DELIVERABLE

Project Acronym: DM2E

Grant Agreement number: ICT-PSP-297274

Project Title: Digitised Manuscripts to Europeana

Deliverable D3.2: Prototyping Platform Implemented

Revision: 1.0

Authors:

Christian Morbidoni (Net7)
Simone Fonda (Net7)

Project co-funded by the European Commission within the ICT Policy Support Programme		
Dissemination Level		
P	Public	x

Revision history and statement of originality

Revision	Date	Author	Organisation	Description
0.1	20.02.2013	Christian Morbidoni; Simone Fonda	Net7	Initial draft version
0.2.	27.02.2013	Christian Morbidoni	Net7	Some additions
0.3	28.02.2013	Violeta Trkulja	UBER	Some additions and revision
0.4	28.02.2013	Christian Morbidoni	Net7	Some additions
0.5	28.02.2013	Violeta Trkulja	UBER	Final revision
Final 1.0	28.02.2013	Stefan Gradmann	UBER / KU Leuven	Approval of Final 1.0

Statement of originality:

This deliverable contains original unpublished work except where clearly indicated otherwise. Acknowledgement of previously published material and of the work of others has been made through appropriate citation, quotation or both.

Contents

List of Figures

1 Role and scope of this deliverable

This deliverable presents the current state of the prototype platform being developed in WP3. It also provides links to online demonstrations of the software components and documentation.

The basic building blocks have been developed and tested and are currently under evaluation within the Wittgenstein Brown Book experiment. From this and other communities, continuous feedback is being received, which will inevitably require the tools themselves to evolve in the next period.

Furthermore, the platform will be expanded during the DM2E project, in two main directions:

- integrate with the Work Package 2 platform which will make available for annotation all the published EDM (European Data Model) representations of cultural objects produced in the first phase of the project.
- implement end-user applications and demonstrate the reuse of structured data coming both from the original cultural objects metadata and from the annotations made by scholars using Pundit. One of the goals of the DM2E project is to involve a wide community of developers and digital humanists in this process (Work Package 4).

2 Prototype platform overview

2.1 The software ecosystem

The prototype platform produced in WP3 is an *ecosystem* of software components. They interact with each other - mostly using web APIs – and provide building blocks for implementing end-user workflows. The most important parts of the workflows (also known as primitives) are annotation and (semantic) augmentation. By annotating and augmenting the content, scholars can collect items of interest and use them to add meaning, or context, to the content itself.

The system supports deep linking of text and images to the Linked Data Web and to controlled vocabularies and taxonomies. It allows the creation of semantically rich annotations using typed relations, for example connecting a portion of a writing to another one asserting that they disagree with each other. The knowledge collaboratively created by users, which maintains provenance and attribution to annotators, forms a big semantic network, which the system manages and exposes in RDF format.

Specific components provide REST APIs and Linked Data interfaces to consume and produce such collaborative knowledge, fostering future development of additional components, like for example visualizations and exploration tools.

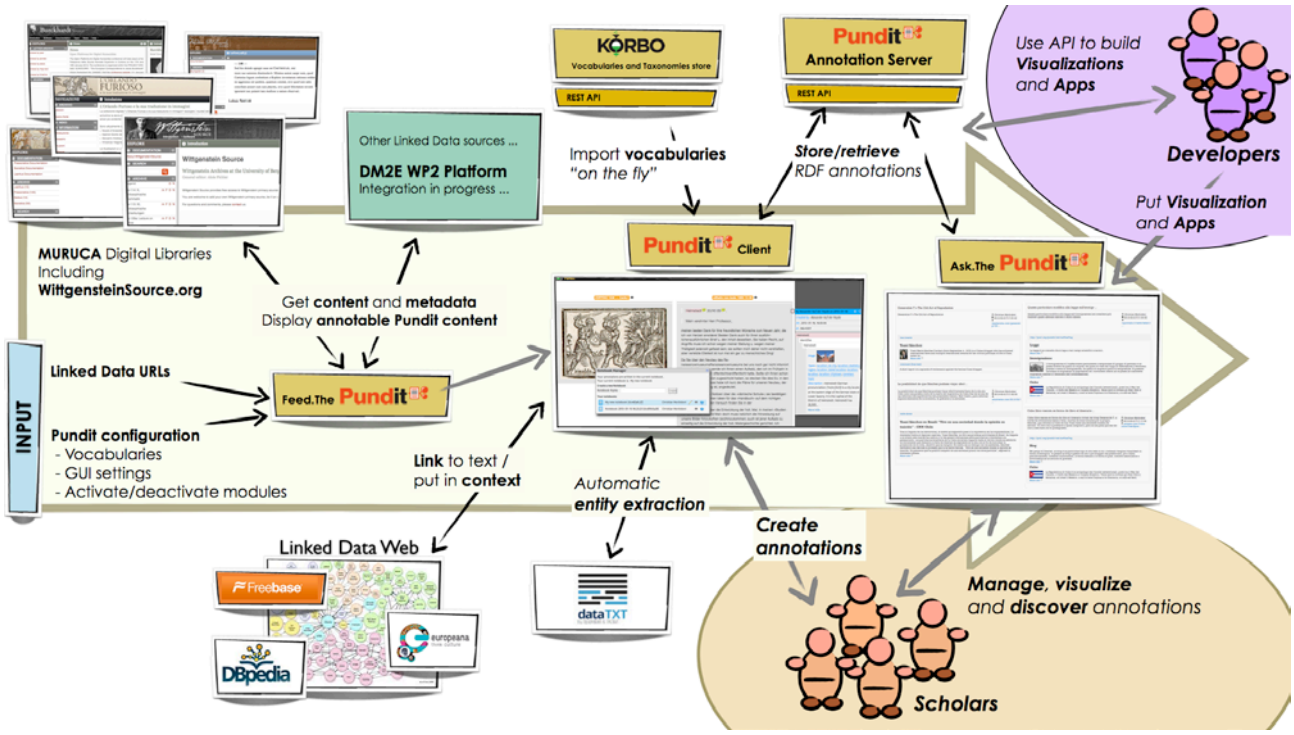

Figure 1: The software ecosystem overview

2.2 Workflow and components interaction

The system takes as input a Linked Data representation of the resource to be annotated. This can be done by typing a URL in the Feed.thePund.it simple web GUI (Figure 4), or by following hyperlinks from a particular Digital Library (DL), as it happens in Muruca DLs.

In WittgensteinSource.org, for example, while browsing the original transcriptions, published by the Wittgenstein Archive Institute of Bergen, users can switch to annotation mode following a dedicated link containing the Linked Data URL of the visualized resource.

