The RDFa Content Editor – From WYSIWYG to WYSIWYM

Ali Khalili and Sören Auer

Universität Leipzig, Institut für Informatik, AKSW,
Johannisgasse 26, 04103 Leipzig, Germany
{lastname}@informatik.uni-leipzig.de
http://aksw.org

Abstract. Recently practical approaches for managing and supporting the life-cycle of semantic content on the Web of Data made quite some progress. However, the currently least developed aspect of the semantic content life-cycle is the user-friendly manual and semi-automatic creation of rich semantic content. In this paper we present the RDFaCE approach for combining WYSIWYG text authoring with the creation of rich semantic annotations. WYSIWYG text authoring is meanwhile ubiquitous on the Web and part of most content creation and management workflows. Our approach is based on providing four different views to the content authors: a classical WYSIWYG view, a WYSIWYM (What You See Is What You Mean) view making the semantic annotations visible, a fact view and the respective HTML/RDFa source code view. The views are synchronized such that changes made in one of the views automatically update the others. They provide different means of semantic content authoring for the different personas involved in the content creation life-cycle. For bootstrapping the semantic annotation process we integrate five different text annotation services. We evaluate their accuracy and empirically show that a combination of them yields superior results.

Keywords: RDFa, semantic content authoring, text annotation, rNews

1 Introduction

Recently practical approaches for managing and supporting the life-cycle of semantic content on the Web of Data made quite some progress. On the backend side, a variety of triple stores were developed and their performance and maturity improves steadily. Similarly tools and algorithms for linking data and schemata are progressing and approaches are deployed for the use on the emerging Web of Data. The quantity of semantic content being made available on the Data Web is rapidly increasing, mainly due to the use of automated knowledge extraction techniques or due to the semantic enrichment and transformation of existing structured data. Despite many interesting showcases (e.g. Sindice, Parallax or PowerAqua), we still lack more user friendly and scalable approaches for the exploration, browsing and search of semantic content. However, 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 paper we present the *RDFaCE* approach for combining WYSIWYG text authoring with the creation of rich semantic annotations. WYSIWYG text authoring is meanwhile ubiquitous on the Web and part of most content creation and management workflows. It is part of Content Management Systems, Weblogs, Wikis, fora, product data management systems and online shops, just to mention a few. Our goal with this work is 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. We devise the concept of a *What You See Is What You Mean* (WYSIWYM) view, which extends the WYSIWYG view with highlighted semantic annotations. In addition a fact view helps content authors, curators as well as content and knowledge engineers to quickly review and possibly revise the semantic annotations. The rationale behind our WYSIWYM concept is that we have 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. Hence, WYSIWYM is an extension of the WYSIWYG concept, where users can observe and manage the semantic annotations right in the familiar context of the WYSIWYG environment. Depending on the focus of work (or the persona using the tool) the various RDFaCE views provide a different balance between the visualization of the semantic annotations and the formated textual content.

Despite the improved usability of semantic content creation we aim to achieve with this work, the manual semantic annotation of longer articles can still be a tedious and time-consuming task. In order to kick-start the annotation, we integrated five automatic annotation services into RDFaCE. During an extensive evaluation of these services, we observed that each individual tool has either weaknesses regarding precision, recall or the support of certain information domains. Our evaluation also reveals, that a substantial increase in precision *and* recall can be achieved, when combining the output of *several* services.

The contributions of this work are in particular:

1. An architecture and implementation of an RDFa 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.

The remainder of this article is structured as follows: Section 2 describes the architecture of the system. Section 3 presents the different views for semantic text authoring. In Section 4, the combination of different NLP APIs for bootstrapping the semantic annotation process is discussed. Section 5 introduces three RDFaCE use cases. Related work is presented in Section 6 and finally Section 7 concludes with an outlook on future work.

