Perseids Platform Analysis: curation and annotation of the Shakespeare His Contemporaries Corpus (subset of TCP/EEBO)

October 20, 2014

Bridget Almas, Gernot Höflechner

Currently the Perseids Platform enables the creation of "publications" (from existing XML editions or from scratch) which can be composed of one or more of the following types of files:

- TEI XML Transcriptions
- TEI XML Translations
- Treebanks (morpho-syntactic annotations)
- Translation Alignments
- Simple triple based annotations using varying ontologies, serialized per the Open Annotation data model
    - examples of annotations of this type include identification of text reuse, quotations and named entities
- Commentary annotations, expressed in Markdown and also serialized per the Open Annotation data model

Publications can go through one or more review workflows, whereby the entire publication is reviewed by a single board, or different components can be reviewed by different boards according to document type, etc. More traditional, pre-configured review boards as well ad-hoc communities are both supported. A general overview of the currently supported review workflows in Perseids can be found at http://sites.tufts.edu/perseids/workflows/approval-workflows/.

Currently, CTS compatibility is a prerequisite for deployment of a text on the Perseids platform. At a minimum, this means a CTS URN identifier must be assigned to the text and it must appear in a CTS TextInventory served by an instance of the CTS API. Identification of the canonical citation scheme, and its mapping to an XPath within the text, is required for passage-centric functionality.

CTS support is a core element of the Perseids platform and serves a number of different purposes:

- CTS Text Inventories provide the catalog(s) of texts which are available for curation and annotation on the platform, and also those which can serve as the content of an annotation (e.g. in the case of text reuse, where an annotation identifications a quotation which appears in one text, and its source in another). These Text Inventories can be retrieved from an instance of a CTS API which may be either local to the Perseids deployment, or remote.

- The citation scheme for a text as encoded in the metadata of the CTS TextInventory also drives passage-centric functionality on the platform, enabling texts to be both edited and annotated at the passage level.
- Annotations may also reference specific ranges of text within a passage, via the CTS sub-referencing scheme. We use CTS URN sub references rather than XML identifiers as the targets of our annotations so that they can be applied more broadly across different editions of a work, and so that they can be technology independent and possibly outlast changes in digitization and serialization techniques. There are issues with this if the underlying text is not exactly the same across all editions or istokenized differently, but the goal is to provide a scheme which is robust enough to allow for automatic calculation of the correct target in most cases, given some context, and also allows for identification and manual adjustment where automated methods fail.

Although the texts in the TCP/EEBO corpus do not currently leverage the CTS protocol, it is not inconceivable that they could. A sample CTS URN scheme for "*The Honest Lawyer"* (text A6863 in the SHC sample) could look something like the following:

urn:cts:shc:sheppard.A68363.folger-en1

In this example, the namespace '*shc*' might be defined as applying to the corpus of texts which make up the Shakespeare His Contemporaries corpus. The textgroups might be attributed authors, in this example, *sheppard.* Work identifiers can be opaque or semantically meaningful – in this example I just used the identifier assigned in the sample given to me, *A68363.* And finally, I've created an edition identifier, *folger-en1,* to represent this specific digital edition of the text. This too can be opaque or semantically meaningful, here I've chosen a pattern similar to what we use on Perseus, including an identifier for the publisher, language and sequence. A fourth component, for the exemplar, could be used to reference very specific versions of this particular edition (e.g. you might use in the exemplar component *experiment, origspell,* or *stdspell* if you wanted to reference the different encodings provided in the SHC sample).

With the above URN identifier then, an annotation which targeted the first instance of the word "Cuckold" in Act 1 Line 1 of the Folger edition of "*The Honest Lawyer"* would look like

urn:cts:shc:sheppard.A68363.folger-en:1.1@Cuckold[1]

It should be noted though that CTS doesn't itself provide the means to access all parts of a text, only those pieces which are part of the "canonical citation scheme". So for plays, this means that cast lists, stage directions, etc. would all be considered separate and need to be identified and served via another means. One option would be to treat everything that is not part of the canonical citation scheme as stand-off markup, or annotations of one sort or another. However, this may not make sense in all cases, particularly for dramatical works where things like stage instructions, cast lists, etc. are integral parts of the work. This is an area that needs further thought, particularly for a corpus like SHC.

Note also that the SoSOL component of the platform supports other identifier schemes and catalog approaches, and can be fairly easily extended for new ones, so the CTS aspect of Perseids is not a prerequisite for use of this component of the platform. In addition, many of the tools we use on Perseids are CTS- aware but do not require it. For example, Arethusa, the annotation framework which we use for treebank annotations and which will play an increasingly important role in the platform as described further below, will check the document identifier for a treebank document to see if it contains a CTS URN. If so, it will look up the standard abbreviation for the document (e.g. Hom. Il.) in an external service and display it to the user. If not, it will simply display the document identifier unadorned.

