DARIAH Winter School in Prague

Open Data Citation for Social Sciences and Humanities

24th to 28 of October 2016

Session 4: Persistent Identification

4-Persistent Identification

Persistent Identifiers (PIDs)

Ondřej Košarko, Institute of Formal and Applied Linguistics, Faculty of Mathematics and Physics, Charles University, Czech Republic

Why PIDs?

- Making data available but it then requires referencing and interlinking
    - resources
    - semantic definitions
- Reproducibility and reuse of results: collaboration on data can improve it or at least verify the results.
- Location on the web is not a good identifier. It will stop working. You upload your data and get a link to it, but the problem is that URL or links will stop working - I am not even saying that they *might* stop working, it will stop working, it is a matter of time.
    - The content can be changed at will: When you share a link, the target of the link can be changed, not necessarily to cheat others, but to improve data and correct it but this has an impact for a paper written years before for example. In this case it is not possible anymore to access what was there.
    - Names can lead to certain expectations: URLs often contain semantic words that can lead to some conclusions, like the “final_version.pdf” and “final_final_version.pdf” extensions

Good identifier

- Persistence
    - Longevity: in an imaginable future, we should still be able to get sense of what the attached idea was.
    - Commitment of involved actors for the future.
- Uniqueness of the identifiers: You should not be able to change it in the future, otherwise nobody can reproduce your work and it might even not be findable anymore => One ID for one “object” and ideally one “object” has only one ID (but that can’t be guaranteed across multiple PID systems or even in some systems on their own)

PID systems

There are in fact different systems because people have different idea of what persistent identifiers should do beyond the identification, the persistence and the uniqueness.

- URI/URN (IETF standard)
- Handles (DOI)
- PURL

- ARK
- Info-URI
- XRI

Different concepts

- Naming schema, like URNs

urn:oid:0.9.2342.19200300.100.1.3
Eventually http://oid-info.com/get/0.9.2342.19200300.100.1.3
Urn:oasis:names:specification:docbook:dtd:xml:4.1.2
But it might not land on something that you consider as an authoritative resource if you are not even sure that this is a real identifier, because it might not be easy to find.

- Resolution system

It provides a way to give you a location, like web driven proxies, a mean to see the resource that is assigned the identifier. It means that if your browser is enabled with a DOI or Handle plugin, you could click the DOI and find the resource. If not, you can still use the proxy that is provided and that does the resolution. So when you click a link, you end up on the resource.

    - DOI Handbook: doi:10.1000/182 or http://dx.doi.org/10.1000/182
    - Mark Twain's sketches, New and Old: hdl:loc.gdc/scd0001.00162117695 or http://hdl.handle.net/loc.gdc/scd0001.00162117695

- Services around DOIs

With DOIs, you have an infrastructure built around like Crosscite does, it allows to get a citation just by providing a DOI, so you don't have to download the resource. See Crossref's Auto-Update for ORCID records.

- Parts/fragments

It is useful for continuous data: some DOI providers also offer identification for fragments of a described resource. If you have PID assigned to an audio document for example, one approach is to create another identifier for a specific part is this audio file.

- Costs and ease of use

With handles, when you are on the provider side and you want to start providing handles to your users, (you are not using someone else repository) it might take some time because you have to communicate with the global handle registry and pay some fees (50$ a year). With DOI, the pricing policy differs, if you want to attach PID to a great number of documents, DOIs might come expensive.

Versions and PIDs

The rules are up to the PID providers.
If PIDs should provide uniqueness, how can you assign one PID to two different versions of the same document? One objection might be: how do you keep track of the versions? Where is which version? Which is the newer? You can keep it in your metadata, use the DublinCore relation "isreplacedby" and provide information about the other resource that was previously used. But with PIDs you might not always get what you expect:

- What is a substantial change
- If the point of PIDs is persistence and uniqueness, shouldn't new versions "automatically" get different PID?