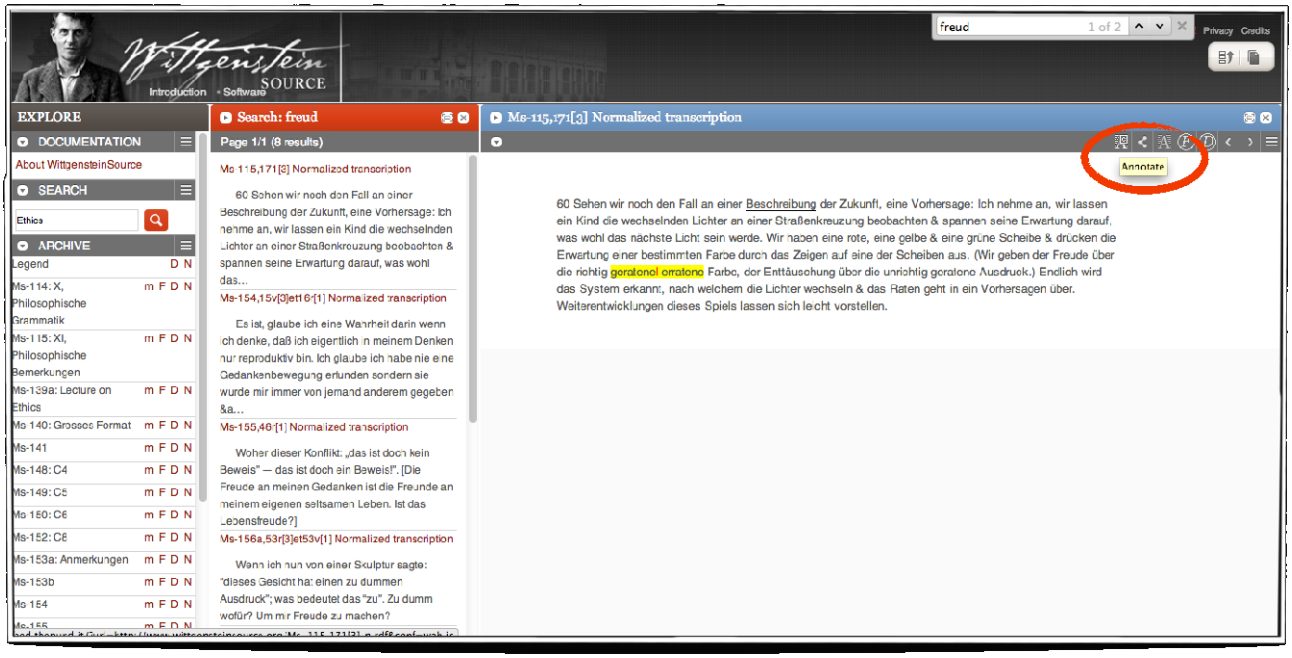

Figure 2: WittgensteinSource.org: the annotate button

This link brings the user to **Feed.ThePund.it,** who downloads and parses the RDF data, in order to present the resource into an annotation environment. Two different contents can be annotated at the same time by specifying two RDF description URLs, making annotations between different resources easy to do, as illustrated in Figure 3.

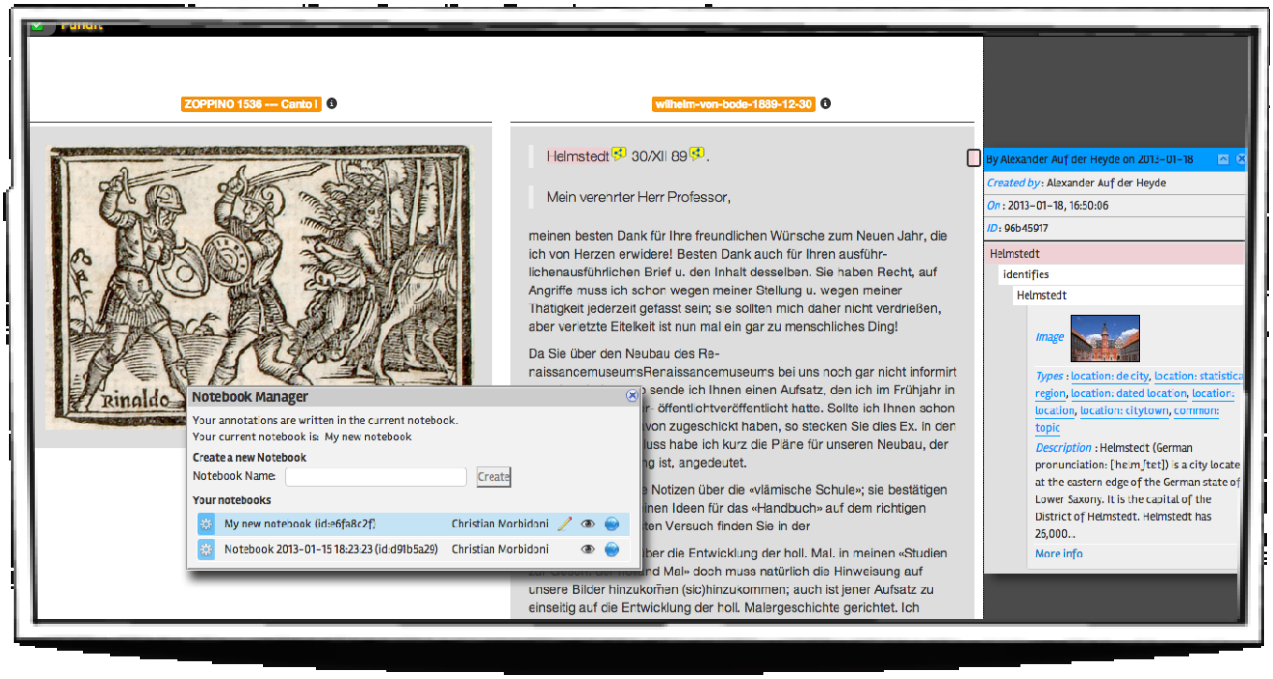

Figure 3: The Pundit annotation environment. In the example an image from FuriosoSource and a text from BurkhardtSource (two Muruca DLs) are shown. The side bar shows an annotation linking to a place mentioned in the text.

