11-2017

Separating Markup from Text

Ronald I. Greenberg
Loyola University Chicago, Rgreen@luc.edu

George K. Thiruvathukal
Loyola University Chicago, gthiruv@luc.edu

Separating Markup from Text

Chicago Colloquium on Digital Humanities & Computer Science

November 19, 2017

Ronald I. Greenberg and George K. Thiruvathukal
rig@cs.luc.edu and gkt@cs.luc.edu

Digital Humanities Texts

Modern digital texts, even with classical/historical content, often come in complicated formats (some more complicated than others) such as HTML, DOC, DOCX, PDF, and XML (DocBook, TEI) to indicate both display specifications such as

- Font (e.g., Arial, Times Roman, etc.)
- Typeface (e.g., bold, italic, etc.)
- Typesize
- Color

and logical organizational elements such as

- Title
- Heading
- Abstract
- Stage instruction

Often, classical texts are augmented with additional content, for example, line numbers.

Computerized Analyses of Original Text

Markup can interfere.

So just strip it out, right?

Not so simple: Existing tools produce inconsistent results and especially will have trouble distinguishing auxiliary content such as line numbers or an editor’s introduction.

Standard Unix Tools Removing HTML Markup: Some Issues

- `nohtml`
  - Leaves content in place from "`<style>`" and "`<script>`" elements.
  - Retains entities such as "` `".
  - Strips content looking akin to an HTML element even if tag name is illegal for HTML and even XML.
- `html2text`
  - Inserts its own stuff to simulate display, e.g., a line of "`=`"s for "`<hr>`", "`*`" for bullets, "`alt`" content for images, etc.
- Sometimes HTML files have syntax errors that browsers can handle in a passable way but that will confuse programs that strip markup.
- As previously noted, line numbers, editorial emendations, etc. cause difficulties.

Example http://www.folgerdigitaltexts.org/html/Ham.html

Last bit, simplified with respect to non-printing characters in the files

nohtml

```
FTLN 4166 Becomes the field but here shows much amiss.

FTLN 4167 Go, bid the soldiers shoot.

They exit, marching, after the which, a peal of ordnance are shot off.
```

html2text

```
FTLN 4166 Becomes the field but here shows much amiss.

FTLN 4167 Go, bid the soldiers shoot.

They exit, [text from the Folio not found in the Second Quarto]marching, after the which, a
peal of ordnance are shot off.[text from the Folio not found in the Second Quarto]

================================================================================
```

Possible Fixes

- Adopt and enforce strict conventions for adding markup to text.
  - Good luck!
- Store markup separately from text.
  - Still need conventions but perhaps a better chance of retaining inviolate original text.
  - Seems like an easier approach to provide various types of markup with the same base text.
  - Need a mechanism to quickly combine markup with text for pretty display.
  - Desirable to provide semi-automated tools for separating markup from text in the existing corpus of digital files with embedded markup.

A Prototype

Format of each line of markup file is: `|charnum|markup|type|eltnum`

where

- `charnum` is the number of the text character after which the markup appears
- `markup` is the markup
- `type` is one of:
  - `B`: beginning (start) tag (e.g., `div`, `p`, `a`)
  - `E`: end tag (e.g., `/div`, `/p`, `/a`)
  - `V`: void (empty) tag (e.g., `hr`, `br`, `img`)
  - `M`: markup element; i.e., content between start and end tags is all markup (e.g., `head`, `script`, `noscript`, `style`)
- `eltnum` is a count of which markup element we are processing

Why This Format?

- Very general format that can be used for HTML or any other sort of markup.
- Very easy to use this to merge the markup into the text. (<100 lines of C code.)
- Easy to pair beginning and end tags if desired and easier to work with than if we would try to store tags in pairs.

Example: End of hamlet.txt2html

```
240643|<a name="line-SD5.2.449.1">|B|27956
240644|</a>|E|27956
240644|<span class="stage right">|B|27957
240644|<span id="line-SD 5.2.449.1" title="SD 5.2.449.1">|B|27958
240655|<img src="fdt-texta-l.png" class="imgTextX" alt="text from the Folio not found
in the Second Quarto" title="text from the Folio not found in the Second Quarto">|V|27959
240663|</span>|E|27958
240663|<span id="line-SD 5.2.449.1" title="SD 5.2.449.1">|B|27960
240691|</span>|E|27960
240691|<br>|V|27961
240693|<span id="line-SD 5.2.449.1" title="SD 5.2.449.1">|B|27962
240715|<img src="fdt-texta-r.png" class="imgTextX" alt="text from the Folio not found
in the Second Quarto" title="text from the Folio not found in the Second Quarto">|V|27963
240715|</span>|E|27962
240715|</span>|E|27957
240715|<br>|V|27964
240717|<br>|V|27965
240719|<br>|V|27966
240721|</div>|E|24808
240723|</div>|E|22594
240725|<hr>|V|27967
240727|<br>|V|27968
240729|</div>|E|11
240731|</div>|E|10
240733|</body>|E|4
240735|</html>|E|2
```

Prototype to Separate HTML (or XML) Markup from Text

- With some simplifying assumptions about HTML/XML file (e.g., proper nesting, and every “<” begins a tag ending at next “>”, less than 250 lines of C code).
- Newlines, spaces, punctuation treated as text, but could also treat as markup.
- Has been used for successful “round trip” from, e.g., Folger `hamlet.html` to `hamlet.txt` (text) and `hamlet.txt2html` (markup) and then back to `hamlet.html`.
- Other than treating entities and markup elements in a better way than the `nohtml` program, however, the `.txt` file looks similar, e.g., with line numbers and editorial comments included.

Better Separation of Markup from Existing Digital Files

We envision a tool incorporating standard technologies used by web browsers to parse HTML/XML documents that can present the user with a browser display and easily-used options to flag various types of markup elements as belonging to the true base text or not, and then to perform the separation.

Markup Transformation

We also envision this approach providing an environment in which it is easy to transform one type of markup to another, e.g., HTML to DOCX, etc., and to offer the user the choice of applying whichever type of markup is desired at a given time.