Text Encoding and Semantic Representation

XML / TEI (2)

marilena.daquino2@unibo.it | https://github.com/marilenadaquino/tesr_dhdk

Outline

Named entities

Facsimiles

Critical apparatus

Homework, all good?

Quiz time!

Did you answer all the questions? Anything to discuss?

https://forms.gle/DNt9cPCxuUZzwxue8

Named entities

People, places, dates
Authority files

Named entities

People, places, and others

So far we have seen elements representing **metadata** and the **logical structure** of a text. TEI also provides elements to record **occurrences of real-world entities** (or abstract concepts) in the text. See the guidelines

Why is this useful? **indexes** of annotated entities can be created automatically (people, dates, places, etc.) and XML documents can be interrogated with appropriate languages to return lists of entities, their occurrence, etc.
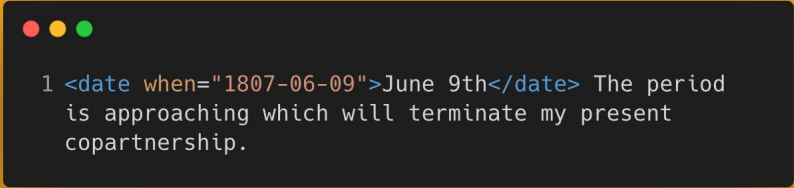
Named entities

Dates

Dates are often referenced in natural language. Nonetheless, a machine readable version of the date can be recorded via the element <date> and its attribute @when, using a standard format yyyy-mm-dd.

If the date is uncertain, other attributes can be used to frame the period.

```
1 <date when="1807-06-09">June 9th</date> The period
  is approaching which will terminate my present
  copartnership.
```

@when	supplies the value of the date or time in a standard form, e.g. yyyy-mm-dd.
@notBefore	specifies the earliest possible date for the event in standard form, e.g. yyyy-mm-dd.
@notAfter	specifies the latest possible date for the event in standard form, e.g. yyyy-mm-dd.
@from	indicates the starting point of the period in standard form, e.g. yyyy-mm-dd.
@to	indicates the ending point of the period in standard form, e.g. yyyy-mm-dd.

Named entities

People

While an element <name> exists and can be used to annotate any proper name, <persName> specifies the name belongs to a person.

It accepts several children elements, such as <forename>, <surname> and <roleName> to further distinguish substrings.

```
1  <persName>
2  <forename type="first">Franklin</forename>
3  <forename type="middle">Delano</forename>
4  <surname>Roosevelt</surname>
5  </persName>
```

Named entities

Organisations

Similarly to people, organisations can be annotated via <orgName> and substrings can be further defined.

```
1 <orgName type="acronym">BBC</orgName>
```

```
1 <orgName type="regional">
2 <orgName>IBM</orgName>
3 <country>UK</country>
4 </orgName>
```

Named entities

Places

Places have dedicated elements to record names <placeName>, location and coordinates <geo>.

```
1 <placeName>Rome</placeName>
2 <location>
3   <geo>41.891775, 12.486137</geo>
4 </location>
```

Named entities

Identifiers and referents

In a text, several different strings may refer to the same real-world entity. To **preserve the relation between occurrences and the related entity**, identifiers can be used to bind them.

```
1 We went on holiday in <placeName>Rome</placeName>
2 ...
3 Despite it is a beautiful <placeName>city</placeName>, it is
  rather chaotic.
```

Named entities

Identifiers and referents

The attribute **@key** is used to record an externally-defined string identifying the referent (e.g. the english name of a place in an Atlas).

```
1 We went on holiday in <placeName key="Rome">Rome</placeName>
2 ...
3 Despite it is a beautiful <placeName key="Rome">city</placeName>, it
  is rather chaotic.
```

Named entities

Identifiers and referents

Projects may use @key to refer to the same terms or not, therefore the attribute **@ref** or **@sameAs** are preferred instead.

@ref records one or more **URLs** of web resources describing the entity at end, e.g. Wikipedia entries.

@sameAs records **URIs** identifying the entity.

