WebAnnotator, an Annotation Tool for Web Pages

Xavier Tannier

LIMSI-CNRS
Univ. Paris-Sud, 91403 Orsay, France
xavier.tannier@limsi.fr

Abstract

This article presents WebAnnotator, a new tool for annotating Web pages. WebAnnotator is implemented as a Firefox extension, allowing annotation of both offline and inline pages. The HTML rendering fully preserved and all annotations consist in new HTML spans with specific styles. WebAnnotator provides an easy and general-purpose framework and is made available under CeCILL free license (close to GNU GPL), so that use and further contributions are made simple. All parts of an HTML document can be annotated: text, images, videos, tables, menus, etc. The annotations are created by simply selecting a part of the document and clicking on the relevant type and subtypes. The annotated elements are then highlighted in a specific color. Annotation schemas can be defined by the user by creating a simple DTD representing the types and subtypes that must be highlighted. Finally, annotations can be saved (HTML with highlighted parts of documents) or exported (in a machine-readable format).

Keywords: Annotation, Web, Tool

1. Introduction

Annotating documents is an occupation that many researchers are familiar with in the field of information technologies, and especially in natural language processing. It is a tedious but necessary task for resource construction, corpus studies, machine learning or system evaluation. Consequently, a huge number of annotation tools have been developed for various applications. All of these tools have the same purpose: making annotation process faster and easier.

However we think that a convenient and general tool for annotating Web pages is still missing. What we expect from such a tool is mainly that it allows to keep HTML rendering for annotators, and to provide an appropriate saving format, so that it preserves HTML (often not well-formed) structure and augments it with machine-readable markup for new annotations.

In this paper, we present WebAnnotator, a Firefox extension providing an easy and general-purpose framework for annotating Web pages. We first describe what kinds of annotation needs exist and what kinds of tools have been developed to fill them (Section 2.). We then present the functionalities of WebAnnotator and show screenshots of the tool (Section 3.).

2. Annotating Web Pages

Needs for manually annotating Web pages exist in many applications. The structured and multimedia information contained in HTML documents interest a lot of researchers and workers in information technology. Also, information retrieval, information extraction and natural processing processing are obviously very concerned by unstructured information.

Among lots of applications where annotation of Web pages is useful or necessary, one can cite:

- Text tagging, *i.e.* any exercise of information extraction involving for example named entity tagging, pattern learning, etc. Annotation can then be used for evaluation and/or for machine learning.
- Image tagging, for any kind of applications (*e.g.* image retrieval, recognition, etc.).
- Web page cleaning, *i.e.* detection of ads, menus, metadata, blog posts, tables...
- etc.

For all these different applications, tools are built. These tools are often application-specific (see for example (LAW, 2009; LAW, 2011)). Most of them are not even publicly distributed. Among existing, publicly available tools, we can cite GATE (Cunningham et al., 2011), Callisto (Day et al., 2004), Glozz (Widlöcher and Mathet, 2009) MAE/MAI (Stubbs, 2011), Knowtator (Ogren, 2006), Boemie (Paliouras et al., 2011). Among them, only the latest is able to annotate HTML pages with appropriate rendering, but it is not open-source and runs only on Windows-operated machines.

3. WebAnnotator

WebAnnotator is open-source and freely available at the following URL:
`https://addons.mozilla.org/en-US/firefox/addon/webannotator/`[1]

WebAnnotator has been designed in order to deal with four needs that are specific to Web pages:

1. Annotate online pages, without having to store them beforehand.
2. Maintain the visual rendering of HTML pages, so that the annotated document is exactly the same as document seen by regular Web users.
3. Allow annotation of any element in the page, including text, images, menus, etc. in order to stay as general as possible.

[1] French and US-English locales are currently available.