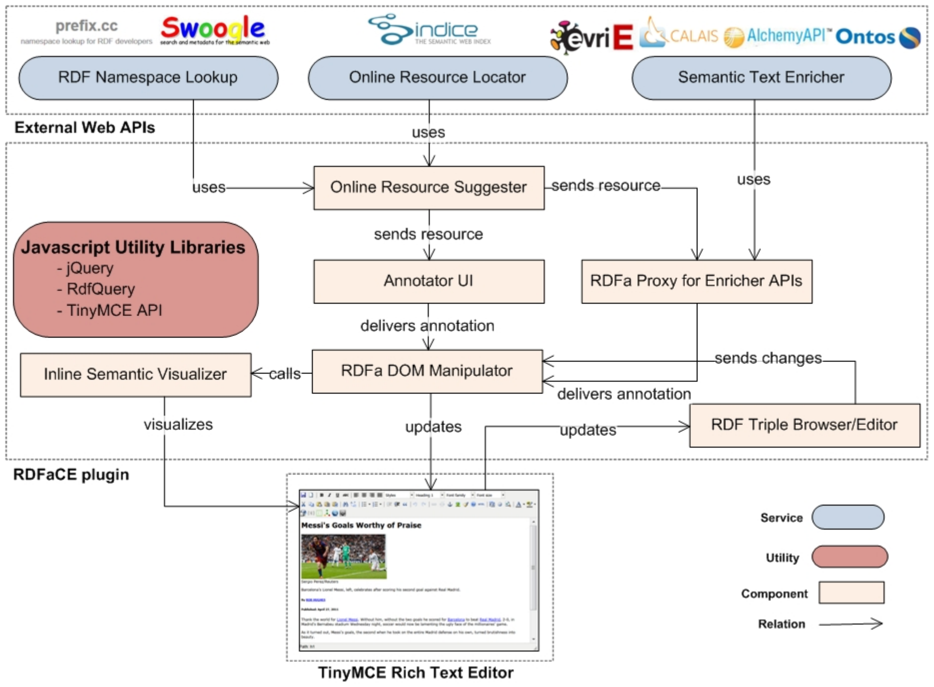


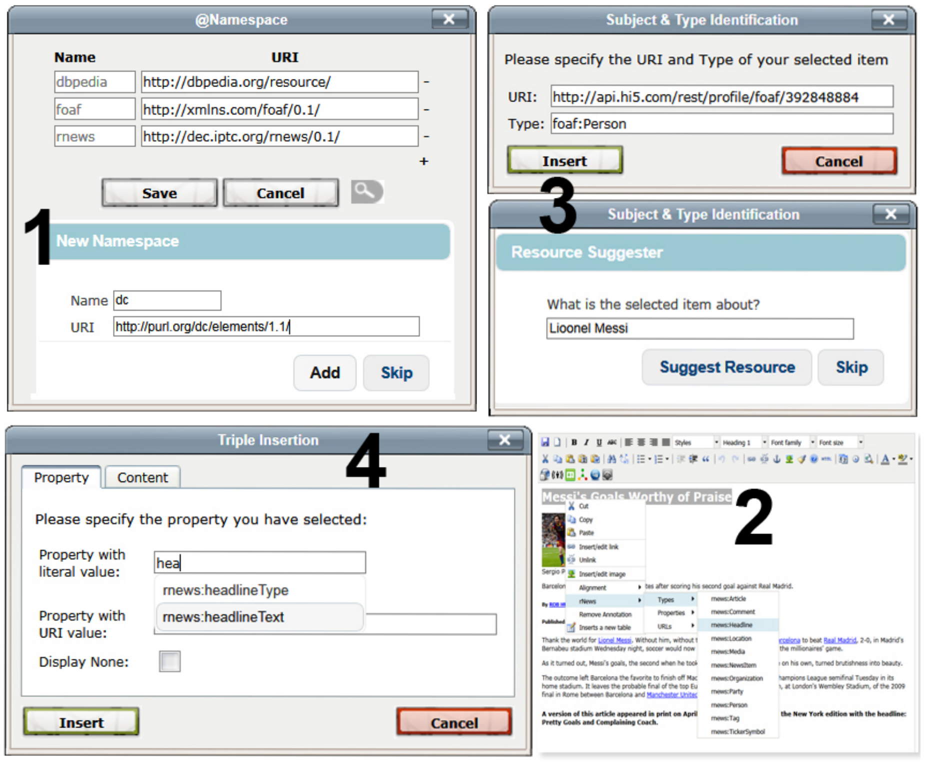
Fig. 1. RDFaCE system architecture.

2 Architecture and Implementation

The RDFaCE system architecture is depicted in Figure 1 and consists of three layers. The foundation layer on which we ground the RDFaCE plugin includes the *TinyMCE Rich Text Editor*[1]. This open source HTML editor was chosen because it is very flexible to extend and is used in many popular Content Management Systems (CMS), blogs, wikis and discussion forums, etc. Therefore, by focusing efforts on this one particular editor, it is possible to quickly propagate accessible RDFa authoring practices to a number of other tools [10]. The RDFaCE implementation is open-source and available for download together with an explanatory video and online demo at `http://aksw.org/Projects/RDFaCE`. RDFaCE includes the following components:

Annotator UI. This component uses the TinyMCE API as well as jQuery UI to create user friendly user interfaces for RDFa content editing. As shown in Figure 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

[1] `http://tinymce.moxiecode.com`


@Namespace

Name	URI
dbpedia	http://dbpedia.org/resource/
foaf	http://xmlns.com/foaf/0.1/
rnews	http://dec.iptc.org/rnews/0.1/

New Namespace

Name: dc

URI: http://purl.org/dc/elements/1.1/

Subject & Type Identification

Please specify the URI and Type of your selected item

URI: http://api.hi5.com/rest/profile/foaf/392848884

Type: foaf:Person

Resource Suggester

What is the selected item about?