Multiple values are separated by white spaces.

```
1 We went on holiday in <placeName ref="https://en.wikipedia.org/wiki/Rome
  https://viaf.org/viaf/251380480">Rome</placeName>
2 ...
3 Despite it is a beautiful <placeName ref="https://en.wikipedia.org/wiki/Rome
  https://viaf.org/viaf/251380480">city</placeName>, it is rather chaotic.
```

Named entities

URIs

URI stands for **Uniform Resource Identifier.** Similar to URL (Uniform Resource Locator), URIs identify something. A URL identifies a web resource, while a URI identifies a real-world entity (or concept), which can be described in several web resources (in turn identified by URLs).

A URL tells us where a web resource is located (i.e. on which server) but not what it is about

A URI tells us what the identifier is about (e.g. a person) and where it is located.

Named entities

Authority control lists

In some cases, a URI identifies both a real-world entity (e.g. a place) and a web resource describing that entity. Authority control files mint such type of URIs so that projects can univocally refer to those to identify the entities, regardless of language, homonyms, etc.

Some well-known authorities are **VIAF, Wikidata, geonames, Getty vocabularies** (AAT, ULAN, TGN)

E.g. **Rome**: https://viaf.org/viaf/251380480/ , http://www.wikidata.org/entity/Q220 , https://www.geonames.org/3169070 , https://vocab.getty.edu/page/tgn/7000874

Named entities

Why do authorities matter?

Keeping track of identifiers provided by central authorities in your data is fundamental to further process data in future scenarios.

1. **Facilitate data exchange.** If you merge your edition with that of others, it will be easier to recognise which people, places, etc. are shared across editions.
2. **Enable data enrichment.** You can query central authorities to get more information about those entities (e.g. biographies) and integrate that information in your data or application (e.g. a website)

Named entities

Local authority files

Recording URIs everytime an occurrence of a name appears is cumbersome and time-consuming.

A good practice is to create a local authority file in the <teiHeader> of the very same XML file of your edition (or in another one).

Authority lists can be created for:

- People (listPerson),
- Places (listPlace),
- Organisations (listOrg),
- Events (listEvent),
- Relations (listRelations),
- Bibliographic References (listBibl)

Named entities

Example

Notice the local identifier MG first defined in the element person/@xml:id and then recalled in @ref, preceded by #.

This linking mechanism is used also in other contexts, in XML/TEI, and in HTML.

```xml
 1 <teiHeader>
 2   ...
 3   <profileDesc>
 4     <particDesc>
 5       <listPerson type="cited-author">
 6         <person xml:id="MG" sameAs="http://viaf.org/viaf/97859168
 7                               http://it.dbpedia.org/resource/Michail_Gorbačёv">
 8           <persName xml:lang="it">Michail Gorbačev</persName>
 9           <persName xml:lang="en">Michail Gorbačev</persName>
10           <birth when="1931"/>
11         </person>
12       </listPerson>
13     </particDesc>
14   <profileDesc>
15   ...
16 </teiHeader>
17 <text>
18   ...
19   <p>(Crisi del comunismo. <lb/>
20     <persName ref="#MG">Gorbaciov</persName> [<persName ref="#GGC">Cesare</persName>])
21   </p>
22   ...
23 </text>
```

https://projects.dharc.unibo.it/bufalini-notebook/static/bufalini_quaderno.xml

Exercise

Search for people in a XML file

Open the XML/TEI file https://projects.dharc.unibo.it/bufalini-notebook/static/bufalini_quaderno.xml

Search for William Shakespeare (#WS)

- Under which type of listPerson is he listed?

- What links to authorities are provided?

- How many and which works of his are mentioned in the body of the text?

Facsimiles

IIIF

JSON

Facsimile

Digital facsimile

Usually, texts encoded in XML/TEI are the result of a transcription, either from an original analog source or from a digital version of it.

A **digital facsimile** is a collection of images, usually 1 for each page, of the source and their metadata.
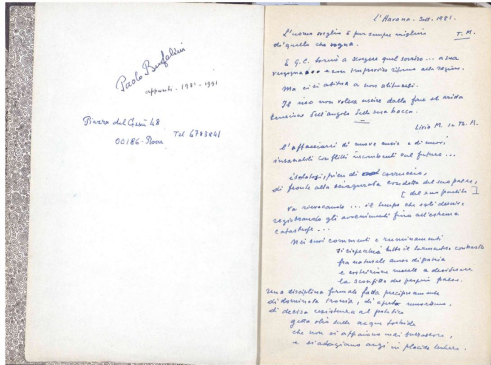
Facsimile

Pagination

When transcribing the text, the editor keeps track of the original pagination using the milestone element <pb> to identify the page breaks.

