case (i.e. e-prescribing) for achieving content interoperability. Providing a consistent connection between patients, physicians, pharmacists, pharmaceutical researchers and drug companies is a crucial step towards enhancing the quality of knowledge management and thereby e-health services in the pharmaceutical domain. With Pharmer, we presented in this chapter another implementation of the WYSIWYM interface model for realizing *Semantic Prescriptions* as intelligent medical prescriptions to improve the integration and interoperability of e-prescribing systems with other e-health services. Semantic prescriptions include the important meta-data about the content of a prescription, which will increase the awareness of their consumers.

Chapter 9

# Conclusions and Future Work

> "In the end, people are persuaded not by what we say, but by what they understand."
> — *John C. Maxwell*

This chapter provides an overview on the answers to the research questions behind this thesis and summarizes the main results of this work. It then discusses the future directions in which we intend to move further to extend and broaden the research conducted in the contributed areas.

## 9.1. Answers to Research Questions

In this section we revisit the research questions discussed in Section 1.3 and provide a summary of the answers and contributions:

### RQ1. What are existing approaches for user-friendly semantic content authoring?

Based on the starting point of the authoring process which can be ontologies (with upper level of expressiveness) or unstructured content (with lower level of expressiveness), we can classify the existing approaches for SCA into categories *Top-Down* and *Bottom-Up*. The bottom-up approaches usually known as semantic annotation or semantic markup techniques aim to annotate existing documents using a set of predefined ontologies. The top-down approaches usually called ontology population techniques aim to create semantic content based on a set of initial ontologies, which are extended during the population process. The tools which employ the bottom-approach are more appropriate for end users with no or limited knowledge of the domain on which the annotations or semantic structures are applied. Tools that adopt the top-down approach usually need users to have knowledge of the corresponding domain as well as ontology concepts.

A predefined set of *quality attributes* comprising standard UI types and features can be employed to evaluate the strengths and weaknesses of existing SCA systems. Quality attributes address different aspects of designing and developing user-friendly SCA systems: Essential, foundational quality attributes for an SCA system are, in particular, *usability*, *generalizability*, *customizability* and *evolvability*. Support of *collaboration*, *interoperability* and *scalability* are quality attributes required when an SCA system is employed in a community-driven environment with large amount of users, systems and interactions. *Automation* and *proactivity* are quality

attributes which facilitate usability of SCA systems especially for non-skilled users. *Portability* and *accessibility* are, as our survey indicated, not well addressed by the SCA-related literature so far and demand more investigations.

### RQ2. How can we bind user interface elements to semantic representation data models?

In order to facilitate the creation of semantically-enriched content, we need to provide suitable UIs which are compatible with the elements of our underlying semantic representation data model. In addition to an in-depth analysis of the elements of existing semantic representation models such as tree-based, graph-based and hyper-graph-based models, we performed an extensive review of the existing UI elements and techniques for visualization, exploration and authoring of text, images and videos. Finding the possible bindings between the semantic representation data models and UI techniques for visualization, exploration and authoring of content led us to the development of a novel interface model called *WYSIWYM* (What You See Is What You Mean). WYSIWYM aims to standardize interfaces for SCA systems. In order to facilitate, enhance and customize the WYSIWYM model, a set of helper components, which implement cross-cutting aspects such as automation, recommendation and collaboration are integrated into the WYSIWYM model.

### RQ3. How can we integrate semantic content authoring features into the current authoring tools on the Social Web?

Integrating SCA features into the current content authoring process on the Social Web, facilitates the promotion of structured content on the Web to a great extent. WYSIWYG (What You See Is What You Get) text authoring is meanwhile ubiquitous on the Web and part of most content creation and management workfows such as CMSs, Weblogs, Wikis, product data management systems and online shops. In this thesis we introduced *RDFaCE* approach as a transition from WYSIWYG to WYSIWYM. The rationale is to provide an environment to the user, which she is sufficiently familiar with, but at the same time enables her to understand, access and work with semantically-enriched content. We implemented RDFaCE as a plugin for existing WYSIWYG implementations which could be installed and employed on the Social Web without much additional effort.

### RQ4. How can we exploit semantically-enriched content for content analysis?

Semantically-enriched content can be exploited to deal with the current analytical information imbalance (cf. Section 6.1). In this thesis, we introduced *conTEXT* as a Mashup platform for text analytics. conTEXT combines services for NLP (e.g. named entity recognition and relation extraction), sentiment analysis, visualization, exploration and feedback to exploit semantically-enriched content for text analysis. conTEXT employs the WYSIWYM interface model to enable ordinary Web users

to perform sophisticated NLP tasks. Instant benefits provided by different analytics views in conTEXT act as an incentive for users to adopt semantic annotations and to take NLP feedback into account. Users receive more precise analytics results as they contribute to the refinement of automatically annotated content.

**RQ5. How can we apply crowdsourcing & collaborative content authoring techniques to the process of semantic content authoring?**

One of the main drivers to increase the amount of structured content on the Web is harnessing the power of crowds contributing to the Social Web on a daily basis. Addressing the crowdsourcing and collaboration aspects of SCA, requires new extensions to our proposed WYSIWYM interface model. By introducing the *WikiApp* data model in this thesis we aimed to provide a refinement of traditional entity-relationship data model which considers users and content revisions as its first class citizen. Based on the WikiApp model, we developed a platform called *SlideWiki* for collaborative authoring of highly-structured e-learning content. SlideWiki implements our proposed WYSIWYM interface together with collaboration helper components for authoring of semi-structured content in an implicit and user-friendly manner.

**RQ6. How can we apply semantic content authoring to a domain-specific use case for achieving content interoperability?**

In this thesis, we introduced *Pharmer* as a domain-specific implementation of the WYSIWYM model. Pharmer enables physicians to author semantic medical prescriptions as intelligent prescriptions which know about their own content. With Pharmer we investigated how semantically-enriched content can be applied to facilitate content interoperability in the domain of health-care services. Pharmer utilizes real-time drug tagging for user-friendly creation of structured medical prescriptions and to enable shared decision-making among physicians, pharmacists, researchers, Pharma and insurance companies.

## 9.2. Summary of the Results

Upon completion of this thesis, the main research question behind this work, namely "*How can we enable user-friendly manual and semi-automatic creation of rich semantic content?*" needs to be answered. As shown in Figure 9.1, in order to facilitate authoring of semantically-enriched documents, we need to extend the existing authoring approaches with appropriate user interfaces for semantic content authoring. This extension should be carried out with minimal effort so that user is not distracted from the normal process of content authoring. In this thesis, we proposed a *semantics-based user interface model* which provides a *binding* between existing semantic representation data models and existing user interfaces for visualization, exploration and authoring of content. The proposed model is

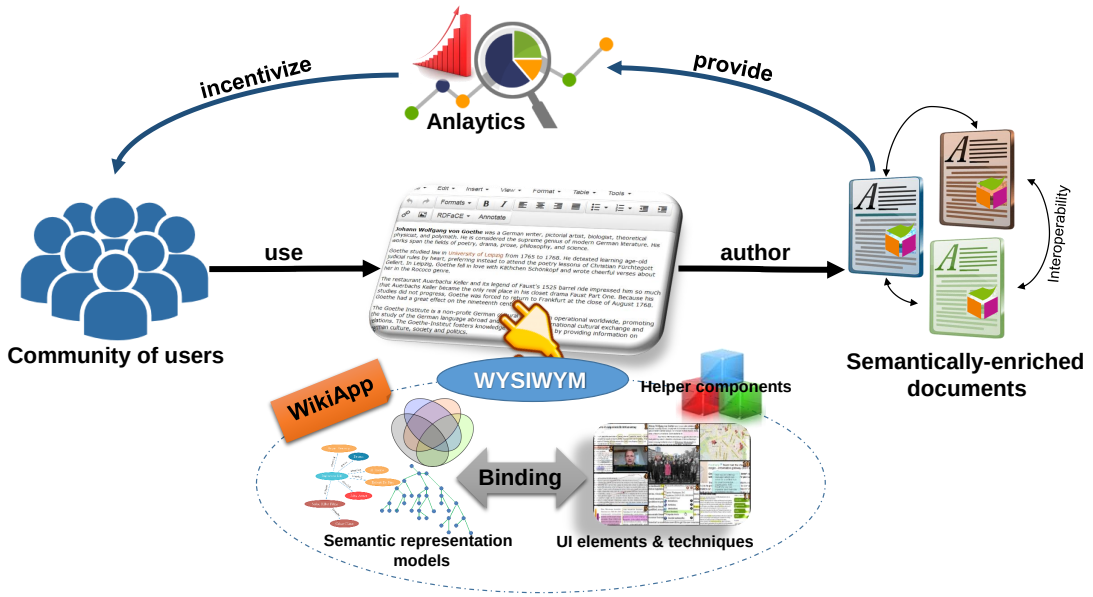


Figure 9.1.: User-friendy manual & semi-automatic creation of rich semantic content.

called *WYSIWYM* (What You See Is What You Mean) and aims to *standardize* semantic authoring user interfaces. WYSIWYM model offers a set of *helper components* to deal with the cross-cutting aspect such as automation, proactivity, accessibility and personalization. In order to deal with the *community of users* and content revisions, we enriched WYSIWYM with an appropriate data model called *WikiApp* which supports collaboration and crowdsourcing. Furthermore, to *incentivize* users to adopt semantic content authoring, different views for *text analytics* are provided to users.