Such an annotation environment uses the **Pundit** client **(thePund.it)**: a javascript application that supports fine granular selection and annotation of text and images. With Pundit, fragments of text and portions of images can be linked to corresponding entities in the web of data. For example, a text describing a place can be connected to an entity which identifies univocally the place itself (e.g. http://www.freebase.com/view/en/ancona). In addition, an automatic entity extraction service (https://spaziodati.3scale.net) matches relevant keywords from a selected text, suggesting corresponding entities which could be linked.

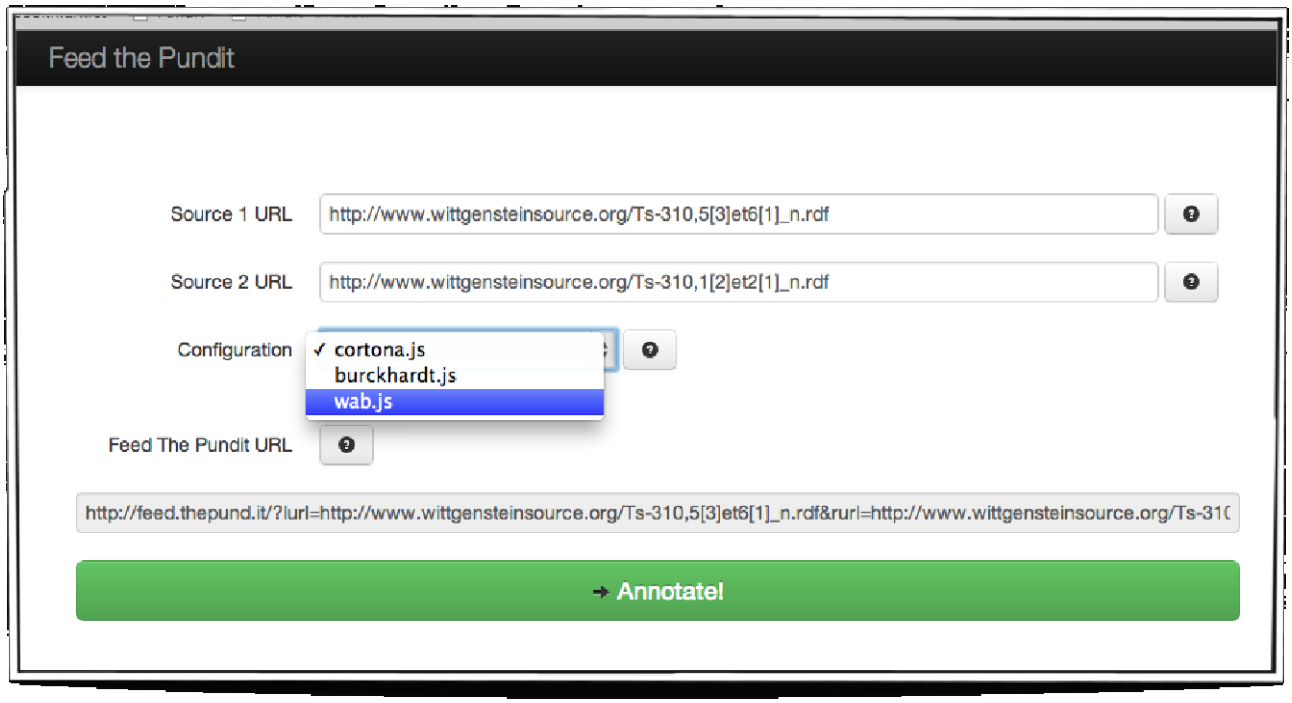

Figure 4: The simple web user interface of Feed.The.Pund.it

Pundit data model is flexible and it can be configured "on the fly" to include custom vocabularies and taxonomies. Vocabularies are expressed in JSON and, although they can be stored anywhere on the web, in the DM2E prototype they are stored in **Korbo**, which provides a basic API and a simple web GUI to create and retrieve hierarchical vocabularies (Figure 5). This enables different communities of scholars to use different data sets. Vocabularies, along with other configuration data, can be given as input to the Feed.thePund.it API, thus delegating to the clients the proper configuration of the desired annotation environment.

