The MONK Project Final Report

John Unsworth and Martin Mueller

September 2, 2009

I. Brief description of the project and purpose of the grant:

In the original proposal for the MONK (Metadata Offer New Knowledge) project, we envisioned three phases to this project:

1) Assembling a substantial testbed (on the order of millions of words)* of literary texts in English, from the beginning of the history of print to the early 20th century; combining functionality from WordHoard and Nora (two projects previously funded by the Andrew W. Mellon Foundation) in a new web-based interface; and integrating MONK as much as possible as an application layer in SEASR (Software Environment for the Advancement of Scholarly Research), the open-source data-analysis infrastructure that succeeds D2K, and that is also funded by the Andrew W. Mellon Foundation (see http://www.seasr.org/).
2) Doing some proof-of-concept work on social software capabilities for MONK, including the sharing of intermediate work-products (for example, pre-processed sub-collections selected by one user and then shared with others), sharing of results, annotation and correction of data, etc.. Part of this second phase was also projected to include working with a small number of libraries and publishers to provide the tools we have built with existing large collections.
3) Deploying the MONK tools in a distributed environment that would allow scholars to do text-mining across multiple large collections.

* The actual testbed is upward of 100 million words

We requested funding for the first and second phases, and estimated that the third phase was beyond scope for this round of funding, though outcomes in this round should provide a use-case for projects interested in the issues involved in distributed text-mining.

Project participants included:

University of Alberta:

- Matt Bouchard
- Carlos Fiorentino
- Piotr Michura
- Mike Plouffe
- Milena Radzikowska
- Bernie Roessler
- Stan Ruecker+
- Kirsten Uszkalo
- Cheryl Wilkinson

University of Illinois Urbana-Champaign:
- Amit Kumar
- John Unsworth+
- Xin Xiang

University of Maryland:
- Tanya Clement
- Anthony Don
- Matthew Kirschenbaum+
- Greg Lord
- Catherine Plaisant+
- Martha Nell Smith

McMaster University:
- James Chartrand
- Andrew MacDonald
- Stefan Sinclair

National Center for Supercomputing Applications
- Loretta Auvil
- Bernie A'cs
- Duane Searsmith

University of Nebraska, Lincoln:
- Brian Pytlik Zillig
- Steve Ramsay+
- Sara Steger (University of Georgia)

Northwestern University:
- Philip "Pib" Burns
- Martin Mueller
- John Norstad
- Joe Paris
- Bill Parod
- Bob Taylor

+ denotes working group ("cell") leader

II. Progress achieved and challenges encountered since the last reporting period:

Overview:

The fullest source of information about the MONK project is its public web site, which can be found at http://monkproject.org/. A version of this report will be posted there, and it already contains downloadable software, downloadable data sets, running versions of software, documentation for users and developers, tutorials, and a complete snapshot of the wiki that project participants used to communicate during the course of the last two years.

The texts used in MONK come from a variety of archives that were encoded by libraries following the TEI's P-4 Guidelines. They add up to a corpus of ~2,500 texts (~150 million words) and can be described as an "L-shaped corpus," where the horizontal leg provides coverage across multiples genres in the century from the birth of Queen Elizabeth to the death of King James (1533-1625), while the vertical leg provides coverage of one genre, fiction, across four centuries.

For users of public domain materials, MONK provides quite good coverage of 19th century American fiction, downloadable as TEI P-5 files, with or without part-of-speech annotation, or available for exploration in the user interfaces developed by the MONK project. The full corpus will be accessible only to CIC institutions, or possibly other universities that are subscribers to the Text Creation Partnership and Chadwyck-Healey databases, at least until the middle of the next decade, when the TCP texts will pass into the public domain. At that point, publicly available texts of reasonably high quality will include just about any text of English letters before 1800 that has ever been of interest to scholars.

The table below describes the collections that make up MONK, and gives summary information about the number of works each collection contains, the number of authors represented by those works, and the number of words in the collection—as well as some information about access restrictions on each collection.

Collection	Works	Authors	Words	availability
DocSouth	113	68	8.6 million	public
Early American Fiction	111	16	5.2 million	public
EEBO	691	281	39.4 million	restricted until after 2015
ECCO	1077	196	34.2 million	restricted until after 2015
19th century fiction	250	102	39.4 million	restricted
Shakespeare	42	1	0.9 million	public
Wright American Fiction 1850-75	301	159	23.5 million	public
Total	2585	806	151.5 million	

The 2,500 source files for MONK add up to just over a gigabyte. The linguistically annotated files take up 26 gigabytes. The MySQL databases with its indexes and precomputed data takes up 180 gigabytes. The MONK datastore runs on a fairly ordinary server that costs about $6,000.00.