Revisiting our user scenario in Section 1.1, Alice can exploit semantically-enriched job posts (either by annotating unstructured job posts or by authoring structured job posts from the scratch) to create different UIs for content exploration and visualization. For example, she will see a taxonomy of all the Data Science related skills together with their mentions in IT jobs posted on LinkedIn or other job posting Websites. She can then easily extract the most demanded Data Science skills from the collected IT jobs and can publish them as structured content on their online magazine to be reusable by other Web users too.

## 9.3. Impact

The main goal of this research was to facilitate and promote semantic content authoring among Web end-users. To achieve this goal, we developed and released several open-source tools and platforms dealing with this task in both general and domain-specific use cases (cf. Appendix A). Most of these tools have been already

actively used by users on the Web. With these tools, we envision the following impacts:

**Alleviating the Semantic Web's chicken-and-egg problem.** Recently we could observe a significant increase of the amount of structured data publishing on the Web. However, this increase can be attributed primarily to article metadata being made available and already to a much lesser extend to just a few entity types (people, organizations, products) being prevalent [Bizer et al., 2013]. As a consequence, we still face the chicken-and-egg problem to truly realize the vision of a Web, where large parts of the information are available in structured formats and semantically annotated. Before no substantial amount of content is available in semantic representations, search engines will not pick up this information and without better search capabilities publishers are not inclined to make additional effort to provide semantic annotations for their content. The latter is particularly true for unstructured and semi-structured content, which is much more difficult to annotate than structured content from relational databases (where merely some templates have to be adopted in order to provide e.g. RDFa).

RDFaCE and conTEXT can help to overcome this problem, since they provide instant benefits (i.e. SEO and text analytics views) to users for creating comprehensive semantic annotations.

**Democratizing the NLP usage.** With conTEXT, natural language processing technology is made more accessible, so that ordinary users can use sophisticated text analytics with just a few clicks. RDFaCE and Pharmer allow ordinary users to exploit NLP by one click (or even in real-time without any click while user is writing) for automatic content annotation. This was achieved by abstracting from a particular technology (e.g. by using the NIF format) and by supporting sophisticated content visualizations and exploration employing the WYSIWYM model and the data-driven document metaphor. As a result, ordinary users can observe the power of NLP and semantic technologies with minimal effort. By directly showing the effect of semantic annotations and demonstrating the benefits for improved navigation, exploration and search, users will gain a better understanding of recent technology advances.

**Harnessing the power of feedback loops.** Thomas Goetz states in his influential WIRED Magazin article [Goetz, 2011]: 'Provide people with information about their actions in real time, then give them a chance to change those actions, pushing them toward better behaviors.' With RDFaCE and conTEXT, we give users direct feedback on what information can be extracted from their works using NLP services. At the same time we want to incorporate their feedback and revisions of the automatic semantic annotations back in the NLP processing loop. Incorporating user feedback was so far not much in the focus of the NLP community. With RDFaCE and conTEXT, we aim to contribute to changing this. We argue,

that NLP technology achieving, for example, 90% precision, recall or f-measure, might not fulfill the requirements of a number of potential use cases. When we can increase the quality of the NLP through user feedback, we might be able to substantially extend the range of potential NLP applications. The user feedback here serves two purposes: One the one hand, it directly increases the quality of the semantic annotation. On the other hand, it can serve as input for active learning techniques, which can further boost precision and recall of the semantic annotation.

**Enabling the collaborative creation of structured multilingual educational content.** The creation of high-quality educational content is a time and resource consuming task. The task requires even more resources if there is a need to offer the content in different languages. With SlideWiki, we propose to exploit the power of crowd to author semantically structured educational content available in a number of different languages. By employing the WYSIWYM model and by semantic structuring of content, we split the content into reusable elements in such a way, that each of them fully covers an individual piece of knowledge thereby enabling educational content reuse and re-purposing on the Web.

## 9.4. Limitations and Future Directions

The work presented in this thesis included both research and engineering parts. In the following, we describe the limitations and future work with regards to the main contributions of this thesis:

**UIs for semantic content authoring.** While there are many benefits of systematic reviews, they also bear some limitations and validity threats originating from human errors. The main threats to validity of our systematic review are twofold: correct and thorough selection of the studies to be included as well as accurate and exhaustive selection of quality attributes together with their corresponding UI features. With the increasing number of works in the area of semantic content authoring we can not guarantee to have captured all the material in this area. The scope of our review is restricted to the scientific domain. Therefore, some tools or approaches employed in the industry might have not been included in our primary studies. Furthermore, since the review process was mainly performed by one researcher a bias is possible. In order to mitigate a potential subjective bias, the review protocol and results were checked and validated by a senior researcher and other colleagues experienced in the context of Semantic Web.

As future work, we envision strategies to semi-automatically improve the realization of the quality attributes discussed in Section 3.9, for example, using active machine learning for better integration with approaches delivering automatic suggestions. Also extending the support for integration of multi-media and multi-modal semantic annotation (e.g. of images and multimedia content) is a promising research direction. Addressing open research and technology challenges

such as accessibility, handling complexity in UIs, formal and systematic methods for user interface evaluation and UIs for ubiquitous devices are other interesting areas for future research.

**WYSIWYM model.** With regards to the limitations of our proposed WYSIWYM model, though we attempted the bindings to be fairly complete, new UI elements might be developed or additional data models (or variations of the ones considered) might appear. In this case, the bindings should be updated.

As future work, we envision to adopt a model-driven approach to enable automatic implementation of WYSIWYM interfaces by user-defined preferences. This will help to reuse, re-purpose and choreograph WYSIWYM UI elements to accommodate the needs of dynamically evolving information structures and ubiquitous interfaces. We also aim to bootstrap an ecosystem of WYSIWYM instances and UI elements to support structure encoded in different modalities, such as images and videos. Creating live and context-sensitive WYSIWYM interfaces which can be generated on-the-fly based on the ranking of available UI elements is another promising research venue.

**Integrating semantic content authoring into the current content authoring tools.** Integrating SCA systems into other applications like speech recognition and question-answering systems for improving the accuracy and quality of results is an important area of future work in this context. At the moment, intelligent mobile assistants (e.g. *Siri*[1] for the iPhone) only allow delegation of certain programmed tasks (e.g. making restaurant reservations, getting movie tickets, etc.) by invoking certain predefined web services. Employing semantically enriched content in the UI of mobile personal agents will extend their capability to inquiry the open Web of Data thereby achieving more efficient and effective results.

**Exploiting semantically-enriched content for content analysis and instant user gratification.** In future, we plan to extend work on conTEXT along several directions:

- *Enhancing the NLP feedback.* We aim to investigate, how user feedback can be used across different corpora. We consider the harnessing of user feedback by NLP services an area with great potential to attain further boosts in annotation quality. On a related angle, we plan to integrate revisioning functionality, where users can manipulate complete sets of semantic annotations instead of just individual ones. In that regard, we envision that conTEXT can assume a similar position for text corpora as have data cleansing tools such as OpenRefine for structure data.