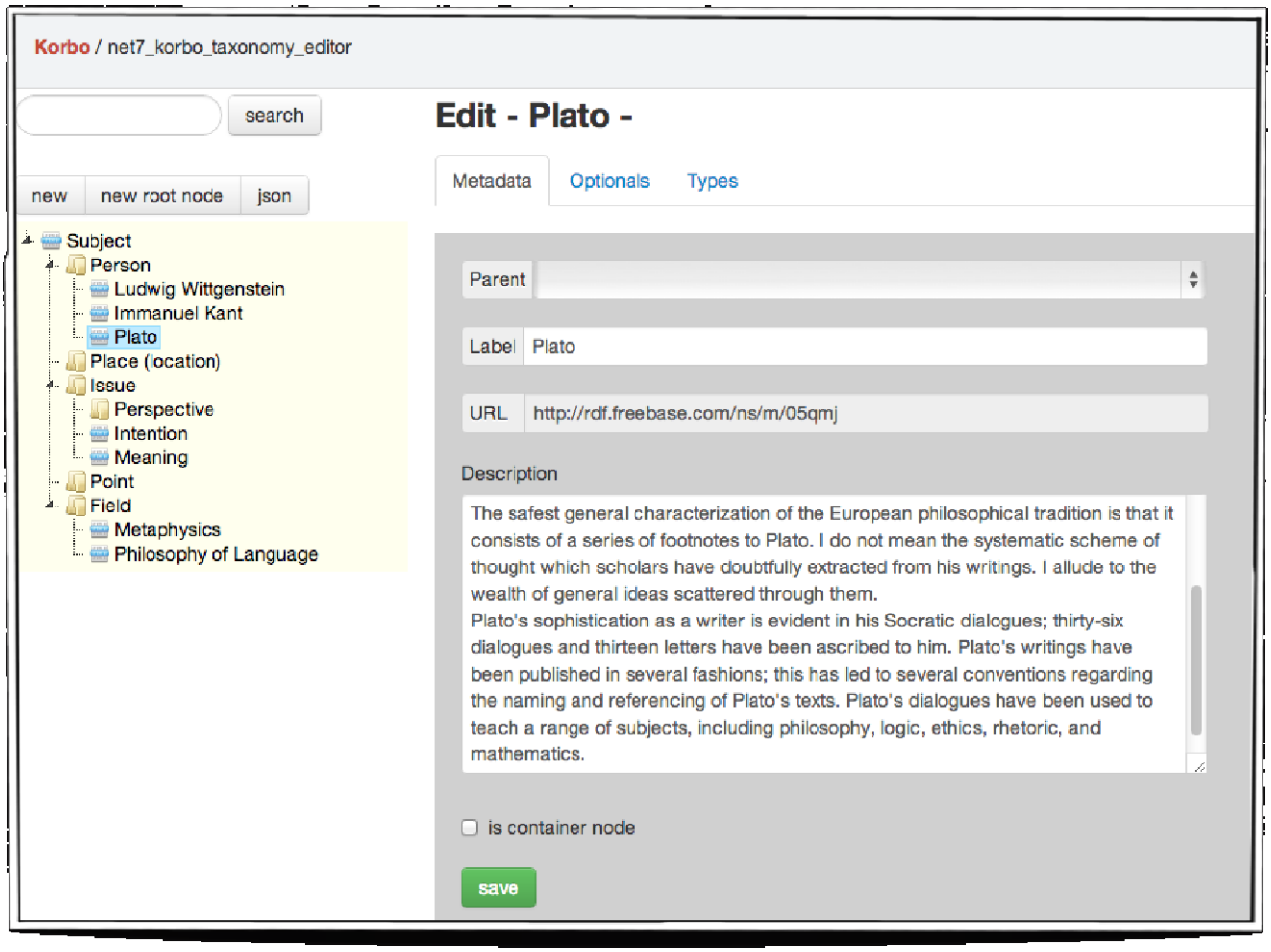

Figure 5: A screenshot of the Korbo taxonomy editing web GUI

In the **Wittgenstein pilot** the set of vocabularies which will be used are extracted from the WAB ontology. This allows, for example, to link a sentence to “Sigmund Freud” (a Person) or to the concept of “Rational Choice” (an “Issue” in WAB ontology’s parlour).

Figure 6: Pundit and the Triple Composer

Moreover, Pundit allows the creation of advanced semantically structured annotations by using its ***triple composer***. It basically allows creating subject-predicate-object statements that represent a relation between two items. They can be text fragments, images, portions of images, persons or any other type of objects defined in a vocabulary. This is illustrated in Figure 6.

The triple composer has tree box, one for the subject, one for the predicate and one for the object of the statement. These boxes can be filled with “items” in different ways, e.g., by drag & drop from the “vocabulary” tab or by directly searching a string inside all the known data sources (custom vocabularies and external services). In Figure 7, we show how to create a simple triple asserting that a piece of text refers to Sigmund Freud.

Furthermore, the triple composer has a type checking mechanism that can be opportunely configured for given domain. Such a mechanism allows to control what “kind” of items can be put in relation by each single predicate. For example, one could configure the system so that the “is author of” relation requires a person as subject of the statement and a “book” as object.

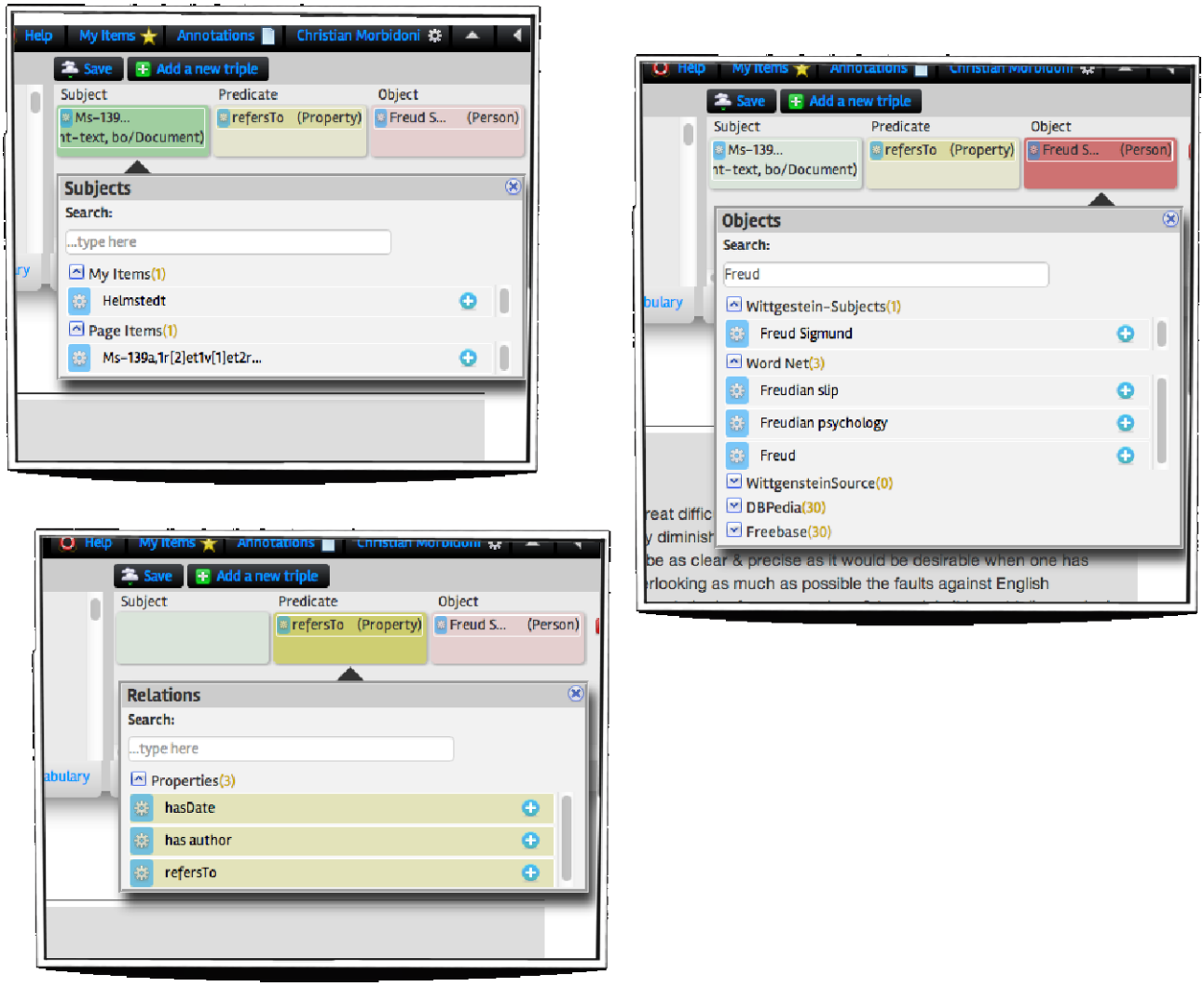

Figure 7: The triple composer in action

In other words, by annotating scholars collaboratively compose a knowledge network where web contents are semantically connected. This is stored in the **Pundit Annotation Server**, who is also in charge of maintaining the association between the author of an annotation and the semantic information that the annotation conceives (e.g. a set of triples). Annotations are grouped in notebooks, which can be kept private or set to publicly readable.

