

WYSIWYM authoring of structured content based on Schema.org

Ali Khalili and Sören Auer

University of Leipzig, Institute of Computer Science, AKSW Group,
Augustusplatz 10, D-04009 Leipzig, Germany
`{lastname}@informatik.uni-leipzig.de`,
<http://aksw.org>

Abstract. Structured data is picking up on the Web, particularly in the search world. Schema.org, jointly initiated by Google, Microsoft, and Yahoo! provides a hierarchical set of vocabularies to embed metadata in HTML pages for an enhanced search and browsing experience. RDFa-Lite, Microdata and JSON-LD as lower semantic techniques have gained more attention by Web users to markup Web pages and even emails based on Schema.org. However, from the user interface point of view, we still lack user-friendly tools that facilitate the process of structured content authoring. 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 how humans interact with information. In this paper we present RDFaCE as an implementation of WYSIWYM (What-You-See-Is-What-You-Mean) concept for direct manipulation of semantically structured content in conventional modalities. RDFaCE utilizes on-the-fly form generation based on Schema.org vocabulary for embedding metadata within Web documents. Furthermore, it employs external NLP services to enable automatic annotation of entities and to suggest URIs for entities. RDFaCE is written as a plugin for TinyMCE WYSIWYG editor thereby can be easily integrated into existing content management systems.

Keywords: WYSIWYG, WYSIWYM, Semantic Content Authoring

1 Introduction

Structured data is picking up on the Web, particularly in the search world. Semantic structuring of content provides a wide range of advantages compared to unstructured information. It facilitates a number of important aspects of information management:

- For *search and retrieval* enriching documents with semantic representations helps to create more efficient and effective search interfaces, such as faceted search [18] or question answering [9].

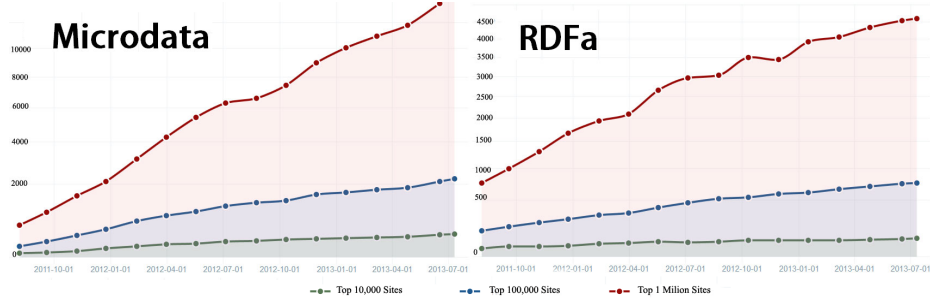


Fig. 1. Microdata and RDFa web usage trends over a historical time period on a large selection of Websites queried by BuiltWith (<http://trends.builtwith.com>).

- 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 [3].
- 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 [2].
- 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 [15].
- For *reusability* and *interoperability* enriching documents with semantic representations facilitates exchanging content between disparate systems and enables building applications such as executable papers [11].

Schema.org, jointly initiated by Google, Microsoft, and Yahoo! provides a collection of shared schemas that webmasters can use to markup their pages in order to enable enhanced search and browsing experiences recognized by major search providers. RDFa, Microdata and JSON-LD as lower semantic techniques have gained more attention by Web users to markup Web pages and even emails based on Schema.org (cf. [10] and Figure 1). However, in order for users to truly benefit from this semantic techniques, we need ways to author, visualize and explore unstructured *and* semantic information in a user-friendly manner. 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 how humans interact with information.

In this paper we present WYSIWYM (What-You-See-Is-What-You-Mean) concept for direct manipulation of semantically structured content in conventional modalities. Our WYSIWYM concept formalizes the binding between semantic representation models and UI elements for authoring, visualizing and exploration. As an implementation of WYSIWYM model suitable for lower semantic techniques, we showcase RDFaCE interface. RDFaCE utilizes on-the-fly

form generation based on Schema.org vocabulary for embedding metadata within Web documents. Furthermore, it employs external NLP services to enable automatic annotation of entities and to suggest URIs for entities. RDFaCE is written as a plugin for TinyMCE WYSIWYG editor thereby can be easily integrated into existing content management systems.

The remainder of this article is structured as follows: In Section 2, we describe the background of our work and discuss the related work. Section 3 describes the fundamental WYSIWYM concept proposed in the paper. In Section 4, we introduce RDFaCE as an implemented WYSIWYM interface. Section 5 explains some use cases of WYSIWYM authoring of structured content based on Schema.org. Finally, Section 6 concludes with an outlook on future work.