- *Creating a flexible end-user NLP ecosystem.* At the moment, conTEXT platform relies mainly on DBpedia Knowledge Base (KB) to extract the

[1] http://www.siri.com

Named Entities and to provide different views for text analytics. Nevertheless, there are many use cases that require to change or extend the underlying KB for providing more elaborated and domain-specific views on content. Enabling users with mechanisms to modify the underlying KB with minimal efforts will bring a high impact to the conTEXT end-user NLP ecosystem. In the envisioned ecosystem, users can either create their own KB or reuse existing KBs provided by knowledge engineers and domain experts. Once the underlying KB changes, all the other components (e.g. NER tools or analytics views) must adapt to this change. Also, in this direction we plan to provide conTEXT as a composite Web service, where each component of the system such as input processors, NLP services and analytics views can be created, shared and reused by Web users in a modular way. In this dynamic and flexible ecosystem, a user's content is continuously ingested and processed, the user is informed about updates and thus the semantic representations of the content evolve along with the content itself.

**Applying crowdsourcing & collaborative authoring techniques to the process of semantic content authoring.** Our first direction for future work in this context is to implement a completely SCORM-compliant LCMS and authoring tool, based on the SlideWiki. This will allow us to exchange the content with other SCORM-compliant LCMSs. Also, in a real e-learning scenario, learners come from different environments, have different ages and educational backgrounds. These heterogeneities in user profiles are crucial to be addressed when enhancing the crowdlearning concept. New approaches should provide the possibility to personalize the learning process. Thus, our second direction is providing the personalized content based on initial user assessments. The third direction for the future work is to support the annotation of learning objects using standard metadata schemes. We aim to implement the *LRMI*[2] metadata schemes to facilitate end-user search and discovery of educational resources.

**Exploiting semantically-enriched content for content interoperability in domain-specific use cases.** Regarding future work, we envision to extend the Pharmer application towards different modalities, such that the annotation of images and other medical objects is supported. Furthermore, we aim to integrate the other existing linked open datasets (e.g. related to publications, laboratories or insurance documents) into the Pharmer to extend its stakeholders.

[2]Learning Resource Metadata Initiative: www.lrmi.net/

Appendix A

# Software Release History

The following software releases were made during the thesis:

- **RDFaCE** (https://bitbucket.org/ali1k/rdface & http://wordpress.org/plugins/rdface/)
  - 0.4 - released 2014-5-11 - bug fixes, compatibility with TinyMCE 4.0 and Wordpress 3.9
  - 0.3 - released 2013-4-15 - schema.org edition
  - 0.2 - released 2012-3-6 - rdface-lite with support for rNews vocabulary
  - 0.1 - released 2011-7-8 - initial version
- **conTEXT** (https://github.com/AKSW/context)
  - 0.3 - released 2014-5-1 - support for replacing the DBpedia ontology with user's ontology, support for selecting a subgraph of DBpedia
  - 0.25 - released 2014-4-20 - support for real-time analysis of Twitter streams
  - 0.2 - released 2014-4-1 - support for social media sign-in, added Twitter, LinkedIn, Facebook and G+ input sources, added sentiment analysis view
  - 0.1 - released 2014-1-17 - initial version
- **SlideWiki** (https://github.com/AKSW/SlideWiki)
  - 0.1 - released 2013-9-24 - initial version
- **Pharmer** (https://code.google.com/p/pharmer)
  - 0.1 - released 2012-12-1 - initial version

# List of Abbreviations

**API** Application Programming Interface, pp. 38, 80, 82–84, 88, 89, 91, 92, 95, 99, 106, 108, 114

**ATAG** Authoring Tool Accessibility Guidelines, p. 59

**ATR** Automatic Term Recognition, pp. 25, 26

**BOA** BOotstrapping linked datA, p. 107

**CMS** Content Management System, pp. 7, 81, 91, 156

**CSS** Cascading Style Sheets, pp. 63, 83, 95, 134

**CURIE** Compact URI, p. 18

**D3** Data-Driven Document, pp. 108, 114

**DOM** Document Object Model, pp. 82–84

**DSL** Domain-Specific Language, pp. 123, 124

**EL** Entity Linking, pp. 25, 26

**ER** Entity-Relation, pp. 10, 63, 121

**FOAF** Friend Of A Friend, pp. 47, 68

**FOX** Federated knOwledge eXtraction Framework, p. 107

**HCI** Human Computer Interaction, pp. 35, 75

**IRI** Internationalized Resource Identifier, p. 17

**JSON** JavaScript Object Notation, pp. 17, 94, 114, 135, 137

**KB** Knowledge Base, pp. 26, 161, 162

**KE** Keyword Extraction, pp. 25, 26

**KR** Knowledge Representation, p. 4

**LCMS** Learning Content Management Systems, pp. 125, 127, 136, 162

**LD** Linked Data, pp. 17, 27, 106, 141–143

**LOD** Linked Open Data, pp. 141–143, 145, 151

**LODD** Linked Open Drug Data, pp. 8, 140, 142, 143, 150, 152

**LOM** Learning Objects Metadata, p. 136

**LOR** Learning Objects Repository, pp. 135, 136

**MVC** Model-View-Controller, pp. 114, 121, 124, 131, 136

**NER** Named Entity Recognition, pp. 9, 25, 26, 103, 162

**NIF** NLP Interchange Format, pp. 8, 27, 108, 114, 159

**NLP** Natural Language Processing, pp. 1, 3, 7, 9, 25, 27, 38, 78, 80, 83, 84, 88, 89, 95, 99, 102–104, 106–108, 114, 118, 145, 147, 156, 157, 159–162

**OCA** One Click Annotation, pp. 57, 64

**OCW** OpenCourseWare, p. 126

**OPML** Outline Processor Markup Language, p. 135

**OWL** Web Ontology Language, pp. 27, 35

**PbE** Programming by Example, p. 43

**POS** Part-Of-Speech, p. 25

**QUIM** Quality in Use Integrated Measurement, p. 42

**RDBMS** Relational Database Management System, p. 25

**RDF** Resource Description Frameworka, pp. 8, 12–19, 24, 25, 27, 35, 47, 56–58, 63, 68, 69, 83, 100, 105, 107, 119, 122, 124, 136, 143, 146

**RDFa** Resource Description Framework in Attributes, pp. 18, 19, 35, 37, 47, 54, 58, 67, 69, 82–84, 86, 87, 92, 95, 96, 98–100, 108, 146, 159

**RSS** Rich Site Summary, pp. 106, 135

**SCA** Semantic Content Authoring, pp. 5, 8, 9, 28, 32, 35, 38, 39, 41–48, 52, 54, 59–61, 99, 155–157, 161

**SCAUI** Semantic Content Authoring User Interface, pp. 35, 58–60

**SCORM** Sharable Content Object Reference Model, pp. 126, 162

**SDM** Shared Decision Making, p. 149

**SEO** Search Engine Optimization, pp. 7, 110, 159

**SIOC** Semantically-Interlinked Online Communities, pp. 47, 68

**SKOS** Simple Knowledge Organization System, pp. 1, 35, 47, 68

**SPARQL** SPARQL Protocol and RDF Query Language, pp. 8, 12, 24, 25, 35, 106, 119, 124, 145, 147

**SUS** System Usability Scale, pp. 117, 137, 153

**UI** User Interface, pp. 3–9, 28, 32, 39, 41–45, 47, 52, 54–66, 68, 71, 73–75, 78, 79, 82, 83, 86, 95, 98, 102, 105, 110, 145, 146, 155, 156, 158, 160, 161

**URI** Uniform Resource Identifier, pp. 13–15, 18, 24, 27, 45, 68, 78, 83, 84, 97, 98, 107, 114, 141, 142

**URL** Uniform Resource Locator, pp. 14, 19, 136

**W3C** World Wide Web consortium, pp. 12, 13, 20, 24, 142

**WCAG** Web Content Accessibility Guidelines, p. 59

**WKF** Wikification, pp. 25, 26

**WYSIWYG** What You See Is What You Get, pp. 7, 9, 58, 74, 80, 81, 86, 101, 132, 156

**WYSIWYM** What You See Is What You Mean, pp. 2, 6, 7, 9, 10, 62–67, 78–81, 86, 87, 96, 101, 102, 108, 119, 131, 132, 139, 140, 146, 154, 156–161

# List of Tables

# List of Figures

# Bibliography

[ADL, 2011a] ADL (2011a). Scorm 2004 4th edition specification. http://www.adlnet.gov/scorm/scorm-2004-4th/.

[ADL, 2011b] ADL (2011b). Scorm users guide for programmers. http://www.adlnet.gov/wp-content/uploads/2011/12/SCORM_Users_Guide_for_Programmers.pdf.