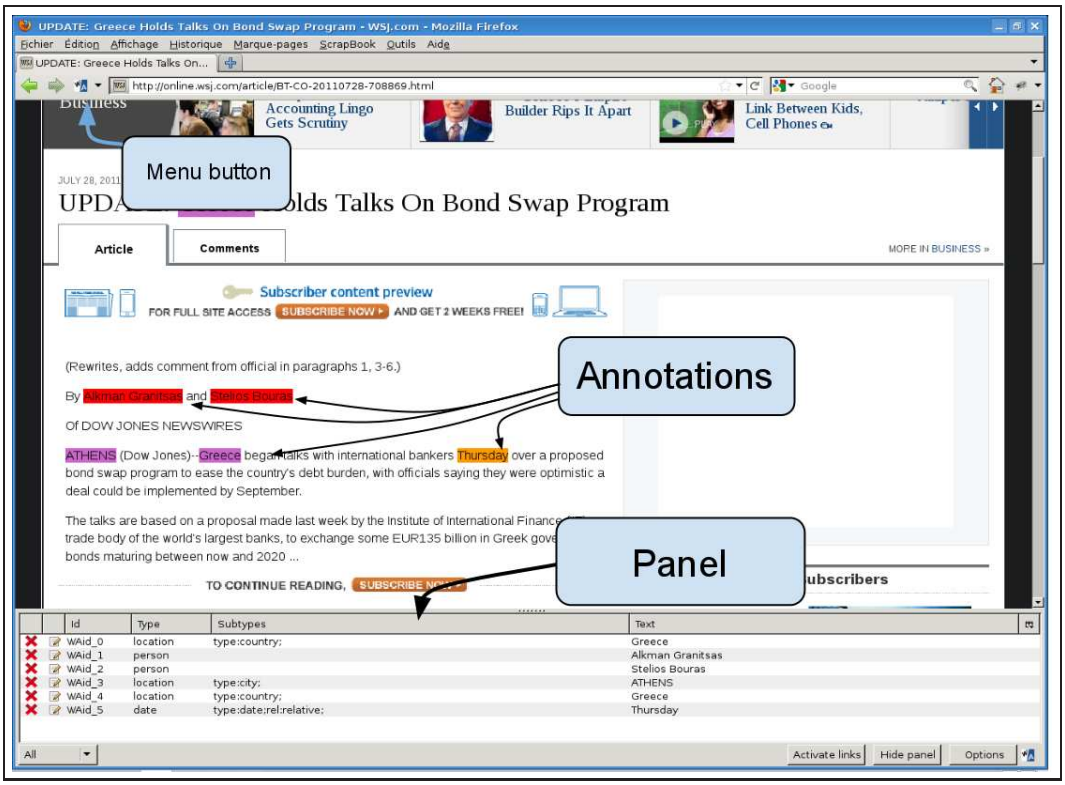

Figure 1: General view of WebAnnotator annotation session.

4. Allow a both human- and machine-readable saving format. This is particularly difficult since HTML pages have already markup structure, and this structure is very often ill-formed. Also, annotations are likely to spread over several HTML elements, but also to overlap some of these elements. For this reason, a "classical" XML markup that would bound the annotated element is not possible.

In order to fulfill the first two needs in a simple manner, we choose to build a Firefox add-on. Firefox is a very commonly used Web browser, and Firefox users often use add-ons to extend their browser functionalities. Furthermore, no from-scratch tool will ever be as good as a major Web browser for rendering Web pages. Building a Firefox add-on with Javascript and XUL is made quite easy by many online tutorials and forums[2].

A consequence of this choice is that everything that can be selected in Firefox can be annotated. This includes text, lists, tables, images, videos, ads, menus. Only non-contiguous parts of a document cannot be grouped into a single annotation tag.

As for the fourth need (saving format), this will be dealt with in Section 3.3.

3.1. Creating Annotation Schema

The procedure for creating a personal annotation schema is inspired from Callisto (Day et al., 2004). The user simply creates a DTD, the common format used to represent the structure of XML files, in order to describe the elements she wants to annotate, as well as any number of attributes she wants to associate to these elements.

Contrary to Callisto, the tool supports the `required` or `implied` attribute features, as well as list or free-text attribute values.

Figures 2 and 3 show two examples of NLP-related user-defined DTDs that can be used in WebAnnotator: chunking and named entity tagging.

Once the DTD file is parsed, the user specifies a task name and description, as well as her personal choices of highlighting colors if she wants, and the tool is ready to start an annotation session.

3.2. Annotating Pages

The general view of an annotation session under WebAnnotator is illustrated in Figure 1. Options are accessible by a menu button as well as textual menu items. Annotations are shown directly in the Web page, while a summary of already annotated parts of document is available in a panel at the bottom of the browser.

When selecting a part of the page, a small rectangle will pop-up near the selection, waiting for the user to select the type of tag she wants to associate to the element (see Figure 4). If this element contains specific attributes, the user can now choose values, as described by the loaded DTD (see Figure 5).