Lioonel Messi

Triple Insertion

Please specify the property you have selected:

Property with literal value: hea

rnews:headlineType

rnews:headlineText

Property with URI value:

Display None:

Fig. 2. Annotation user interface.

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 DOM manipulator.

RDFa DOM Manipulator. This component is responsible for manipulating the Document Object Model (DOM) according to the desired RDFa annotation. The simplest solution to add RDFa attributes to content is using `<span>` tags. For each new annotation, a new `<span>` can be created containing related RDFa 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 annotated content which might be overlapping. To show the value of RDFa 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.

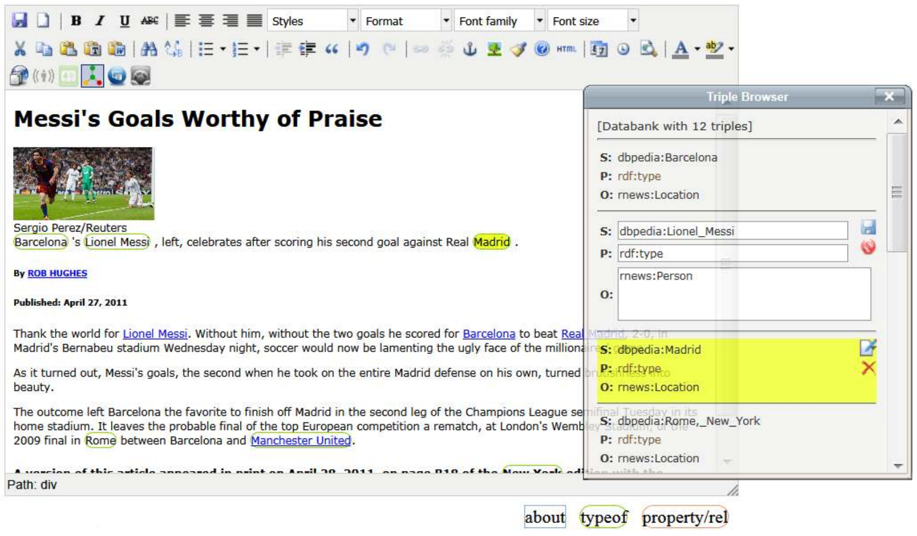
RDF Triple Browser and Editor. This component extracts the RDF triples embedded in the text and provides the edit and delete functionality for these triples. This component is in a mutual relation to rich text editor and is dynamically updated when a new annotation is added to the text (also the text editor is updated when a triple is modified here). When user edits or deletes a triple, these changes are delivered to RDFa 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 (a detailed explainable follows below). 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 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 output. Therefore, we need to convert their generated output into RDFa. To do this, the RDFa 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 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[2] and RDFQuery[3] to implement their functions. To facilitate semantic annotation of

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



Messi's Goals Worthy of Praise

Sergio Perez/Reuters
Barcelona 's Lionel Messi , left, celebrates after scoring his second goal against Real Madrid .

By ROB HUGHES

Published: April 27, 2011

Thank the world for Lionel Messi. Without him, without the two goals he scored for Barcelona to beat Real Madrid, 2-0, in
Madrid's Bernabeu stadium Wednesday night, soccer would now be lamenting the ugly face of the millionaires

As it turned out, Messi's goals, the second when he took on the entire Madrid defense on his own, turned brutishness into
beauty.

The outcome left Barcelona the favorite to finish off Madrid in the second leg of the Champions League semifinal Tuesday in its
home stadium. It leaves the probable final of the top European competition a rematch, at London's Wembley Stadium, of the
2009 final in Rome between Barcelona and Manchester United.