[Adrian et al., 2010] Adrian, B., Hees, J., Herman, I., Sintek, M., and Dengel, A. (2010). Epiphany: Adaptable rdfa generation linking the web of documents to the web of data. In Cimiano, P. and Pinto, H., editors, *Knowledge Engineering and Management by the Masses*, volume 6317 of *Lecture Notes in Computer Science*, pages 178–192. Springer Berlin / Heidelberg. 10.1007/978-3-642-16438-5-13.

[Ankolekar et al., 2007] Ankolekar, A., Krötzsch, M., Tran, T., and Vrandecic, D. (2007). The two cultures: mashing up web 2.0 and the semantic web. In *WWW '07: Proceedings of the 16th international conference on World Wide Web*, pages 825–834, New York, NY, USA. ACM Press.

[Araujo et al., 2010] Araujo, S., Houben, G.-J., and Schwabe, D. (2010). Linkator: Enriching web pages by automatically adding dereferenceable semantic annotations. In *Web Engineering*, volume 6189 of *Lecture Notes in Computer Science*, pages 355–369. Springer.

[Auer et al., 2012a] Auer, S., Bühmann, L., Dirschl, C., Erling, O., Hausenblas, M., Isele, R., Lehmann, J., Martin, M., Mendes, P., Nuffelen, B., Stadler, C., Tramp, S., and Williams, H. (2012a). Managing the life-cycle of linked data with the lod2 stack. In Cudré-Mauroux, P., Heflin, J., Sirin, E., Tudorache, T., Euzenat, J., Hauswirth, M., Parreira, J., Hendler, J., Schreiber, G., Bernstein, A., and Blomqvist, E., editors, *The Semantic Web – ISWC 2012*, Lecture Notes in Computer Science, pages 1–16. Springer Berlin Heidelberg.

[Auer et al., 2012b] Auer, S., Demter, J., Martin, M., and Lehmann, J. (2012b). Lodstats – an extensible framework for high-performance dataset analytics. In Teije, A., Völker, J., Handschuh, S., Stuckenschmidt, H., d'Acquin, M., Nikolov, A., Aussenac-Gilles, N., and Hernandez, N., editors, *Knowledge Engineering and Knowledge Management*, volume 7603 of *Lecture Notes in Computer Science*, pages 353–362. Springer Berlin Heidelberg.

[Auer et al., 2009] Auer, S., Dietzold, S., Lehmann, J., Hellmann, S., and Aumueller, D. (2009). Triplify: Light-weight linked data publication from relational databases. In *WWW2009, Spain*. ACM.

[Auer et al., 2006] Auer, S., Dietzold, S., and Riechert, T. (2006). Ontowiki – a tool for social, semantic collaboration. In Cruz, I., Decker, S., Allemang, D., Preist, C., Schwabe, D., Mika, P., Uschold, M., and Aroyo, L., editors, *The Semantic Web - ISWC 2006*, volume 4273 of *Lecture Notes in Computer Science*, pages 736–749. Springer Berlin / Heidelberg. 10.1007/11926078-53.

[Auer et al., 2013] Auer, S., Khalili, A., and Tarasowa, D. (2013). Crowd-sourced open courseware authoring with slidewiki.org. *International Journal of Emerging Technologies in Learning (iJET)*, 8(1).

[Beckett, 2004] Beckett, D. (2004). RDF/XML syntax specification (revised). http://www.w3.org/TR/2004/REC-rdf-syntax-grammar-20040210/.

[Benson et al., 2010] Benson, E., Marcus, A., Howahl, F., and Karger, D. (2010). Talking about data: Sharing richly structured information through blogs and wikis. In *The Semantic Web – ISWC 2010*, volume 6496 of *Lecture Notes in Computer Science*, pages 48–63. Springer.

[Berners-Lee and Connolly, 2011] Berners-Lee, T. and Connolly, D. (2011). Notation3 (N3): A readable RDF syntax. http://www.w3.org/TeamSubmission/n3/.

[Berners-Lee et al., 2001] Berners-Lee, T., Hendler, J., and Lassila, O. (2001). The semantic web. *Scientific American*, 284(5):34–43.

[Berners-Lee et al., 2007] Berners-Lee, T., Hollenbach, J., Lu, K., Presbrey, J., d'ommeaux, E. P., and m.c. schraefel (2007). Tabulator redux: Writing into the semantic web. http://eprints.ecs.soton.ac.uk/14773/. Tabulator Redux tech report.

[Bishop et al., 2011] Bishop, B., Kiryakov, A., Ognyanoff, D., Peikov, I., Tashev, Z., and Velkov, R. (2011). OWLIM: A family of scalable semantic repositories. *Semantic Web*, 2(1):1–10.

[Bizer et al., 2013] Bizer, C., Eckert, K., Meusel, R., M?hleisen, H., Schuhmacher, M., and V?lker, J. (2013). Deployment of rdfa, microdata, and microformats on the web - a quantitative analysis. In *12th International Semantic Web Conference, 21-25 October 2013, Sydney, Australia, In-Use track*.

[Bizer et al., 2009] Bizer, C., Heath, T., and Berners-Lee, T. (2009). Linked Data - The Story So Far. *International Journal on Semantic Web and Information Systems (IJSWIS)*, 5(3):1–22.

[Bizer and Schultz, 2009] Bizer, C. and Schultz, A. (2009). The berlin sparql benchmark. *Int. J. Semantic Web Inf. Syst.*, 5(2):1–24.

[Blankenship and Ruona, 2009] Blankenship, S. and Ruona, W. (2009). Exploring knowledge sharing in social structures: Potential contributions to an overall knowledge management strategy. *Advances in Developing Human Resources*, 11(3).

[Bostock et al., 2011] Bostock, M., Ogievetsky, V., and Heer, J. (2011). D3 data-driven documents. *Visualization and Computer Graphics, IEEE Transactions on*, 17(12):2301–2309.

[Breslin et al., 2009] Breslin, J., Passant, A., and Decker, S. (2009). *The Social Semantic Web*. Springer-Verlag, Heidelberg.

[Broekstra et al., 2002] Broekstra, J., Kampman, A., and van Harmelen, F. (2002). Sesame: A generic architecture for storing and querying RDF and RDF schema. In *ISWC*, number 2342 in LNCS, pages 54–68. Springer.

[Buffa et al., 2008] Buffa, M., Gandon, F., Ereteo, G., Sander, P., and Faron, C. (2008). Sweetwiki: A semantic wiki. *Web Semantics: Science, Services and Agents on the World Wide Web*, 6(1):84 – 97. Semantic Web and Web 2.0.

[Burel et al., 2009] Burel, G., Cano1, A. E., and Lanfranchi, V. (2009). Ozone browser: Augmenting the web with semantic overlays. volume 449 of *CEUR WS Proceedings*.

[Camara et al., 1999] Camara, G., Souza, R. C. M., Monteiro, A. M., Paiva, J., Garrido, J., Câmara, G., Cartaxo, R., Souza, M. D., Miguel, A., Monteiro, V., Carlos, J., Garrido, P. D., Processamento, D., and Dpi, I. (1999). Handling complexity in gis interface design. In *In: Proceedings of the I Brazilian Workshop on GeoInformatics, Campinas, São*.

[Casey and Olivera, 2011] Casey, A. J. and Olivera, F. (2011). Reflections on organizational memory and forgetting. *Journal of Management Inquiry*, 20(3):305–310.

[Chang et al., 2013] Chang, K. S.-P., Myers, B. A., Cahill, G. M., Simanta, S., Morris, E., and Lewis, G. (2013). Improving structured data entry on mobile devices. In *Proceedings of the 26th Annual ACM Symposium on User Interface Software and Technology*, UIST '13, pages 75–84, New York, NY, USA. ACM.

[Charles et al., 1997] Charles, C., Gafni, A., and Whelan, T. (1997). Shared decision-making in the medical encounter: What does it mean? (or it takes at least two to tango). *Social Science & Medicine*, 44(5):681 – 692.

[Chen and Babar, 2011] Chen, L. and Babar, M. A. (2011). A systematic review of evaluation of variability management approaches in software product lines. *Information & Software Technology*, 53(4):344–362.

[Chu et al., 2009] Chu, H.-C., Chen, M.-Y., and Chen, Y.-M. (2009). A semantic-based approach to content abstraction and annotation for content management. *Expert Systems with Applications*, 36(2, Part 1):2360 – 2376.

[Clark et al., 2008] Clark, K. G., Feigenbaum, L., and Torres, E. (2008). SPARQL Protocol for RDF. World Wide Web Consortium, Recommendation REC-rdf-sparql-protocol-20080115 http://www.w3.org/TR/2008/REC-rdf-sparql-protocol-20080115.