The Pundit Annotation Server provides two types of APIs, public and authenticated, to access annotations and retrieve corresponding semantic data. Third party applications can be built on top of these APIs, for example to produce a number of different applications or visualizations for users annotations.

One of these applications, is **Ask.ThePund.it**, a generic notebook browser that allows users to access their own and other's annotations. Ask.ThePundit, currently in its first alpha release, supports a basic visualization mode and the ability to create, delete and edit personal notebooks. While the application is completely decoupled from the Pundit Client, it relies on the same OPEN ID authentication mechanism (provided by the Annotation server). This means users logged into pundit are also automatically logged into ask and vice versa.

In Figure 8, we show a notebook with annotations of web resources involving Yoani Sanchez, a famous Cuban blogger (a book, one of her blog posts, etc.). Annotations contain textual comments and references to other entities (e.g. a person as author of the blog, the "Immigration" topic as related to the blog post) that the annotator decided to add. Annotation are always linked to the original annotated content on the web.

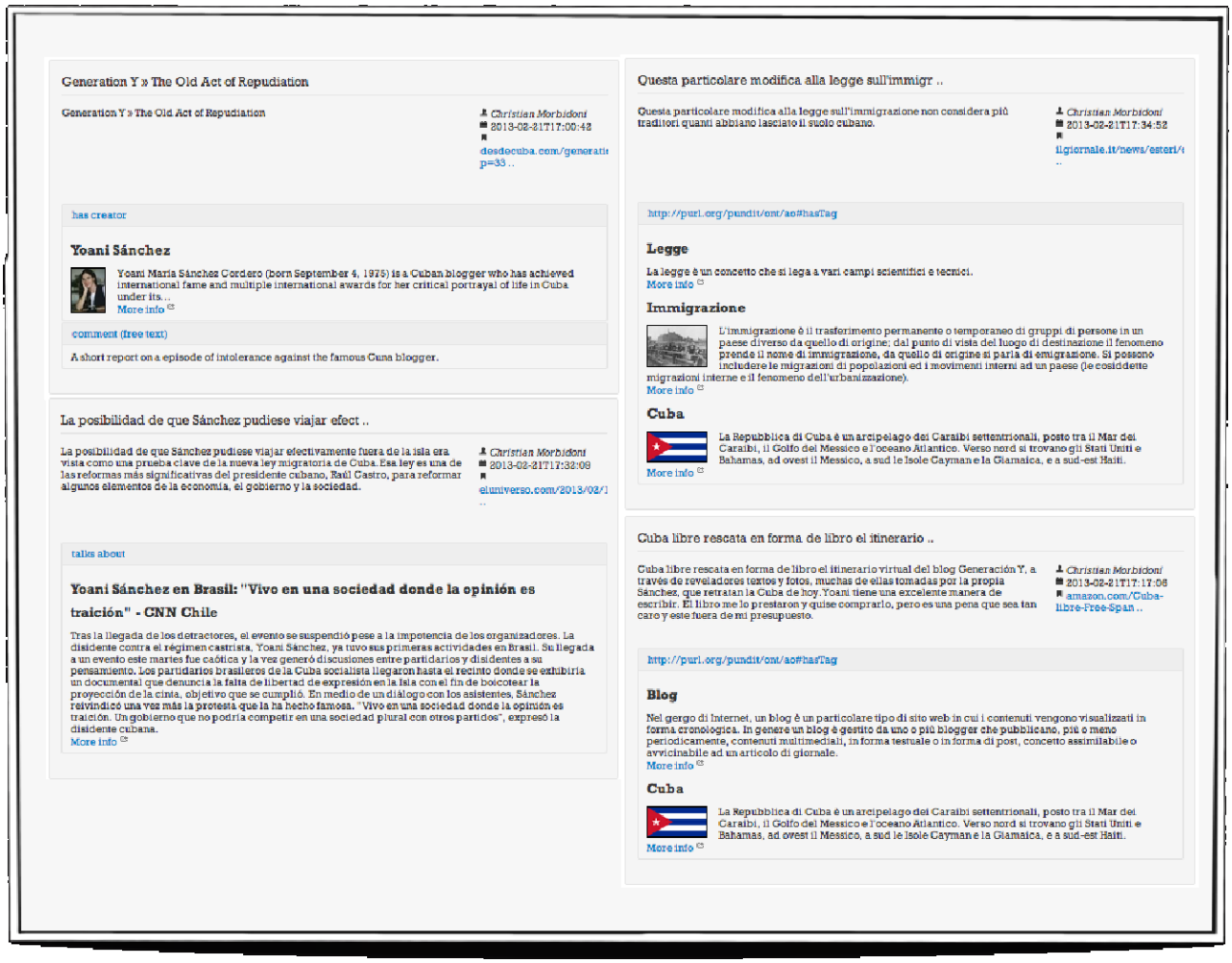

Figure 8: An example notebook shown in Ask.ThePund.it

Ask.ThePundit will possibly provide in the future an integrated portal, working as a collector of data visualizations and applications. One of the goals of the DM2E project in the next period is to prototype demonstrative visualizations and to involve the digital humanities community in developing new and interesting ones.

At the current stage some experiments has been done in visualizing scholarly annotations and simple prototypes has been developed (see section 3).

2.3 The "annotable content"

In order to get basic metadata and pointers to an annotable content, Pundit expects as input a dereferenciable URL with RDF data following the format shown in the following example.

RDF Example:

```
<rdf:RDF
      xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
      xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
      xmlns:bibo="http://purl.org/ontology/bibo/"
      xmlns:edm="http://www.europeana.eu/schemas/edm/"
      xmlns:dc="http://purl.org/dc/elements/1.1/"
      xmlns:pundit="http://purl.org/net7/pundit/vocab#">
      <rdf:Description
            rdf:about="http://www.wittgensteinsourcevps.org/
            Ms141,6[3]">
            Paragraph 16 of Ms-141
            <rdfs:comment>Paragraph Ms-141,6[3]</rdfs:comment>
            <dc:title>Ms-141,6[3]</dc:title>
            <pundit:hasAnnotableVersionAt
            rdf:resource="http://www.wittgensteinsourcevps.org/Ms-
            141,6[3]_n.html"/>
            <pundit:nextResource rdf:resource="http://www.wittgensteins
            ourcevps.org/Ms-141,6[4]et7[1]_n.rdf"/>
            <pundit:prevResource rdf:resource="http://www.wittgenstein
            sourcevps.org/Ms-141,6[2]_n.rdf"/>
      </rdf:Description>
</rdf:RDF>
```

The only assumption made by Pundit is the existence of a triple with the predicate

pundit:hasAnnotableVersionAt

which points to an HTML representation of the content which will be annotated:

http://www.wittgensteinsourcevps.org/Ms-141,6[3]_n.html

In order to work well with Pundit, such HTML chunk of content should follow a simple specification, documented in *Appendix A*. This allows content portions to be properly used in RDF annotations and to establish a link back to the original source.