MONK components and architecture:

MONK consists of a datastore, middleware, an analytics engine, and various user-interfaces, of which the MONK Workbench is the most developed. The MONK project also spent time on some related proof-of-concept work (like faceted browsing for selecting worksets from large collections, or using Zotero to pass those collections into the Workbench). The datastore was produced by an ingest process that used XSL routines collectively referred to as Abbot, a part-of-speech tagger called Morphadorner, and a database loader called Prior, all of which were developed wholly or in part during the MONK project. MONK middleware handles traffic passing back and forth among the user interface, the datastore, and the analytics engine. The analytics engine is SEASR (the Software Environment for Advancement of Scholarly Research), and it takes information from the user (for example, ratings of texts in a supervised learning scenario), combines that with the actual data from the datastore, and runs user-specified statistical routines (Naïve Bayes, etc.) to produce text-mining results.

Building a MONK Datastore:

Using techniques described in more detail below, the source texts were converted into a common interchange format and were linguistically annotated in a manner that virtually levels orthographic, morphological, and dialectal variance across the texts. The goal in this part of MONK has been to create a document space in which every word or phrase in any document becomes comparable with any word or phrase in any other document and the variables of author, date, genre, place of origin, lexical, grammatical, prosodic, or narratological status. Consistent and unified metadata, including word-level metadata, are the key to a deeper grasp of substantive difference in the underlying texts.

The linguistically annotated texts were next moved into a relational datastore that exposes textual data, metadata, and many precomputed counts of textual objects in a coherently structured 'object model' written in Java. Communication with this object model hapens via a proxy server, which is the gateway through which different user interfaces can approach and explore MONK's query potential.

Data-herding with Abbot

"In theory, there is no difference between theory and practice—but in practice, there is."

--Jan L. A. van de Snepscheut

One of the declared goals of the Text Encoding Initiative has been to create digitally encoded texts that are 'machine-actionable' in the sense of allowing a machine to process the differences that human readers negotiate effortlessly in moving from a paragraph, stanza, scene etc. in one book to a similar instance in another. American university libraries have developed a six-level hierarchy of encoding texts that is theoretically interoperable, but as we discovered very early in MONK, in practice, these texts do not actually interoperate. Encoding projects at Virginia, Michigan, North Carolina, and Indiana certainly share family resemblances, but it is also obvious that in the design of these projects local preferences or convenience always took precedence over ensuring that 'my texts' will play nicely with 'your texts'. And aside from simple interoperability, there is even less affordance for extensibility: none of the archives seriously considered the possibility that some third party might want to tokenize or linguistically annotate their texts.

In fairness to these past practices, though, several points need to be made. The MONK texts come from encoding projects that date back to the nineties, and it would not have been easy for a project director/librarian to imagine that a quite ordinary professor of English could store and manipulate all the TEI archives created at Michigan, Virginia, North Carolina, and Indiana on the quite ordinary computer provided by his university without pushing its limits. Nor was it easy to imagine that from technical perspectives of speed or storage the linguistic annotation of very large corpora would be a relatively trivial task. It is also true that the P4 Guidelines, however variably observed, were a lot better than nothing; and there is the fact that during the MONK project, there was a major version release in the TEI. TEI P5 is the first version to be based on native XML. It is not backwardly compatible and makes more extensive use of general protocols in the XML world. We approached the task of making our texts "MONK compatible" from the perspective of creating a P5-based interchange format that would not only serve our purposes but might become a model for others. (Conceptually, this part of the project is similar to the 'Kernkodierung' or 'base line encoding' in the German TextGrid project).

We called this part of the project 'Abbot'. It involved a variety of shell scripts written by Stephen Ramsay, but at its heart it uses a method developed by Brian Pytlik Zillig involving a set of master XSLT stylesheets that write second-level XSLT stylesheets that, in turn, transform a given text into the MONK version of TEI-P5. MONK's version of TEI-P5 is a close cousin of TEI-Lite. We called it TEI-Analytics (TEI-A for short) to stress the fact that its major goal was to facilitate analytical routines across a variety of corpora. TEI-A incorporates a subset of elements for linguistic annotation and was, we note, responsible for broadening the content model of the <w> element in P5. The TEI-A schema is documented at http://segonku.unl.edu/teianalytics/TEIAnalytics.html

Most of the challenges for Abbot focused on what theologians of an earlier era called 'adiaphora' or 'non-necessaria'—things like the treatment of hyphenated words at the end of a line or page. What are trivia from the reader's perspective are major stumbling blocks in workflows that aim at creating a document space in which texts of different origins can be treated as members of a single corpus.

