Pundit 2.0

Christian Morbidoni\textsuperscript{a}, Alessio Piccioli\textsuperscript{b}
\begin{flushleft}
\textsuperscript{a}Università Politecnica delle Marche, Department of Information Engineering, Ancona, Italy \\
\textsuperscript{b}Net7 S.R.L, Pisa, Italy
\end{flushleft}

Abstract. This paper presents and demonstrates Pundit, a semantic annotation tool that combines powerful annotation functionalities - covering comments, tagging, entities markup and linking, composition of rich semantic statements - with private/public annotation sharing and retrieval via REST API. The adoption of a flexible data model, based on RDF and an extension of the Open Annotation model, allows, on one hand, to make annotations easily available as graphs, and, on the other, to support high semantic expressivity. With Pundit, structured semantic knowledge can be created by semantic interlinking among items in a web page, such as text or images, and LOD resources as well as custom annotation vocabularies. Pundit is designed as a configurable annotation service. It addresses online annotation communities by allowing customization of vocabularies and user interface, deployment of domain specific annotation environments, and easy connection of such environments with existing web sites or applications. We demonstrate Pundit with an example use case and supported by an online demo. Finally, we briefly show how results of semantic annotations can be reused in additional components.

Keywords: Semantic Annotation, Resource Description Framework, Semantic Web

1. Introduction

Bookmarking, tagging and annotating content from the web are common activities since the early days of the Web. Web users needs to remember and organize links but also to take notes and add meaning and information. Today the Web is the biggest and most widely used information source in many contexts, including research, scholarship and (data) journalism. Annotating content to a fine granular level, by adding metadata and structured knowledge, is becoming more and more a real need.

Humanities scholars, for example, need to collect papers or digital manuscripts on the web, but also to add comments and perhaps highlight entities mentioned in a text, classify them according to some domain thesaurus, or even establish relations among concepts and texts. In fact, specialized communities working with manuscripts would benefit, for example, from a solid way to annotate precise regions of manuscripts and link them to transcriptions.

Annotation requires effort and is always associated to some expected benefit or added value. For single scholars or users these might range from the curation of a document collection, the production of report or papers, to produce categorizations or time/spatial visualizations.

In other cases annotations by a controlled group of scholars are used to produce high quality metadata. This might have the objective to improve exploration capabilities in digital libraries.

In both cases there is a need for annotation tools that goes beyond simple commenting and tagging facilities. The information is often hidden in texts or media and needs to be made explicit, appropriately captured by tools and perhaps linked to specific vocabularies or ontologies.

This requirement for uniform, machine readable representation for annotations has been well received by the Semantic Web and Linked Data communities, and most specifically by initiatives as Open Annotation, the Europena Data Model, and others.

The Open Annotation data model \cite{12} notably advanced the state of the art in terms of structured representation and interoperability among annotation systems. On the other hand, the requirements in terms of schemas and ontologies are in fact domain dependent. This is the case for annotation vocabularies, as different communities needs to refer to very specific concepts or the re-use existing schemas or thesauri. Tools should take this diversity into account and be able to adapt to specific data-driven requirements.

In this paper we present and demonstrate Pundit 2.0, a semantic annotation system that assist users in
creating semantically structured data by annotating web documents. Different levels of complexity are supported and annotations can range from semantic tags to free composition of relations among annotated resources.

Pundit aims at fostering the re-use of knowledge created in communities, by making annotations available via REST APIs and in the form of RDF graphs, making it easier to create visualizations and reuse data in different contexts.

Pundit implements a client-server architecture. It provides a Javascript client component that realizes the user annotation environment, and a server side component where annotations are stored and made accessible. Pundit 2.0 is the evolution of previous systems and versions (see section 2.3). The tool went through a major revision in 2014. The Pundit architecture has been reengineered to provide the annotation environment as-a-service via REST API. Furthermore, Pundit 2.0 features a completely new user interface, based on AngularJS\(^1\), as well as notable improvements in annotation functionalities.

In this paper we first overview the system, in section 2, and its architecture, section 2.4. In section 3 we introduce the annotation server and the implemented annotation data model, while in section 4 the Pundit 2.0 annotation environment is demonstrated with the support of sample use cases and a live online demo.

While Pundit per se is an annotation-authoring tool, the added value of producing semantic annotations is that of favor re-use of their semantics. In section 5 we discuss how Pundit can be integrated with other systems and the annotations exploited, e.g., to collaboratively populate faceted search or other kinds of visualizations and exploration tools.