The attribute @facs records the path to the file representing the page at hand.
Guidelines

```xml
<TEI xmlns="http://www.tei-c.org/ns/1.0">
 <teiHeader>
<!--...-->
 </teiHeader>
 <text>
  <pb facs="page1.png"/>
<!-- text contained on page 1 -->
  <pb facs="page2.png"/>
<!-- text contained on page 2 -->
 </text>
</TEI>
```

Facsimile

Manage images

Usually images are served along with the files of the digital edition of the text.
When working **locally**, images are saved in the same folder of the XML files. Similarly, when **publishing online** a digital edition, images are often stored together with the HTML of the text.

Sometimes, images are stored in **institutional repositories**, maybe more than one, they are in different formats, shared with different licenses, and cannot be re-published by third parties (e.g. in my digital edition).

Facsimile

IIIF

IIIF is a standard that universalizes the way archived images are described and retrieved on the web.

It is the "definitive solution" to publish images via dedicated server applications to store and serve **images with metadata** and that allow users to access **advanced features** when browsing (e.g. cropping, zooming, collate images, cite, annotate, access metadata).

International
Image
Interoperability
Framework

Facsimile

IIIF: is a digital repository

IIIF community has created a few server applications to store images and their metadata, according to IIIF specifications.

Here a list of IIIF servers: https://iiif.io/get-started/image-servers/

Disclaimer

You are <u>not</u> required to install a IIIF server for your project

We are going to see how to query and use images published via a IIIF server. If applicable, you can use images and data published via a IIIF provider in your project.

However, you are not required to publish your images/data via IIIF.

Facsimile

IIIF: is an image API

The image API (Application Programming Interface) is a program that interacts with the IIIF server and **serves images** to users or external applications that request the image URL.

The image URL is composed of a few parameters that can be modified by the user/application, e.g.
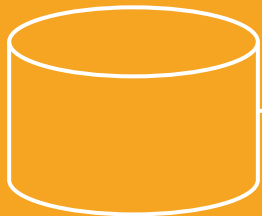
{scheme}://{server}{/prefix}/{identifier}/{region}/{size}/{rotation}/{quality}.{format}
https://stacks.stanford.edu/image/iiif/hd288rd1737%252F0090_496-34/full/full/0/default.jpg

https://	{scheme}://
stacks.stanford.edu/	{server}
image/iiif/	{/prefix}/
hd288rd1737%252F0090_496-34/	{identifier}/
full/	{region}/
full/	{size}/
0/	{rotation}/
default	{quality}
.jpg	.{format}

Facsimile

IIIF: image API

IIIF server
stores the images and serves them on the web

IIIF images API
A way to serve images on demand

Image URL scheme
{scheme}://{server}{/prefix}/{identifier}/{region}/{size}/{rotation}/{quality}.{format}

Browser / Application
Asks for an image via its URL.

https://iiif.bodleian.ox.ac.uk/iiif/image/497d8383-1580-419b-91bf-20452f3fdbe9/full/full/0/default