Linguistic annotation with Morphadorner

After conversion to a TEI-A format texts were linguistically annotated with Morphadorner, a tool developed by Phil Burns, using the NUPOS tag set designed by Martin Mueller. The basic goal here has been to develop a common descriptive framework for written English from Chaucer onward. Annotation with Morphadorner involves the tokenization of a text and the description of every word token in terms of

- its lemma
- its standard spelling
- its part of speech

Thus a form like 'louyth' appears as

<w reg="loveth", lem="love", pos="vvz">louyth</w>

There are several distinct problems that need solving if you want to provide a common metalanguage for a diachronical, dialectically, and generically diverse corpus. Commonly used tag sets (Penn Treebank, CLAWS) assume standardized spelling and use the apostrophe as a token splitter for the possessive case (Mary's) and contracted forms (don't). But before the eighteenth century the apostrophe is not a dependable marker of possessive forms, and the language is full of contracted forms that are not explicitly marked, such as 'nilt' (wilt thou not), 'nas' (was not). In 19th century fiction, contracted dialect forms are often written as a single word (dinna, didna).

In its MONK implementation Morphadorner proceeds on the assumption that the tokenizer should not sunder what the typesetter has joined. Spellings like "can't", 'didna', "nilt", or "th'earth" are treated as single tokens, with the orthography reflecting a perception of linguistic reality that marks some difference, however slight, from the reality reflected in the spellings "the earth" or "did not".

The consequence of this decision is that single tokens may have compound description. The possessive case, however, is treated like a simple case marker, which it historically is. Forms like 'dinna' or "won't" are treated as negative verb forms. Words like 'never', 'nothing', or 'nowhere' also have a negative marker. This has the advantage that the degree of negation becomes an easily searchable phenomenon, whether or not it is expressed in a distinct word.

Morphadorner explicitly marks sentence boundaries, thus allowing for the extraction and analysis of sentences from a corpus. This depends on a distinction between various functions of the period mark—a very tricky task in any form of written English, but particularly hard in Early Modern English with its confusing practices of abbreviation and the uses of the period mark in Roman numerals.

From the morphadorned text to the datastore

The upshot of the previous paragraph is that from the workflow that leads through Abbot to morphadorned files you can construct coherent linguistic corpora of indefinite size and move English texts from the late Middle Ages to our day in the level plane of a single document space where any word(s) here can be compared with any word(s) there. What you do with a set of texts created in this fashion is another question. The MONK datastore is one answer. It takes about 30 hours to build the entire datastore, and given its design of interlocking indexes, to change anything is to rebuild everything. Linguistic annotation is a divisible task, however, in the sense that it can be done text-by-text or it can be distributed across different computers. The Morphadorner program used for MONK handles between 12 and 18 million words per hour per CPU.

To create this datastore, morphadorned texts were ingested into a MySQL database, using an object model written in Java. You could write direct SQL queries against that database, but it is designed to be explored through its Java object model. This object model is very fully documented at http://gautam.lis.illinois.edu:8080/monk/servlet, a test app site that includes a number of example queries. While not designed with an end user in mind, this test site is the best way to find out about the affordances of the datastore, which go considerably beyond the routines that are currently enabled in the user interface (see below).

For each work ingested, the datastore receives two pieces of information: the linguistically annotated text and a kind of "property sheet" that provides information about the author, genre, and circulation date of the work. Information about the author, in turn, includes data about birth, death, and origin. Some of these data can be (and indeed were) extracted from the teiHeader of each work. Some data had to be supplied externally. Bibliographical data in the teiHeader do not give you reliable information about the author's sex, origin or the work's genre or date. For instance, the header for the Jew of Malta tells you correctly that it was published in 1633. But the work dates to ~1590. In MONK every work is assigned a "circulation date," which is the best estimate for the time at which it became available.

The data in this property sheet could (and perhaps should) be integrated into the teiHeader of each work, but for us it was simpler to treat them separately. They govern much of the query potential of the data, and they are the criteria by which users construct work sets for comparison or analysis.