The following is an example of a compliant HTML chunk. It contains a paragraph, a picture and a caption.

```
<div class="pundit-content"
about="http://example.org/contents/123">
<!-- HTML goes here. -->
<p>This is a named content and contains both text and a picture</p>
<img src="http://example.org/pictires/pictire123.png" />
<p><em>Caption:</em> this is a caption.</p>
</div>
```

The content is included into a DIV element, with a class attribute specifies it is a Pundit Content and that its dereferenciable URL is the one specified in the about attribute. Such URL will be used by Pundit to hook the annotation to the appropriate piece of content.

A more advanced use of the named content mechanism, allows the DL to provide some additional metadata to Pundit, through a REST service whose specification is currently being worked. This would allow the user to directly annotate entities such as an entire

chapter, or paragraph, given that they are presented into a named content which follow the specs to provide the additional metadata.

2.4 DM2E WP2 Platform

Integration with the WP2 platform, which in DM2E publishes and exposes as Linked Data a big amount of digital objects from DM2E partner, is one of the goals of the incoming stage of the project.

In principle, the digital objects collected in the WP2 platform should provide a RDF representation with all of the needed information to initialize the Pundit annotation environment.

At the time of writing the REST API of the WP2 platform has been announced and the integration activity will start with the Wittgenstein EDM data.

3 Building on top of the knowledge graph

As mentioned, the annotations produced by users with Pundit contain structured RDF data and are merged to the initial graph (including basic metadata about digital objects provided by the data sources) to form a bigger knowledge graph.

One of the upcoming research directions in this project is that of understanding how this knowledge graph can be used, visualized and explored, to bring added value to scholars who are studying the digital materials.

In this section we discuss two different demonstrations of how existing open tools can be used to provide insights and visualize such knowledge.

3.1 Browsing the graph with LodLive.it

In the Wittgenstein Brown Book experiment, an initial ontology with main classes, properties and links among Wittgenstein's writings, has been released by WAB.

As the ontology is formalized in RDF/OWL, it is straightforward to store it into a triplestore and connect it to LodLive.it (http://lodlive.it, a recent open-source Linked Data exploration tool) through a SPARQL endpoint. The result is shown in Figure 9, where a page of the Brown Book is connected, with typed links, to all the related nodes in the graph.

This interactive application is an interesting exploration tool as it allows to follow relations and further expand nodes to highlight paths in the graph. Going from node to node makes in fact possible to discover new information and, more interestingly, to do it virtually at web scale. If properly configured, when a resource comes from DBPedia.org, users can expand it and browse the DBpedia graph as it was an "extension" of the local one, showing new and possibly unexpected information.

Figure 9: A portion of the Wittgenstein knowledge graph shown in LodLive.it

As scholars add annotations, the initial graph grows to include new relations that users have created.

For example let us consider the annotation illustrated in Figure 10. With these screenshots we show the process of creating a triple asserting that a text from the Brown Book has one of the “Issues” formalized in the WAB ontology as subject.

This bit of new knowledge can be visualized immediately by reloading the LodLive.it page, which leads to the graph illustrated in Figure 11.

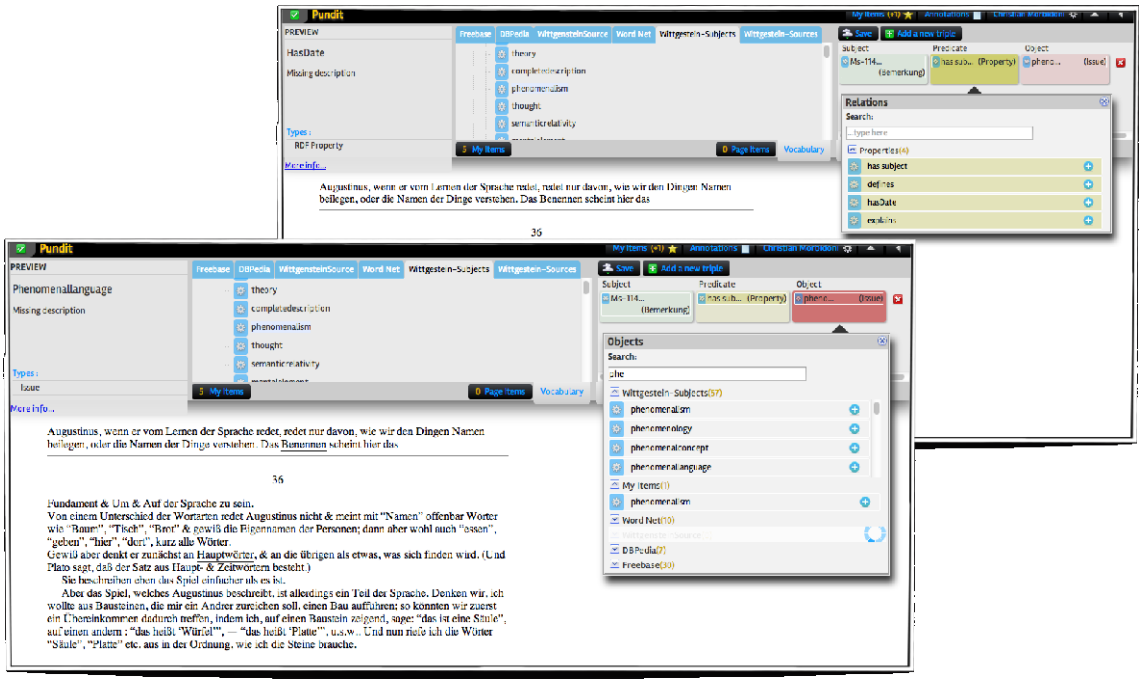

Figure 10: An example annotation made on Wittgenstein Brown Book

Figure 11: A new triple added by a user's annotation is shown in LodLive.it

3.2 A demonstrative application: EdgeMaps

Recently the digital humanities community attentions have been captured by interactive graph visualizations such as Edgemaps (http://mariandoerk.de/edgemaps).