2 Related Work

Binding data to UI elements. There are already many approaches and tools which address the binding between data and UI elements for visualizing and exploring semantically structured data. Dadzie and Rowe [4] present the most exhaustive and comprehensive survey to date of these approaches. For example, *Fresnel* [12] 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 CSS.

Parallax, Tabulator, Explorer, Rhizomer, Sgvizler, Fenfire, RDF-Gravity, IsaViz and *i-Disc for Topic Maps* are examples of tools available for visualizing and exploring semantically 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 many approaches and tools which allow binding semantic data to UI elements within unstructured content (cf. our comprehensive literature study [6]). As an example, *Dido* [5] 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*¹ framework. *Loomp* as a prove-of-concept for the *One Click Annotation* [1] strategy is another example in this context. Loomp is a WYSIWYG web editor for enriching content with RDFa annotations. It employs a partial mapping between UI elements and data to hide the complexity of creating semantic data.

WYSIWYG and WYSIWYM. 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) displayed on-screen during editing appears in a form closely corresponding to its appearance when printed or displayed as a finished

¹ <http://simile-widgets.org/exhibit/>

product. WYSIWYG text authoring is meanwhile ubiquitous on the Web and part of most content creation and management workflows (e.g. content management systems, weblogs, wikis). 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 [17,16].

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*² 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. [13] 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 user interface (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 paper targets the novel aspect of integrated visualization, exploration and authoring of unstructured and semantic content. The rationale of our WYSIWYM concept is to enrich the existing WYSIWYG presentational view of the content with UI components revealing the *semantics* embedded in the content and enable the exploration and authoring of semantic content. Instead of separating presentation, content and meaning, our WYSIWYM approach aims to integrate these aspects to facilitate the process of *Semantic Content Authoring*.

Schema.org Tools. **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 *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 [14]. Much of the vocabulary on schema.org is inspired by existing vocabularies such as *FOAF*³, *GoodRelations*⁴, *OpenCyc*⁵, *rNews*⁶, etc.

² <http://www.lyx.org/>

³ <http://www.foaf-project.org>

⁴ <http://www.heppnetz.de/projects/goodrelations>

⁵ <http://www.cyc.com/platform/opencyc>

⁶ <http://dev.iptc.org/rNews>

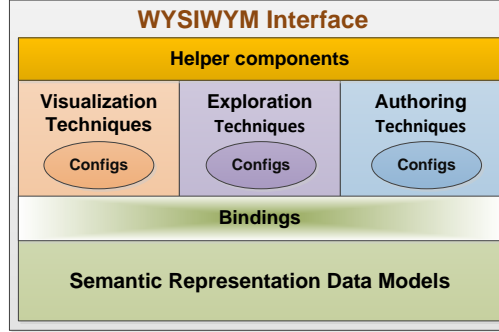


Fig. 2. Schematic view of the WYSIWYM model.

Schema.org annotations can be written using markup attributes in HTML such as *RdFa*⁷ and *Microdata*⁸ or in pure JSON as *JOSN-LD*⁹. Google recently published two Webmaster tools namely *Data Highlighter* and *Structured Data Markup Helper* (SDMH)¹⁰ for content annotation based on Schema.org. SDMH is very similar to *RdFaCE* described in this paper but with the following differences: *RdFaCE* unifies the authoring and publishing of the structured content on the Web. It can be integrated into the existing content management systems to deal with the structured content from the scratch. On the contrary, SDMH as a standalone tool offers a separate additional step to publish metadata on websites. The *RdFaCE* approach is very flexible and can be configured to cover all the schemas on Schema.org as well as other existing vocabularies. SDMH at the moment only supports a limited set of schemas (i.e. Articles, Events, Local Businesses, Movies, Products, Restaurants, Software Applications, TV Episodes) from Schema.org. *RdFaCE* also offers automatic content annotation feature which is not supported by SDMH.

3 WYSIWYM concept

In this section we introduce the fundamental WYSIWYM (What-You-See-Is-What-You-Mean) 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 for software engineers, user interface and domain experts to communicate efficiently and effectively. It provides insights into and an understanding of the requirements as well as corresponding

⁷ <http://www.w3.org/TR/rdfa-syntax/>

⁸ <http://www.w3.org/TR/microdata/>

⁹ <http://json-ld.org/>

¹⁰ <http://www.google.com/webmasters/markup-helper/>

UI solutions for proper design and implementation of semantic content management applications. Secondly, it allows to evaluate and classify existing user interfaces according to the conceptual model in a defined way. This will highlight the gaps in existing applications dealing with semantic content.