[Databank with 12 triples]

S: dbpedia:Barcelona
P: rdf:type
O: rnews:Location

S: dbpedia:Lionel_Messi
P: rdf:type
O: rnews:Person

S: dbpedia:Madrid
P: rdf:type
O: rnews:Location

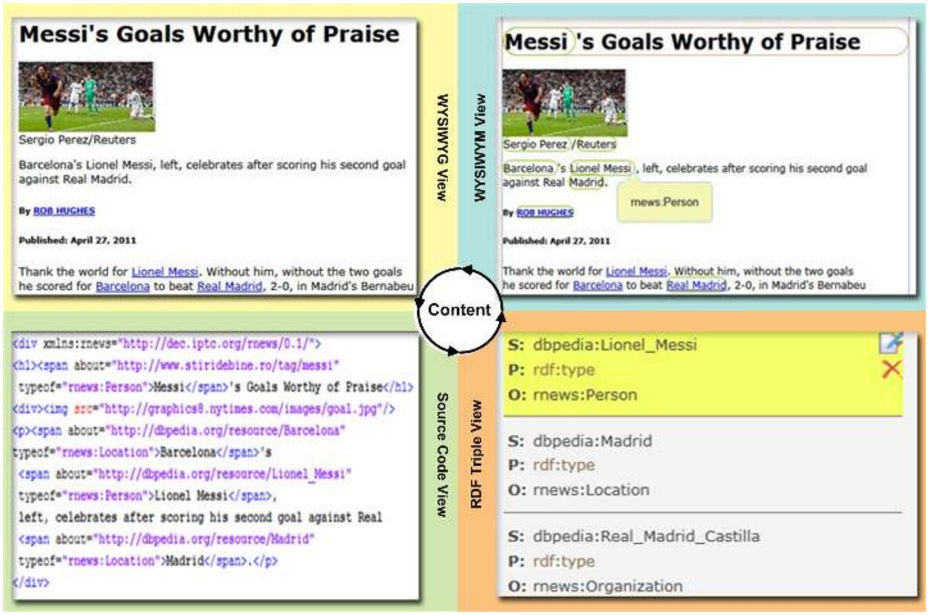
S: dbpedia:Rome,_New_York
P: rdf:type
O: rnews:Location

Fig. 3. The main RDFaCE WYSIWIM view (left) with the RDF Triple Browser and Editor (right).

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

[4] `http://swoogle.umbc.edu/`
[5] `http://sindice.com`



Messi's Goals Worthy of Praise

Sergio Perez/Reuters

Barcelona's Lionel Messi, left, celebrates after scoring his second goal against Real Madrid.

By ROB HUGHES

Published: April 27, 2011

Thank the world for Lionel Messi. Without him, without the two goals he scored for Barcelona to beat Real Madrid, 2-0, in Madrid's Bernabeu

```
<div xmlns:rnews="http://dec.iptc.org/rnews/0.1/">
<h1><span about="http://www.stiridebine.ro/tag/messi"
 typeof="rnews:Person">Messi</span>'s Goals Worthy of Praise</h1>
<div><img src="http://graphics8.nytimes.com/images/goal.jpg"/>
<p><span about="http://dbpedia.org/resource/Barcelona"
typeof="rnews:Location">Barcelona</span>'s
 <span about="http://dbpedia.org/resource/Lionel_Messi"
 typeof="rnews:Person">Lionel Messi</span>,
 left, celebrates after scoring his second goal against Real
 <span about="http://dbpedia.org/resource/Madrid"
 typeof="rnews:Location">Madrid</span>.</p>
</div>
```

S: dbpedia:Lionel_Messi
P: rdf:type
O: rnews:Person

S: dbpedia:Madrid
P: rdf:type
O: rnews:Location

S: dbpedia:Real_Madrid_Castilla
P: rdf:type
O: rnews:Organization

Fig. 4. The four views for semantic text authoring.

the *OpenCalais*, *Ontos*, *Alchemy*, *Extractiv* and *Evri* NLP APIs[6] to enrich the text.

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 4 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 syncronized so that applying changes in one of the views automatically updates other views.

WYSIWYG View. The What-You-See-Is-What-You-Get view 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 Web or offline.

[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`

In particular WYSIWYG text authoring is already part of Content Management Systems, Weblogs, Wikis, fora, product data management systems and online shops, just to mention a few.

WYSIWYM View. The What-You-See-Is-What-You-Mean view is an extension of the WYSIWYG view, which highlights named entities and other semantic information. The highlighting is realized with special CSS3 selectors for the RDFa 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 WYSIWIM 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 overly.