Facsimile

IIIF: image API

https://iiif.bodleian.ox.ac.uk/iiif/image/497d8383-1580-419b-91bf-20452f3fdbe9/full/full/0/default
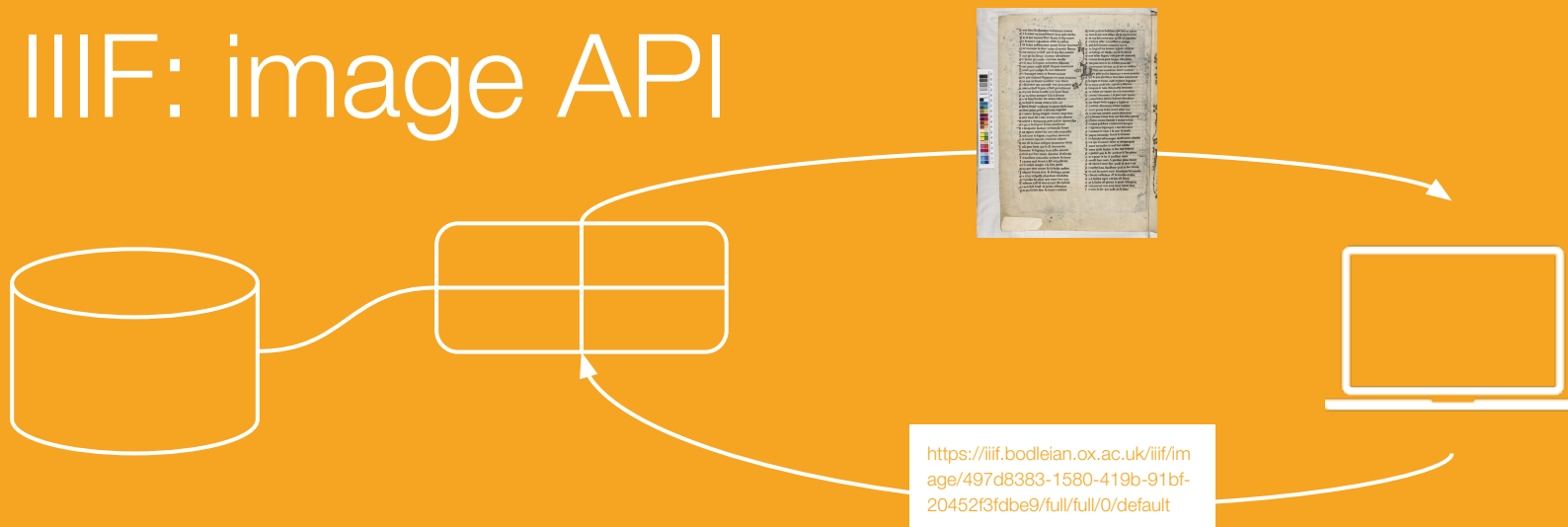
The user / application asks for the URL to the server, which communicates with its API, reads the parameters in the URL (e.g. quality, format) and returns the image file.

Facsimile

IIIF: image API

The image API can also return a **JSON-LD file** including technical information about the image.

{scheme}://{server}{/prefix}/{identifier}/info.json

See docs:
https://iiif.io/api/image/3.0/#51-image-information-request

```
1  {
2    "@context" : "http://iiif.io/api/image/2/context.json",
3    "@id" : "http://iiif.thembi.me/10r/",
4    "protocol" : "http://iiif.io/api/image",
5    "width" : 1944,
6    "height" : 2592,
7    "sizes" : [
8      { "width" : 121, "height" : 162 },
9      { "width" : 243, "height" : 324 },
10     { "width" : 486, "height" : 648 }
11   ],
12   "tiles" : [
13     { "width" : 256, "height" : 256, "scaleFactors" : [ 1, 2, 4, 8, 16 ] }
14   ],
15   "profile" : [
16     "http://iiif.io/api/image/2/level1.json",
17     { "formats" : [ "jpg" ],
18       "qualities" : [ "native","color","gray" ],
19       "supports" : ["regionByPct", "regionSquare", "sizeByForcedWh",
   "sizeByWh","sizeAboveFull","rotationBy90s","mirroring"] }
20   ]
```
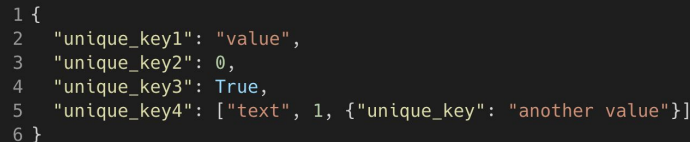
JSON

A lightweight format

JavaScript Object Notation (JSON) is a lightweight format for data-interchange. Like XML, JSON is a standard language for **exchanging data on the web**, which is meant to be easy to read for both humans and machines, and that it is not meant to be visualised on the web (like e.g. HTML).

NB. Does not it resemble a Python dictionary?

```json
1 {
2   "unique_key1": "value",
3   "unique_key2": 0,
4   "unique_key3": True,
5   "unique_key4": ["text", 1, {"unique_key": "another value"}]
6 }
```

JSON

JSON-LD

It's a particular type of JSON.