- The versions can be linked in metadata

Granularity
The question is to know if there is some minimal size from which you should assign a PID or not. If you are working on textual resources, you need to refer to one particular character and that resource because it is some medieval text and it is the only appearance of the character you have found, so it makes sense to assign a PID to it. Basically, the PID system does not limit what the objects are, but the provider might because of the rules that can be set up in such a way that you won't fulfil it. With repositories, if you are uploading a file that contains a character, it might be odd but still if you are able to provide descriptive metadata, it makes sense.

Persistence
It is the idea that PID is persistent, not the resource itself. The resource is made persistent by being added in an archive that guarantees that it will take care of it for years to come. PIDs provide a way to make the resolution possible, not the resource itself. In certain cases it makes sense to withdraw a resource, the PID should remain as the descriptive metadata (and explain the withdraw: claim not true, institute disappeared, etc.).

Shortref.org: **(Moving away from repositories to) Citing arbitrary views of data**

How to cite the use of your research data on one specific aspect which might be representative of the phenomena but not for the complete picture => Pictures are better than words. For example, if the dataset and the querying service are online with such an URL: https://lindat.mff.cuni.cz/services/pmltq/#!/treebank/pdt30/query/IYWgdg9gJgpgBAEgK4AcUwE5wFwF44DaAUAM7AC2MIALhjFcCYgEYzUDu9YANEaJLEQAbCJyx5CcALrSA3H3DR4CVhy458BOAEI4wMAGMYJahCzI0mXgAo41s7AwgUdI7CYX0GPmChwHmCAAZhBCIuwe4ZgAIERmdgFOIWGiHqhePn6Jzq4w7sKiMXDR0kREQA/result/svg. You can use a shortener even if it might have the previously mentioned issues. Shortref.org service (handle) is part of LINDAT/CLARIN, the Centre for Language Research Infrastructure in the Czech Republic.

Metadata and url:
- Assign a PID (handle) to an url, or make it usable - even if the service is down
- Possible to change the location: We keep a track
- Additional metadata
- Healthcheck on the location

For users:
- Fill the form: URL & metadata
- Save the token for future updates
- Use the handle as reference

For services:
- Use REST API: provide metadata and URL, store the token

- Show the PID
- Just a click away

Overview:

1. Creation
2. Monitoring: Health check, the locations are periodically polled. After certain number of failures, an error page is shown: http://shortref.org/resource_down.html#hdl=11346/FFF-TN7H
3. Resolution

Contact

Ondřej Košarko, Institute of Formal and Applied Linguistics, Faculty of Mathematics and Physics, Charles University
Ondřej Košarko is a programmer working at the Institute of Formal and Applied Linguistics (UFAL), Prague, Czech Republic. He is one of the developers behind LINDAT/CLARIN repository. The repository is based on DSpace and has been modified to meet the needs of CLARIN centers. This modified version is now deployed in several member institutions. He is also responsible for parts of shortref.org, a tool to ease persistent data citation, and various other bits and pieces like this guide for choosing the adequate licence.
Institutional websites: http://lindat.cz & http://ufal.mff.cuni.cz/
Email: kosarko@ufal.mff.cuni.cz