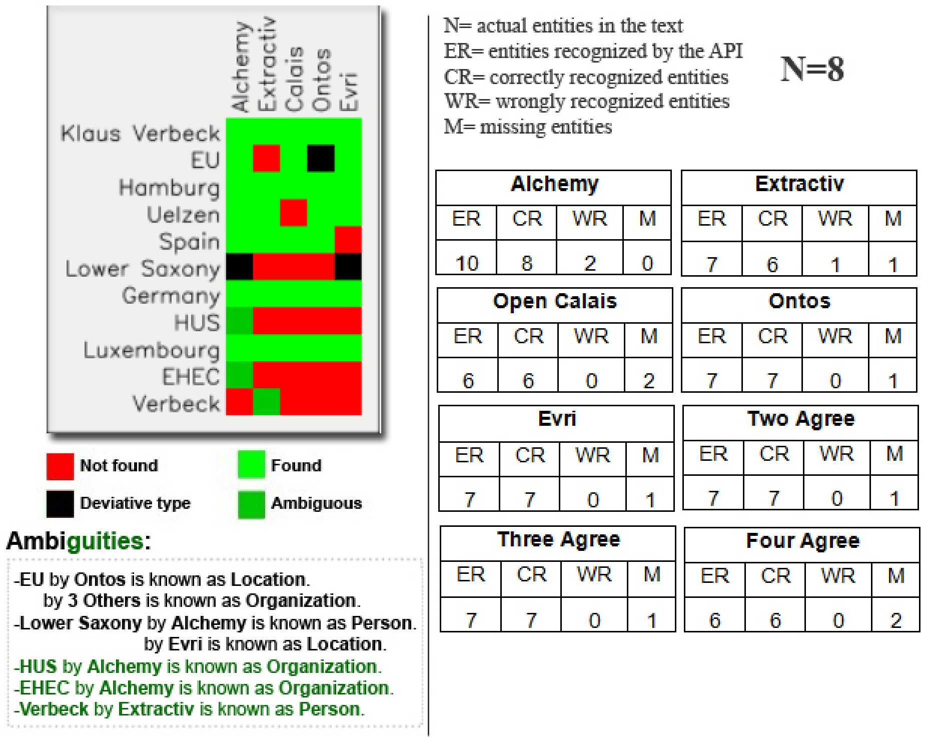
RDF Triple View. This view summarizes all the facts, which can be extracted from the annotated text. It provides a deeper semantic view when compared to WYSIWYM view. There might be some triples not visible in the WYSIWYM view (e.g. annotations hidden using the CSS `display:none` style) but visible in 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 view is (as all other views) updateable, i.e. facts can be directly deleted, which results in the removal of the corresponding RDFa 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 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 WYSIWIM visualization.

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 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 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 which 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.



	Alchemy	Extractiv	Calais	Ontos	Evri
Klaus Verbeck	🟩	🟩	🟩	🟩	🟩
EU	🟩	🟥	🟩	⬛	🟩
Hamburg	🟩	🟩	🟩	🟩	🟩
Uelzen	🟩	🟩	🟥	🟩	🟩
Spain	🟩	🟩	🟩	🟩	🟥
Lower Saxony	⬛	🟥	🟥	🟥	⬛
Germany	🟩	🟩	🟩	🟩	🟩
HUS	▨	🟥	🟥	🟥	🟥
Luxembourg	🟩	🟩	🟩	🟩	🟩
EHEC	▨	🟥	🟥	🟥	🟥
Verbeck	🟥	▨	🟥	🟥	🟥

🟥 Not found 🟩 Found

⬛ Deviative type ▨ Ambiguous

Ambiguities:

-EU by Ontos is known as Location.
by 3 Others is known as Organization.
-Lower Saxony by Alchemy is known as Person.
by Evri is known as Location.
-HUS by Alchemy is known as Organization.
-EHEC by Alchemy is known as Organization.
-Verbeck by Extractiv is known as Person.

N= actual entities in the text
ER= entities recognized by the API
CR= correctly recognized entities
WR= wrongly recognized entities
M= missing entities

N=8

Alchemy			
ER	CR	WR	M
10	8	2	0

Extractiv			
ER	CR	WR	M
7	6	1	1

Open Calais			
ER	CR	WR	M
6	6	0	2

Ontos			
ER	CR	WR	M
7	7	0	1

Evri			
ER	CR	WR	M
7	7	0	1

Two Agree			
ER	CR	WR	M
7	7	0	1

Three Agree			
ER	CR	WR	M
7	7	0	1

Four Agree			
ER	CR	WR	M
6	6	0	2

Fig. 5. Generated results of different NLP APIs for article #1.