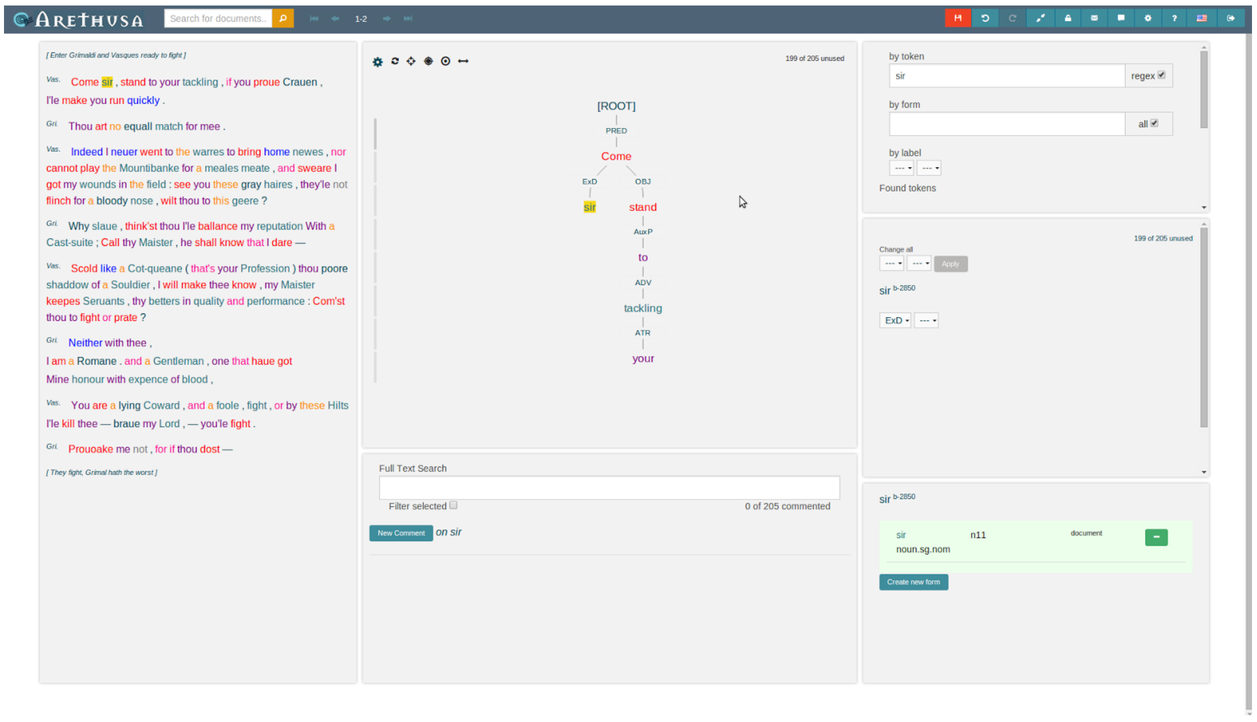
Another prerequisite for deployment of an XML text for curation on the platform is adherence to a validating schema, expressed either in XML Schema or Relax-NG. Currently in SoSOL the identifier scheme used for a document is a bit too tightly coupled to the definition of the validating schema for documents using that identifier scheme, requiring some duplication of effort to make general functionality available across document types. This should be fixed in order to improve the extensibility of the platform. Since the Perseus texts are transitioning to the EpiDoc TEI schema, we have focused on enabling new functionality for EpiDoc documents. So, while we do already have initial support for documents adhering to the TEI-Analytics schema, enabling the new functionality for TEI-Analytics files is lagging a bit.

All changes made on the platform are made in the text files and/or the annotation files directly, and saved as commits in a git repository. Our philosophy is try to work on the source files themselves in all cases, rather than extracting data to a relational database and then trying to merge it back in or reconstruct the text from there. In the coming year we will also be experimenting with an approach for managing annotations simultaneously as documents in git and as named graphs in a triple or quad store.

While a Perseids workflow can simply end with the individual files in a publication being committed to the master git repository, we also plan to automatically produce disseminations which merge the parts of a publication into a single coherent whole, e.g. for publication as a web site, ingest into an institutional repository, or inclusion in an ePortfolio. This work is not yet complete but it is in scope for the current Perseids grant ending in September 2015, for at least one or two model publication and display types. We also do already have prototype dissemination code for publications which combine all of these different annotation types to represent the final data in a dynamic HTML 5 interface with annotations linked to and accessible from the source text, by way of CTS URN references (e.g. see http://perseids.org/sites/bodin/ and http://perseids.org/sites/berti_demo/).