[Cobb and Steele, 2011] Cobb, J. and Steele, C. (2011). Association learning management systems. http://www.tagoras.com/docs/Tagoras-Association-LMS-Report-Overview.pdf.

[Cunningham et al., 2011] Cunningham, H., Maynard, D., Bontcheva, K., Tablan, V., Aswani, N., Roberts, I., Gorrell, G., Funk, A., Roberts, A., Damljanovic, D., Heitz, T., Greenwood, M. A., Saggion, H., Petrak, J., Li, Y., and Peters, W. (2011). *Text Processing with GATE (Version 6)*.

[Dadzie and Rowe, 2011] Dadzie, A.-S. and Rowe, M. (2011). Approaches to visualising linked data: A survey. *Semantic Web*, 2(2):89–124.

[d'Aquin et al., 2008] d'Aquin, M., Motta, E., Dzbor, M., Gridinoc, L., Heath, T., and Sabou, M. (2008). Collaborative semantic authoring. *Intelligent Systems, IEEE*, 23(3):80 –83.

[Davis et al., 1993] Davis, R., Shrobe, H. E., and Szolovits, P. (1993). *AI Magazine*, (1):17–33.

[Deligiannidis et al., 2007] Deligiannidis, L., Kochut, K. J., and Sheth, A. P. (2007). RDF data exploration and visualization. In *CIMS 2007*, pages 39–46. ACM.

[Di Iorio et al., 2010] Di Iorio, A., Musetti, A., Peroni, S., and Vitali, F. (2010). Ontology-driven generation of wiki content and interfaces. *New review of hypermedia and multimedia*, 16(1-2, SI):9–31.

[Dyba et al., 2007] Dyba, T., Dingsoyr, T., and Hanssen, G. K. (2007). Applying systematic reviews to diverse study types: An experience report. In *Proceedings of the First International Symposium on Empirical Software Engineering and Measurement*, ESEM '07, pages 225–234, Washington, DC, USA. IEEE Computer Society.

[Ennals et al., 2007] Ennals, R., Brewer, E. A., Garofalakis, M. N., Shadle, M., and Gandhi, P. (2007). Intel mash maker: join the web. *SIGMOD Record*, 36(4):27–33.

[Erling and Mikhailov, 2007] Erling, O. and Mikhailov, I. (2007). RDF support in the virtuoso DBMS. In Auer, S., Bizer, C., Müller, C., and Zhdanova, A. V., editors, *CSSW*, volume 113 of *LNI*, pages 59–68. GI.

[Ermilov et al., 2011] Ermilov, T., Heino, N., Tramp, S., and Auer, S. (2011). Ontowiki mobile – knowledge management in your pocket. In *8th Extended Semantic Web Conference (ESWC2011)*.

[Ferrucci and Lally, 2004] Ferrucci, D. and Lally, A. (2004). Uima: an architectural approach to unstructured information processing in the corporate research environment. *Nat. Lang. Eng.*, 10(3-4):327–348.

[Fitzpatrick, 1998] Fitzpatrick, R. (1998). Strategies for evaluating software usability. *Methods*, 353(1).

[Fonstad, 2005] Fonstad, N. O. (2005). Tangible purposes and common beacons: The interrelated roles of identity and technology in collaborative endeavors. In *OKLC (Organizational Learning, Knowledge and Capabilities)*.

[Frosterus et al., 2011] Frosterus, M., Hyvönen, E., and Laitio, J. (2011). Datafinland—a semantic portal for open and linked datasets. In *The Semanic Web: Research and Applications*, volume 6644 of *LNCS*, pages 243–254. Springer.

[Galanter et al., 2013] Galanter, W., Falck, S., Burns, M., Laragh, M., and Lambert, B. L. (2013). Indication-based prescribing prevents wrong-patient medication errors in computerized provider order entry (cpoe). *Journal of the American Medical Informatics Association*, 20:477–481.

[Geiger et al., 2011] Geiger, D., Rosemann, M., and Fielt, E. (2011). Crowdsourcing information systems : a systems theory perspective. In *Australasian Conference on Information Systems (ACIS) 2011*, Sydney, Australia.

[Gerber and Ngonga Ngomo, 2011] Gerber, D. and Ngonga Ngomo, A.-C. (2011). Bootstrapping the linked data web. In *1st Workshop on Web Scale Knowledge Extraction @ ISWC 2011*.

[Glaser and Strauss, 1967] Glaser, B. G. and Strauss, A. L. (1967). *The Discovery of Grounded Theory: Strategies for Qualitative Research*. Aldine de Gruyter, New York, NY.

[Goetz, 2011] Goetz, T. (2011). Harnessing the power of feedback loops. *WIRED Magazine*.

[González et al., 2011] González, A. R., García-Crespo, Á., Palacios, R. C., Berbís, J. M. G., and Jiménez-Domingo, E. (2011). Using ontologies in drug prescription: The semmed approach. *IJKBO*, 1(4):1–15.

[Greenfield, 2006] Greenfield, A. (2006). *Everyware: The Dawning Age of Ubiquitous Computing*. New Riders Publishing, Berkeley, CA.

[Haase et al., 2010] Haase, P., Eberhart, A., Godelet, S., Mathäß, T., Tran, T., Ladwig, G., and Wagner, A. (2010). The information workbench. interacting with the web of data. In *3rd Future Internet Symposium (FIS2010)*.

[Hachey, 2011] Hachey, G. (2011). Semantic web user interface: A systematic survey. Master's thesis, Athabasca University.

[Haller and Abecker, 2010] Haller, H. and Abecker, A. (2010). imapping: a zooming user interface approach for personal and semantic knowledge management. *SIGWEB Newsl.*, pages 4:1–4:10.

[Hasida, 2007] Hasida, K. (2007). Semantic authoring and semantic computing. In Sakurai, A., Hasida, K., and Nitta, K., editors, *New Frontiers in Artificial Intelligence*, volume 3609 of *Lecture Notes in Computer Science*, pages 137–149. Springer. 10.1007/978-3-540-71009-7-12.

[Heese et al., 2010] Heese, R., Luczak-Rösch, M., Oldakowski, R., Streibel, O., and Paschke, A. (2010). One click annotation. In *Scripting and Development for the Semantic Web (SFSW)*.

[Heflin, 2004] Heflin, J. (2004). OWL Web Ontology Language Use Cases and Requirements. Technical report, W3C.

[Heino et al., 2009] Heino, N., Dietzold, S., Martin, M., and Auer, S. (2009). Developing semantic web applications with the ontowiki framework. In *Networked Knowledge - Networked Media*, volume 221 of *Studies in Computational Intelligence*, pages 61–77. Springer, Berlin / Heidelberg.

[Heino et al., 2011] Heino, N., Tramp, S., and Auer, S. (2011). Managing web content using linked data principles – combining semantic structure with dynamic content syndication. In *Proceedings of the 35th Annual IEEE International Computer Software and Applications Conference (COMPSAC 2011)*. IEEE Computer Society.

[Heinrich et al., 2012] Heinrich, M., Lehmann, F., Springer, T., and Gaedke, M. (2012). Exploiting single-user web applications for shared editing: a generic transformation approach. In *WWW 2012*, pages 1057–1066. ACM.

[Heitmann et al., 2009] Heitmann, B., Kinsella, S., Hayes, C., and Decker, S. (2009). Implementing semantic web applications: reference architecture and challenges. In *5th International Workshop on Semantic Web-Enabled Software Engineering*.

[Hellmann et al., 2013] Hellmann, S., Lehmann, J., Auer, S., and Brümmer, M. (2013). Integrating nlp using linked data. In *12th International Semantic Web Conference, 21-25 October 2013, Sydney, Australia*.

[Herzig and Ell, 2010] Herzig, D. and Ell, B. (2010). Semantic mediawiki in operation: Experiences with building a semantic portal. In *The Semantic Web – ISWC 2010*, volume 6497 of *Lecture Notes in Computer Science*, pages 114–128. Springer. 10.1007/978-3-642-17749-1-8.

[Hong and Chi, ] Hong, L. and Chi, E. H. Annotate once, appear anywhere: collective foraging for snippets of interest using paragraph fingerprinting. CHI '09, pages 1791–1794. ACM.

[Howe, 2006] Howe, J. (2006). The rise of crowdsourcing. *Wired Magazine*, 14(6).

