Encoding and Designing for the Swift Poems Project

Jonathan Swift and the Text Encoding Initiative
Introductions

James Woolley
(Emeritus) Frank Lee and Edna M. Smith Professor of English at Lafayette College

Stephen Karian
Associate Professor of English at the University of Missouri

James R. Griffin III
Digital Library Developer at the Lafayette College Libraries
Overview of the Swift Poems Project

- Woolley and Karian seek to archive poems attributed to Jonathan Swift
- Beginning in 1987, this has involved:
  - Identifying and Cataloging Primary Sources
  - Transcription
    - Copy-typing, encoding, and annotating the primary sources
    - This method has **not** relied upon the usage of the TEI Collation
  - Collation
    - Identifying the copy-text (and variant texts) for any given poem
Overview of the Swift Poems Project

● The Libraries at Lafayette
  ○ In 2009 Woolley consulted with the Libraries for assistance with the Project
  ○ Visual Resources Curator (Paul Miller) developed a set of Microsoft Access Databases
    ■ These structure the catalogs maintained by Woolley and Karian

● In 2012, the NEH awarded a Scholarly Editions Grant for the Project
  ○ This includes supporting the development of a digital edition
  ○ This will be integrated into volumes of the *Cambridge Edition of the Works of Jonathan Swift*
  ○ Digital Scholarship Services (Department in the Libraries)
    ■ Agreed to support this project formally using a planned migration to Fedora Commons
    ■ Griffin joined Digital Scholarship Services in the role of digital library developer
Identifying and Cataloging Primary Sources

● Identifying the Sources
  ○ Sources are 18th century printed and manuscript texts
    ■ To date, over 6500 manuscripts have been identified and cataloged
  ○ Few digital surrogates for the printed and manuscript texts are available
    ■ Of these only a restricted set aren’t protected under copyright

● Cataloging the Sources
  ○ Bibliographic metadata are structured for each source
    ■ Elements are extracted from external catalogs (e.g. English Short Title Catalogue)
  ○ An author attribution may be specified (but it is rarely authoritative)
  ○ Not all sources are cataloged with an authoritative title
    ■ An internal identifier is used to reference poems as a result
Transcribing the Primary Sources

As he whom MDUL\poe\bus in his Ire
Hath MDUL\blasted with Poetick Fire.
What Hope of Custom in the MDUL\Fair MDUL,
While not a Soul demands your more?
Where you have nothing to produce,
For private Life, or publick Use?
<MDUL\Court, City, Country> want you not;
You cannot bribe, betray, or plot.
For Poets Law makes no Provision;
The Wealthy have you in Desision.
Of State-Affairs you cannot smother;
Are awkward when you try to flatter.
Your Portion, taking MDUL\left MDUL\round,
<PN>:MDUL\Paid to the Poet Laureat, which Place was given to one MDUL\Cibber MDUL, a Player> was just one annual Hundred Pound.
Now not so much as in Remainder
Since MDUL\Cibber MDUL brought in an Attainder;
Nor ever fixt by Right Divine
<MDUL\Grubstreet> Line.
<MDUL\Monarch's Right> on <MDUL\Grubstreet> Line.
Poor starvling Bard, how small thy Gains!
How unproportion'd to thy Pains!
And here a MDUL\Smile MDUL comes put in:
Though MDUL\Chickens MDUL take a Week to fatten,
The Guests in less than half an Hour
Will more than half a Score devour.
So, after toiling twenty Days,
To earn a Stock of Pense and Praise,
Thy Labours groan the Critick's Prey,
Are swallow'd o'er a Dish of Tea;
Gone, to be never heard of more;
Gone, where the MDUL\Chickens MDUL went before.
How shall a new Attempter learn
Of different Spirits to discern MDUL?
And how distinguish, which is which,
The Poet's Vein or scribbling Itch?
Then hear an old experienc'd Sinner
Instructing thus a young Beginner.

On Poetry A Rapsody (Poem 640–35D–)
Transcribing the Primary Sources

- The transcripts themselves are created using the *Nota Bene* application
  - *Nota Bene* encodes textual structure using a system of tags termed as “mode codes”
    - «MDUL»: This mode encodes italicized style rendering «MDNM»
    - «MDBO»: This mode encodes black letter «MDNM»
  - The researchers have further extended this system to support editorial annotation:
    - Lorem «MDUL» add·caret «MDNM» · this text added with a caret · ipsum
    - Dolor «MDUL» del «MDNM» · this was deleted · sit amet
  - Not all instances of annotative markup require mode code tags:
    - \pasted·over\
    - \printed text\
Accessing and Preserving the Transcripts

- Accessing the Nota Bene comes with challenges
  - The Nota Bene release used by the researchers has been 3.0 (released in 1988)
  - Accessing the Nota Bene directly would require a virtualized environment for Microsoft DOS
  - The Nota Bene transcripts are managed as text/plain media resources

- The Text Encoding Initiative P5
  - Provides a robust data model
  - Standardized and open format for interchange
  - A more effective solution for preservation
Encoding the Transcripts

