

How to enrich TEI corpus with Wikidata

An OpenRefine and XSL transformation workflow

Celian RINGWALD
Slides CC-BY 4.0

Two Case Studies:

1. The CraikSiteIndex



University of Calgary

2. Devonshire Manuscript



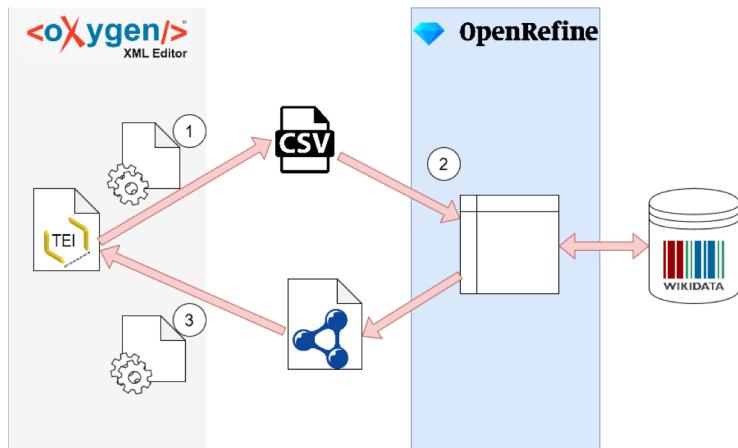
University of Victoria

Software requirements



- [Oxygen Editor](#)
- [OpenRefine 3.2](#) (**the version is important**)
- OpenRefine's [RDF XML extension](#)
 - **The installation is rather involved, try to follow the instructions carefully.**

The global approach : a mix of scripted and manual steps



1. We will export content we want to enrich from the TEI file as a CSV file [via XSLT]
2. We will use this file for getting via OpenRefine the interesting information that we want and then exporting it into a RDF file [manual]
3. Finally we will create a base XSLT containing a xsl map from the RDF file that we can use to inject data back into the TEI file [via XSLT]

But as every encoded project has different goals and each TEI file is different, we will have to adapt our process to each new TEI file!

1. The Craik Site Index



I.o. The Craik Site Index - [file description](#)

```
<TEI xmlns="http://www.tei-c.org/ns/1.0">
  <teiHeader> ...
  </teiHeader>
  <text>
    <body>
      <div type="DigitalCraikTeam"> ...
      </div>
      <div type="PastEditors"> ...
      </div>
      <div type="Pets"> ...
      </div>
      <div type="HistoricalPeople"> ...
      </div>
      <div type="FictionalCharacters"> ...
      </div>
      <div type="StubEntriesPeople"> ...
      </div>
      <div type="Organizations"> ...
      </div>
      <div type="StubEntriesOrganizations"> ...
      </div>
      <div type="HistoricalPlaces"> ...
      </div>
      <div type="StubEntriesPlaces"> ...
      </div>
      <div type="Events"> ...
      </div>
      <div type="Repositories"> ...
      </div>
      <div type="Periodicals"> ...
      </div>
      <div type="WorksByCraik"> ...
      </div>
      <div type="WorksByOthers"> ...
      </div>
      <div type="StubEntriesTitles"> ...
      </div>
    </body>
  </text>
</TEI>
```

TEI file listing :

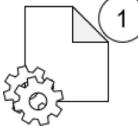
- Different type of peoples (DigitalCraikTeam, PastEditors, HistoricalPeople, FictionalCharacters, StubEntriesPeople)
- Organisations
- Places
- Events
- Literary works
- Periodicals



Current focus of the Tutorial



The « blahblah_person.extension » files



I.1. The Craik Site Index – TEI TO csv : which information to get ?

A Note About People

Some people could have a lot of homonyms!
For exemple, let's take a look on Walter Scott's Wikipedia page :

Walter Scott (disambiguation)

From Wikipedia, the free encyclopedia

Sir [Walter Scott](#) (1771–1832) was a Scottish poet and novelist.

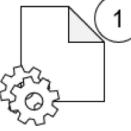
[Walter Scott](#) may also refer to:

Nobility [\[edit\]](#)

- [Walter Scott, 4th Baron of Buccleuch \(1549–1574\)](#)
- [Sir Walter Scott, 1st Lord Scott of Buccleuch \(1565–1611\)](#), Scottish nobleman and border reiver
- [Walter Scott, 1st Earl of Buccleuch \(before 1606–1633\)](#), Scottish nobleman
- [Sir Walter Scott of Braxholme and Buccleuch](#) (c. 1495–1552), nobleman of the Scottish Borders
- [Walter Scott, Earl of Tarras \(1644–1693\)](#), Scottish nobleman
- [Sir Walter Scott, 1st Baronet, of Beauclerc \(1826–1910\)](#), English building contractor and publisher
- [Walter Montagu Douglas Scott, 5th Duke of Buccleuch \(1806–1884\)](#), British politician and nobleman
- [Walter Montagu Douglas Scott, 8th Duke of Buccleuch \(1894–1973\)](#), politician and Conservative peer
- [Walter Scott, Earl of Dalkeith \(born 1984\)](#), British nobleman

- For this reason will have to extract the maximum amount prosopographic data to describe them from the TEI for the CSV file that we will use as input for OpenRefine !

```
<person xml:id="DMC" sex="2">
  <persName>
    <surname type="married">Craik</surname>
    <surname type="maiden">Mulock</surname>
    <forename>Dinah</forename>
    <forename>Maria</forename>
  </persName>
  <birth when="1826"/>
  <death when="1887"/>
  <occupation>Writer</occupation>
  <nationality>English</nationality>
  <event type="marriage" when="1865">
    <label>Marriage</label>
    <desc><persName ref="#DMC">Dinah Maria Mulock</persName> married <persName
      ref="#GeorgeCraik">George Lillie Craik.</persName></desc>
  </event>
</person>
```