The datastore is most readily seen as an inventory in which every word occurrence is a lowest-level object. It its described in terms of its lemma and part of speech. It inherits the properties of the work of which it is part (e.g. a poem written by a female writer in the 1570's). It inherits some properties of its immediate neighbourhood. If in the XML source its immediate ancestor was an <l> tag it is classified as verse. If not, it is classified as prose. It is XML ancestor was an element like <note>, <speaker>, <front> or the like, it is classified as 'paratext' and excluded from default counts. Thus the count of 'king' in Hamlet includes only the cases where 'king' is spoken by a character. In this inventory a lot of parts are precomputed. A search for 'king' in plays between 1590 and 1600 sums the counts for each play rather than counts each occurrence from scratch. The many 'count objects' in the datastore account for the fact that it is seven times as large as the annotated texts on which it is based.

While many of the searches in MONK are based on a 'bag of words' model in which a text is reduced to an inventory of word tokens with counts, the datastore 'knows' something about a word's neighbours. Linguists have found that the distribution of part-of-speech trigrams across a text tells you much about it. For each work the datastore keeps track of its POS trigrams, just as it keeps track of its lemma bigrams, whether 'in the' or 'beauteous majesty'. Any word in the datastore also knows about its neighbour on the left or right, and it is in principle possible to look for indefinite sequences of spellings or POS tags, but these are not precomputed and therefore take longer to retrieve.

The MONK Workbench and other interface experiments

Some of this query potential is exposed in the current user interface, which is based on a "workbench" metaphor and

- allows for defining and storing 'projects'
- has flexible methods for defining 'work sets', i.e. collections of works or work parts that serves as the objects of analysis
- supports several statistical routines, run through SEASR—in particular Naive Bayes, Naive Bayes with Decision Tree, and Dunning's log likelihood ratio, for comparing and classifying different works or collections.
- allows users to save result sets or export them for use in other environments (Excel, ManyEyes, etc)

The MONK workbench is written in Javascript, with underlying MONK middleware written in Java, and it communicates with a (local or remote) installation of SEASR to run its analytic routines. SEASR and the Workbench both use the MONK middleware to communicate with the datastore, which can also be local or remote.

The Workbench itself is component-based, highly extensible, and well documented, including documentation for component developers and a video tutorial on using the Spket Javascript editor to produce MONK components. Extensive tutorial and help documentation for users of the MONK workbench is available from within the interface or at

http://gautam.lis.illinois.edu/monkmiddleware/public/tutorial/index.html

Reading this documentation would probably be the best way to get an in-depth sense of what the user-interface allows, and comparing the interface *functionality* to the *features* made visible at the test app site

http://gautam.lis.illinois.edu:8080/monk/servlet

would be the best way to get a sense of the potential of the datastore not yet realized in the interface. Alternately, you could experiment with the Workbench itself, at http://gautam.lis.illinois.edu/monkmiddleware/public/index.html, using public domain collections. The full MONK datastore is available but password-protected at http://monk.lis.uiuc.edu/monkmiddleware: once the InCommon integration (described below) is complete, the entire MONK datastore will be available in the Workbench to users at most CIC institutions using their own usernames and passwords, and that facility will be linked at http://monkproject.org/

Other interfaces to the datastore were developed during the course of the MONK project, and those include:

- TeksTale, an interface for fast, unsupervised clustering that allows list-based, graph-based, or tree-based visualizations of results, along with word-clouds to show which words were most determinative in clustering, and a tabular display of word-frequency data, for each cluster. See http://devadatta.lis.illinois.edu:1719/TeksTale/index.action for a live demonstration with public domain collections.
- A Flamenco-based faceted browser for assembling collections, and a Firefox plugin for Zotero that allows Zotero to store those collections and then deliver the collection metadata to MONK as a workset. Flamenco is an open-source faceted browser that was developed at Berkeley and funded by the National Science Foundation; Zotero is an open-source bibliographic tool developed at George Mason and funded by the Andrew W. Mellon Foundation. See http://monk.lis.uiuc.edu/cgi-bin/flamenco.cgi/monkpub/Flamenco for a live demonstration with public domain collections.

III. Significant board, management or staff changes since the last reporting period:

None.