In a popular demo, influences among philosophers are shown in a conceptual map that helps understanding and exploring paths in the history of philosophy. The demo shows

influence relations coming from Freebase, a well know general-purpose Linked Data repository.

While for a “generic” user such a visualization is enough, we can’t probably say the same when it comes to scholars that consider such relations as the core of their studies and might legitimately ask: “Why exactly you say that Marx influences Gramsci?”, ”What is the evidence of that in the actual primary sources?” or “Who affirms that?”. Based on this idea we customized Pundit by including a relations vocabulary extracted from the CiTO ontology (http://purl.org/spar/cito). The relation set includes predicates like “cites” and “quotes”, as well as other more specific like “discusses”, “cites as sources”, “agrees with”, etc.

We then used the bookmarklet version of Pundit, that allows to run the tool on generic web pages without installing anything, to annotate primary sources on the web, hosted by the Wikisource.org project (which collects non copyrighted materials from a variety of authors in an open data portal).

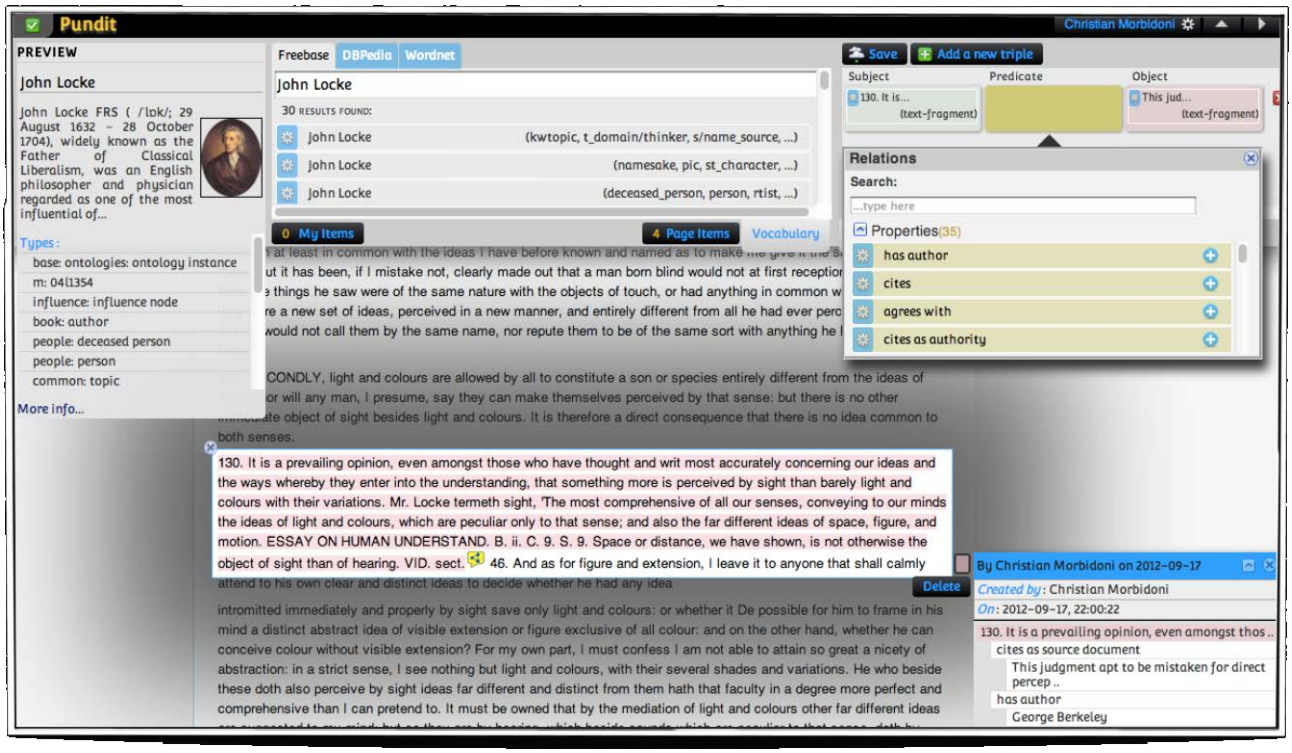

Figure 12: An example annotation in Pundit, ready to be visualized in the EdgeMaps demo.

Finally we extended Edgemaps’ code to load influence relations from a Pundit user notebook instead of grabbing them from Freebase.
Each time the user creates a relation using some of the properties from the CiTO ontology, connecting two texts from different philosophers, a corresponding edge is created in the edgemap.

In this new visualization (see Figure 13), each time two philosophers are connected by an “influenced by” relation, the corresponding annotations are shown so that the scholar can immediately get evidence of “why the relation is there”, deciding to agree or not.
It is also possible to load multiple notebooks from different scholars, enabling a collaborative scenario, where annotation authorship is always tracked back and each user can decide what notebook to see or trust.