2. Pundit overview

Pundit is an annotation system that provides users with a graphical environment to create, collect and share semantic annotations on web documents.

The system provides a uniform user interface for creating different annotations with different levels of expressivity. Annotations are represented as set of types relations among different types of items. Items can be elements of a web page (e.g. text, images), entities on the LOD cloud or entries in custom domain vocabularies.

Annotations are collected in notebooks and can be made public or private. The Pundit server exposes the content notebooks as a semantic graphs and accessible via REST APIs.

Pundit can complement existing systems by providing simple ways to integrate the annotation environment in digital libraries or web sites.

The system is designed as a configurable annotation service. Custom vocabularies and annotation functionalities can be easily deployed into personalized annotations environments.

Figure 1 shows Pundit in action. Annotations are shown in the sidebar and refer to objects in the page.

The main facts about Pundit are:
- It implements a data model based on named graphs and OA, where annotations are set of triples and preserve provenance and context. The representation is flexible with respect to aggregating and mixing annotations, and at the same time Pundit can complement existing systems by providing simple ways to integrate the annotation environment in digital libraries or web sites.

\(^1\)https://angularjs.org/
time naturally captures different semantics and expressivity in annotations.

- **Granularity of annotations.** Annotations can be made on text and image fragments, by drawing polygons. Fragments of web resources are treated as resources themselves and can be put in relation, e.g. by linking a portion of a manuscript page to its transcription or a text to an other in a different web page.

- **Expressivity of annotations.** Composing semantic annotations as a matter of fact requires more effort to users if compared to simple tagging. Pundit implements a uniform model to capture annotations with different complexities. The simple templating mechanism, illustrated in Section 4.3.1, allows to easily extending Pundit with custom annotation facilities to address specific needs and speed up repetitive tasks. The same, uniform user interface (the triple composer) supports advanced users in free composition of semantic relations.

- **Custom deployments and configurations.** Pundit is designed to be itself a framework to build custom annotation tools. Many aspects of the annotation environment can be configured via a JSON file, including tuning of annotation modules, editing of custom annotation vocabularies, and connection to external search or information retrieval services. JSON configurations are loaded at run-time so they can be easily hosted and maintained by third parties.

- **Annotation as-a-service.** Pundit is provided as-a-service, via the feed.thepund.it API. This simple solution enables easy basic integration with mostly any web application. Both content to be annotated and configurations are passed as parameter to the Feed API. In practice, the system supports many real world use cases where some administrator or leader (power user) sets the annotation environment for his users or collaborators. With Pundit this can be done by editing and putting online the configuration file and by adding a hyperlink to the Feed API from the web content that one would like to make annotatable.

- **Bookmarklets.** In cases where the web pages to be annotated are not under the control of the user (a.g. wiki pages or blogs), Pundit can be packed and distributed as a bookmarklet. This is a simple way of loading javascript applications into web browsers. A Pundit bookmarklet loads Pundit on top of any web page.

### 2.1. Annotations in Pundit

We consider the definition of annotation given in Open Annotation Model, which can be summarized as “the act of creating associations between distinct pieces of information”.

In Pundit such pieces of information can be texts, documents, entries in a vocabulary or catalogue (including LOD resources), single annotated sentences or paragraphs, images, etc. Annotations are therefore associations among such different kind of resource in the form of RDF triples. Multiple annotations possibly created collaboratively form a semantic overly on top of annotated web documents, as illustrated in Figure 2. Once such overlay is created, links can be exploited in Pundit to browse among resources, and by other applications to provide other means for exploring the content.

![Figure 2](image_url)

**Figure 2.** A possible semantic overly created by semantic annotations. A text (subject) is marked has been written by (predicate) Immanuel Kant (object); a portion of an image (e.g. a stamp) depicts Kant, or a manuscript is transcribed in a given Web page.

### 2.2. Sharing annotations

Pundit is a collaborative annotation tool where annotations are stored in personal notebooks. Users can have different notebooks that can be made public, thus making annotations visible to other users. While Pundit supports basic management of personal notebooks and sharing, it delegates search and exploration of public notebooks to additional components. Ask², developed in the DM2E project, is an example of such an application where users can discovery, aggregate and analyze public notebooks and enclosed annotations.