I.1. The Craik Site Index -TEI TO CSV : XSLT file – PART I

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" xmlns:xs="https://schema.org/"
  xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:tei="http://www.tei-c.org/ns/1.0"

  xpath-default-namespace="http://www.tei-c.org/ns/1.0" version="3.0">
<xsl:output method="text" encoding="utf-8" />

<xsl:param name="delim" select=";" />
<xsl:param name="quote" select='"' />
<xsl:variable name="newline" select="
" />

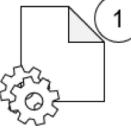
<xsl:template match="/">
  <xsl:value-of select="$quote"/>
  <xsl:text>Xml_id</xsl:text>
  <xsl:value-of select="$quote"/>
  <xsl:value-of select="$delim"/>
  <xsl:value-of select="$quote"/>
  <xsl:text>Gender</xsl:text>
  <xsl:value-of select="$quote"/>
  <xsl:value-of select="$delim"/>
  <xsl:value-of select="$quote"/>
  <xsl:text>Fornames</xsl:text>
  <xsl:value-of select="$quote"/>
  <xsl:value-of select="$delim"/>
  <xsl:value-of select="$quote"/>
  <xsl:text>Surnames</xsl:text>
  <xsl:value-of select="$quote"/>
  <xsl:value-of select="$delim"/>
  <xsl:value-of select="$quote"/>
  <xsl:text>Role</xsl:text>
  <xsl:value-of select="$quote"/>
  <xsl:value-of select="$delim"/>
  <xsl:value-of select="$quote"/>
  <xsl:text>BirthDate</xsl:text>
  <xsl:value-of select="$quote"/>
  <xsl:value-of select="$delim"/>
  <xsl:value-of select="$quote"/>
  <xsl:text>DeathDate</xsl:text>
  <xsl:value-of select="$quote"/>
  <xsl:value-of select="$newline"/>
  <xsl:apply-templates select="descendant::div[@type != 'DigitalCraikTeam' and @type != 'PastEditors' and @type != 'Pets']/listPerson"/>
</xsl:template>

<xsl:template match="listPerson">
  <xsl:apply-templates select="person[@xml:id]"/>
</xsl:template>
```

} CSV marks definitions

} CSV header

} Apply it to each listPers, where
@type is not linked to past
editors, the DH team or pets



I.1. The Craik Site Index -TEI TO CSV : XSLT file – PART II

```
<xsl:template match="listPerson">
    <xsl:apply-templates select="person[@xml:id]"/>
</xsl:template>

<xsl:template match="person">
    <xsl:value-of select="$quote"/>
    <xsl:value-of select="@xml:id"/>
    <xsl:value-of select="$quote"/>
    <xsl:value-of select="$delim"/>
    <xsl:value-of select="$quote"/>
    <xsl:apply-templates select="@sex"/>
    <xsl:value-of select="$quote"/>
    <xsl:value-of select="$delim"/>
    <xsl:apply-templates select="persName"/>
    <xsl:value-of select="$delim"/>
    <xsl:value-of select="$quote"/>
    <xsl:value-of select="occupation" separator="," />
    <xsl:value-of select="$quote"/>
    <xsl:value-of select="$delim"/>
    <xsl:value-of select="$quote"/>
    <xsl:value-of select="birth/@when"/>
    <xsl:value-of select="$quote"/>
    <xsl:value-of select="$delim"/>
    <xsl:value-of select="$quote"/>
    <xsl:value-of select="death/@when"/>
    <xsl:value-of select="$quote"/>
    <xsl:value-of select="$delim"/>
    <xsl:value-of select="$newline"/>
</xsl:template>
```

We only take the person with an ID, to ensure we can find them again when we return to the TEI file

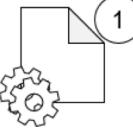
The xml id

Refers to « gender » fitted template

Refers to a persName template

if more than one occupation we bind them with a comma

Get birth and death dates



I.1. The Craik Site Index – TEI TO csv : XSLT file – PART III

```
<xsl:template match="@sex">
  <xsl:variable name="gender" as="xs:string">
    <xsl:choose>
      <xsl:when test=". eq '1'">man</xsl:when>
      <xsl:when test=". eq '2'">woman</xsl:when>
    </xsl:choose>
    <xsl:value-of select="$gender"/>
  </xsl:variable>
</xsl:template>
<xsl:template match="persName">
  <xsl:value-of select="$quote"/>
  <xsl:value-of select="forename" separator="," />