Canonical Text Services

Christopher Blackwell, Furman University

CTS Implementation has been developed through the Humboldt chair at the University of Leipzig. This presentation will be an implementation of the values that Ondřej Košarko just articulated.
CTS is a protocol for identifying and retrieving passages of text by means of machine actionable canonical citation. This is something we have developed for the Homer multitext project since the 2000. In this context, CTS is not necessarily traditional citation, but unique, unambiguous citation values that are independent of technology or format allowing to cite a physical book, a digital text in TEI-XML or not. Ideally a citation should capture the semantics of the text, what we call the citation hierarchy and the bibliographic hierarchy.
I am a classicist and a lot of our texts have been read and deeply cared about for a long time and they have good traditional schemas of citation, which map nicely into the digital realm, for example Homer, Iliad, book 1, line 1. Some texts don't have a traditional schema of citation at all because they might be 20th century texts or they have a traditional schema of citation that doesn't work well in the digital realm.
CTS consist in two parts: the probably most important is URNs, the identifiers, machine actionable citation for identifying passages of text. There is also a CTS service protocol: request and response for retrieving information about a passage of text; http using CTS URNs.
CTS is based on a model of text that is an ordered hierarchy of citation objects, we call this OHCO2, pronounced "ochio 2". CTS is not a cataloguing application, it is not an editing application, it is not a search engine, it is not a browsing, reading or commenting application, but we have found it to be useful as a component in all of this things. We found CTS URNs to be extremely useful for expressing the results of automated textual analysis and also the work of human editors. CTS is not limited to TEI-XML text, it is not limited to digital text; CTS URNs can identify passages of text in physical volumes. CTS can't say everything about the history, nature and meaning of a text, which is why we write scholarships on commentary, it is just for identifying and retrieving passages of a text.

Data model: Ordered Hierarchy of Citation Objects

According to this model, a text consists of citation objects. In a traditional format, each of this is a precise identification of a citation object. The original OHCO goes back in 1990 and consider a text as an ordered hierarchy of content objects. It immediately caused violent civil wars and people started to fight about what was content. For example, is markup content? Is whitespace content? If you have two whitespaces in a digital text is that one content or two?
Our approach was to back it up and think in term of citation objects. In CTS, the citation object has textual content and it can be mixed content or just plain text. By separating the textual content from the citation object, we have a lot of flexibility. We can have textual content itself or embedded with some kind of analysis in our text. We can add it without altering the citation and the text doesn't even have to look like text. For example, in one

digital exemplar of one of Iliad text, you might not be interested in the content of the Greek corpus but in metrical values.

Ordered hierarchy of citation object
It is ordered because you read from the beginning to the end, the sequence matters.
It is a hierarchy, text may be organised by containing elements, ex. Iliad book 1 contains 611 citation objects, which can be poetic lines. The hierarchy may be only one level deep or many levels deep, different sections of a text may have hierarchies of differing depth. CTS is good at this and we use XML, so we know the citation objects, headings and paragraphs, etc.

Text in a bibliographic hierarchy
CTS also put text in a bibliographic hierarchy which is a sort of FRBR (Functional Requirements for Bibliographic REcords). In CTS, text belongs to text group that may be author or not. Notional works become real when you have versions of them. The Iliad is an abstraction, but you sometimes want to talk about real things, like post translation of Iliad, specific editions and texts of the Iliad. A text may also be an exemplar, which is a specific instance of a version, for example Thomas Jefferson's personal copy of the edition of Iliad, in which he wrote notes. In a digital realm, we see the idea of an exemplar as a specific text derived from a version. So, if we do diplomatic edition of the Iliad, which is a *version*. From that, if we produce a normalised version, we would call that a *digital exemplar*. It has a clear *relationship* to a version, it depends on it.
The CTS URNs are our effort to capture all of this semantics and bibliographic hierarchy and citation hierarchy as precisely or imprecisely as we need to in a machine actionable citation. This *arbitrary identifier* happened to be the numbers that the TLG canon of Greek authors and works used. In this authority list, TLJ0012 is homeric epic, TLJ001 is the Iliad, and a particular version MSA, and then citation book 1, line 1. You can then mix and match the components of this, ex.: Homer, Iliad, no version, 1.1. So this is the Iliad in general book 1, line 1, with no specific version in mind. It identifies any version of the Iliad that has a book 1, line 1. There is a manuscript in Venice that begins with book 16, this doesn't cite that manuscript, but any version out there that does have whether in English or French, etc., that has a book 1, line 1. We can express *ranges*: from book 1 line 1 to book 1 line 10. There is no reason to believe that it will results in 10 citation objects, they look like numbers but they are just arbitrary numbers. Besides, you can have *mixed range*: from book 1 line 600 through the end of book 2.
With CTS and this *sub reference*, we can identify more specific strings of text within the citation element and, at least in the Homer multitext implementation of CTS, we don't retrieve on this. If you put this on the CTS server and with a "get passage" request, you will get all of book 1 line 1 and then it is your problem to find the first instance of the string meaning in it.