[Huynh et al., 2003] Huynh, D., Quan, D., and Karger, D. R. (2003). User interaction experience for semantic web information. In King, I. and Máray, T., editors, *WWW (Posters)*.

[Huynh et al., 2007] Huynh, D. F., Karger, D. R., and Miller, R. C. (2007). Exhibit: lightweight structured data publishing. WWW '07, pages 737–746, New York, NY, USA. ACM.

[Johnson, 2014] Johnson, J. (2014). *Designing with the Mind in Mind, Second Edition: Simple Guide to Understanding User Interface Design Guidelines*. Morgan Kaufmann Publishers Inc.

[Jungermann, 2009] Jungermann, F. (2009). Information extraction with rapidminer. Proceedings of the GSCL Symposium Sprachtechnologie und eHumanities.

[Kandel et al., 2011] Kandel, S., Paepcke, A., Hellerstein, J., and Heer, J. (2011). Wrangler: interactive visual specification of data transformation scripts. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, pages 3363–3372. ACM.

[Karger et al., 2009] Karger, D. R., Ostler, S., and Lee, R. (2009). The web page as a wysiwyg end-user customizable database-backed information management application. In *UIST 2009*, pages 257–260. ACM.

[Karger and Quan, 2005] Karger, D. R. and Quan, D. (2005). What would it mean to blog on the semantic web? *Web Semantics: Science, Services and Agents on the World Wide Web*, 3(2-3):147 – 157.

[Khalili and Auer, 2013a] Khalili, A. and Auer, S. (2013a). User interfaces for semantic authoring of textual content: A systematic literature review. *Web Semantics: Science, Services and Agents on the World Wide Web*, 22(0):1 – 18.

[Khalili and Auer, 2013b] Khalili, A. and Auer, S. (2013b). Wysiwym authoring of structured content based on schema.org. In Lin, X., Manolopoulos, Y., Srivastava, D., and Huang, G., editors, *The 14th International Conference on Web Information Systems Engineering (WISE 2013)*, volume 8181 of *Lecture Notes in Computer Science*, pages 425–438. Springer Berlin Heidelberg.

[Khalili and Auer, 2014] Khalili, A. and Auer, S. (2014). Wysiwym – integrated visualization, exploration and authoring of semantically enriched un-structured content. *Semantic Web Journal*.

[Khalili et al., 2012a] Khalili, A., Auer, S., and Hladky, D. (2012a). The rdfa content editor - from wysiwyg to wysiwym. In *2012 IEEE 36th Annual Computer Software and Applications Conference (COMPSAC)*, pages 531–540.

[Khalili et al., 2014] Khalili, A., Auer, S., and Ngomo, A.-C. N. (2014). context – lightweight text analytics using linked data. In *11th Extended Semantic Web Conference (ESWC 2014)*, pages 628–643. Springer International Publishing Switzerland.

[Khalili et al., 2012b] Khalili, A., Auer, S., Tarasowa, D., and Ermilov, I. (2012b). Slidewiki: Elicitation and sharing of corporate knowledge using presentations. In Teije, A., Völker, J., Handschuh, S., Stuckenschmidt, H., d'Acquin, M., Nikolov, A., Aussenac-Gilles, N., and Hernandez, N., editors, *The 18th International Conference on Knowledge Engineering and Knowledge Management (EKAW 2012)*, volume 7603 of *Lecture Notes in Computer Science*, pages 302–316. Springer Berlin Heidelberg.

[Khalili and Sedaghati, 2013a] Khalili, A. and Sedaghati, B. (2013a). Semantic medical prescriptions – towards intelligent and interoperable medical prescriptions. In *IEEE Seventh International Conference on Semantic Computing (ICSC 2013)*, pages 347–354.

[Khalili and Sedaghati, 2013b] Khalili, A. and Sedaghati, B. (2013b). A wysiwym interface for semantic enrichment of e-prescriptions using linked open drug data. *International Journal On Advances in Life Sciences*, 5(3,4):204 – 213.

[Kitchenham, 2004] Kitchenham, B. (2004). Procedures for performing systematic reviews. Technical report, Keele University and NICTA.

[Kiyavitskaya et al., 2009] Kiyavitskaya, N., Zeni, N., Cordy, J. R., Mich, L., and Mylopoulost, J. (2009). Cerno: Light-weight tool support for semantic annotation of textual documents. *Data & Knowledge Engineering*, 68(12):1470 – 1492. Including Special Section: 21st IEEE International Symposium on Computer-Based Medical Systems (IEEE CBMS 2008) - Seven selected and extended papers on Biomedical Data Mining.

[Klebeck et al., 2011] Klebeck, A., Hellmann, S., Ehrlich, C., and Auer, S. (2011). Ontosfeeder – a versatile semantic context provider for web content authoring. In *The Semanic Web: Research and Applications*, volume 6644 of *Lecture Notes in Computer Science*, pages 456–460. Springer.

[Kock et al., 2009] Kock, E. D., Biljon, J. V., and Pretorius, M. (2009). Usability evaluation methods : Mind the gaps. *Evaluation*, pages 122–131.

[Krötzsch et al., 2007] Krötzsch, M., Vrandečić, D., Völkel, M., Haller, H., and Studer, R. (2007). Semantic Wikipedia. *Journal of Web Semantics*, 5(4):251–261.

[Kurki and Hyvönen, 2010] Kurki, J. and Hyvönen, E. (2010). Collaborative metadata editor integrated with ontology services and faceted portals. In *1st Workshop on Ontology Repositories and Editors for the Semantic Web*.

[Lane et al., 2010] Lane, N., Miluzzo, E., Lu, H., Peebles, D., Choudhury, T., and Campbell, A. (2010). A survey of mobile phone sensing. *Communications Magazine, IEEE*, 48(9):140–150.

[Lauesen, 2005] Lauesen, S. (2005). *User Interface Design: A Software Engineering Perspective*. Addison Wesley.

[Leuf and Cunningham, 2001] Leuf, B. and Cunningham, W. (2001). *The Wiki way: quick collaboration on the Web*. Addison-Wesley, London.

[Lewis and Sauro, 2009] Lewis, J. and Sauro, J. (2009). The Factor Structure of the System Usability Scale. In *Human Centered Design*, volume 5619 of *LNCS*, pages 94–103.

[Lin et al., ] Lin, J., Thomsen, M., and Landay, J. A. A visual language for sketching large and complex interactive designs. CHI '02, pages 307–314. ACM.

[Loecken et al., 2012] Loecken, A., Hesselmann, T., Pielot, M., Henze, N., and Boll, S. (2012). User-centred process for the definition of free-hand gestures applied to controlling music playback. *Multimedia Syst.*, 18(1):15–31.

[Lohmann et al., 2008] Lohmann, S., Heim, P., Auer, S., Dietzold, S., and Riechert, T. (2008). Semantifying requirements engineering – the softwiki approach. In *Proceedings of the 4th International Conference on Semantic Technologies (I-SEMANTICS '08)*, J.UCS, pages 182–185.

[Lopez et al., 2011] Lopez, V., Uren, V., Sabou, M., and Motta, E. (2011). Is question answering fit for the semantic web? a survey. *Semantic Web ? Interoperability, Usability, Applicability*, 2(2):125–155.

[Luczak-Roesch, 2009] Luczak-Roesch, R. H. M. (2009). Linked data authoring for non-experts. In *WWW WS on Linked Data on the Web (LDOW2009)*.

[Makhoul et al., 1999] Makhoul, J., Kubala, F., Schwartz, R., and Weischedel, R. (1999). Performance measures for information extraction. In *In Proceedings of DARPA Broadcast News Workshop*, pages 249–252.

[Melissa Conrad Stoppler, 2012] Melissa Conrad Stoppler, M. (2012). http://www.medicinenet.com/script/main/art.asp?articlekey=55234.

[Mendes, 2013] Mendes, P. (2013). *Adaptive Semantic Annotation of Entity and Concept Mentions in Text*. PhD thesis, Department of Computer Science and Engineering, Wright State University.

[Mendes et al., 2011] Mendes, P. N., Jakob, M., García-Silva, A., and Bizer, C. (2011). Dbpedia spotlight: shedding light on the web of documents. In *Proceedings of the 7th International Conference on Semantic Systems*, I-Semantics '11, pages 1–8, New York, USA. ACM.

[Miles and Huberman, 1994] Miles, M. B. and Huberman, M. (1994). *Qualitative Data Analysis: An Expanded Sourcebook(2nd Edition)*. Sage Publications, Inc, 2nd edition.