  <xsl:value-of select="$quote"/>
  <xsl:value-of select="$delim"/>
  <xsl:value-of select="$quote"/>
  <xsl:if test="surname">
    <xsl:choose>
      <xsl:when test="surname[@type='maiden'] and surname[@type='married']">
        <xsl:value-of select="surname[@type='maiden']/text()"/>
        <xsl:text> / </xsl:text><xsl:value-of select="surname[@type='married']/text()" />
      </xsl:when>
      <xsl:when test="count(surname) > 1">
        <xsl:value-of select="surname" separator="," />
      </xsl:when>
      <xsl:otherwise>
        <xsl:value-of select="surname"></xsl:value-of>
      </xsl:otherwise>
    </xsl:choose>
  </xsl:if>
  <xsl:value-of select="$quote"/>
</xsl:template>
</xsl:stylesheet>
```

...

Match and replace gender id by a more explicit expression

Template for the persName div

Specific process for the surname, in order to have each surname @type appear in the output in the same order



I.1. The Craik Site Index - getting the CSV output result

In Oxygen's XSLT view, specify files' names...

And go !



I.2. The Craik Site Index - OpenRefine - PART I: importing data

- First import the in OpenRefine, check encoding of the CSV file, and finally add a tag to your project.

Create a project by importing data. What kinds of data files can I import? TSV, CSV, XLS, Excel (xls and.xlsx), JSON, XML, RDF as XML, and Google Docs documents are all supported. Support for other formats can be added with OpenRefine extensions.

Get data from
This Computer
Web Addresses (URLs)
Clipboard
Database
Google Data

Locate one or more files on your computer to upload:
Browse... No files selected.
Next >

Start Over | Configure Parsing Options

Project name: 2_craikSiteIndex_person.csv | Tags | Create Project >

Xmt_id	Gender	Fornames	Surnames	Role	BirthDate	DeathDate	Column(s)
1.	DMC	woman	Dinah, Maria	Mulock / Craik	Winter	1826	1887
2.	ScottWalter	man	Water	Scott	Winter		
3.	Alderman	man	Henry	Alden			
4.	PhryeJohn	man	John	Phrye			
5.	CarrollEdward	woman	Craig	Osgood			
6.	OgrodJames	woman	James, Ripley	Osgood			
7.	LowSampson	man	Sampson	Low	Publisher	1797	1866
8.	MulockTom	man	Thomas, Mellar	Mulock		1827-11-18	
9.	MulockThomas	man	Thomas, Samuel	Mulock	Writer, Clergyman	1789	1869-08-11
10.	MulockKathiah	woman	Kathiah	Mulock		1800-01-23	1886-01-03
11.	MulockBenjamin	woman	Benjamin, Robert	Mulock	Engineer, Artist	1829-09-19	1887-01-13
12.	MulockJane	woman	Jane	Mulock		1801	1879-12-25
13.	MulockEmily	woman	Emily	Mulock	Gentlewoman	1801	1885-06-02
14.	MulockSarah	woman	Sarah	Mulock		11/18/1801	1900-06-07
15.	MulockFrances	woman	Frances	Mulock		1807	1882
16.	MulockAnna	woman	Ann	Mulock		1809	1886-12-01
17.	MulockJulia	woman	Julia, Hoblyn	Mulock / Hoblyn		1812	1886-06-28
18.	MulockHannah	woman	Hannah	Mulock		1786-07-09	1859
19.	HallSamuelCarter	man	Samuel, Carter	Hall	Editor, Writer	1800	1839
20.	HallAnneMeredith	woman	Anne, Merideth	Hall	Writer	1800	1851
21.	HallJohn	man	John	Hall	Democrat	1800	1856
22.	MulockEliza	woman	Eliza, Emily	Miles / MacDugger		1808	1901
23.	MulockEdward	man	Edward, John	Miles		1801	1930
24.	MulockHenry	woman	Henry, Alexander	Miles		1806	1942
25.	MulockJohn	woman	John	Miles		1802	1922
26.	MulockCatherine	woman	Catherine, Emily	Miles / Py		1806-05-29	1933-04-01
27.	MulockJohn	woman	John, Esther	Miles		1854	1940-04-14
28.	MulockJulia	woman	Julia, Miles	Miles		1855	1935-06
29.	MulockGeorge	woman	George	Miles	Journalist	1832-09-20	1911-10-28
30.	MulockJohn	man	John, Westland	Miles	Writer	1819	1870
31.	MulockEleanor	woman	Eleanor, Jane	Potts / Marion		1870	
32.	MulockJohn	woman	John, Marion	Potts / Marion	Writer	1850-01-13	1887-02-14
33.	JonesEsther	woman	Esther	Marie	Journalist		
34.	GatesElizabeth	woman	Elizabeth	Gates			
35.	DobellSydney	woman	Sydney	Dobell	Writer	1824	1874
36.	DobellDobell	woman	Dobell	Dobell		1830	1917
37.	DobellDobell	woman	Dobell	Dobell		1847	1921
38.	DobellWalter	man	Walter	Dobell		1876-01-30	1903-09-03
39.	DobellJessie	woman	Jessie, Dobell	Dobell	Writer		
40.	RiviereDobell	man	Brian	Riviere	Artist		

Parse data as
CSV / TSV / separator-based files

Line-based text files
Fixed-width field texts
PC-Axis text files
JSON files
MARC files
JSON-LD files
RDF/IN files
RDF/IN/Triples files

Character encoding
Character separator
Separator character

Columns are separated by
 commas (CSV)
 tabs (TSV)
 custom :
Escape special characters with \