We use *Precision,Recall* and *F-measure* [8] as metrics for evaluating the correctness of the recognized entities found by each API as well as combined APIs:

$$Recall = \frac{Correctly\ Recognized\ Entities}{Actual\ Entities\ in\ the\ Text} \tag{1}$$

$$Precision = \frac{Correctly\ Recognized\ Entities}{Entities\ Recognized\ by\ the\ API} \tag{2}$$

$$F = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{3}$$

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 analysed 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) 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 (1), Precision (2) and F-Score (3) were calculated for each API as well as for various combinations of APIs. Table 1 presents the detailed results of these metrics for each API as well as for the situation when 2, 3 or 4 of the APIs agree on a recognized entity. The table also shows the average value of these metrics for each category of the analyzed texts as well as the average for all analyzed texts. In Figure 6 the results are visualized as bar charts.

The results 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. Evry. 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 (Figure 6). 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 Use Cases

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 complementary use cases for RDFaCE which exemplify the versatility of the approach.

[7] Available at `http://rdface.aksw.org/samples/`

Table 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

5.1 rNews

rNews[8] is a proposed standard for using RDFa to annotate HTML documents with news-specific metadata. rNews is proposed by International Press Telecommunications Council (IPTC)[9] 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.

RDFaCE is well suited for the rNews vocabulary. It provides an auto suggestion feature for the classes and properties defined in the rNews vocabulary. RDFaCE also provides a context menu for the rNews vocabulary so that users can easily annotate their news articles using the rNews vocabulary. Furthermore,

[8] `http://dev.iptc.org/rNews`
[9] `http://www.iptc.org/`

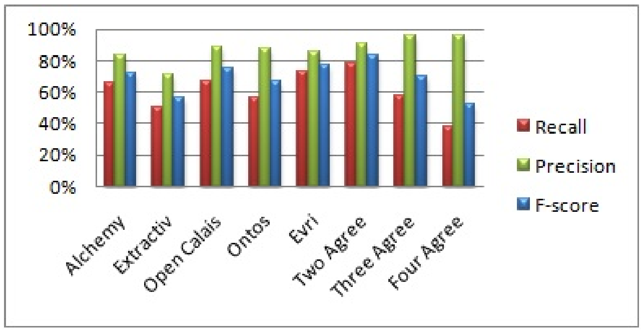


Fig. 6. Average Precision, Recall and F-score for each API and their combination.

RDFaCE 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.

The adaption of RDFaCE for rNews are performed in a way, that the same strategies are easily applicable for similar domain specific annotation vocabularies. RDFaCE can be configured to employ an annotation vocabulary for auto suggestion, context menu and NLP API output mapping.

5.2 OntoWiki

OntoWiki[10] [2] 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. Furthermore, it enables intuitive authoring of semantic content, with an inline editing mode for editing RDF content. OntoWiki supports two complementary edit strategies for the knowledge base: a) *Inline editing* which enables users to edit the smallest possible 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. Ontowiki uses RDFAuthor [9] to make generated RDFa views editable.

Since OntoWiki is other than other semantic Wikis based on the RDF data model in the first place (i.e. facts/triples are the main artifacts instead of annotated wiki texts), the semantic annotation of texts is not possible. By integrating RDFaCE into Ontowiki text literals containing rich text can now be enriched with semantic annotations. These annotations can subsequently automatically extracted and explicitly represented in the knowledge base being edited with OntoWiki. In addition to the form-based editing of information, RDFaCE provides a WYSIWYM editor component to OntoWiki for selecting and editing rich text including RDFa annotations. Thus, RDFaCE contributes significantly

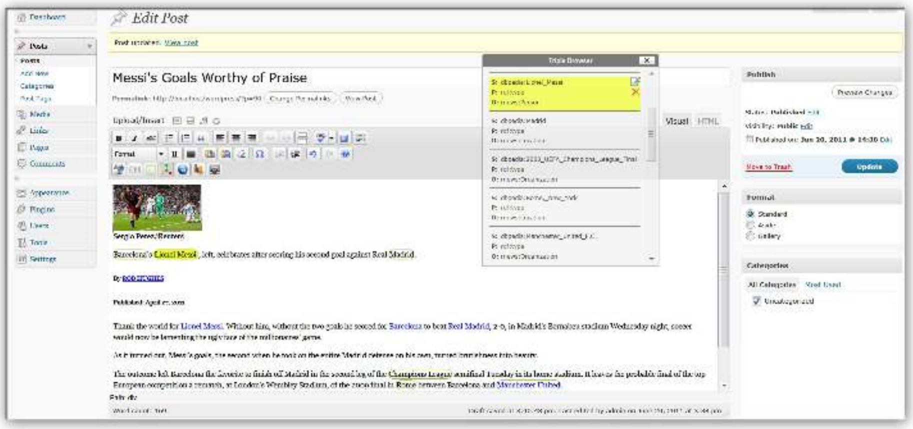
[10] `http://ontowiki.net`

to OntoWiki's aim of facilitating a pay-as-you-go knowledge engineering strategy: users can firstly add textual content, annotated this textual content in a second step and later on explicitly materialize the facts, which can be extracted from the annotations and thus attain a rich semantic representation.