IV. Recent publications, news articles, or other materials related to the grant:

Most importantly, two dissertations that used MONK as a centrally important research tool were successfully defended in May of 2009, one on American literature, by Tanya Clement at the University of Maryland, and the other on British literature, by Sara Steger at the University of Georgia. Beyond that, there is the extensive software and documentation produced in and published by this project, including:

- Software:
    - HTML Search/Browse Access to the MONK Datastore
    - TeksTale: Clustering and Word Clouds (log in with user: guest / password: guest)
    - Flamenco faceted browsing of MONK Collections
    - MONK Project plug-in for Zotero (use with Flamenco to build Zotero collections you can import into MONK as worksets; plug-in ver. 0.1.2 does not work with Zotero ver. 2)
- Downloadable texts, schemas, and source code
- Documentation for:
    - Users of The MONK Workbench (see also these training videos on classification and comparison in the MONK Workbench)

- Developers interested in creating components for the MONK Workbench (and a screencast on Using the Spket editor)
- Abbot
- The MONK datastore
- Morphadorner (also available as a PDF file)

- Javadocs for:
  - The MONK Datastore
  - Morphadorner
  - Workbench JSDoc
- Schema documentation for TEI Analytics

Last but not least, there are the following journal articles, conference papers, and blog posts, listed in rough chronological order, either feature MONK as a tool or engage it as an example and were published since our last MONK report to Mellon:

"Library as virtual abbey"
Robert Fox
OCLC Systems & Services
Volume 24, Issue 2, 2008
DOI:10.1108/10650750810875421

"Visualizing Repetition in Text"
Stan Ruecker, Milena Radzikowska, Piotr Michura, Carlos Fiorentino and Tanya Clement
CHWP A.46, publ. July 2008
http://www.chass.utoronto.ca/epc/chwp/CHC2007/Ruecker_etal/Ruecker_etal.htm

"Late Nights at the Scriptorium: Interim Results from the Interface Cell of the MONK Project"
Sinclair, S., Macdonald, A., Bouchard, M., Plouffe, M., Giacometti, A., Kumar, A., Radzikowska, M., Ruecker, S., Michura, P., Fiorentino, C., Kirschenbaum, M. and Plaisant, C.,
Proceedings of the Canadian Digital Humanities Conference (2008)

TEI-Analytics and the MONK Project
Martin Mueller
TEI Annual Members Meeting, 2008
Kings College, London
http://www.cch.kcl.ac.uk/cocoon/tei2008/programme/abstracts/abstract-169.html

"MONK project expands text analysis online literature archives"
Sara Gilliam
The Scarlet, April 24, 2008
University of Nebraska-Lincoln
http://www.unl.edu/scarlet/archive/2008/04/24/story1.html

"Dozens of Little Radio Stations: Getting Technologies Talking in the MONK Workbench."
Andrew McDonald, Amit Kumar, Matt Bouchard, Alejandro Giacometti, Matt Patey, Milena Radzikowska, Piotr Michura, Carlos Fiorentino, Stan Ruecker, Catherine Plaisant, and Stefan Sinclair.
2008 Chicago Colloquium on Digital Humanities and Computer Science
http://lucian.uchicago.edu/blogs/dhcs2008/schedule/program/session-1/

"'A thing not beginning and not ending': using digital tools to distant-read Gertrude Stein's The Making of Americans"
Tanya E. Clement
Literary and Linguistic Computing 2008 23(3):361-381; doi:10.1093/llc/fqn020

"Digital Shakespeare, or towards a literary informatics"
Martin Mueller
Shakespeare, 1745-0926, Volume 4, Issue 3, 2008, pp 284-301.

"Using the Web as corpus for self-training text categorization"
Rafael Guzmán-Cabrera1, Manuel Montes-y-Gómez, Paolo Rosso and Luis Villaseñor-Pineda
Information Retrieval
Volume 12, Number 3 / June, 2009
DOI10.1007/s10791-008-9083-7
Tuesday, December 23, 2008

"Text-Grid and MONK"
Martin Mueller
DATA: Digitally Assisted Text Analysis, February 9, 2009
http://literaryinformatics.northwestern.edu/?q=node/21

"Have you heard of the MONK Project- for analyzing texts?"
Writing Studies & the University Libraries, February 24, 2009
http://blog.lib.umn.edu/katep/infolit/2009/02/have_you_heard_of_the_monk_pro.html

"TEI Analytics: converting documents into a TEI format for cross-collection text analysis"
Brian L. Pytlik Zillig
Literary and Linguistic Computing 2009 24(2):187-192; doi:10.1093/llc/fqp005

"What's Being Said Near "Martha"? Exploring Name Entities in Literary Text Collections,"
Vuillemot, R., Clement, T., Plaisant, C., Kumar, A.,
Proceedings of IEEE VAST, 2009

"The Story of One: Humanity scholarship with visualization and text analysis,"
Clement, T., Plaisant, C., Vuillemot, R.,

Proceedings of the Digital Humanities Conference (DH 2009).