---

² Open Annotation - http://www.openannotation.org/spec/core/
³ Ask online demo: http://demo-cloud.ask.thepund.it/
In general the possibility of re-using annotation’s semantics makes it possible to get benefits and concrete results out of the annotation process. While, it is true that semantic annotations inevitably require more user effort and commitment if compared to simpler annotation tasks, we believe that making it easier to build visualizations and informative explorations of content can in fact bring such benefits and possibly motivate users in annotating. We discuss additional concrete examples in section 5.

2.3. History and status of Pundit

The first version of the software was developed in 2009 by the Semedia group in Ancona Italy and delivered as result of the Semlib EU project, as an evolution of a previous desktop annotation tool and knowledge management tools, Swikynotes and DBin [14]. After that, Net7 S.R.L. further developed the tool in collaboration with Semedia and applied it in different research and commercial projects.

New requirements were recently gathered in the context of the DM2E\(^4\) project and subsequent improvements lead to a major revision of Pundit. In collaboration with the Excellency Cluster and Berlin Humboldt University, a new user interface has been designed and implemented in Pundit 2.0.\(^5\)

Pundit is now an open-source project\(^6\) and is provided under a double license (open and commercial). Every major revision of the Pundit server is released under open license after an embargo period\(^6\). Demos and other information can be found on the project website at http://thepund.it. The demo used in this paper can be found at http://purl.org/pundit/demo/all.

2.4. System architecture and functioning

The simple and modular architecture of Pundit is composed by three distinct modules, as shown in Figure 3:

a) A client side annotation library that can be configured and deployed as a bookmarklet or, more traditionally, included in web sites as a javascript library.

b) A server side component built in Java and implementing a set of REST APIs on top of a triplestore.

c) A service API, called Feed, that instantiates a given instance of the pundit client on one specific target page.

Figure 3. The different components of the system and their interaction. The graph also shows how user interact with the components: by configuring, deploying and distributing annotation clients, and by authoring and exploiting annotations.

Two kind of user can be distinguished with respect to how they interact with the component and what is their goal. End users are scholars, professional or occasional web surfers that creates annotations on web content. They can be “independent” users, e.g. using the public stable release of Pundit\(^7\), or they can belong to institutions or communities and more probably using specialized clients. Such specialized clients can be created by power users configuring and deploying Pundit. They maintain online configuration files and/or domain vocabularies, and make the customized annotation tool available to their users. They do so via Feed APIs, by deploying bookmarklets, or by including a javascript library in their web sites.

3. Pundit Server

3.1. Data model

The simple schema in Figure 4 shows the RDF representation of a sample annotation. While no peculiarity exist with respect to OA for what concerns metadata and context of annotations, Pundit implements a particular methodology to represent the oa:Body of an annotation. As in Pundit annotations are sets of triples, rather than text or links, we use a named graph as body of annotations. We call this the body-graph of the annotation.

\(^4\) http://dm2e.eu/

\(^5\) https://github.com/net7/pundit2

\(^6\) https://github.com/net7/pundit-server

\(^7\) http://thepund.it/try-pundit/
Figure 4. A schema of the RDF data model adopted by the Pundit server.

This methodology is discussed in more detail in [11], and has been recently added to the OA extension specification\(^8\). Its main benefits are:

- It naturally enables clients to freely use existing or custom annotation ontologies and vocabularies, as annotation server is agnostic to schema used in a body-graph.
- It provides a simple and flexible way to mix annotations from different users and/or to collect them, without loosing track of the provenance of each piece of data created by users.
- It allows notebooks to be exported and explored as regular RDF graphs, created collaboratively by users.

3.1.1. The Items Graph

When displaying annotations body-graphs, an application needs appropriate context and at least basic metadata. By linking a text to a DBpedia entity, for example, a single triple is created in the body-graph, like the one in Example 1. However, for displaying the annotation a set of additional triples are desirable, e.g., a label, a description, types, etc.

As distinct users can annotate the same entities, such additional information could be centralized to avoid redundancy. However, in doing so the visualization of single user annotations may be affected when metadata changes. For example consider the case where different communities or even single users refer to the very same resource in their annotations, but they disagree on the label, description or other details.

Pundit takes a conservative approach and associates an additional items graph to each annotation.

\[^8\] http://www.openannotation.org/spec/extension/#NamedGraph

This allows to clearly separating those triples that the user actually created (triples in the body-graph) from those that an application needs to display the resources mentioned in the body-graph (items-graph). In doing so a potentially big number of triples are duplicated across annotations, and thus across named graphs in the triplestore.