Figure 2 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 the 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 is 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 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 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 t ;*
- *H is a set of helper components.*

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*. Each semantic representation model comprises a number of representation elements, such as various types of entities and relationships. For example, elements of a typical graph-based data model are:

- Instances – e.g. **Warfarin** as a drug.
- Classes – e.g. **anticoagulants drug** for Warfarin.
- 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.
- Literal property values – e.g. the halflife for the Amoxicillin.
 - Value – e.g. **61.3** minutes.
 - Language tag – e.g. **en**.
 - Datatype – e.g. **xsd:float**.

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 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. For example, *Framing and Segmentation* (i.e. different borders, overlays and backgrounds), *Text formatting* (i.e. different colors, fonts, text size, etc.) and *Marking* (i.e. icons appended to text or image) are some highlighting techniques. Associating deals with techniques that visualize the relation between some parts of an object. *Line connectors* and *Arrow connectors* are two examples of associating techniques. Detail-view includes techniques which reveal detailed information about a part of an object. *Callouts* as strings of text connected to a part of text giving information about that part are an example of detail-view techniques.

Exploration. To increase the effectiveness of visualizations, users need to be capable to dynamically 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. *Zooming* (i.e. changing the scale of the viewed area in order to see more or less detail), *Faceting* (i.e. accessing information organized according to a faceted classification system), *Bar layouts* (i.e. using vertical nested bars to indicate the hierarchy of entities) and *Expandable callouts* (i.e. interactive and dynamic callouts to explore the properties and relations of an entity) are some examples of available exploration techniques.

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 semantic documents. *Form editing*, *Inline editing*, *Drawing*, *Drag and drop*, *Context menu* and *(Floating) Ribbon editing* are some examples of authoring techniques.

Helper components. In order to facilitate, enhance and customize the WYSIWYM model, we utilize a set of helper components, which implement cross-cutting aspects. A helper component acts as an extension on top of the core functionality of the WYSIWYM model. For example, *Automation* is a helper component which 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. *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.

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.




				Graph-based (e.g. RDF)						
				Instance	Class	Relationships between entities	Literal property values			
							Value	Language tag	Datatype	
				No binding	Partial binding	Full binding				
										
				* If value is available in the text.						
UI categories				UI techniques						
Visualization	Highlighting	Framing and segmentation (borders, overlays, backgrounds)								
		Text formatting (color, font, size etc.)								
		Marking (appended icons)								
	Associating	Line connectors						*		
		Arrow connectors						*		
	Detail view	Callouts (infotips, tooltips, popups)								
Exploration	Zooming									
	Faceting									
	Bar layout									
	Expandable callouts									
Authoring	Form editing									
	Inline edit									
	Drawing									
	Drag and drop									
	Context menu									
	(Floating) Ribbon editing									

Fig. 3. Possible bindings between user interface & RDF semantic representation model.

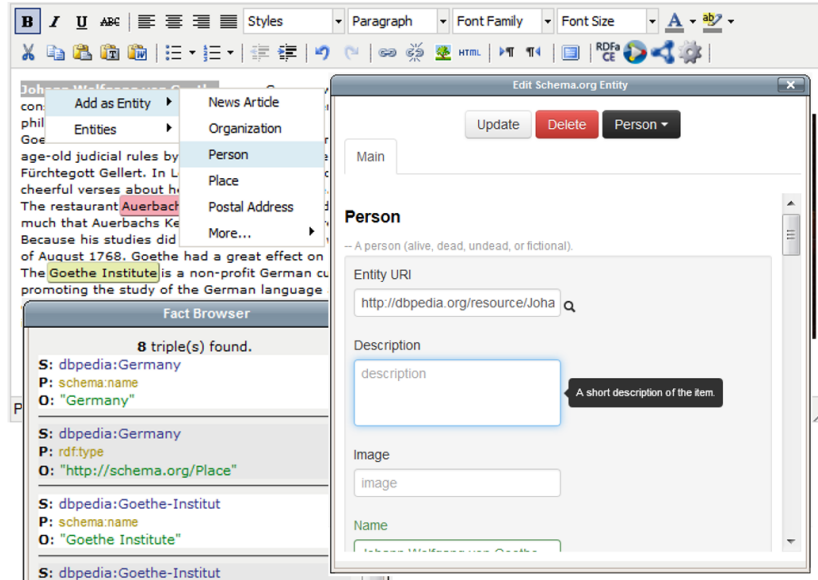


Fig. 4. An screenshot of RDFaCE authoring environment.