"DH09 Tuesday, session 3: Use Cases Driving the Tool Development in the MONK Project"
Digilib: The digital library blog at Boston University
http://digilib.bu.edu/blogs/digilib/2009/06/dh09-tuesday-session-3-use-cases-driving-the-tool-development-in-the-monk-project/

Text-Mining and Humanities Research
John Unsworth
Microsoft Faculty Summit, July 2009
Redmond, Washington
http://research.microsoft.com/en-us/um/redmond/events/fs2009/presentations/Unsworth_John_DigitalHumanities.pptx

V. Plans and goals for the future:

Integrating MONK with InCommon

The CIC Library heads have provided MONK with up to $15,000 to effect the integration of the MONK Workbench and Flamenco faceted browser with the InCommon authentication framework that CIC CIOs have recently adopted. InCommon is a shibboleth-based framework for authentication across multiple institutions, and we believe that MONK will be the first library service to be brought up under this framework. This corresponds to one of the stated goals of phase two, so we are glad to report that it will be accomplished soon. This will make it possible for us to provide access to the entire MONK datastore to researchers across the Big Ten, and that research use should, in turn, provide valuable information for librarians, publishers, and the disciplines. MONK co-PIs Martin Mueller and John Unsworth, as well as some library representatives, are scheduled to have a conversation in September with representatives of Gale, the publisher who partners with the University of Michigan on the Text Creation Partnership, to talk about how Gale might support such research use in data communities, or scholarly neighborhoods, and how it might work with scholars and with libraries in the context of this support.

Affordances and limits of the datastore

The datastore has been tested with 2,500 texts adding up to 150 million words. We think it will scale up to 250 million words before running into performance problems. That is a lot of words or not very many, depending on how you look at it. It is little more than a rounding error in terms of what is on Google's servers. But the work of many scholarly communities takes place in much smaller textual neighbourhoods. A fiction corpus of 1001 novels from Sidney's Arcadia to Joyce's Ulysses would add up to about 150 million words. The Chadwyck-Healey English Poetry database has only 90 million words. Every English play from *Gorboduc* to *Juno and the Paycock* that was ever reprinted or attracted some other notice would fit comfortably into this container.

The point of these cases is very simple. If you think of the datastore as a container with certain affordances and then think of an interface that explores all or most of its affordances in a user-friendly manner, there are quite a few scholarly neighbourhoods that can be accommodated generously with particular instances of it.

Error rates in Morphadorner

Any analytical routine performed on an annotated corpus depends on the quality of the underlying data, and users need to have a clear sense of where the errors and how much they matter. POS taggers working with modern English have an error rate of ~ 3%. Morphadorner performs at that level with texts that are like modern English in most regards. The error rate is higher in texts or text regions that contain dialect or unusual orthographic variance.

The accuracy of a POS tagger is critically dependent on the quality of the training data. For the MONK texts we used training data that were derived from the hand-corrected versions of Shakespeare and Spenser. These data, supplemented by various lexical data, were used to tag a dozen 19th century English novels, including *Moby Dick* and *Uncle Tom's Cabin*. Hand-corrected versions of those texts became the training data for tagging the bulk of English and American fiction, as well as the 18th century texts. For the 16th and 17th century texts, the WordHoard training data were supplemented by Mary Wroth's *Urania*, Painter's *Palace of Pleasure*, and North's *Plutarch*.

The further away the test data are from the training the more error-ridden they are likely to be. In the current run, 4600 occurrences of the spelling 'Ile' are erroneous identified as instances of the noun 'isle', when in fact they are a contracted form of "I will". The training data did not include Early modern plays in their original spellings, but they did include 'ile' as a variant spelling of 'isle'. Martin Mueller is currently engaged in a review of the 300 Early modern plays in MONK. This will lead to better training data, and in a second run many errors beyond the plays will be caught. But the identification and correction of error is fundamentally an iterative business. It not easy to decide how bad is 'good enough'. That is a powerful argument for a framework of user-driven error correction. If users care enough and you make it easy for them to spot and report errors, they will fix them. If they don't care, the errors do not matter. This is a matter for future work and future proposals, but MONK provides necessary underpinning for that work.