The Perseids platform currently integrates different tools for annotations of different types, and users switch between them for different tasks. However, we intend to implement, and have the means to develop in the Arethusa Annotation Framework, support for working across multiple annotation types and documents at a single time. Arethusa's event-driven design allows for different annotation tools to listen for changes and actions on a token and respond as appropriate for their use case. For example, a change to the morphological analysis of a token can trigger behavior which links a grammatical construct to the sentence in which it occurs. We intend in the future to use Arethusa to support use cases such as correcting typos in a transcription while annotating the morphology, and having the typographical correction automatically cascade to the different representations of the text managed by the platform. It should be noted that Arethusa is highly configurable and extensible and is not tied to any particular back-end for retrieval or storage of texts. It is therefore conceivable that Arethusa could be used as a curation and annotation environment for the TCP/EEBO texts on its own without use of the Perseids back-end.

Although the currently deployed version of Arethusa works primarily on treebank documents, we have work in progress which will enable it to work on TEI XML files directly. We have begun prototyping this with the EEBO SHC sample corpus, extracting the lemma and morphological attributes in the @ana attributes and make them available for correction and saving the change directly back to the source TEI document.
Integrating the corpus with Arethusa automatically grants access to the framework’s other plugins. The following screenshot shows an EEBO text in an environment with the treebanking, the comment, the morphology and the search plugin active.



[Enter Grimaldi and Vasques ready to fight]

Vas. Come sir , stand to your tackling , if you proue Crauen ,
I'le make you run quickly .

Gri. Thou art no equall match for mee .

Vas. Indeed I neuer went to the warres to bring home newes , nor cannot play the Mountibanke for a meales meate , and sweare I got my wounds in the field : see you these gray haires , they'le not flinch for a bloody nose , wilt thou to this geere ?

Gri. Why slaue , think'st thou I'le ballance my reputation With a Cast-suite ; Call thy Maister , he shall know that I dare —

Vas. Scold like a Cot-queane (that's your Profession) thou poore shaddow of a Souldier , I will make thee know , my Maister keepes Seruants , thy betters in quality and performance : Com'st thou to fight or prate ?

Gri. Neither with thee ,
I am a Romane . and a Gentleman , one that haue got
Mine honour with expence of blood ,

Vas. You are a lying Coward , and a foole , fight , or by these Hilts
I'le kill thee — braue my Lord , — you'le fight .

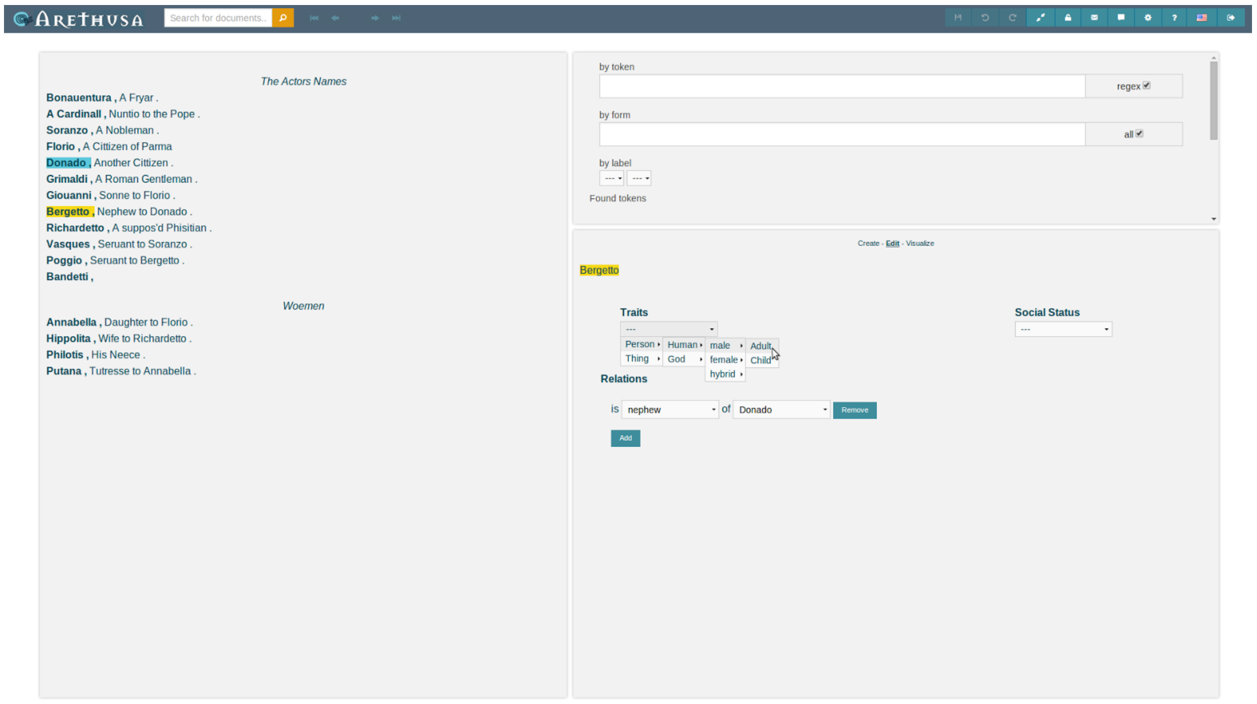
Gri. Prouoake me not , for if thou dost —