CTS Service

It is an http request, you have a URL to the service. Request equal "get capabilities", will return a catalog of text that the service knows about and can offer. And we have schemas

that define how that catalog works. These are the three most important CTS requests with what you can get everything done:

- get valid reff
- given the URN: will give every valid citation define by the URN. This is useful in cases when you have fragmentary text, ex. manuscript that begins with book 16 and other text where you need to know where the citations are.
- get passage given the URN: (also get passage + get first reference)

Text for CTS

Any text uses pieces that can be identified by Canonical Citation CTS compliant. TEI-XML works great, it is a little complicated because you have to do some processing, but it works great. The Homer multitext with our work starts with TEI-XML and we process them into RDF statement. The complete expression of OCHO2 model book 1 line 2 of our edition of the Iliad has a series of RDF statements. We are confident on the data model because we regularly start with XML and bring it to RDF and back into XML fragments for serving. This kind of round tripping suggests that this model is actually capturing the semantics of the text.
The simplest possible CTS text will be two column tab delimited/separated values files where you have an URN and text content. The URN captures the citation hierarchy and you have an ordered hierarchy of citation objects and this will be a good CTS text. Over the years, we have implemented CTS on a lot of ways, google app, engine, existDB, etc.

Homer multitext implementation of CTS

It is a downloadable virtual machine. The fundamental difference between our implementation and what Matt Munson is going to show you, it that is works with TEI-XML and keeps the file as XML that allows constant integration that way; our involves RDF backends, so it is two different approaches.

Recommended reading

Amy H. Blackwell & Christopher W. Blackwell, Hijacking Shared Heritage: Cultural Artifacts and Intellectual Property Rights, 13 Chi. - Kent J. Intell. Prop. 137 (2013). Available at: http://scholarship.kentlaw.iit.edu/ckjip/vol13/iss1/6

Contact

Christopher W. Blackwell holds a B.A, summa cum laude from Marlboro Collect in Vermont, USA. He holds a Ph.D. from Duke University, where he was the William H. Willis Fellow in Classics. Since 1995 he has been on the faculty of Classics at Furman University in South Carolina, USA. He served as Chair of the Classics Department for 14 years, until 2015, and is currently the Louis G. Forgione University Professor. Since 2001 he has been Project Architect, with Neel Smith, of the Homer Multitext, a project of the Center for Hellenic Studies of Harvard University under the editorship of Casey Dué and Mary Ebbott. With Smith, Blackwell is co-creator of the Canonical Text Services protocol and the CITE Architecture for identification and retrieval of scholarly resources by canonical citation in networked environments. Blackwell has led several digitization projects and has

collaborated with scholars in the U.K., Italy, Germany, the Netherlands, Greece, and Croatia. He has published two books on the history of Alexander the Great, and articles on topics in Classics, Computer Science, Intellectual Property Law, and Botany.

Academic website:
http://www.furman.edu/academics/classics/about/Pages/FacultyandStaff.aspx
Email: christopher.blackwell@furman.edu

Workshop: CTS with CapiTainS, Hook(Test), Nemo, and Nautilus

Matt Munson, Humboldt Chair of Digital Humanities, University of Leipzig

HookTest, Nemo, and Nautilus Setup

1. Instal Docker up and running
2. Instal Hooktest, create a virtual machine
3. Nemo nautilus

This repository contains the documents that will be edited during the CTS Workshop at the DH 2016 Conference in Krakow. Nautilus is a python based CTS API using XML TEI files following CapiTainS guidelines. Nemo is a user interface, starting to grow as a CMS (Content Manager System), which reads its core data from standard CTS API calls (and so is compliant with any standard CTS API normally) and is starting, as of the beta of 1.0.0, to accept annotation resources such as treebank and images in the form of plugins. Hooktest is a software developed to make unit tests on XML repository regarding CapiTainS Compliances.