3.3. Saving annotations

Saving annotations made on a Web page is a challenge. As an HTML page already has a given structure, the saving format should be compatible with this existing structure. For example, a simple beginning and ending tag bounding each

[2] `https://addons.mozilla.org/developers/`

```
<!-- Chunking                                                -->
<!--   Three high-level annotations types : NP, VP, PP -->
<!ELEMENT NP (#PCDATA)>
<!ELEMENT VP (#PCDATA)>
<!ELEMENT PP (#PCDATA)>
```

Figure 2: Example of a simple DTD schema for basic chunking. Allowed types are `NP`, `VP` and `PP`, no subtypes.

```
<!-- Four high-level annotations types : person, org, location, date -->
<!ELEMENT person (#PCDATA)>
<!ELEMENT org (#PCDATA)>
<!ELEMENT location (#PCDATA)>
  <!-- Attributes for locations, no default -->
  <!ATTLIST location type (river|mountain|city|country) #IMPLIED>
<!ELEMENT date (#PCDATA)>
  <!-- Attributes for location              -->
  <!ATTLIST date type (date|time|duration) #REQUIRED
                 rel (absolute|relative) absolute
                 value CDATA #IMPLIED>
```

Figure 3: Example a DTD schema for named entity tagging. Allowed types are `person`, `org`, `location` and `date`. Type `location` has a optional attribute `location` that can take the values *river*, *mountain*, *city* or *country*. Type `date` has two required types: `type` and `rel`. This latest has a default value *absolute*. The optional subtype `value` is a free-text attribute.

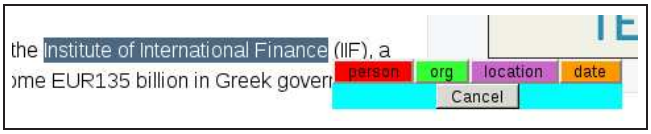

Figure 4: Creating a new annotation.

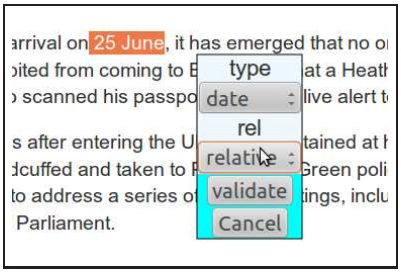

Figure 5: Choosing attribute values for an annotation.

annotation is not possible, since it would potentially overlap HTML elements, which is not acceptable. It is even also possible that two annotations overlap. On the other hand, separating totally annotations from the text, for example with AIFF format, as proposed by Callisto, GATE or Glozz for example, is not desirable in our opinion. Such a stand-off markup format is useful for separating totally markup from text, so that text remains clear and analyzable. However, in our case, annotation markup is just one more markup in the document. Furthermore, annotations in Web pages can be strongly related to rendering and context, and keeping them all together makes sense. For these reasons, we think that separated annotations would add unnecessary complexity to the further automatic processing.
Also, saving the annotations should not lead to no longer being able to read the document on a Web browser, with

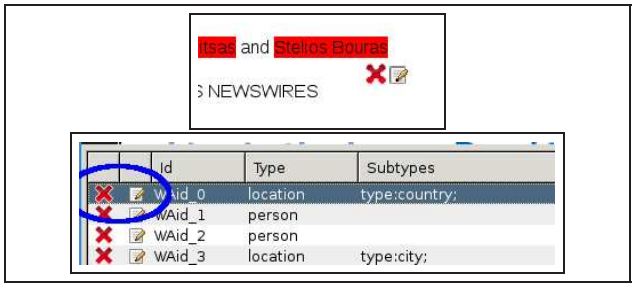

Figure 6: Two ways of modifying annotations: near the highlighted part of document (up) or from the panel (bottom).

colored annotations and HTML rendering.
As a result, two output formats are available in WebAnnotator, namely save format and export format:

- **Save format** leads to pretty much the same file as if we had chosen "Save as" in the regular Firefox File menu, except that a small clean-up is achieved in the WebAnnotator-specific tags in the HTML file. This format allows to keep the visual information of annotated parts of documents. With this format, it is also possible to come back to annotating the document later, which is not the case with the export format. However, save format makes difficult the machine reading and automatic use of the document. Indeed, a same annotation can result in several HTML span tags, in order to avoid overlaping of HTML elements. However, all spans coming from a single annotation have the same `WA-id` attribute, which make the automatic merging easier (see Figure 7).
- **Export format** replaces HTML span tags by two empty elements, `WA_Start` and `WA_End`, sharing

HTML rendering of annotation on *"by importing"*:

Annotation schemas can be specified to WebAnnotator by **importing a DTD**

Original HTML:

```
Annotation schemas can be specified to
WebAnnotator by <b>importing a DTD</b>
```

Saved as:

```
Annotation schemas can be specified
to WebAnnotator <span wa-id="1"
wa-type="PP" wa-subtypes=""
class="WebAnnotator_PP">by</span> <b><span
wa-id="1" wa-type="PP" wa-subtypes=""
class="WebAnnotator_PP">importing</span> a
DTD</b>
```

Figure 7: Save format. When the annotation overlaps HTML elements, several `span` tags are created (with the same id) so that the element seems continuously highlighted to the user.