Column names (comma separated)

Ignore first
 0 line(s) at beginning of file
 1 line(s) as column headers
 Discard initial
 0 row(s) of data
 Load at most
 0 row(s) of data
 Use character
 to enclose cells containing column separators

Parse cell text into numbers, dates, ...
 Store blank rows
 Store blank cells as nulls
 Store source (file names, URLs) in each row

567 rows

Show 5 rows, records 1 to 25 of 567 rows

All	id	Gender	Fornames	Surnames	Role	BirthDate	DeathDate
1.	DMC	woman	Dinah, Maria	Mulock / Craik	Winter	1826	1887
2.	ScottWalter	man	Water	Scott	Winter		
3.	Alderman	man	Henry	Alden			
4.	PhryeJohn	man	John	Phrye			
5.	CarrollEdward	woman	James, Ripley	Osgood			
6.	OgrodJames	woman	James, Ripley	Osgood			
7.	MulockTom	man	Thomas, Mellar	Mulock		1827-11-18	
8.	MulockThomas	man	Thomas, Samuel	Mulock	Writer, Clergyman	1789	1869-08-11
9.	MulockKathiah	woman	Kathiah	Mulock		1800-01-23	1886-01-03
10.	MulockBenjamin	woman	Benjamin, Robert	Mulock	Engineer, Artist	1829-09-19	1887-01-13
11.	MulockJane	woman	Jane	Mulock		1801	1879-12-25
12.	MulockEmily	woman	Emily	Mulock	Gentlewoman	1801	1885-06-02
13.	MulockSarah	woman	Sarah	Mulock		11/18/1801	1900-06-07
14.	MulockFrances	woman	Frances	Mulock		1807	1882
15.	MulockAnna	woman	Ann	Mulock		1809	1886-12-01
16.	MulockJulia	woman	Julia, Hoblyn	Mulock / Hoblyn		1812	1886-06-28
17.	MulockHannah	woman	Hannah	Mulock		1786-07-09	1859
18.	HallSamuelCarter	man	Samuel, Carter	Hall	Editor, Writer	1800	1839
19.	HallAnneMeredith	woman	Anne, Merideth	Hall	Writer	1800	1851
20.	HallJohn	man	John	Hall	Democrat	1800	1856
21.	MulockEliza	woman	Eliza, Emily	Miles / MacDugger		1808	1901
22.	MulockEdward	man	Edward, John	Miles		1801	1930
23.	MulockHenry	woman	Henry, Alexander	Miles		1806	1942
24.	MulockJohn	woman	John	Miles		1802	1922
25.	MulockCatherine	woman	Catherine, Emily	Miles / Py		1806-05-29	1933-04-01
26.	MulockJohn	woman	John, Esther	Miles		1854	1940-04-14
27.	MulockJulia	woman	Julia, Miles	Miles		1855	1935-06
28.	MulockGeorge	woman	George	Miles	Journalist	1832-09-20	1911-10-28
29.	MulockJohn	man	John, Westland	Miles	Writer	1819	1870
30.	MulockEleanor	woman	Eleanor, Jane	Potts / Marion		1870	
31.	MulockJohn	woman	John, Marion	Potts / Marion	Writer	1850-01-13	1887-02-14
32.	JonesEsther	woman	Esther	Marie	Journalist		
33.	GatesElizabeth	woman	Elizabeth	Gates			
34.	DobellSydney	woman	Sydney	Dobell	Writer	1824	1874
35.	DobellDobell	woman	Dobell	Dobell		1830	1917
36.	DobellDobell	woman	Dobell	Dobell		1847	1921
37.	DobellWalter	man	Walter	Dobell		1876-01-30	1903-09-03
38.	DobellJessie	woman	Jessie, Dobell	Dobell	Writer		
39.	RiviereDobell	man	Brian	Riviere	Artist		

Important for next steps: unselect checkboxes here



I.2. The Craik Site Index

OpenRefine - PART II : adding a column based on another

Secondly we will create a column containing first and last names
 > it will be the base of our Wikidata request !

Edit column > Add column based on this column...

All	Xml_id	Gender	Forenames	Surnames	Role	B
1.	DMC	woman	Facet	Mulock / Craik	Writer	1826
2.	ScottWalter	man	Text filter	Scott	Writer	
3.	AldenHenry	man	Edit cells	Alden		
4.	PhayreJohn	man	Edit column	Phayre		
5.	CairdMrsEdward	woman	Transpose		Split into several columns...	
6.	OsgoodJames		Sort...		Add column based on this column...	
7.	LowSampson	man	View		Add column by fetching URLs...	97
8.	MulockTom	man			Add columns from reconciled values...	27
9.	MulockThomas	man	Reconcile		Rename this column	89
10.	MulockDinah	woman			Remove this column	94
11.	MulockBen	man	Benjamin, Robert	M	Move column to beginning	01
12.	MulockJane	woman	Jane	M	Move column to end	01
13.	MulockEmily	woman	Emily	M	Move column left	87
14.	MulockEliza	woman	Elizabeth	M	Move column right	07
15.	MulockFrances	woman	Frances	M		06
16.	MulockAnn	woman	Ann	M		
17.	MulockAlicia	woman	Alicia, Bonne	Mulock / Hoblyn		1812



Thanks to the GREL language :

Add column based on column Fornames

New column name

On error set to blank store error copy value from original column

Expression

`cells["Fornames"].value + " " + cells["Surname"].value` No syntax error.

Preview	History	Starred	Help
row value	cells["Fornames"].value + " " ..		
1. Dinh, Maria	Dinh, Maria Mulock / Craik		
2. Walter	Walter Scott		
3. Henry	Henry Alden		
4. John	John Phayre		
5. null	null		
6. James, Ripleh	James, Riple Osgood		
-	-	-	-

OK Cancel