Example 1

<table>
<thead>
<tr>
<th>Body-graph</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Items-graph</th>
</tr>
</thead>
<tbody>
<tr>
<td>dbpedia:Horror rdf:type dbpedia-owl:Genre.</td>
</tr>
<tr>
<td>dbpedia:Horror rdfs:label &quot;Horror Movies&quot; @en.</td>
</tr>
<tr>
<td>dbpedia:Horror rdfs:description &quot;...&quot; @en.</td>
</tr>
<tr>
<td>voc:identifies rdf:type rdf:Property.</td>
</tr>
<tr>
<td>voc:identifies rdf:type &quot;identifies&quot; @en.</td>
</tr>
<tr>
<td>voc:identifies rdfs:description &quot;...&quot; @en.</td>
</tr>
<tr>
<td>:fragment-text-555 rdf:type :TextFragment</td>
</tr>
<tr>
<td>:fragment-text-555 :hasSelector :selector-text-345-351</td>
</tr>
<tr>
<td>:fragment-text-555 :hasSource <a href="http://example.org/texts/example.htm">http://example.org/texts/example.htm</a></td>
</tr>
</tbody>
</table>

Note that Pundit does not support centralization of items metadata. This, along with other requirements and possible improvements are currently being evaluated for the next development phase.

3.2. Notebooks

Like many other web annotation systems, Pundit supports collections of annotations, called notebooks. Access and sharing of annotations are managed at notebook level; each of them can be set to public or private. Switching among notebooks, users can decide at any time to what notebook annotations will be wrote to.

Notebooks are collections of annotations, and annotations contain sets of triples. We call the knowledge graph of a notebook the graph composed by all the triples in the body-graphs of its annotations. Such graph can be retrieved via server API a set of triples. Additionally, the Pundit server provides a SPARQL endpoint to query each single notebook as a graph. The address of each endpoint is published at:

\{root\}/open/notebooks/\{notebook-id\}/sparql

Where root is the URL or a Pundit server and notebook-id indicates to id of a notebook (and indirectly its URL). The notebook has to be public and the function is provided only via the open API.
3.3. APIs and documentation

Annotations and notebooks can be consumed in the form of RDF triples via the server API.

While public notebooks are exposed via the open API, operations like creating, editing annotations or sharing notebooks are available over authenticated API. The documentation is available at :

http://purl.org/pundit/doc/pundit-server-api

The Talis RDF/JSON format\(^9\) is used to make RDF easily consumable by Javascript applications. It is to be noticed that, while such a format is very easy to handle as triples are simply represented as hierarchies of object, predicate, objects of triples, the Semantic Web community today is more oriented toward LOD-specific standards like JSON-LD\(^10\).

4. The Pundit Client: Online Demo and Main functional components

In this section we present the main functionalities of the annotation environment provided by Pundit via its client component. We do this driven by simple demonstrative use cases and supported by a live demo available at the following address:

http://purl.org/pundit/demo/all

As in Pundit all annotation have an author, if you want to see and create annotation you need to login by using an existing OpenId account, the most common being a Google account, to do so click on the Login link on the top left in the Pundit toolbar.

4.1. Anno-matic with DataTXT

Let us first suppose you are a movie fan and you visit some blog where you find an interesting article\(^11\).

As you read the article you find mentions to entities such as movies, actors, movie genres and so on. Retrieving additional information would help both to understand the text and to create meaningful annotations.

A first approach to semantic annotation is often that of identifying mentions of relevant entities, as happens in the Burckhardt-Source project, where Pundit is being used to mark persons, places and artworks in a corpus of correspondences\(^12\).

![Figure 5. Anno-matic. Entities recognized in the text can be discarded or saved as annotations in the current notebook.](https://dvcs.w3.org/hg/rdf/raw-file/default/rdf/json/index.html)

In the last years, fast and accurate tools have been made available to perform Named Entity Recognition (NER) and linking, as, for example, TextRazor\(^13\), AlchemyAPI\(^14\) and Dandelion/DataTXT\(^15\). In Pundit we created a built-in support for DataTXT, developed by SpazioDati, Trento, and based on previous research work on entity extraction from Wikipedia [18]. By selecting the suggestion mode in the Pundit sidebar, and selecting a paragraph of text, you can get automatic suggestions of entities mentioned. Mentions are highlighted in the text and a simple UI is provided to check, discard or confirm annotations, as illustrated in

**Figure 5.** Once confirmed, annotations are saved to your current personal notebook and are immediately visible in the sidebar.