[Morsey et al., 2011] Morsey, M., Lehmann, J., Auer, S., and Ngonga Ngomo, A.-C. (2011). Dbpedia sparql benchmark – performance assessment with real queries on real data. In *ISWC 2011*.

[Muller et al., 2011] Muller, W., Rojas, I., Eberhart, A., Haase, P., and Schmidt, M. (2011). A-r-e: The author-review-execute environment. *Procedia Computer Science*, 4:627 – 636. ICCS 2011.

[Myers, 1998] Myers, B. A. (1998). A brief history of human-computer interaction technology. *interactions*, 5(2):44–54.

[Möller et al., 2006] Möller, K., Bojars, U., and Breslin, J. (2006). Using semantics to enhance the blogging experience. In Sure, Y. and Domingue, J., editors, *The Semantic Web: Research and Applications*, volume 4011 of *Lecture Notes in Computer Science*, pages 679–696. Springer Berlin Heidelberg.

[Navarro-Galindo and Samos, 2010] Navarro-Galindo, J. L. and Samos, J. (2010). Manual and automatic semantic annotation of web documents: the flersa tool. In *Proceedings of the 12th International Conference on Information Integration and Web-based Applications Services*, iiWAS '10, pages 542–549, New York, NY, USA. ACM.

[Ngomo et al., 2013] Ngomo, A.-C., Kolb, L., Heino, N., Hartung, M., Auer, S., and Rahm, E. (2013). When to reach for the cloud: Using parallel hardware for link discovery. In Cimiano, P., Corcho, O., Presutti, V., Hollink, L., and Rudolph, S., editors, *The Semantic Web: Semantics and Big Data*, volume 7882 of *Lecture Notes in Computer Science*, pages 275–289. Springer Berlin Heidelberg.

[Ngomo et al., 2011] Ngomo, A.-C. N., Heino, N., Lyko, K., Speck, R., and Kaltenböck, M. (2011). Scms - semantifying content management systems. In *ISWC*, pages 189–204.

[Nielsen, 2012] Nielsen, J. (2012). Introduction to usability.

[Nielsen and Molich, 1990] Nielsen, J. and Molich, R. (1990). Heuristic evaluation of user interfaces. In *Proceedings of the SIGCHI conference on Human factors in computing systems: Empowering people*, CHI '90, pages 249–256, New York, NY, USA. ACM.

[O'Donoghue et al., 2010] O'Donoghue, S. I., Horn, H., Pafilis, E., Haag, S., Kuhn, M., Satagopam, V. P., Schneider, R., and Jensen, L. J. (2010). Reflect: A practical approach to web semantics. *Web Semantics: Science, Services and Agents on the World Wide Web*, 8(2-3):182 – 189.

[Oviatt et al., 2000] Oviatt, S., Cohen, P., Wu, L., Vergo, J., Duncan, L., Suhm, B., Bers, J., Holzman, T., Winograd, T., Landay, J., Larson, J., and Ferro, D. (2000). Designing the user interface for multimodal speech and pen-based gesture applications: state-of-the-art systems and future research directions. *Hum.-Comput. Interact.*, 15(4):263–322.

[Patel and Khuba, 2009] Patel, D. R. and Khuba, S. A. (2009). Realization of semantic atom blog. *Journal of Computing*, 1:34 – 38.

[Paulheim and Probst, 2010] Paulheim, H. and Probst, F. (2010). Ontology-enhanced user interfaces: A survey. *International Journal on Semantic Web and Information Systems (IJSWIS)*, 6:2.

[Perdrix et al., 2009] Perdrix, F., García, R., Gil, R., Oliva, M., and Macías, J. A. (2009). Semantic web interfaces for newspaper multimedia content management. In *New Trends on Human-Computer Interaction*, pages 1–10. Springer London. 10.1007/978-1-84882-352-53.

[Petrucka et al., 2013] Petrucka, P., Bassendowski, S., Roberts, H., and James, T. (2013). mhealth: A vital link for ubiquitous health. *Online Journal of Nursing Informatics (OJNI)*, 17:2675.

[Pietriga et al., 2006] Pietriga, E., Bizer, C., Karger, D. R., and Lee, R. (2006). Fresnel: A browser-independent presentation vocabulary for rdf. In *ISWC*, LNCS, pages 158–171. Springer.

[Power et al., 1998] Power, R., Scott, D., and Evans, R. (1998). What You See Is What You Meant: direct knowledge editing with natural language feedback. In *European Conference on Artificial Intelligence (ECAI)*, pages 677 – 681.

[Preotiuc-Pietro et al., 2012] Preotiuc-Pietro, D., Samangooei, S., Cohn, T., Gibbins, N., and Niranjan, M. (2012). Trendminer: an architecture for real time analysis of social media text. http://people.eng.unimelb.edu.au/tcohn/papers/trendminer+ramss+2012.pdf.

[Prud'hommeaux and Seaborne, 2008] Prud'hommeaux, E. and Seaborne, A. (2008). SPARQL query language for RDF. http://www.w3.org/TR/rdf-sparql-query/.

[Puustjärvi and Puustjärvi, 2006] Puustjärvi, J. and Puustjärvi, L. (2006). The challenges of electronic prescription systems based on semantic web technologies. In *ECEH*, pages 251–261.

[Quint and Vatton, 2007] Quint, V. and Vatton, I. (2007). Structured templates for authoring semantically rich documents. In *Proceedings of the 2007 international workshop on Semantically aware document processing and indexing*, SADPI '07, pages 41–48, New York, NY, USA. ACM.

[Riechert et al., 2010] Riechert, T., Morgenstern, U., Auer, S., Tramp, S., and Martin, M. (2010). The catalogus professorum lipsiensis – semantics-based collaboration and exploration for historians. In *Proceedings of the 9th International Semantic Web Conference (ISWC2010)*, Lecture Notes in Computer Science, Shanghai / China. Springer.

[Rizzo and Troncy, 2011] Rizzo, G. and Troncy, R. (2011). Nerd : a framework for evaluating named entity recognition tools in the web of data.

[Ronallo, 2012] Ronallo, J. (2012). HTML5 Microdata and Schema.org. *The Code4Lib Journal*, (16).

[Ross and Nisbett, 1991] Ross, L. and Nisbett, R. E. (1991). *The person and the situation : perspectives of social psychology / Lee Ross, Richard E. Nisbett.* Temple University Press Philadelphia.

[Ruiz-Rube et al., 2010] Ruiz-Rube, I., Cornejo, C. M., Dodero, J. M., and García, V. M. (2010). Development issues on linked data weblog enrichment. In Sánchez-Alonso, S. and Athanasiadis, I. N., editors, *Metadata and Semantic Research*, volume 108 of *Communications in Computer and Information Science*, pages 235–246. Springer. 10.1007/978-3-642-16552-8-22.

[Sah et al., 2007] Sah, M., Hall, W., Gibbins, N. M., and Roure, D. C. D. (2007). Semport - a personalized semantic portal. In *18th ACM Conf. on Hypertext and Hypermedia*, pages 31–32.

[Sahoo et al., 2009] Sahoo, S. S., Halb, W., Hellmann, S., Idehen, K., Jr, T. T., Auer, S., Sequeda, J., and Ezzat, A. (2009). A survey of current approaches for mapping of relational databases to rdf. http://www.w3.org/2005/Incubator/rdb2rdf/RDB2RDF_SurveyReport.pdf.

[Saleem et al., 2013] Saleem, M., Padmanabhuni, S. S., Ngonga Ngomo, A.-C., Almeida, J. S., Decker, S., and Deus, H. F. (2013). Linked cancer genome atlas database. In *Proceedings of I-Semantics*.

[Samwald et al., 2011a] Samwald, M., Jentzsch, A., Bouton, C., Kallesoe, C., Willighagen, E., Hajagos, J., Marshall, M., Prud'hommeaux, E., Hassanzadeh, O., Pichler, E., and Stephens, S. (2011a). Linked open drug data for pharmaceutical research and development. *Journal of Cheminformatics*, 3(1).

[Samwald et al., 2011b] Samwald, M., Jentzsch, A., Bouton, C., KallesÃ¸e, C. S., Willighagen, E., Hajagos, J., Marshall, M. S., Prudâ€™hommeaux, E., Hassenzadeh, O., Pichler, E., and Stephens, S. (2011b). Linked open drug data for pharmaceutical research and development. *Journal of Cheminformatics*, 3(19).