```
cells["Forname"].value+" "+cells["Surname"].value
```



I.2. The Craik Site Index - OpenRefine - PART III : linking data with Wikidata

Wikidata	Surnames
Facet	Craik
Text filter	Mulock / Craik
Edit cells	Scott
Edit column	Alden
Transpose	Phayre
Sort...	Caird
View	Osgood
Reconcile	Low
Facets	Mulock
Actions	Mulock
Copy reconciliation data...	
Use values as identifiers	

Then select Wikidata service and select « human » as reconciliation type...

Reconcile column "Wikidata"

Reconcile each cell to an entity of one of these types:

- human Q5
- painting Q3305213
- biographical article Q19369937
- publisher Q2085381

Also use relevant details from other columns:

Column	Include? As Property
Xml_Id	<input type="checkbox"/> Q5
Gender	<input checked="" type="checkbox"/> sex or gender
Fornames	<input checked="" type="checkbox"/> given name
Surnames	<input checked="" type="checkbox"/> family name
Role	<input checked="" type="checkbox"/> occupation
BirthDate	<input checked="" type="checkbox"/> date of birth
DeathDate	<input checked="" type="checkbox"/> date of death
Column 8	<input type="checkbox"/>

Reconcile against type:

Reconcile against no particular type

Auto-match candidates with high confidence

Maximum number of candidates to return:

Add Standard Service... Start Reconciling Cancel

And select each field that could be a clue for finding our entities in Wikidata
with its Wikidata type

Now sit back & wait for things to happen...

Reconcile cells in column Wikidata to type Q5
3% complete Cancel



I.2. The Craik Site Index - OpenRefine - PART IV: getting more links !

For this tutorial just filter the already matched data

Wikidata: judgment change

3 choices Sort by: name count

(blank) 277
matched 118
none 172

Facet by choice counts

If you want to go further you can, match every person manually to ensure that you are getting accurate content and that OpenRefine hasn't mismatched anyone in your file.

Then add column from reconciled values

Wikidata Surnames Role

Mulock / Craik	Writer
Alden	

Facet Text filter Edit cells Edit column Transpose Sort... View Reconcile

- ▶ Facet
- ▶ Text filter
- ▶ Edit cells
- ▶ Edit column
 - ▶ Split into several columns...
 - ▶ Add column based on this column...
 - ▶ Add column by fetching URLs...
 - ▶ Add columns from reconciled values...
- ▶ Transpose
- ▶ Sort...
- ▶ View
- ▶ Reconcile

Henry Alexander Miers
Choose new match

Francis Charles Miers
Choose new match

John Westland Marston
Choose new match

Elizabeth Gaskell
Choose new match

And add ISNI , VIAF and Getty IDs !

Add Property Preview

ISNI Select an item from the list:
ISNI International Standard Name Identifier for an identity. Format: 4 blocks P213

VIAF ID Wikidata Select an item from the list:
VIAF ID identifier for the Virtual International Authority File database [format: u] P214

Getty Wikidata Select an item from the list:
ULAN ID identifier from the Getty Union List of Artist Names P245



I.2. The Craik Site Index - OpenRefine - PART V: concatenate it with URLs

ISNI

VIAF ID

0000108957219	46793285
---------------	----------

Facet

Text filter

Edit cells

- Transform...
- Common transforms
- Fill down
- Blank down
- Split multi-valued cells...
- Join multi-valued cells...
- Cluster and edit...
- Replace

<http://www.isni.org/000>

On error: keep original set to blank store error

OK Cancel

Custom text transform on column ISNI

Expression: `"http://www.isni.org/" + replace(value, " ", "") + ".xml"`

Language: General Refine Expression Language (GREL)

Preview History Starred Help

row	value	transformed value
1.	0000 0001 0895 7219	http://www.isni.org/0000000108957219.xml
7.	0000 0000 4039 6637	http://www.isni.org/0000000040396637.xml
19.	0000 0000 8077 6490	http://www.isni.org/0000000080776490.xml
23.	0000 0003 8353 9986	http://www.isni.org/0000000383539986.xml
24.	0000 0000 8196 2535	http://www.isni.org/0000000081962535.xml
25.	null	Error: replace expects 3 strings, or 1 string, 1 regex, and 1 string
26.	0000 0000 0112 0084	http://www.isni.org/0000000001120084.xml

On error: keep original set to blank store error

Re-transform up to 10 times until no change

OK Cancel

For deleting spaces in ISNI IDs

Machine readable urls :

ISNI : [http://www.isni.org/\\$ID.xml](http://www.isni.org/$ID.xml)

Getty : [http://vocab.getty.edu/ulan/\\$ID.rdf](http://vocab.getty.edu/ulan/$ID.rdf)

VIAF : [https://viaf.org/viaf/\\$ID/rdf.xml](https://viaf.org/viaf/$ID/rdf.xml)

WikidataID	ISNI	VIAF ID	UALAN ID
https://www.wikidata.org/wiki/Q231155	http://www.isni.org/0000000108957219	https://viaf.org/viaf/46793285	
https://www.wikidata.org/wiki/Q5722891	http://www.isni.org/000000001502993	https://viaf.org/viaf/20174323	
https://www.wikidata.org/wiki/Q7419233	http://www.isni.org/0000000040390537	https://viaf.org/viaf/21131048	
https://www.wikidata.org/wiki/Q2218970	http://www.isni.org/0000000087774840	https://viaf.org/viaf/57216250	
https://www.wikidata.org/wiki/Q265204	http://www.isni.org/0000000091381968	https://viaf.org/viaf/59344020	
https://www.wikidata.org/wiki/Q343666	http://www.isni.org/0000000383539986	https://viaf.org/viaf/70092022	
https://www.wikidata.org/wiki/Q376708	http://www.isni.org/0000000091962335	https://viaf.org/viaf/10612317	
https://www.wikidata.org/wiki/Q4976588	http://www.isni.org/0000000091962335	https://viaf.org/viaf/14444221	
https://www.wikidata.org/wiki/Q11912	http://www.isni.org/0000000091962335	https://viaf.org/viaf/39977538	
https://www.wikidata.org/wiki/Q229522	http://www.isni.org/0000000091962335	https://viaf.org/viaf/68944196	
https://www.wikidata.org/wiki/Q3644145	http://www.isni.org/0000000091962335	https://viaf.org/viaf/12176155	http://vocab.getty.edu/ulan/500000000000000000
https://www.wikidata.org/wiki/Q2470305	http://www.isni.org/000000007040411	https://viaf.org/viaf/14300596	http://vocab.getty.edu/ulan/500000000000000000



I.2. The Craik Site Index - OpenRefine - PART VI: getting the Wikidata ID ?

For the moment we have an html link in the Wikidata columns...How can I get only the machine readable Wikidata URI?

Add column based on column Wikidata

New column name **WikidataID**

On error set to blank store error copy value from original column

Expression Language General Refine Expression Language (GREL)