4.2. Comments and tags

Other useful and common things that you might want to do is taking some note and perhaps tagging some passage in the movie review.

The so-called semantic tagging is an improvement over simple tagging as it link to entities in a dataset rather than to simple strings of characters. In practice, when a search connection to a vocabulary or to a LOD dataset is established, performing semantic tagging does not present significant different for users, if not even advantages, as, for example, nicely choose them from a catalogue with previews and descriptions.

As previously explained, annotations in pundit are set of triples connecting texts, images and entities, thus commenting and tagging is only a subset of the


\(^10\) [http://json-ld.org/](http://json-ld.org/)

\(^11\) [For example: https://cinephilefix.wordpress.com](https://cinephilefix.wordpress.com)

\(^12\) [http://burckhardtsource.org/](http://burckhardtsource.org/)

\(^13\) [http://www.textrazor.com](http://www.textrazor.com)

\(^14\) [http://www.alchemyapi.com/](http://www.alchemyapi.com/)

\(^15\) [https://dandelion.eu/products/datatxt/](https://dandelion.eu/products/datatxt/)
possible annotations. We will see 4.3.1 how Pundit supports comments and semantic tags trough annotation templates (Figure 6 a) and b)).

4.3. Semantic Annotations with Pundit

To continue our use case, let us now consider you are the owner of a movie blog. Your blog have a community of authors. Posts in your blog contain information about movies directors, actors, genre, year of release, etc., but this is expressed in natural language and you need to make it explicit to build coherent metadata. As your blog also contains links to external articles in related websites, you might want to annotate them with the same information.

You then want to involve collaborators so that annotation work can be distributed. Annotations made by different persons will have to use the same structure and will have to be as much as possible consistent. For example you want to have a uniform representation for dates, or avoid errors by constraining the “director” field to be a Person in DBpedia, or one of the directors listed in vocabulary you maintain.

Let us show how Pundit supports this scenario.

4.3.1. Annotation templates

To make it easier and quicker for user to create annotations with a common structure, Pundit allows the definition of custom annotation templates. In Figure 5 c), we show a possible movie template. Once such template is available, it can be selected by a drop-down menu on the toolbar and activated. From this moment on every time the user selects a text a pre-configured annotation is created. User is then requested to fill-in a form by searching in DBpedia or in controlled vocabularies. Each field in the form corresponds to a RDF statement, where the entity mentioned in the text is the subject and specific predicates are used to link it to an author, a release date and a genre.

A template also allows to constraint values for each statement. For example only resources of type Person will be allowed in the first statement.

A very simple template is the one in Figure 5 b), corresponding to a semantic tag. In this case the subject of the triple is a selected sentence or paragraph in the page and the predicates is pre-configured, e.g., to restrict possible values to a list of topics.

Templates provide a uniform interface supporting different kinds of annotations, including basic ones like commenting and multiple relations among different entities.

A template is composed by a set of RDF triples with non-grounded resources and a set of rules, specifying possible entities types and data-types for each triple.

4.3.2. Triples composer

If no template is selected, the triple composer can be used to freely create statements and save them as annotation. Upon selecting a text or an image, you can use them as subject or object of a statement. The triple composer helps you to complete the triples. Items previously used or bookmarked in the My Items folder are also available in the respective tab, and can be used in your annotations.

In any case, available predicates and entities are constrained by the configuration, and more precisely by the annotation vocabulary.

4.3.3. Annotation vocabularies

The annotation vocabularies are custom controlled sets of resources and/or predicates. Pundit loads them at run time following HTTP links specified in the

\[\text{http://conf.thepund.it/V2/templates/ptp}\]
configuration file. In this way, annotation vocabularies can be easily hosted and maintained by a power user on their web sites. Pundit defines a JSON format for vocabularies to be correctly parsed, for more details please refer to the online documentation.17

An additional module is provided in Pundit to enable collaborative editing of one annotation vocabulary. This allows a user community to be able to create new entries in a vocabulary if they do not exist. While activating this feature can lead to possibly uncontrolled folksonomies especially in large communities, in Pundit this effect is mitigated by constraining to predefined categories encoded in annotation vocabularies and by the support for re-using entities from existing datasets.