Future uses linguistically annotated TEI-A files?

The 'morphadorned' TEI-A files were designed as the input for the MONK datastore. But the procedures for generating them have a wider range of applicability, and it is worth sketching future projects that can take them as their point of departure. We can say with some confidence that we have created the groundwork for an 'English Diachronic Annotated Corpus' (EDDAC), a very large and public domain archive of written English from Caxton's Troy book, the first printed book in English (1473) to Joyce's Ulysses

(1922) or beyond if Congress ever touches the sacred date of current copyright.

Opportunities and problems with TCP texts

The foundation of EDDAC would no doubt be the digitized texts in the Text Creation Partnership, which will pass into the public domain at some point in the next decade and will by then include some 40,000 works published in the British Isles or America before 1800. That corpus will include just about any text from before 1800 that has been or is likely to be of more than casual scholarly interest.

We processed 1,800 of the 20,000 or so currently available texts and have probably encountered and solved most of the problems involved in processing the rest, leaving aside a small percentage of outliers that would require special treatment or can be ignored.

While the Text Creation Partnership is a magnificent project, it is also the case that many of the current texts have serious deficiencies. They are full of gaps, words or letters that the transcribers could not read, or were instructed to ignore (languages in non-Roman alphabets). Because of the idiosyncratic and inconsistent treatment of end-of-line hyphens the texts are riddled with words that are wrongly split or wrongly joined. Considered as diplomatic transcription of their sources, the current texts are not nearly as good as they should be. They are obvious candidates for a process of distributed and collaborative data curation. Oddly enough, it is in some ways easier to do this with a linguistically annotated text than with the plain file. Morphadorner, for instance, has a 'vertical' output format in which every word token is surrounded by left and right context, together with the lemma, the POS tag, and a unique sequential identifier that allows you sort and resort the text in various ways, concentrating on incomplete or missing words, parts of speech, etc. 'Error-forcing' techniques of this kind do a good job of identifying and clustering similar types of errors, making their correction easier and more accurate. Northwestern undergraduates who volunteered to correct the particularly error-ridden transcription of Marlowe's *Tamburlaine* had no difficulty deciphering most of the words the transcribers could not read. They took their laptops to a computer lab, looked at the vertical screen on their computer, at the EEBO page image on the lab computer screen, and entered the corrected word in a correction column on their vertical file. This process generates a tuple associating a unique word_id with a particular type of correction. It is not hard to envisage a robust and network-based framework in which thousands of such suggestions for correction lead to substantial improvements in the texts that people care about for one reason or another. In fact, a proof-of-concept development of such a framework, resembling 'community annotation projects' in genome research, is underway at Northwestern. It will use the vertical output format of MorphAdorner with a Django-based interface.

Creating digital editions from 19th and early 20th century OCA texts

For texts after 1800, OCR texts from the Open Content Alliance are very promising candidates for supplementing EDDAC. It is attractive to think of digital surrogates that allow modern users to experience, say, Bleak House in ways that range seamlessly from the page image that is a simulacrum of its original materiality to a 'bag of words' model that highlights distinctive lexical or syntactic qualities of this text when read against a larger corpus.

During a practicum in the spring of 2009, Katrina Fenlon, a graduate student at GSLIS did some interesting experiments with Tim Cole and Martin Mueller. What would it take to convert the 'white space' XML of an OCR text into a TEI-A file that can be linguistically annotated and become part of EDDAC? How much manual checking and tweaking is necessary to produce a structurally sound representation of the text? She thinks the process can be reduced to half an hour, which is not much time for a text that has some value to some users. The very extensive collection of 19th century English fiction in the UIUC library makes an excellent guinea pig for further testing and would supplement the extensive archive of publicly available 19th century American fiction.

Improving the Abbot workflow

If you think of the Abbot workflow as a procedure for converting existing texts to compatible TEI-P5 versions, it will take some additional work. Two examples from the TCP make the case. In the SGML source files the common old spelling of 'the' as a 'y' followed by a superscript 'e' is represented as 'y^e'. An XML transformation changes this to 'y^e'. In the MONK environment that was a typograpical accidental without interest, and we replaced it directly with 'the'. The TCP texts use character entities for early modern brevigraphs, such as '&abper; for 'per', and we resolved those without trace.