HTML rendering of annotation on *"by importing"*:

Annotation schemas can be specified to WebAnnotator by **importing a DTD**

Exported as:

```
Annotation schemas can be specified
to WebAnnotator <WA_Start WA-id="1"
wa-type="PP" wa-subtypes="" />by
<b>importing<WA_End WA-id="1" /> a DTD</b>
```

Figure 8: Export format. A beginning and an ending empty elements are created.

the same `WA-id` attribute and containing all types and subtypes information. This makes automatic processing easier, but we can no longer visualize or re-annotate exported WebAnnotator files.

Exporting pages would result in valid XML if all Web pages were well-formed XHTML documents. It is of course far from being the case, but at least these additional tags do not insert any new ill-formed elements. An example is given at Figure 8.

3.4. What WebAnnotator does not do

- The main "missing" functionality of WebAnnotator is that it does not allow annotation of relations between tagged elements, as does Glozz for example. It would be a nice extension but seems very difficult to add as a Firefox add-on with a nice user interface. Indeed, representing relations by lines, curves or arrows between two elements is not possible in classic HTML. The only simple way to store relations would be to represent them as a table that would update when clicking on elements.

- Not all valid DTDs are accepted. In particular, nested elements are not recognized. It is the case of most other annotation tools.

- Automatic pre-annotation, or use of thesaurus or ontologies, as in Protégé or Knowtator, is not possible. This would constitute a nice additional functionality and is in the top list of further extensions. However, the annotation format is very simple, and it is quite easy to save the Web page, then preprocess it by adding the relevant spans, and finally open the enriched file in WebAnnotator.

- Finally, a few known minor bugs are listed on the add-on home page. The extension will be maintained as far as possible and users should not hesitate to contact the author if they find any annoying bug or wish to participate to the add-on evolution.

4. Conclusion

We presented WebAnnotator, a new annotation tool for Web pages, based on a Firefox extension. It allows user-defined annotation schemas, various options and keeps the HTML rendering during annotations.
This tool is free, open-source and not application-specific. It is made available under CeCILL free license (close to GNU). It can be modified and extended by any developer.

5. References

Hamish Cunningham, Diana Maynard, Kalina Bontcheva, Valentin Tablan, Niraj Aswani, Ian Roberts, Genevieve Gorrell, Adam Funk, Angus Roberts, Danica Damljanovic, Thomas Heitz, Mark A. Greenwood, Horacio Saggion, Johann Petrak, Yaoyong Li, and Wim Peters. 2011. *Text Processing with GATE (Version 6)*. University of Sheffield Department of Computer Science.

D. Day, R. Kozierok, C. McHenry, and L. D. Riek. 2004. Callisto: A Configurable Annotation Workbench. In European Language Resources Association (ELRA), editor, *Proceedings of the Fourth International Language Resources and Evaluation (LREC'04)*, Lisbon, Portugal, May.

2009. *Proceedings of the 3rd Linguistic Annotation Workshop*, Suntec, Singapore, aug.

2011. *Proceedings of the 5th Linguistic Annotation Workshop*, Portland, USA, jun.

Philip V. Ogren. 2006. Knowtator: a protégé plug-in for annotated corpus construction. In *Proceedings of the 2006 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, pages 273–275, Morristown, NJ, USA. Association for Computational Linguistics.

Georgios Paliouras, Constantine D. Spyropoulos, and George Tsatsaronis, editors. 2011. *Knowledge-Driven Multimedia Information Extraction and Ontology Evolution*. Springer.

Amber Stubbs. 2011. MAE and MAI: Lightweight Annotation and Adjudication Tools. In LAW2011 (LAW, 2011).

Antoine Widlöcher and Yann Mathet. 2009. La plate-forme Glozz : environnement d'annotation et d'exploration de corpus. In *Actes de la Conférence Traitement Automatique des Langues Naturelles (TALN 2009, poster)*, Senlis, France, jun.