```
http://www.wikidata.org/wiki/Special:EntityData  
"+cell.recon.candidates[0].id+".rdf"
```

No syntax error.

Preview History Starred Help

row	value
1.	Dinah, Maria Mulock / Craik http://www.wikidata.org/wiki/Special:EntityData/Q2331155.rdf
7.	Sampson Low http://www.wikidata.org/wiki/Special:EntityData/Q7410238.rdf
19.	Samuel, Carter Hall http://www.wikidata.org/wiki/Special:EntityData/Q2218070.rdf
23.	Edward, John Miers http://www.wikidata.org/wiki/Special:EntityData/Q5343666.rdf
24.	Henry, Alexander Miers http://www.wikidata.org/wiki/Special:EntityData/Q375078.rdf
25.	Francis Miers http://www.wikidata.org/wiki/Special:EntityData/Q49576588.rdf

OK Cancel

Call it « WikidataID »
Important for the next steps!

"http://www.Wikidata.org/wiki/Special:EntityData/"
+cell.recon.candidates[0].id+".rdf"
> This would introduce problems with directly matched rows...

"http://www.Wikidata.org/wiki/Special:EntityData/"
+cell.recon.match.id+".rdf"



I.2. The Craik Site Index - OpenRefine - PART VII: Exporting results as a RDF file

Why are using the RDF export schema?

Because XLST allows us to transform only XML files and RDF is a specific kind of XML,

To create the RDF XML we need to define first a schema in which data will be structured : let's do it!

- 1- reset RDF default schema
- 2- Edit the RDF skeleton
- 3- Replace URI by the XML_id
- 4- add the field of interest
that we want to extract
(WikidataID, ISNI Cell, VIAF ID Cell, ULAN ID)
- 5- define each property
as « rdfs:seeAlso »
- 6- Save
& Export > RDF as RDF/XML

The screenshot shows the OpenRefine interface with the following steps highlighted:

1. In the top-left corner, the "Edit RDF skeleton..." option is selected.
2. In the top-right corner, the "Export" dropdown menu is open, showing various export options like "Tab-separated value", "Comma-separated value", "HTML table", "Excel (.xls)", etc. The "RDF as RDF/XML" option is circled and highlighted in blue.
3. In the main panel, under "Available prefixes:", the "Xml_id URI" and "Add type" buttons are circled.
4. In the main panel, the "WikidataID Cell", "ISNI Cell", "VIAF ID Cell", and "ULAN ID Cell" checkboxes are circled.
5. In the main panel, the "rdfs:seeAlso" property is defined with four recursive arrows, circled.
6. In the bottom-right corner, the "Save" button is circled, and a yellow arrow points from the "RDF as RDF/XML" option in the export menu down to the save button.

The main panel displays the "RDF Schema alignment" configuration with the following text:
The RDF schema alignment skeleton below specifies how the RDF data that will get generated from your grid-shaped data. The cells in each record of your data will get placed into nodes within the skeleton. Configure the skeleton by specifying which column to substitute into which node.
Base URI: http://localhost:3333/ [Edit](#)

Available prefixes: rdf owl rdfs foaf [+ Add](#) [Manage](#)

RDF skeleton [RDF Preview](#)

Add another root node

OK Cancel

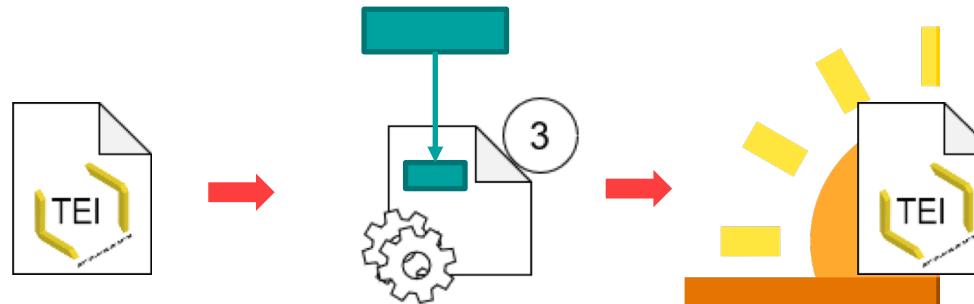
3_craikSiteIndex_rdf_person.rdf

I.3. The Craik Site Index - a last two step process

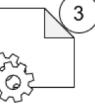
The last part!

In this section we will present [a first xsl file](#) for mapping the data we've found in Wikidata as an xsl variable.

Then we will use this variable in a xsl file created for copying the content of our original file and inject new information into the original site index based on our xsl variable



> In other words, we will include one file in another and keep both in same directory



I.3. The Craik Site Index - From RDF to a xsl:map variable

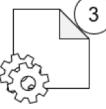
```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" xmlns:schema="https://schema.org/"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:map="http://www.w3.org/2005/xpath-functions/map"
  exclude-result-prefixes="xsl schema rdf xs"
  xmlns:axsl="http://www.w3.org/1999/XSL/TransformAlias"
  version="3.0">
  <xsl:output method="xml" encoding="UTF-8" indent="yes"/>
  <xsl:namespace-alias stylesheet-prefix="axsl" result-prefix="xsl"/>
  <xsl:template match="/rdf:RDF">
    <axsl:stylesheet>
      <axsl:variable name="viaf_ref" as="map(xs:string, xs:string)">
        <axsl:map>
          <xsl:for-each select="rdf:Description/child::*">
            <xsl:if test="contains(text(), 'viaf')">
              <axsl:map-entry key="'{replace(parent::node()//@rdf:about, 'http://localhost:3333/', '')}' select=".
                .'">
              </xsl:if>
            </xsl:for-each>
          </axsl:map>
        </axsl:variable>
      <axsl:variable name="wikidata_ref" as="map(xs:string, xs:string)">
        <axsl:map>
          <xsl:for-each select="rdf:Description/child::*">
            <xsl:if test="contains(text(), 'wikidata')">
              <axsl:map-entry key="'{replace(parent::node()//@rdf:about, http://localhost:3333/, '')}' select=".
                .'">
              </xsl:if>
            </xsl:for-each>
          </axsl:map>
        </axsl:variable>
    </axsl:stylesheet>
  </xsl:template>
</xsl:stylesheet>
```

} Namespace declaration

} Alias is used for the xsl tag that we will write in the output.
We are using xsl to write an xsl file!

} And we create a map for linking each authority URI to
the xml:id...

You could have to change availability of
your localhost port. Use the port number
from the OpenRefine RDF file



I.3. The Craik Site Index - Tei to Enriched TEI file

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet
    xmlns="http://www.tei-c.org/ns/1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:map="http://www.w3.org/2005/xpath-functions/map"
    xmlns:tei="http://www.tei-c/ns/1.0"
    xpath-default-namespace="http://www.tei-c.org/ns/1.0"
    exclude-result-prefixes=" xsl tei xs"
    version="3.0">
    <xsl:strip-space elements="*"/>
    <xsl:output method="xml" indent="yes"/>
    <xsl:include href="5_xsl_map_person.xsl"/>

    <xsl:template match="node()| @*">
        <xsl:copy>
            <xsl:apply-templates select="node() | @*"/>
        </xsl:copy>
    </xsl:template>

    <xsl:template match="person[@xml:id]">
        <xsl:variable name="myId" select="@xml:id"/>

        <xsl:copy>
            <xsl:apply-templates select="node() | @*"/>
            <xsl:if test="map:contains($wikidata_ref, $myId)">
                <xsl:element name="idno" >
                    <xsl:attribute name="type">WIKIDATA</xsl:attribute>
                    <xsl:attribute name="ref">
                        <xsl:value-of select="$wikidata_ref($myId)"/>
                    </xsl:attribute>
                </xsl:element>
            </xsl:if>
        </xsl:copy>
    </xsl:template>