The key **@context** includes a reference to an online schema with rules on which keys and types of values to include in the file (like XSD and DTD would work for an XML file). The context.json file is yet another JSON file.

```
 1 {
 2   "@context" : "http://iiif.io/api/image/2/context.json",
 3   "@id" : "http://iiif.thembi.me/10r/",
 4   "protocol" : "http://iiif.io/api/image",
 5   "width" : 1944,
 6   "height" : 2592,
 7   "sizes" : [
 8     { "width" : 121, "height" : 162 },
 9     { "width" : 243, "height" : 324 },
10     { "width" : 486, "height" : 648 }
11   ],
12   "tiles" : [
13     { "width" : 256, "height" : 256, "scaleFactors" : [ 1, 2, 4, 8, 16 ] }
14   ],
15   "profile" : [
16     "http://iiif.io/api/image/2/level1.json",
17     { "formats" : [ "jpg" ],
18       "qualities" : [ "native","color","gray" ],
19       "supports" : ["regionByPct", "regionSquare", "sizeByForcedWh",
   "sizeByWh","sizeAboveFull","rotationBy90s","mirroring"] }
20   ]
```

JSON

JSON-LD

PS/ In the next classes we will see that JSON-LD is a the JSON syntax to record graph data.

```json
1 {
2    "@context" : "http://iiif.io/api/image/2/context.json",
3    "@id" : "http://iiif.thembi.me/10r/",
4    "protocol" : "http://iiif.io/api/image",
5    "width" : 1944,
6    "height" : 2592,
7    "sizes" : [
8       { "width" : 121, "height" : 162 },
9       { "width" : 243, "height" : 324 },
10      { "width" : 486, "height" : 648 }
11   ],
12   "tiles" : [
13      { "width" : 256, "height" : 256, "scaleFactors" : [ 1, 2, 4, 8, 16 ] }
14   ],
15   "profile" : [
16      "http://iiif.io/api/image/2/level1.json",
17      { "formats" : [ "jpg" ],
18        "qualities" : [ "native","color","gray" ],
19        "supports" : ["regionByPct", "regionSquare", "sizeByForcedWh",
  "sizeByWh","sizeAboveFull","rotationBy90s","mirroring"] }
20   ]
```

Exercise

Manipulate a IIIF image URL

Consider the following image

https://images.lib.cam.ac.uk/iiif/MS-ORCS-00001-00002-000-00001.jp2/full/full/0/default.jpg

- Change the parameter in the URL to rotate the image enough to read the text in the frame
- Change the URL to retrieve the info.json file

Facsimile

IIIF: a presentation API

Another API that packages images and metadata, so users know the origin, title of the image, and even what page of a book the image was from, likely in conjunction with the Image API.
In particular, IIIF images come with **manifests**, that is, other JSON files including structured information about an object's metadata (e.g. title, author, date of a book), how to access the image(s) within the object, and the order in which they should appear.

`{scheme}://{host}/{prefix}/{identifier}/manifest`

See the docs: https://iiif.io/api/presentation/2.0/#primary-resource-types-1

Facsimile

IIIF: a manifest

A manifest.json file provides:
- **descriptive information** about an object, e.g. a book,
- the **sequence** (i.e. the list) of objects that compose the main object and in which order these should be visualised by a browsing application
- the **canvases**, like a page or view
- the **content** actually displayed in the canvas, and optionally the **annotations** attached to it

See an example: https://iiif.io/api/presentation/2.1/#c-example-manifest-response

Facsimile

IIIF: a viewer

There are many IIIF viewers. IIIF-compatible viewers generally allow users to **zoom, rotate, and resize image** objects, and play audio/visual files. Some allow **annotation** with text, audio, location, and more. Others allow **comparison** of objects from a single collection side-by-side (or even objects from multiple collections if the object's Manifest is made available to users).
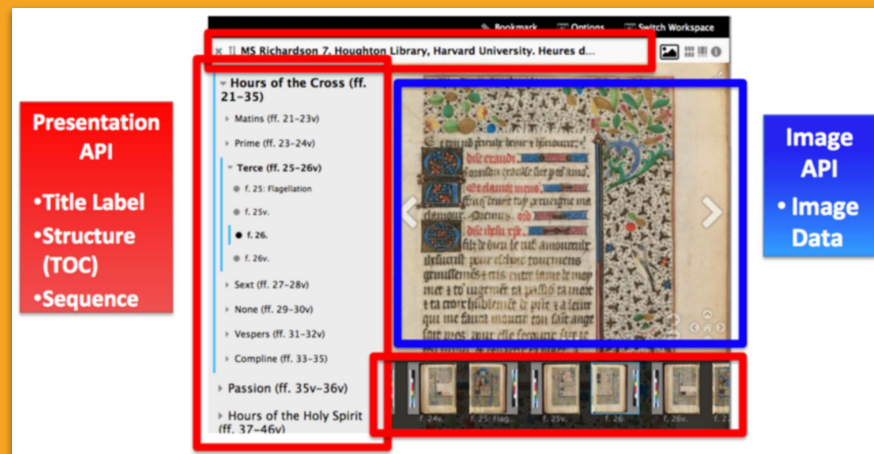