5.3 Wordpress

Wordpress[11] 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) [11].

Wordpress uses TinyMCE as its content editor. That makes it extremely easy to add the RDFaCE plugin for semantic content authoring within this CMS (cf. Figure 7). With the integration of RDFaCE into the Wordpress, the availability of semantically annotated content on the Web can be substantially increased. The semantically annotated content can be indexed by the new generation of search engines, will result in more accurate search results and facilitate the knowledge extraction from the Web in general.



Sergio Perez/Reuters

Barcelona's Lionel Messi, left, celebrates after scoring his second goal against Real Madrid.

Thank the world for Lionel Messi. Without him, without the two goals he scored for Barcelona to beat Real Madrid, 2-0, in Madrid's Bernabeu stadium Wednesday night, soccer would now be lamenting the ugly face of the millionaires' game.

Fig. 7. Integration of RDFaCE into Wordpress.

6 Related Work

Semantic authoring [3] aims to facilitate the composition of intelligent content (i.e. content with explicit semantic structure) based on a set of shared vocabularies. Semantic content is created by annotating the content with appropriate vocabularies. With the emergence of the Web of Data, some efforts have been

[11] `http://wordpress.org`

undertaken to blur the border between authoring and annotation of content as much as possible. *Semantic Wikis* such as, for example, Semantic MediaWiki [6] represent an approach that combines traditional wiki systems with semantic annotation. They provide a collaborative environment for semantic content authoring. Microformats[12], RDFa[13] and most recently Microdata[14] are existing options used by semantic editors to embed the semantic annotations within the Web content. Among them, RDFa as a W3C recommendation is the most expressive and versatile option.

There are already a few RDFa editors available. *RADiFy*[15] is a bookmarklet for annotating a web page with RDFa. As a bookmarklet, it provides a side bar to view, edit and load ontologies. RADiFy provides a list of predefined predicates that can be extended by loading new ontologies. For a user familiar with ontologies and semantic web concepts, it seems to be a useful and simple tool to annotate web pages. However, RADiFy has the following drawbacks: There is no editing option to edit the created triples. Also, the annotation mechanism of simply adding `<span>` tags is not very efficient and in some cases erroneous. For instance, if a link is selected, it adds a `<span>` tag inside the `<a>` tag. In addition, RADiFy does not support the use of the "rel" attribute but uses the "property" attribute for all predicates without considering their values and data types.

WYMeditor[16] is a web-based XHTML editor with support for RDFa. WYMeditor's main concept is to leave details of the document's visual layout, and to concentrate on its structure and meaning, while trying to give the user as much comfort as possible (similar as WYSIWYG editors). The emphasis on strict XHTML compliance, makes it a good option to add RDFa annotations but its RDFa editor does not include many features. It considers blocks as the unit of annotation which makes it difficult to annotate a part of text within a block. It also uses different background colors to distinguish different annotations which can be problematic in cases when there are too many overlapping annotations. Furthermore, editing and deleting annotations in a WYSIWYG way is not supported.

Loomp[17] [7] is a tool representing a prove-of-concept for the One Click Annotation (OCA) strategy. OCA is a concept addressing the issue of creating semantic content by non-expert users [4]. 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. At the time

[12] `http://microformats.org`
[13] `http://www.w3.org/TR/rdfa-syntax/`
[14] `http://www.w3.org/TR/microdata/`
[15] `http://duncangrant.co.uk/radify/`
[16] `http://www.wymeditor.org`
[17] `http://loomp.org`

of writing, we were not able to find any demo of Loomp to further investigate its implemented features.

RDFauthor[18] is another related tool that aims at editing RDFa contents. The RDFauthor approach is based on the idea of making arbitrary XHTML views with integrated RDFa annotations editable [9]. RDFauthor converts an RDFa-annotated view directly into an editable form thereby hiding the RDF and related ontology data models from novice users. 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.

OntosFeeder[19] and *Epiphany*[20] are two partially related tools. Epiphany is a service that annotates web pages automatically with Linked Data by creating an RDFa enhanced versions of the input HTML page [1]. OntosFeeder [5] is also a TinyMCE plugin which uses the Ontos Web API to provide context information (in RDFa) to be integrated into content management systems. These tools use the same annotation mechanism as RADiFy with the above mentioned issues. Furthermore, they do not provide editing functionality for RDFa generated content. They 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.