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 UI technique ui ($ui \in V \cup X \cup T$) and c is a configuration $c \in C_{ui}$.

Figure 3 gives an example of possible bindings between the user interface (rows) and RDF semantic representation model (columns)¹¹. These possible bindings were obtained from a comprehensive survey of existing tools and an analysis of possible connections between a specific UI element and a semantic model element. The shades of gray in a certain cell indicate the suitability of a certain binding between a particular UI and data model element. Each binding can have a configuration to customize its implementation. For example, special borders or background styles, bar styles or a related icons can be defined for each semantic entity type.

Once a selection of data models and UI elements was made and both are bound to each other 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 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 ($T_I \subset T$);
- H_I is a selection of helper components ($H_I \subset H$);
- b_I is a binding between a particular occurrence of a data model element and a visualization, exploration and/or authoring technique¹².

4 RDFaCE WYSIWYM interface for structured content authoring based on Schema.org

RDFaCE (RDFa Content Editor)[8] is a WYSIWYM interface for semantic authoring of textual content. It is implemented on top of the *TinyMCE*¹³ rich text editor. RDFaCE extends the existing WYSIWYG user interfaces to facilitate semantic authoring within popular CMSs, such as blogs, wikis and discussion forums. The RDFaCE implementation is open-source and available for download together with an explanatory video and online demo at <http://rdface.aks.org>. Since releasing the RDFaCE, the tool has been downloaded over 2000 times and the online demo page has received more than 4000 unique visits. RDFaCE

¹¹ A more complete list of bindings are available at [7]

¹² 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.

¹³ <http://www.tinymce.com>

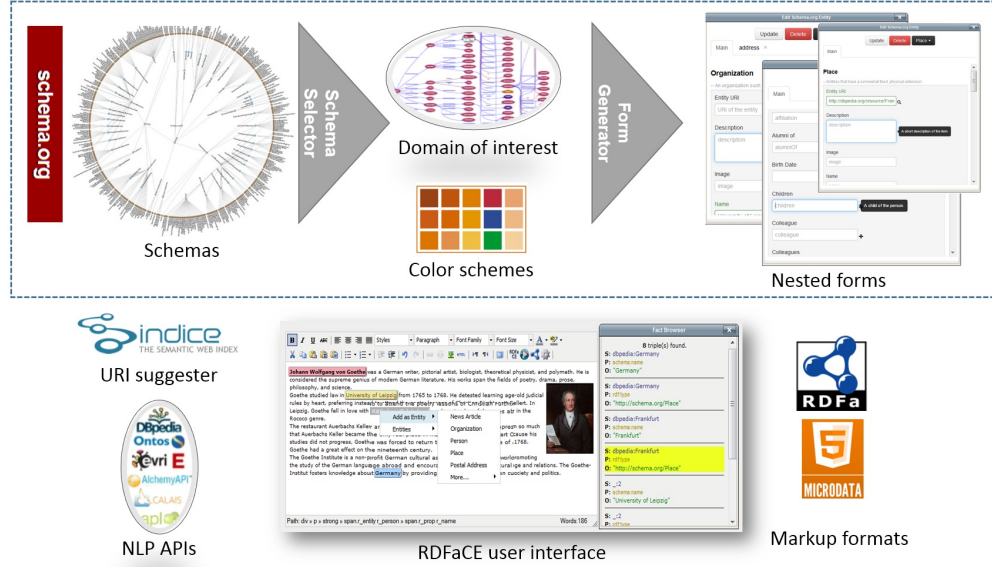


Fig. 5. Configuration steps in RDFaCE.

as a WYSIWYM instantiation supports Microdata and RDFa serializations of RDF data model based on the bindings defined in Figure 3. For visualization, it uses framing using backgrounds to highlight the semantic entities within text content. It also supports callouts in form of dynamic tooltips to reveal the type and property values of semantic entities. For exploration, it supports faceting based on the type of entities, properties and values. For authoring, it employs different methods such as form editing, context menu and ribbon editing(cf. Figure 4). As helper component, RDFaCE utilizes automation and recommendation based on external Web services.

4.1 Configuration

Figure 5 presents the configuration steps in RDFaCE. 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 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.