[Sauer, 2006] Sauer, C. (2006). What you see is wiki – questioning WYSIWYG in the Internet age. In *Proceedings of Wikimania 2006*.

[Schaffert, 2006] Schaffert, S. (2006). Ikewiki: A semantic wiki for collaborative knowledge management. In *Enabling Technologies: Infrastructure for Collaborative Enterprises, 2006. WETICE '06. 15th IEEE International Workshops on*, pages 388–396.

[Schaffert et al., 2008] Schaffert, S., Bry, F., Baumeister, J., and Kiesel, M. (2008). Semantic wikis. *IEEE Software*, 25(4):8–11.

[Schaffert et al., 2009] Schaffert, S., Eder, J., Grünwald, S., Kurz, T., Radulescu, M., Sint, R., and Stroka, S. (2009). Kiwi - a platform for semantic social software. In *SemWiki*.

[Seaman, 1999] Seaman, C. B. (1999). Qualitative methods in empirical studies of software engineering. *IEEE Trans. Software Eng.*, 25(4):557–572.

[Seffah et al., 2006] Seffah, A., Donyaee, M., Kline, R. B., and Padda, H. K. (2006). Usability measurement and metrics: A consolidated model. *Software Quality Control*, 14(2):159–178.

[Sheu et al., 2010] Sheu, P., Yu, H., Ramamoorthy, C. V., Joshi, A. K., and Zadeh, L. A. (2010). *Semantic Computing*. Wiley-IEEE Press.

[Shneiderman, 2000] Shneiderman, B. (2000). Creating creativity: user interfaces for supporting innovation. *ACM Trans. Comput.-Hum. Interact.*, 7(1):114–138.

[Simperl, 2012] Simperl, E. (2012). Crowdsourcing semantic data management: Challenges and opportunities. In *Proceedings of the 2Nd International Conference on Web Intelligence, Mining and Semantics*, WIMS '12, pages 1:1–1:3, New York, NY, USA. ACM.

[Siorpaes and Simperl, 2010] Siorpaes, K. and Simperl, E. (2010). Human intelligence in the process of semantic content creation. *WORLD WIDE WEB-INTERNET AND WEB INFORMATION SYSTEMS*, 13(1-2, SI):33–59.

[Spiesser and Kitchen, ] Spiesser, J. and Kitchen, L. Optimization of html automatically generated by wysiwyg programs. In *WWW 2004*, pages 355–364.

[Tarasowa et al., 2014] Tarasowa, D., Auer, S., Khalili, A., and Unbehauen, J. (2014). Crowd-sourcing (semantically) structured multilingual educational content (cosmec). *Open Praxis*, 6(2).

[Tarasowa et al., 2013] Tarasowa, D., Khalili, A., Auer, S., and Unbehauen, J. (2013). Crowdlearn: Crowd-sourcing the creation of highly-structured e-learning content. In Foley, O., Restivo, M. T., Uhomoibhi, J. O., and Helfert, M., editors, *CSEDU*, pages 33–42. SciTePress.

[Thórisson et al., 2010] Thórisson, K., Spivack, N., and Wissner, J. (2010). The semantic web: From representation to realization. In *Transactions on Computational Collective Intelligence II*, volume 6450 of *Lecture Notes in Computer Science*, pages 90–107. Springer.

[Tramp et al., 2010] Tramp, S., Heino, N., Auer, S., and Frischmuth, P. (2010). Rdfauthor: Employing rdfa for collaborative knowledge engineering. In *Knowledge Engineering and Management by the Masses*, volume 6317 of *LNCS*, pages 90–104. Springer.

[Treviranus, 2008] Treviranus, J. (2008). Authoring tools. In Harper, S. and Yesilada, Y., editors, *Web Accessibility*, Human-Computer Interaction Series, pages 127–138. Springer. 10.1007/978-1-84800-050-69.

[Tunkelang, 2009] Tunkelang, D. (2009). *Faceted Search (Synthesis Lectures on Information Concepts, Retrieval, and Services)*. Morgan and Claypool Publishers.

[Uren et al., 2006] Uren, V., Cimiano, P., Iria, J., Handschuh, S., Vargas-Vera, M., Motta, E., and Ciravegna, F. (2006). Semantic annotation for knowledge management: Requirements and a survey of the state of the art. *Web Semantics: Science, Services and Agents on the World Wide Web*, 4(1):14 – 28.

[Valaski et al., 2012] Valaski, J., Malucelli, A., and Reinehr, S. (2012). Ontologies application in organizational learning: A literature review. *Expert Systems with Applications*, 39(8):7555 – 7561.

[Valkeapaeae et al., 2007] Valkeapaeae, O., Alm, O., and Hyvoenen, E. (2007). An adaptable framework for ontology-based content creation on the semantic web. *JOURNAL OF UNIVERSAL COMPUTER SCIENCE*, 13(12):1835–1853. 28th Annual Meeting of the Society-of-Behavioral-Medicine, Washington, DC, MAR 21-24, 2007.

[Van Kleek et al., 2007] Van Kleek, M., Bernstein, M., Karger, D. R., and schraefel, m. (2007). Gui — phooey!: The case for text input. In *Proceedings of the 20th Annual ACM Symposium on User Interface Software and Technology*, UIST '07, pages 193–202, New York, NY, USA. ACM.

[W3C, 2004] W3C (2004). Resource description framework (rdf). http://www.w3.org/RDF/.

[W3C, 2009] W3C (2009). W3C semantic web activity. http://www.w3.org/2001/sw/. Última visita 8/6/2010.

[W3Techs, 2011] W3Techs (2011). Usage of content management systems for websites. http://w3techs.com/technologies/overview/content_management/all.

[Wang et al., 2012] Wang, X., Love, P. E., Klinc, R., Kim, M. J., and Davis, P. R. (2012). Integration of e-learning 2.0 with web 2.0. *ITcon - Special Issue eLearning 2.0: Web 2.0-based social learning in built environment*, 17:387–396.

[Wikipedia, 2013] Wikipedia (2013). SPARQL — Wikipedia, The Free Encyclopedia. http://en.wikipedia.org/w/index.php?title=SPARQL&oldid=544624084. [Online; accessed 31-March-2013].

[Williams et al., 2012] Williams, A. J., Harland, L., Groth, P., Pettifer, S., Chichester, C., Willighagen, E. L., Evelo, C. T., Blomberg, N., Ecker, G., Goble, C., and Mons, B. (2012). Open phacts: semantic interoperability for drug discovery. *Drug Discovery Today*, 17(21-22):1188 – 1198.

[Yang et al., 2013] Yang, H., Pupons-Wickham, D., Chiticariu, L., Li, Y., Nguyen, B., and Carreno-Fuentes, A. (2013). I can do text analytics!: designing development tools for novice developers. CHI '13, pages 1599–1608, New York, NY, USA. ACM.

[Yu, 2006] Yu, B. (2006). Cognitive aspects of human-gis interaction : A literature review cognitive aspects of human-gis interaction : A literature review. *Interface*, pages 1–17.

[Yu, 2007] Yu, L. (2007). *Introduction to Semantic Web and Semantic Web services*. Chapman & Hall/CRC, Boca Raton, FL.

[Łazaruk et al., 2012] Łazaruk, S., Kaczmarek, M., Dzikowski, J., Tokarchuk, O., and Abramowicz, W. (2012). Towards the semantic web – incentivizing semantic annotation creation process. In Teije, A., Völker, J., Handschuh, S., Stuckenschmidt, H., d'Acquin, M., Nikolov, A., Aussenac-Gilles, N., and Hernandez, N., editors, *Knowledge Engineering and Knowledge Management*, volume 7603 of *Lecture Notes in Computer Science*, pages 282–291. Springer Berlin Heidelberg.

# Selbständigkeitserklärung

Hiermit erkläre ich, die vorliegende Dissertation selbständig und ohne unzulässige fremde Hilfe angefertigt zu haben. Ich habe keine anderen als die angeführten Quellen und Hilfsmittel benutzt und sämtliche Textstellen, die wörtlich oder sinngemäß aus veröffentlichten oder unveröffentlichten Schriften entnommen wurden, und alle Angaben, die auf mündlichen Auskünften beruhen, als solche kenntlich gemacht. Ebenfalls sind alle von anderen Personen bereitgestellten Materialien oder erbrachten Dienstleistungen als solche gekennzeichnet.

Leipzig, den 26.1.2015

Ali Khalili