mirador

Facsimile

IIIF: all together

The **manifest** tells viewer applications how to display content and in which order.

The **image** API returns the actual image.

Exercise

How many swans?

- Open the URL of the **manifest**
  https://iiif.bodleian.ox.ac.uk/iiif/manifest/ae9f6cca-ae5c-4149-8fe4-95e6eca1f73c.json
- Click on triangles, browse to sequences/0/canvases/456/@id (search "fol. 218r").
- Click on the link and browse the manifest of the **canvas**
- Click on triangles, browse to images/0/resource/@id
- Click on the link and browse the info.json of the **image**
- Modify the URL: remove info.json, replace it with **full/full/0/default** to see the image

How many swans are there in the river?

Critical apparatus

Readings

Methods to record the apparatus

Critical apparatus

Viewpoints

Scholarly editions of texts, especially texts of great antiquity or importance, often record some or all of the known variations among different **witnesses** to the text.

Information about variant readings (whether or not represented by a critical apparatus in the source text) may be recorded in a series of apparatus entries, each entry documenting one variation, or set of readings, in the text.

Guidelines

Critical apparatus

List of witnesses

Information about witnesses is supplied by means of a <listWit> element, including as many <witness> elements as the number of witnesses, each identified with a @xml:id attribute.

```
1 <listWit>
2 <witness xml:id="El">Ellesmere, Huntingdon Library 26.C.9</witness>
3 <witness xml:id="Hg">Hengwrt, National Library of Wales,
4   Aberystwyth, Peniarth 392D</witness>
5 <witness xml:id="Ra2">Bodleian Library Rawlinson Poetic 149
6   (see further <ptr target="http://example.com/msDescs#MSRP149"/>)</witness>
7 </listWit>
```

Critical apparatus

Apparatus entries

Whenever variant readings of a word or text chunks must be recorded, the element **<app>** is used to wrap the readings. The attribute **@wit** records the identifier of the text wherein the variant reading appears.

@resp and @source record the responsible person, and @cert the level of certainty.

```
1 <app>
2   <rdg wit="#El">Experience though noon Auctoritee</rdg>
3   <rdg wit="#La">Experiment thouh noon Auctoritee</rdg>
4   <rdg wit="#Ra2">Eryment though none auctorite</rdg>
5 </app>
```

Critical apparatus

Readings

The app element must include at least one <rdg> element. The **<lem>** element may be used to record the **base text** of the source edition/witness, to indicate the preference of an editor or encoder for a particular reading, or (e.g. in the case of an external apparatus) to indicate precisely to which portion of the main text the variation applies.

```
1 <app>
2 <lem wit="#El #Hg">Experience</lem>
3 <rdg wit="#La" type="substantive">Experiment</rdg>
4 <rdg wit="#Ra2" type="substantive">Eryment</rdg>
5 </app>
```

Critical apparatus

How to annotate apparatus

Three different methods may be used to link a critical apparatus to the text:
- the location-referenced method
- the double-end-point-attached method, and
- the parallel segmentation method

All methods can be used **inline**, meaning variants are recorded in place and are scattered around the **base text**, or external, that is, all apparatus entries are included in an **external** file and a linking method is provided to reference text fragments.

Critical apparatus

How to annotate apparatus

The method for encoding the apparatus must be recorded in the **<encodingDesc>** element

```
1 <variantEncoding method="location-referenced" location="external"/>
```

Critical apparatus

Location-referenced external

Base text

```
1  <text>
2   <body>
3    <div n="WBP" type="prologue">
4     <head>The Prologe of the Wyves Tale of Bathe</head>
5     <l n="1">Experience though noon Auctoritee</l>
6     <l>Were in this world ...</l>
7    </div>
8   </body>
9  </text>
```

External xml or elsewhere in the base text xml

```
1  <app loc="WBP 1">
2   <rdg wit="#La">Experiment</rdg>
3   <rdg wit="#Ra2">Eryment</rdg>
4  </app>
```

The external apparatus references ids of elements in the base text and records variants in other witnesses.