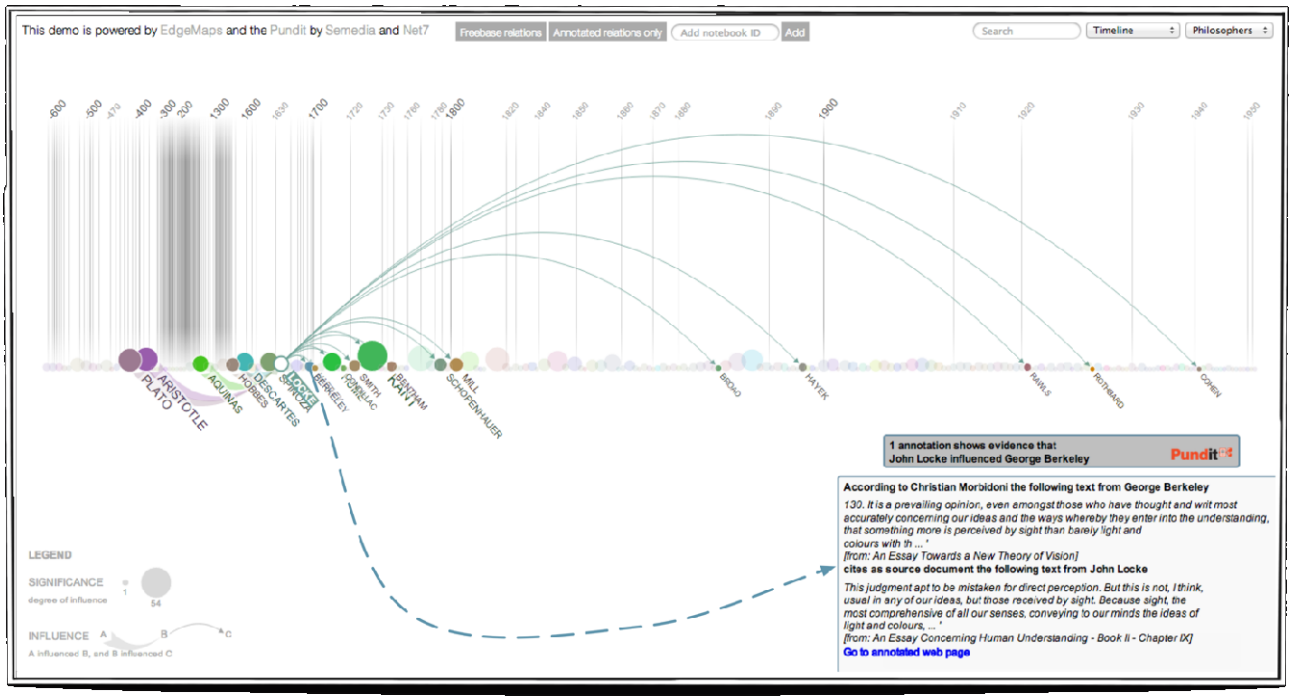

Figure 13: The EdgeMaps demo showing annotations related to a influence relation in the graph.

4 Components and online resources

WittgensteinSource - http://www.wittgensteinsource.org

Is a Digital Library maintained by the Wittgenstein Archive of Bergen and realized with the Muruca technology. It is integrated, in a lightly coupled fashion, with the Pundit annotation system.

Pundit - http://thePund.it

Pundit is a semantic annotation tool that has been adopted and further developed within this project to be used as an augmentation tool by scholars to add information to online cultural objects.

Feed.ThePund.it – http://feed.thePund.it

An application with a very simple Web GUI that serves as an access point to the Pundit annotation environment. It can be fed with one or two URL of annotable objects and a configuration, that sets appropriate vocabularies and environment's parameters.

Ask.ThePund.it – http://ask.thePund.it

A web based annotation explorer, that let users search among public notebooks and among his own notebooks to explore information and manage personal annotations.

Korbo - http://korbo.org

Korbo is a vocabularies store that supports hierarchical structure. Items in a taxonomy can be created from scratch (choosing a name, description, etc.) or "copied" from a Linked Data source. Korbo also provides a web based GUI to manage and edit taxonomies.

All taxonomies and corresponding metadata are accessible via a simple REST API, that is used by Pundit to fetch vocabularies and terms on the fly.

A demonstrative instance of Korbo is installed at http://korbo.netseven.it.

LodLive.it – http://lodlive.it

LodLive is a recent javascript application that connects to one or more SPARQL endpoints and/or Linked Data sets, to provide visual graph exploration.

Edgemaps+Pundit demo – http://thepund.it/apps.php

A demonstrative application that creates a specialized visualization to show influences among philosophers extracted from a user's notebook.

5 Appendix A – Named contents

Pundit is designed to allow annotation of generic web pages. However it provides additional functionalities if the markup of the page adhere to some simple rules.

In order to fully exploit Pundit's capabilities, a web page should use a specific markup to enable Pundit to identify each piece of content that the page is showing. We call these pieces "named contents".A named content is just an HTML element container which clearly specifies the boundaries of the content and assigns a stable identifier (an URL) to it. Example:

```
<div class="pundit-content" about="http://example.org/contents/123">
<!-- HTML goes here. -->  <p>This is a named content and contains both
text and a picture</p>  <img
src="http://example.org/pictires/pictire123.png" />
<p><em>Caption:</em> this is a caption.</p> </div>
```

The CSS class "pundit-content" enables Pundit to recognize named contents. The value of the attribute "about" tells Pundit what named content can be found in that container. The value of the attribute "about" should be a stable URL and should be resolvable to a web page containing just the named content itself. Please note that the container tag, the CSS class name and the attribute used to store the content identifier are fully configurable, allowing maximum flexibility and easy integration with existing pages. The HTML content enclosed in the container element should not change over time.

Why named contents?

Web pages often contains, along with the actual "content" (e.g. a document, a digitized manuscript, a picture) other accessory elements: think about navigation menus, advertising banners, page headers and so on.

Properly marking up the content enables Pundit to ignore such accessory elements, at the same time allowing it to preserve content-annotation relations if such elements change over time (e.g. after a web site has been restyled).

Additionally, web pages often changes location over time. A web page that today is available at http://example.org/paper/mypaper.html, can be moved to http://anotherdomain.org/papers/paper1.html in the future. Assigning identifiers to content, and making them explicit via markup, enables Pundit to hook annotations to content rather than to web pages, allowing it to show the right annotations when the content moves among different web pages.

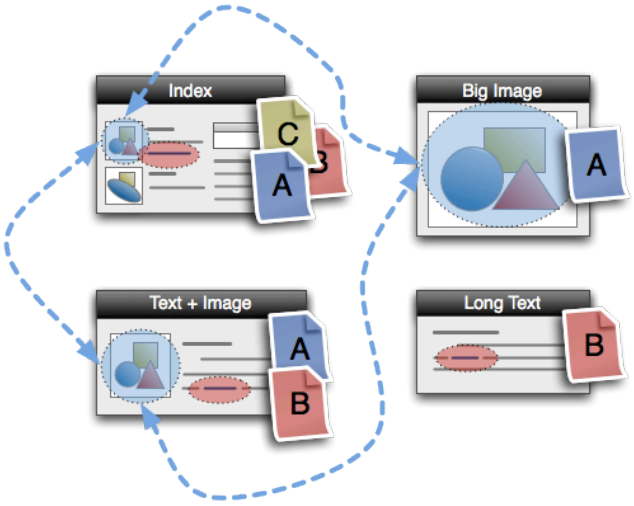

Figure 14. Named contents enclosed into different web pages

Finally, the content within a web page can have a certain degree of granularity. For example, a web page can display multiple pages of a document, or multiple paragraphs of a document page. In some cases such atomic contents can be replicated in several pages. For example an index page often display a short document summary which contains only relevant paragraphs, while another page could contain all of them.

Again, if each paragraph is contained in a named content identified by a stable URL, Pundit will be able to display the right annotations in all of the web pages containing that same content, following the principle “same content, same annotations”.

What if my page does not have such markup? No worries! Pundit will use a pretty decent fallback: it will hook your annotation to the page’s full URL.