[They fight, Grimal hath the worst]

https://raw.githubusercontent.com/latin-language-toolkit/arethusa/master/dist/examples/images/eebo1.png

We have also been prototyping an Arethusa plugin which can use a configurable hierarchical ontology to create semantic annotations on a text. This might be used in the SHC corpus, for example, to annotate prosopographical characteristics of members of the cast. We can envision a scenario where the text is automatically parsed to extract annotations which the end user can then curate. For example, given “Bergetto, nephew of Donado ” a relation between Bergetto and Donado could be automatically proposed, e.g.

Bergetto -is-nephew-of-> Donado
Donado -is-uncle-of-> Bergetto



The Actors Names

Bonauentura , A Fryar .
A Cardinall , Nuntio to the Pope .
Soranzo , A Nobleman .
Florio , A Cittizen of Parma
Donado , Another Cittizen .
Grimaldi , A Roman Gentleman .
Giouanni , Sonne to Florio .
Bergetto , Nephew to Donado .
Richardetto , A suppos'd Phisitian .
Vasques , Seruant to Soranzo .
Poggio , Seruant to Bergetto .
Bandetti ,

Woemen

Annabella , Daughter to Florio .
Hippolita , Wife to Richardetto .
Philotis , His Neece .
Putana , Tutresse to Annabella .

https://raw.githubusercontent.com/latin-language-toolkit/arethusa/master/dist/examples/images/eebo2.png

We can also imagine a number of ways which external visualizations could be used during and after annotation, to highlight and answer specific research questions. For example, looking at when facts about relationships (person b is the mother of person a) are revealed over the course of a play, examining the social status of the characters across a play or entire corpus of plays, and their clustering, etc. We would like to implement such visualizations through an external service called by Arethusa, and allow users to contribute their own views.

Once we have full support for working with TEI XML in Arethusa, and have the ability to produce such visualizations, we expect to be able to use it both as a close-reading environment and an annotation environment, switching seamlessly back and forth between the two. This is generally inline with what Arethusa tries to be. Producing annotations is one goal - but there are also lots of other tools that allow simple annotations of text (mostly in generic input and output formats that need further curation/transformation after they have been produced). Arethusa’s design is driven by the desire to connect annotation tools (manual and automated ones) and dissemination tools under one hood. It should also be a reading environment that tries to work with annotation data. For example, treebanking should be possible both in a view designed specifically for that task as well as in a plain or structured text view. Users could then annotate a text according to dependency grammar while reading, and switch to immediately visualize such annotations in a tree as well. This same concept could be applied to a wide variety of other annotation types. If Arethusa can combine visualizations of such annotations as well as the ability to create them in the first place, it is possible to disseminate annotation data on the fly. This is especially useful when a user detects an annotation error, e.g. by looking at a graph of such annotations. He should be able to correct this error and immediately see the effects of his changes in the visualization - without a need to switch programs, convert data back and forth, reload data etc. Arethusa's architecture is built to support this - it is meant to be understood as a growing platform where people add different tools - a developer can then choose to chain such tools together to provide a good user experience.

Our current assumption for the Perseids back-end component of such an environment is that the text will be served from a CTS Service which retrieves citations directly from the TEI XML and saves annotations on them bilaterally – making them immediately available for querying and use by all (filterable by creator and other contextual and provenance metadata), while also allowing them to go through a separate review/publication workflow in Perseids. Accomplishing this will require the deployment of pieces of infrastructure that we don't yet have in place, and is a bit out of scope for the current Perseids grant, but which we hope will be funded by other future and pending proposals.

For more information on the architecture of the Perseids platform, Arethusa, and use cases and approaches for integrating with these tools, see https://github.com/PerseusDL/perseids_docs/tree/master/integrations.