Figure 8. Annotating regions of pictures with Pundit.

The same method can be used to annotate images as well as precise regions by drawing polygons. Pundit provides full-screen and zooming to work with large images (Figure 8). To use this feature, however, images have to be provided as HTML elements. Unfortunately some web sites use specific widget or application to display images (as often happens in digital libraries).

For lack of space we do not show example of configurations, e.g. to activate/deactivate modules, we suggest interested readers to refer to the online documentation18.

4.4. Integrating the Pundit annotation environment

Once a configuration of pundit is available with proper templates and vocabularies, this should me made available to annotators. The simplest way to connect Pundit to a web application is by using the Feed API. The API, available at http://feed.thepund.it, expects three parameters, passed in standard HTTP GET:

- b, is the URL of the web page to annotate. URLs have to be encoded.
- conf, is the Pundit configuration file. A default is used if no configuration is provided.
- pndurl, an option parameter to load a custom Pundit client installation.

In this setting, connecting Pundit to a blog page to enable annotation capabilities boils down to adding a link to the blog page itself. For example a link to http://feed.thepund.it/?b=https:%2F%2Fcinephilefix, will invoke Feed and load the web page b into the Pundit environment.

You can also decide to simply add Pundit as a Javascript library in your blog to make posts annotatable. However, in some cases you might want users to annotate web pages that are not under your control (e.g. online newspapers). While connecting via the Feed API is still possible, for the convenience of you users Pundit can be deployed as a bookmarklet. This is a little Javascript code that, added to a web browser as a standard bookmark, loads Pundit on any web page.

5. Pundit in use

Finally, let us consider you are a developer and you want to build a faceted search interface to meaningfully browse a corpus of texts, e.g. a collection of letters or papers. You want to provide users with a nice way of exploring the texts based on persons, places and events mentioned in them. All this information, however, is not available in a structured way and has to be carefully created by a team of experts, by annotating the corpus.

In the prototype portal available at http://dm2e-lab1.netseven.it/, a similar use case was put in place at the DM2E event on 18 November at Astrian National Library, Vienna. The aim was that of demonstrating how a group of scholars can collaboratively populate a semantic portal featuring faceted browsing and time-spatial visualization. This involved the following steps:

- Creating a Pundit configuration with new templates and putting it on online.
Making the text corpus annotatable via Pundit. This is done by adding a link to Feed APIs near the texts in the faceted browser.

Gathering metadata from the public Pundit server to populate facets, by connecting to the APIs and feeding it the content of selected notebooks in the form of RDF triples. In this simple setting, each predicate used in an annotation from an “authorized” user is transformed into a facet, where items are populated by possible object values. In practice, this is a simple RDF indexing pattern to implement, even if it affected by clear data scalability limitations for big datasets.

The prototype is based on Solr and Apache-Solr. An indexing script is in charge of synchronizing the Solr index to a RDF dataset via SPARQL queries directly executed on the Pundit annotation server triplestore (the same can be done by contacting the Pundit server REST API). The prototype, illustrated in Figure 9, also support a timeline visualization derived from “date” predicates used in annotations.

A similar approach is used in a pilot project with the Wittgenstein Archive in Bergen, where semantic annotations are being experimented as a way of extending the WAB ontology, made available as a faceted browser to explore the Wittgenstein works, and in the GramsciProject, where they are used to curate online editions of texts.

Ask is a prototype public notebook exploration tool. In Ask users can search for public notebooks hosted on one Pundit server, and explore them as annotations collections. Ask nicely integrates with Pundit by providing direct links to view annotations in context (the web page where they were created) and to further annotate.

An other example of integration of annotation functionalities via Pundit is the Wikidata Annotation tool developed within the Google Summer of Code 2014. This project extends Pundit to be used as a RDF triple editor in WikiData. In this case the Pundit user interface is directly connected to an authenticated WikiData API to insert new triples in the systems. An interesting feature of the tool is that triples created by users comes with a process-able reference to an precise text on the web that is provided by the annotator as a “proof” of the statement. This allows to later analyzing possibly filter the triples based on their truth/relevance.

6. Related works

Semantic annotation systems have been reviewed and compared in literature and formal models have been developed for annotations and annotation systems. In this section we provide a non-exhaustive list of annotation tools related to Pundit.