CapiTainS compliance

(urn:cts:gerLit:ger0001.ger001)

1. Rename the files (ger0001.ger001.opp-{ger,fre}1.xml)
2. Create Repository structure (data/ger0001/ger001/)
    a. Run HookTests
3. Create the __cts__.xml metadata file for the text group ger0001
    a. Run HookTests
4. Create the __cts__.xml metadata file for the work group ger001
    a. Run HookTests
5. Add URN to ger0001.ger001.opp-ger1.xml
    a. Run HookTests
6. Add line numbers
7. Add refsDecl
    a. Run HookTests
8. Check out Nemo!

Hooktest is a continuous integration environment for text. When you do code development with collaborators, you know the continuous integration is where every time you make a change with the code it runs certain test to make sure you didn't break anything. And this is basically what Hook is for text. Every time we make a change to a text, it will test the text against whatever we want to test it against.
Run Epidoc test on the Goethe repository: ./epidoc.sh DH2016-master: Some tests are passed and other failed. The results are exported as Result.json & results.html.
The webpage tells you have two files and none passed (the tests). Then, add metadata file into the repository, description of the text groups (author) and of each work level, describe as precisely as you can what you actually have.

Citation schema

Poem: citation level & line citation level. Make sure that the citation scheme is correctly described and encoded. The citation scheme for a poem is by line, so this poem has 30 lines=> add a number to each lines. Add to the TEI header where are the citations located. Tell the XML parser how to find these two (levels of citation): In the encoding description, in the epidoc, you have a reference declaration, and we say it is CTS and then you have two patterns: one for the line and one for the poem and Xpath tell where these levels are located.
In Kitematic, you can get the Nemo Nautilus, which is the CTS server and it shows that we have one collection (German Literature), within it we have one author, etc. You can also make CTS API, see the documentation.

Some Requests

- .../api/cts/?request=**GetCapabilities**
- .../api/cts/?request=**GetValidReff**&urn=urn:cts:gerLit:ger0001.ger001.opp-ger1&level=2
- .../api/cts/?request=**GetPassage**&urn=urn:cts:gerLit:ger0001.ger001.opp-ger1:1.2
- .../api/cts/?request=GetPassage&urn=urn:cts:gerLit:ger0001.ger001.opp-ger1:1.2**@mit**
- BUT NOT .../read/gerLit/ger0001/ger001/opp-ger1/1.2@mit

And you can also refer to a specific word or letter. It will lookup and return the whole line (to get the context of the occurrence). This allows you to refer to extremely specific parts of a very specific text. Then you can add annotation to this very specific word in this very specific edition that is in this very specific place. If someone wants to look at your annotation and see your work, they will be able to go back to your text and see exactly what you were talking about.

Useful links

- http://capitains.github.io/pages/tutorials
- https://github.com/Capitains/docker-hooktest
- https://github.com/Capitains

Contact

Matt Munson, Humboldt Chair of Digital Humanities, University of Leipzig
Matthew Munson received an MA from the University of Virginia in Religious Studies, his thesis studying the use of the Greek word for law (ν όμ ο ς) in the letters of the Apostle Paul. Before joining the Digital Humanities Team, he worked at the Scholars' Lab at the University of Virginia and in the DARIAH project at the Göttingen Centre for Digital Humanities at the University of Göttingen, Germany. He is currently working on his PhD in Theology in Leipzig studying the automatic extraction of semantic data from biblical texts and the automatic tracking of semantic drift between corpora.
Academix Website: http://www.dh.uni-leipzig.de/wo/team/

Email: munson@dh.uni-leipzig.de