On Poetry A Rapsody (Poem 640-35D-)}
Encoding the Transcripts

- Encoding using the TEI-P5 could not be a manual process
  - The researchers required a system to transform *Nota Bene* into a TEI-XML implementation
    - An API for Ruby using Nokogiri was developed to support this
- Viewing the TEI-XML was of limited value
  - Research techniques driven by Nota Bene require a rendering of the text
  - Styled HTML5 (using Twitter Bootstrap) serves as a minimum viable product
    - Improvements can be rapidly prototyped for the encoding
  - This approach takes inspiration from Agile software development practices
    - The stakeholders have continuously improving (or maturing) prototypes
    - The approach is also draws upon “pair programming” within eXtreme Programming
On Poetry A Rapsody (Poem 640–35D–)
Enriching the Encoded Transcripts

● Limits are obviously present with this approach
  ○ Researchers are not encoding the transcripts using the TEI P5
  ○ The developer for the Ruby API is not a literary scholar

● How can this encoding be made collaborative?
  ○ The developer and the researcher could operate in a shared environment
    ■ This is inspired heavily by the pair-programming technique within eXtreme Programming
  ○ In this case, both the developer and a researcher share a physical working environment
Enriching the Encoded Transcripts

● Collaborative encoding and quality control
  ○ The researchers will identify faults in the rendered transcripts
  ○ The developer can extend the Ruby API, XSL, or styling for the HTML5
  ○ This enables rapid prototyping of the interface
  ○ Delivery time for the researcher (in transcribing the sources) can be increased
    ■ In response, the developer can more readily scope improvement requests

● Textual criticism is still not enabled by this approach
  ○ The researchers must identify variant readings to a given text
  ○ Critical apparata are not explicitly encoded within the Nota Bene transcripts
Collation for the Swift Poems Project

- Collation as a solution
  - Originally the researchers collated the *Nota Bene* transcripts using a FoxPro program
    - Visualization was used to identify variation
    - Tokenization was customized
    - A set of controlled characters (&,~,|,#) symbolized differences in structure
Collation within a Digital Scholarly Edition

- A collation interface was scoped for the digital edition
  - This interface must enable the transition from the legacy collation engine

- Collation features could be extended
  - Lines are still tokenized
    - Initially attempted to preprocess the text and use the Penn Treebank tokenizer
    - Ultimately found that abstracting the tokenizer was simply more effective
  - Alignment is addressed without the use of controlled characters
  - The edit distance between tokens can be calculated

- Experimental features can be introduced
  - Part-of-speech tagging to further enhance textual analysis
    - Currently a pretrained Perceptron tagger is being tested
      - May investigate more performant approaches (e.g. Hidden Markov Model)
Collation within a Digital Scholarly Edition

Collating Variants for the Poem ！W190
Collation within a Digital Scholarly Edition

● The collation can also address flaws in the encoding
  ○ By default, all unencoded Nota Bene markup is stripped from the TEI
    ■ Users will be able to collate the texts and visualize differences
  ○ Optionally Nota Bene can be preserved
    ■ Researchers still retain access to some of the controlled characters
    ■ Researchers and the developer can identify unencoded *Nota Bene* sequences

● A heatmap is currently the supported visualization
  ○ This is a straightforward means of rendering the textual differences
Collation within a Digital Scholarly Edition

Collation for 640–

640–35D– (Copy-Text)

640–36L– (Variant)

640–34L2 (Variant)
Forthcoming Features

● Design Improvements
  ○ Stakeholders have driven the requirements for the UI
    ■ Interviewing and testing for public users must be undertaken
  ○ Extending UI features using JavaScript frameworks
    ■ The digital edition is current implemented in the Tornado framework for Python
    ■ Solutions such as AngularJS and React reduce UI to a set of modular components
      ● They also require a RESTful API to be implemented

● Preservation
  ○ Ingestion of the critically edited reading texts in the TEI-XML
  ○ Lafayette College Libraries is a member of the Project Hydra community
    ■ Migration for other systems (Islandora and DSpace) is underway
    ■ Modeling TEI resources in Hydra could then expose metadata elements in the RDF
Encoding and Designing for the Swift Poems Project

Thank you for your attention

Contact Us
James Woolley (woolleyj@lafayette.edu)
Stephen Karian (karians@missouri.edu)
James R. Griffin III (griffinj@lafayette.edu)
Appendix: Workflow and System Architecture
Appendix: Collation Engine

- Solutions for collating variants of the transcribed texts were explored
  - Juxta
    - Supporting the integration of the Juxta API was given the highest priority
    - Given existing infrastructure and resource concerns there were limitations:
      - Preprocessing and postprocessing the TEI-XML Documents was necessary
      - Juxta itself required performance optimization for our environment
  - CollateX
    - An extremely viable solution
    - Mature (and maintained) Module for Python
    - Interoperability issues in supporting the features of the legacy interface
      - Concerns over whether preprocessing or postprocessing would be required
      - These concerns may not be warranted
Appendix: Collation Engine

- Prototyping a collation application in Python
  - The Tornado framework offered several advantages
    - Support for multiprocessing in collating larger sets of TEI-XML
    - Support for WebSockets (enabling asynchronous updates for a collation job)
  - Python Modules used to extend the features for the collation could be used
    - Natural Language Toolkit (supporting extensible tokenization)
    - NetworkX (supporting the building of stemmatic trees)
  - Integration with API’s for XML databases could also be explored
    - eXistdb
    - Zorba
    - PostgreSQL