The semantic tagging paradigm with linking to LOD has been implemented in several application, as Faviki and Zemanta. One click annotation and CWRC-Writer allow simple markup of entities in text excerpts by choosing between predefined categories (as person, location) or creating new ones.

Systems like Evernote and SpringPad, Clipboard allows bookmarking and clipping as well as to collect clips into thematic collections. Capturing parts of the original source.

Bookmarks and annotation tools explicitly targeted to scholars are Zotero and Mendeley. They provide integration with web browsers, basic annotation functionalities and organization of document into folders or categories.

In general those systems does not support custom semantics in annotation and the data is kept into proprietary systems.

---

9 http://wab.uib.no/wab_philospace.page
10 http://purl.org/net/dm2e/wab/search
21 http://dizionario.gramsciproject.org/
22 http://demo-cloud.ask.thepund.it
23 http://www.mediawiki.org/wiki/Wikidata_annotation_tool/
24 http://www.wikidata.org/wiki/Wikidata:Main_Page
25 https://evernote.com/
Annotea [11] is one of the first systems to implement RDF based annotation, providing both client and server APIs for storing structured data. Its semantic capabilities were limited to Dublin code fields. Europeana Connect Media Annotation Prototype (ECMAP) [9], is an online media annotation suite based on Annotea.

LORE (Literature Object Reuse and Exchange) [6] is a semantic annotation system providing a Mozilla plugin to annotate content within the Aus-e-Lit Project. It implements the concept of compound object which is similar to semantic annotation in Pundit. Compound objects are basically set of interconnected resources and can be linked to web content and created with a visual graph UI.

Domeo [4] implements ontology-based annotation metadata on HTML or XML document targets, using the Annotation Ontology (AO) RDF model, which later converged into the Open Annotation data model specification. It is mainly targeted to and adopted in biomedical and scientific domains.

Semantic Turkey [5], is a semantic management system that shares some basic ideas with Pundit. It allows to capture knowledge from web pages by associating it to reference ontologies, and to edit metadata for each of these recognized resources. While user interaction and data model differs from Pundit, and no support is present in Semantic Turkey to share annotations, there is a clear similarity that both tools aims at creating structured RDF statements from web pages.

For what concerns annotation of images, maps and other media, tools like Annotorious and Recogito reached a very high usability and makes crowdsourcing task like annotating places names in ancient maps, easier. The shared canvas project [12] defined a powerful model for representing annotations on digital images, by introducing the concept of canvas to both addressing specific regions of an image and to abstract from the provenance or the version of images.

Previous version of Pundit are discussed in [2] and [8], while [7] presents a prototypical extension of Pundit to annotate frames of Youtube videos.

7. Conclusions and outlook

In this paper we presented Pundit 2.0, a semantic annotation system designed and developed by Net? in collaboration with Semedia, University of Ancona, and resulted from previous tools and years of experimentation in different projects. A preliminary evaluation of Pundit is included in [2]. For a more extensive evaluation of Pundit 2.0, discussion of experiments in the Digital Humanities domain, use feedback analysis and lesson learnt, please refer to public deliverables of the DM2E project, Work Package 3.

Pundit is meant to annotate web content, but supporting complex web pages represents a technical challenge itself. While basic functioning is supported for most wikis and blogs, some interaction! problems might arise in applying Pundit to complex web pages, e.g. when they include pervasive user interactions. One solution could be to transform content into a uniform simple HTML format, at the cost of potentially loosing page context and information. Furthermore, as Pundit annotate content at a web level, it is very hard if not impossible to annotate objects displayed in proprietary or non-web applications (e.g. Flash, ad-hoc image viewers) if they are not known in advance. In several projects specifications where issued for content providers to expose annotatable content, as in DM2E.

In section 5 we showed how specific annotation templates could be associated to specific search and browsing tools to supported communities in creating coherent data. In such a scenario, a clear coupling emerges between templates and visualization/exploration tools. They should share the same data schema and adopt common vocabularies. This aspect is currently not covered by the Pundit infrastructure. However, we believe that a step forward in this direction would be that of formalizing a model – possibly extending existing ones, as OA - to describe tools and annotation modules in terms of both data schemas and supported functionalities.

This work is partially funded by the GramsciSource project funded by the Italian Ministry of Education under the FIRB action and by the DM2E EU project.
References