```

Namespace declaration

Here we are calling the file created by the xsl in the previous slide (you may have to change the name to match the name of the map_person.xsl file on your local computer)

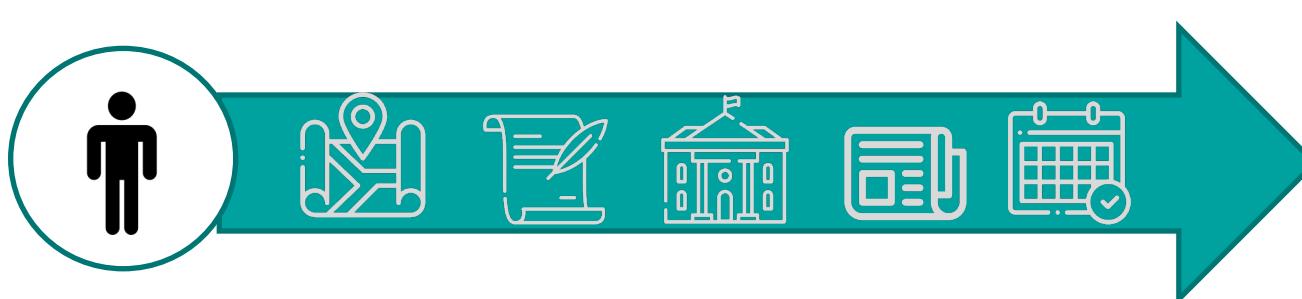
And here check if the xml:id is in the xsl:map of the previous file, if it is the case create a « idno » element with the @type of reference it came from.

...

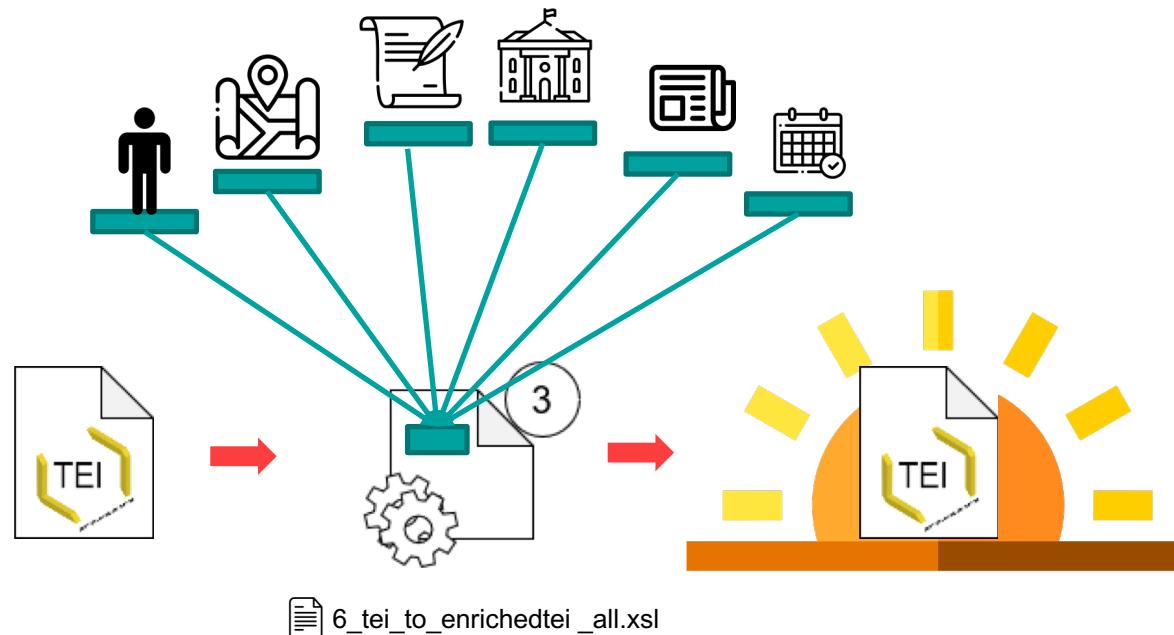


I.3. The Craik Site Index - going further

- We did the process for people cited in the Craik File, (almost) the same process could be done on places, literary works, organizations, periodicals, and historical events in the file. Feel free to make tweaks to the process to enrich these other sections of the file
- Let's take a look on the other files !
(some helpful screenshot are also available in 'going further' directory)