7 Conclusions and Future Work

The user friendly authoring of semantically enriched content is a crucial aspect for the ultimate success of semantic technologies. With RDFaCE we presented in this article an approach and its implementation of a WYSIWYM (What You See Is What You Mean) 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.

We see the work presented in this article as an initial step in a larger research agenda aiming at simplifying the authoring and annotation of semantically enriched textual content. Regarding future work we envision to extend the WISIWYM concept towards different modalities, such that the annotation of images and multimedia object is supported. With regard to the NLP functionality made available through RDFaCE we aim at extending the current implementation towards supporting relationship and keyword extraction.

[18] `http://aksw.org/Projects/RDFauthor`
[19] `http://wordpress.org/extend/plugins/ontos-feeder/`
[20] `http://projects.dfki.uni-kl.de/epiphany/`

Acknowledgments

We would like to thank our colleagues from AKSW research group for their helpful comments and inspiring discussions during the development of RDFaCE. This work was supported by a grant from the European Union's 7th Framework Programme provided for the project LOD2 (GA no. 257943).

References

1. Benjamin Adrian, Jörn Hees, Ivan Herman, Michael Sintek, and Andreas Dengel. Epiphany: Adaptable rdfa generation linking the web of documents to the web of data. In Philipp Cimiano and H. Pinto, editors, *Knowledge Engineering and Management by the Masses*, volume 6317 of *Lecture Notes in Computer Science*, pages 178–192. Springer Berlin / Heidelberg, 2010. 10.1007/978-3-642-16438-513.
2. Sören Auer, Sebastian Dietzold, and Thomas Riechert. Ontowiki – a tool for social, semantic collaboration. In *The Semantic Web - ISWC 2006, 5th International Semantic Web Conference, ISWC 2006*, pages 736–749. Springer, 2006.
3. Kôiti Hasida. Semantic authoring and semantic computing. In Akito Sakurai, Kôiti Hasida, and Katsumi Nitta, editors, *New Frontiers in Artificial Intelligence*, volume 3609 of *Lecture Notes in Computer Science*, pages 137–149. Springer Berlin / Heidelberg, 2007. 10.1007/978-3-540-71009-712.
4. Ralf Heese, Markus Luczak-Rösch, Radoslaw Oldakowski, Olga Streibel, and Adrian Paschke. One click annotation. In *Scripting and Development for the Semantic Web (SFSW)*, May 2010.
5. Alex Klebeck, Sebastian Hellmann, Christian Ehrlich, and Soren Auer. Ontosfeeder – a versatile semantic context provider for web content authoring. In Grigoris Antoniou, Marko Grobelnik, Elena Simperl, Bijan Parsia, Dimitris Plexousakis, Pieter De Leenheer, and Jeff Pan, editors, *The Semanic Web: Research and Applications*, volume 6644 of *Lecture Notes in Computer Science*, pages 456–460. Springer Berlin / Heidelberg, 2011. 10.1007/978-3-642-21064-834.
6. Markus Krötzsch, Denny Vrandečić, Max Völkel, Heiko Haller, and Rudi Studer. Semantic Wikipedia. *Journal of Web Semantics*, 5(4):251–261, 2007.
7. Ralf Heese Markus Luczak-Roesch. Linked data authoring for non-experts. In *Proceedings of the WWW09, Workshop Linked Data on the Web (LDOW2009)*, 2009.
8. John Makhoul, Francis Kubala, Richard Schwartz, and Ralph Weischedel. Performance measures for information extraction. In *In Proceedings of DARPA Broadcast News Workshop*, pages 249–252, 1999.
9. Sebastian Tramp, Norman Heino, Sören Auer, and Philipp Frischmuth. Rdfauthor: Employing rdfa for collaborative knowledge engineering. In Philipp Cimiano and H. Pinto, editors, *Knowledge Engineering and Management by the Masses*, volume 6317 of *Lecture Notes in Computer Science*, pages 90–104. Springer Berlin / Heidelberg, 2010. 10.1007/978-3-642-16438-57.
10. Jutta Treviranus. Authoring tools. In Simon Harper and Yeliz Yesilada, editors, *Web Accessibility*, Human-Computer Interaction Series, pages 127–138. Springer London, 2008. 10.1007/978-1-84800-050-69.
11. W3Techs. Usage of content management systems for websites, June 2011.