How do I represent my corpus?

Some options:
- Text document
- MS word document
- XML
  - TIGER XML
  - Nite XML
  - EXMARALDA
  - PAULA

Markup vs. Annotation

- Generally speaking: annotation is markup
  - Markup: adds extra information to text
  - Linguistic annotation: adds linguistic information to the text, uses markup
  - Non-linguistic information often added in markup, e.g., meta-data
    - i.e., information about the text: author, version, date, information about author, encoding

Corpus mark-up schemes

Meta-data systematically needs to be kept separate from the corpus data

A general idea is to use attributes and corresponding values
- e.g., attribute author has a value William Shakespeare
- We’ll see this encoded in XML in ways such as:
  - `<... author="William Shakespeare" ...>`
  - `<author name="William Shakespeare">`

Text Encoding Standards

- TEI: Text Encoding Initiative
- CES: Corpus Encoding Standard
- XCES: XML version of CES
  - If you annotate a corpus, make it as XCES conform as possible!

Text Encoding Initiative (TEI)

TEI-compliant documents are broken into 2 parts:
1. Header = information about the document
   - File description (<fileDesc>): bibliographic information
   - Encoding description (<encodingDesc>): relationship to the source
   - Text profile (<profileDesc>): non-bibliographic information (setting of the text, languages used, etc.)
   - Revision history (<revisionDesc>): changes made
2. Body = the original text itself

TEI documents can be encoded in different mark-up languages, e.g., SGML and XML
Mark-up languages

Mark-up languages generally have the following aspects

- Tags, which come in both start (e.g., <s>) and end (e.g., </s>) forms
- Attributes and values, encoded within the start tag (e.g., <s num="1">)

<w POS="AT0">the</w>

XML

- eXtensible Markup Language
  - In contrast to HTML, XML does not have built-in "meaning" for labels: you must define your own tags!
  - XML document consists of two parts: DTD & real document
    - Document Type Definition (DTD): defines structure of document
    - XML documents can be validated, i.e. checked against rules in DTD

(MD: An earlier version of these slides had contrast as a typo. I loved it so much I added this note.)

XML Tutorials

- http://www.w3schools.com/xml/note.xml
- http://www.xmlfiles.com/xml/
  - ...

What does an XML file look like?

A simple example:
http://www.w3schools.com/xml/note.xml

A more involved example:
http://www.w3schools.com/xml/simple.xml

Can you figure out the structure?

CES: Corpus Encoding Standard

- TEI is for any text document and, as such, defines approximately 450 elements
  - CES simplifies this system in order to encode language corpora
- CES covers:
  - document-wide mark-up
  - gross structural mark-up, for structural units of text
  - mark-up for sub-paragraph structures
- CES documents can be validated at metalanguage, syntactic, or semantic levels

Encoding

- XML defines the **structure** of a corpus file
  - The text/character **encoding** (e.g., UTF-8) defines the underlying representation of the text
Information on a computer is stored in **bits**.
- A bit is either on (= 1, yes) or off (= 0, no).
- A list of 8 bits makes up a **byte**, e.g., 01001010
- With 8 bits (a single byte), you can represent 256 different characters. Why would we want so many?
  - If you look at a keyboard, you will find lots of non-English characters.
  - With 256 possible characters, we can store every single letter used in English, plus all the things like commas, periods, space bar, percent sign (%), back space, and so on.
- Old encoding system: ASCII

Problems with having multiple encoding systems:
- Conflicts: two encodings can use the same number for two different characters and use different numbers for the same character.
- Hassle: have to install many, many systems if you want to be able to deal with various languages

Unicode tries to fix that by having a single representation for every possible character.

“Unicode provides a unique number for every character, no matter what the platform, no matter what the program, no matter what the language.”
(www.unicode.org)

Version 6.2 has codes for 110,117 characters from alphabets, syllabaries and logographic systems.
- Uses 32 bits – meaning we can store $2^{32} = 4,294,967,296$ characters.
- 4 billion possibilities for each character? That takes a lot of space on the computer!

Unicode has three versions
- UTF-32 (32 bits): direct representation
- UTF-16 (16 bits): $2^{16} = 65536$
- UTF-8 (8 bits): $2^8 = 256$

How is it possible to encode $2^{32}$ possibilities in 8 bits (UTF-8)?
- Several bytes are used to represent one character.
- Use the highest bit as flag:
  - highest bit 0: single character
  - highest bit 1: part of a multi byte character
- Nice consequence: ASCII text is in a valid UTF-8 encoding.