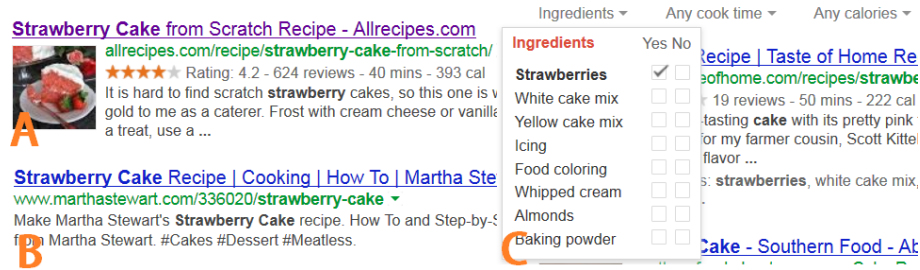


Fig. 6. Search results improved by rich snippets. A: enhanced recipe, B: normal recipe, C: browsing recipes by ingredients, cook time and calories.

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) 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.

4.2 One-click annotation

The annotation steps in RDFaCE are very straightforward. RDFaCE follows the *One Click Annotation* [1] strategy. For automatic annotation, user only needs to press the corresponding RDFaCE button. For manual annotation, user should select some parts of the text and using the context menu, he can choose the appropriate annotations for the selected entities. Context menu is created dynamically based on the user selection. For example if user's selection is inside a NewsArticle schema, user will only see the properties related to the NewsArticle.

RDFaCE includes a fact browser which will reveal the RDF annotation embedded in the text. User can also use tools like Google structured data testing tools¹⁴ to check the validity of embedded annotations.

5 Use Cases: Wordpress and Rich Text Snippets

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

¹⁴ <http://www.google.com/webmasters/tools/richsnippets>

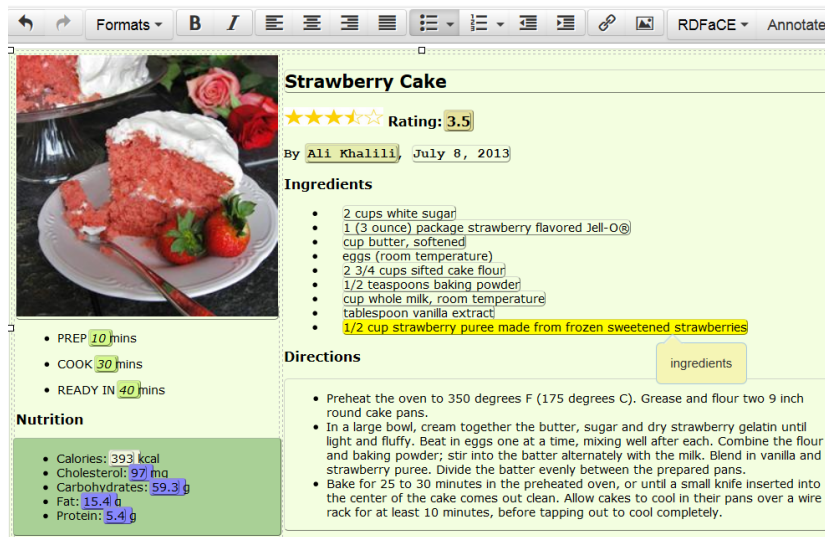


Fig. 7. Using RDFaCE to annotate recipes based on Schema.org recipe schema.

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 6 shows as example of enhanced search results for recipes powered by rich snippets.

RDFaCE comes with a *Wordpress* plugin¹⁵ to enable Weblog authors to create rich snippets based on Schema.org. Wordpress is an open source Weblog tool and publishing platform which is often customized into a Content Management System (CMS). Wordpress is used by 58.5% of websites with known CMS (i.e. 18.8% of all websites¹⁶). Wordpress uses TinyMCE as its content editor. That makes it easy to add the RDFaCE plugin for WYSIWYM authoring of structured content within this CMS. With the integration of RDFaCE into the Wordpress, the availability of semantically annotated content on the Web can be substantially increased.

As an example scenario, Figure 7 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:

¹⁵ <http://wordpress.org/plugins/rdface/>

¹⁶ W3Techs.com, 17 July 2013

- if the selected text already has an HTML tag, metadata will be added as new attributes for the current tag (e.g. Code 1.1 line 2 or 6).
- if the selected text does not have an HTML tag, a new `` or `<DIV>` tag with corresponding attributes will be created (e.g. Code 1.1 line 1 or 3).
- if no text is selected, a new `<META>` tag with corresponding attributes will be created (e.g. Code 1.1 line 17 or 18).