The downside of these shortcuts is that you cannot restore the source text. That was not a concern with MONK. But it is a concern if you think of an archive of compatible texts that are subject to continuing data curation. Whatever changes are made need to be made to the texts that are considered the masterfiles. It is not especially difficult to break down the process of creating TEI-A files, keep its various stages, and apply linguistic annotation to a version of the file that can be traced without loss to the source file. Fixing this problem is a matter of days or weeks rather than weeks or months.

Improving Morphadorner

Morphadorner is very fast: you could theoretically process the entire TCP-EEBO corpus in five hours with five ordinary dual-core desktop machines. You would not want to do this without spending considerably more time on creating more customized training data that would lower the error rate.

In a thoughtful comparative evaluation of a variety of NLP tools, Matthew Wilkens at Rice concluded that Morphadorner is the tool of choice if you want to annotate diachronic literary corpora. It is nice to read this since it was designed precisely for that purpose. Further improvements are largely a matter of creating customized training data—an intrinsically time-consuming task. Some thought could be given to slimming down the output. Morphadorner's standard output is quite verbose. Although there are some options of abbreviated output, there may be some ways of slimming it down further without loss.

A web-based workflow for selecting and ingesting collections

The work done in this project in creating a Flamenco-based faceted browser and Firefox-Zotero plugin are two first steps in the direction of allowing users to assemble the collections with which they want to work. As we move to larger and larger scale in the digital library, it is not going to be possible to have all material processed in advance, as they are in the MONK datastore. Instead, data communities will need to support the ability to select works of interest and submit them to something like the MONK ingest routine, to prepare them for interactive exploration. For uses such as MONK was designed, that ingest routine will need to allow users to check output at various stages of the process, intervene to make adjustments or corrections (to Abbot), choose or develop appropriate training sets (for Morphadorner), and build their own datastores. We are interested in developing this workflow in a web-based interface that would be necessarily modular, since different users might want different tools or have different requirements at different points in the ingest process. We think such web-based workflow will be critical cyberinfrastructure when it comes to working with very large collections.

MONK, HathiTrust, the Google Research Corpus, Bamboo

Speaking of very large collections, MONK co-PI John Unsworth is a member of the recently appointed HathiTrust Research Committee, which is discussing MONK as an example of a research service that might be provided in conjunction with HathiTrust materials. The HathiTrust is a shared digital repository for materials being returned to CIC and California libraries who participate in the Google Books project and in other digitization projects. One outcome of these discussions will be a proposal to establish a research facility for working with the Google research corpus, assuming that the final disposition of Google's legal case with publishers retains the requirement that Google will fund such a facility. Experience from all aspects of the MONK project is already proving useful in the Research Committee's discussions, and MONK will benefit from the discussions as well. Finally, MONK and SEASR have been presented and discussed as examples of tools and services that could be part of Bamboo, the Mellon-funded cyberinfrastructure project. Also included in the Bamboo discussions have been representatives of Centernet, a network of digital humanities centers—the same kind of centers that have been the audience for SEASR's "train-the-trainers" educational efforts. We see these various efforts as converging, in the not very distant future, in a partnership that involves MONK (and many other tools for text analysis), SEASR, Bamboo (possibly in the form of a virtual appliance), around a research corpus of Google and other materials, with digital humanities centers as trusted and authenticated institutional partners, and supercomputing centers as key providers of high-performance computing facilities.

VI. Intellectual property:

MONK software source code is provided for download at http://monkproject.org All of the software except that produced exclusively at Northwestern University comes with the following license terms:

Developed by: The MONK Project
McMaster University National Center for Supercomputing Applications Northwestern University University of Alberta University of Illinois at Urbana-Champaign University of Maryland at College Park University of Nebraska at Lincoln
http://www.monkproject.org

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal with the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimers.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimers in the documentation and/or other materials provided with the distribution.
- Neither the names of the MONK project, nor the names of its contributors may be used to endorse or promote products derived from this Software without specific prior written permission.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE CONTRIBUTORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS WITH THE SOFTWARE.

Software produced exclusively at Northwestern University carries this license:

Copyright (c) 2008, 2009 by Northwestern University.
All rights reserved.
Developed by:
Academic and Research Technologies
Northwestern University
http://www.it.northwestern.edu/about/departments/at/

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal with the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimers.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimers in the documentation and/or other materials provided with the distribution.
- Neither the names of Academic and Research Technologies, Northwestern University, nor the names of its contributors may be used to endorse or promote products derived from this Software without specific prior written permission.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE CONTRIBUTORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS WITH THE SOFTWARE.