Critical apparatus

Location-referenced internal

```
1  <l n="1">Experience
2  <app>
3    <rdg wit="#La">Experiment</rdg>
4    <rdg wit="#Ra2">Eryment</rdg>
5  </app>
6  though noon Auctoritee</l>
7  <l>Were in this world ...</l>
```

The apparatus is recorded inline in the base text. Again, only variants in other witnesses are recorded.

This method, like the previous one, **does not uniquely identify the text fragment** to which the readings refer to.

Critical apparatus

Location-referenced internal

```
1 <l n="1">Experience though noon Auctoritee
2 <app>
3   <lem wit="#El">Experience</lem>
4   <rdg wit="#La">Experiment</rdg>
5   <rdg wit="#Ra2">Eryment</rdg>
6  </app>
7 </l>
8 <l>Were in this world ...</l>
```

The annotated portion can be made explicit by using the **<lem> element** and the witness of the base text is referenced too.

This works well for single words though.

Critical apparatus

Double-end-point-attached

Double end-point attachment permits unambiguous matching of each variant reading against its lemma, since both the beginning and the end of the annotated lemma are marked and unequivocally identified with **@xml:id**. Like the location-referenced method, it can be encoded in the **base text**.

```
1 <l n="1" xml:id="WBP.1">Experience<anchor xml:id="WBP-A2"/> though noon Auctoritee</l>
```

Critical apparatus

Double-end-point-att. external

The apparatus uses the attributes **@from and @to** to record the ids of elements delimiting the lemma.

The lemma runs from the beginning of the element indicated by @from, to the end of that indicated by @to. If no value is given for @to, the lemma runs from the beginning to the end of the element indicated by the @from attribute.

```
1 <app from="#WBP.1" to="#WBP-A2">
2 <rdg wit="#La">Experiment</rdg>
3 <rdg wit="#Ra2">Eryment</rdg>
4 </app>
```

Critical apparatus

Double-end-point-att. inline

If the apparatus is encoded internally, there is no need to repeat the lemma. Otherwise, it should be made explicit.

```
1 <l n="1" xml:id="wbp.1">Experience
2 <app from="#wbp.1">
3   <rdg wit="#La">Experiment</rdg>
4   <rdg wit="#Ra2">Eryment</rdg>
5 </app>
6 though noon Auctoritee</l>
```

```
1 <app from="#WBP.1" to="#WBP-A2">
2   <lem wit="#El #Hg">Experience</lem>
3   <rdg wit="#La">Experiment</rdg>
4   <rdg wit="#Ra2">Eryment</rdg>
5 </app>
```

Critical apparatus

Parallel segmentation

When the base text reading is recorded in an apparatus reading (and not in the transcription), the method is called parallel segmentation.

It must be used **only internally**, but can be translated into the double end-point attachment method without loss of information. It does not handle well overlapping variants.

```
 1 <variantEncoding method="parallel-segmentation"
 2 location="internal"/>
 3 <!-- ... -->
 4 <l n="1">
 5  <app>
 6   <lem wit="#El #Hg">Experience</lem>
 7   <rdg wit="#La">Experiment</rdg>
 8   <rdg wit="#Ra2">Eryment</rdg>
 9  </app> though noon Auctoritee
10 </l>
11 <l>Were in this world ...</l>
```

Exercise

Hallelujah

So many versions of the famous song exist
- 1984 L. Cohen https://genius.com/Leonard-cohen-hallelujah-lyrics (until additional lyrics)
- 1988 L. Cohen https://genius.com/Leonard-cohen-hallelujah-lyrics (additional lyrics + verse 4)
- 1994 J. Buckley https://genius.com/Jeff-buckley-hallelujah-lyrics
- 2008 A. Burke https://genius.com/Alexandra-burke-hallelujah-lyrics

Create a minimal TEI/XML file for a critical edition of the song
- Describe the four witnesses in listWit
- Encode the base text (1984) and annotate the apparatus w/ the parallel-segmentation method