Code 1.1. Example of Microdata annotations generated by RDFaCE

```

1 <div itemscope itemtype="http://schema.org/Recipe">
2   <h2 itemprop="name">Strawberry Cake</h2>
3   <p>By <span itemprop="author">Ali Khalili</span>, July 8, 2013 </p>
4   <h4>Ingredients</h4>
5   <ul>
6     <li itemprop="ingredients">2 cups white sugar</li>
7     <li>...</li>
8   </ul>
9   <p>Preparation time: 10 mins</p>
10  <p>Cooking time: 30 min</p>
11  <p>Ready in 40 min</p>
12  <p>
13    <span itemscope itemtype="http://schema.org/NutritionInformation"
14      itemprop="nutrition">
15      Calories: <span itemprop="calories">393 kcal</span>
16    </span>
17  </p>
18  <meta itemprop="dateCreated" content="2013-07-08">
19  <meta itemprop="prepTime" content="PT15M">
20 </div>

```

6 Conclusion

Bridging the gap between unstructured and semantic 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) concept for direct manipulation of structured content in conventional modalities. RDFaCE employs Schema.org schemas to promote structured content authoring among a wide range of Web users.

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 RDFaCE implementation to support other modalities such as images and multimedia objects. We also envision to perform an extensive usability evaluation of RDFaCE in order to improve its usability as well as functionality.

7 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. One Click Annotation. volume 699 of *CEUR Workshop Proceedings*, February 2010.
2. A. Ankolekar, M. Krötzsch, T. Tran, and D. Vrandečić. The two cultures: mashing up web 2.0 and the semantic web. In *WWW 2007*, pages 825–834.
3. G. Burel, A. E. Cano, and V. Lanfranchi. Ozone browser: Augmenting the web with semantic overlays. volume 449 of *CEUR WS Proceedings*, June 2009.
4. A.-S. Dadzie and M. Rowe. Approaches to visualising linked data: A survey. *Semantic Web*, 2(2):89–124, 2011.
5. D. R. Karger, S. Ostler, and R. Lee. The web page as a wysiwyg end-user customizable database-backed information management application. In *UIST 2009*, pages 257–260. ACM, 2009.
6. A. Khalili and S. Auer. User interfaces for semantic authoring of textual content: A systematic literature review. 2013. to be appear in the journal of Web Semantics.
7. A. Khalili and S. Auer. Wysiwym – integrated visualization, exploration and authoring of un-structured and semantic content. 2013. submitted to the Semantic Web journal.
8. A. Khalili, S. Auer, and D. Hladky. The rdfa content editor. In *IEEE COMPSAC 2012*.
9. V. Lopez, V. Uren, M. Sabou, and E. Motta. Is question answering fit for the semantic web? a survey. *Semantic Web ? Interoperability, Usability, Applicability*, 2(2):125–155, September 2011.
10. P. Mika and T. Potter. Metadata statistics for a large web corpus. In C. Bizer, T. Heath, T. Berners-Lee, and M. Hausenblas, editors, *LDOW*, volume 937 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2012.
11. W. Muller, I. Rojas, A. Eberhart, P. Haase, and M. Schmidt. A-r-e: The author-review-execute environment. *Procedia Computer Science*, 4:627 – 636, 2011. ICCS 2011.
12. E. Pietriga, C. Bizer, D. R. Karger, and R. Lee. Fresnel: A browser-independent presentation vocabulary for rdf. In *ISWC*, LNCS, pages 158–171. Springer, 2006.
13. R. Power, R. Power, D. Scott, D. Scott, R. Evans, and R. Evans. What you see is what you meant: direct knowledge editing with natural language feedback, 1998.
14. J. Ronallo. HTML5 Microdata and Schema.org. *The Code4Lib Journal*, (16), Feb. 2012.
15. M. Sah, W. Hall, N. M. Gibbins, and D. C. D. Roure. Sempert - a personalized semantic portal. In *18th ACM Conf. on Hypertext and Hypermedia*, pages 31–32, 2007.
16. C. Sauer. What you see is wiki – questioning WYSIWYG in the Internet age. In *Proceedings of Wikimania 2006*, 2006.
17. J. Spiesser and L. Kitchen. Optimization of html automatically generated by wysiwyg programs. In *WWW 2004*, pages 355–364.
18. D. Tunkelang. *Faceted Search (Synthesis Lectures on Information Concepts, Retrieval, and Services)*. Morgan and Claypool Publishers, June 2009.