I.4. The Craik Site Index - At the end



2. The Devonshire Manuscript



2.0. The Devonshire Manuscript - file description

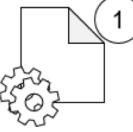
```
<?xml version="1.0" encoding="UTF-8"?><?oxygen RNGSchema="tei_all.rnc" type="compact"?>
<TEI>
  <!-- xmlns:xsi="http://www.w3.org/2001/XMLSchema#" xmlns:svg="http://www.w3.org/2000/svg"
  xmlns:math="http://www.w3.org/1998/Math/MathML" xmlns="http://www.tei-c.org/ns/1.0" -->
  <teiHeader>
    <fileDesc> ...
      ...
    </fileDesc>
    <encodingDesc> ...
      ...
    </encodingDesc>
    <profileDesc>
      <creation> ...
        ...
      </creation>
      <langUsage> ...
        ...
      </langUsage>
    </profileDesc>
    <particDesc> ...
      ...
    </particDesc>
  </profileDesc>
  <revisionDesc> ...
    ...
  </revisionDesc>
</teiHeader>
<text>
  <body>
    <pb n="0v"/>
    <div>
      <note type="editorial">This is the inside front cover.</note>
      <p>Purchased of Thos. Rodd<lb/>11 Nov. 1843.<lb/>He bought it at <name key="NOTT">
        Dr. Nott</name>'s sale.</p>
    </div>
    <pb n="1r"/>
    <div type="collective"> ...
      ...
    </div>
    <pb n="1v"/>
    <pb n="1.1r"/>
    <pb n="1.1v"/>
    <pb n="2r"/>
    <div type="poem" xml:id="LDev001-TM1479" rhyme="20:5x4 aaa8R4">
      <head>
        <bibl><title type="incipit">Take hede be tyme leste ye be spyeðe</title>
          <attributedTo>Sir Thomas Wyatt</attributedTo>
          <type>book</type>>Thomas Wyatt: Complete Poems</title>, page
          <num>151</num>.</bibl>
      <note type="intro">
```

A TEI-XM file describing the **Devonshire Manuscript**:
 A manuscript of copied and original poems by multiple
 authors, compiled in the 16th century.

We will be working with content from the particDesc part of the
 teiHeader/profileDesc tag



```
<particDesc>
  <listPerson type="historical">
    <person xml:id="WYATT" role="poet">
      <persName>
        <ref type="LOC">n 50019425</ref>
        <ref type="DNB">http://www.oxforddnb.com/view/article/30111</ref>
        <roleName type="honorific">Sir</roleName>
        <forename>Thomas</forename>
        <surname>Wyatt</surname>
        <addName type="epithet">the Elder</addName>
      </persName>
      <birth notBefore="1502" notAfter="1504">1503</birth>
      <death when="1542">1542</death>
    </person>
  </listPerson>
</particDesc>
```



2.1. The Devonshire Manuscript – TEI TO CSV : which information to get ?

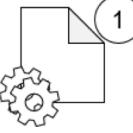
- We have here a lot of imprecise birth and death dates...
- But we have some IDs from :

Ok but we will have to be careful -- some of these were chosen by default during the project

```
<!-- KA: the ref is for the DNB entry for Knyvet, which mentions an altercation with Clere. -->
<person xml:id="CLERE" role="poet">
  <!--
    <ref type="DNB">http://www.oxforddnb.com/view/articleHL/15797</ref>
    <forename>Thomas</forename>
    <surname>Clere</surname>
  -->
</person>
```



Oxford Dictionary of National Biography



2.1. The Devonshire Manuscript– TEI TO CSV : XSLT file – PART I

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  exclude-result-prefixes="xs"
  version="3.0">
  <xsl:output method="text" encoding="utf-8" />

  <xsl:param name="delim" select="';'" />
  <xsl:param name="quote" select="'&quot;'" />
  <xsl:variable name="newline" select="'
'" />

  <xsl:template match="TEI">
    <xsl:value-of select="$quote"/>
    <xsl:text>Xml_id</xsl:text>
    <xsl:value-of select="$quote"/>
    <xsl:value-of select="$delim"/>
    <xsl:value-of select="$quote"/>
    <xsl:text>Names</xsl:text>
    <xsl:value-of select="$quote"/>
    <xsl:value-of select="$delim"/>
    <xsl:value-of select="$quote"/>
    <xsl:text>Role</xsl:text>
    <xsl:value-of select="$quote"/>
    <xsl:value-of select="$delim"/>
    <xsl:value-of select="$quote"/>
    <xsl:text>DNB ID</xsl:text>
    <xsl:value-of select="$quote"/>
    <xsl:value-of select="$delim"/>
    <xsl:value-of select="$quote"/>
    <xsl:text>ILOC ID</xsl:text>
    <xsl:value-of select="$quote"/>
    <xsl:value-of select="$newline"/>
    <xsl:apply-templates select="teiHeader/profileDesc/particDesc/listPerson"/>
  </xsl:template>
  . . .

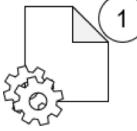
```



CSV marks definitions



CSV header



2.1. The Devonshire Manuscript – TEI TO CSV : XSLT file – PART II

```
<xsl:template match="listPerson[@type='historical']">
    <xsl:apply-templates select="person[@xml:id]"/>
</xsl:template>

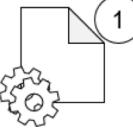
<xsl:template match="listPerson">
    ...
    <xsl:template match="person">
        <xsl:value-of select="$quote"/>
        <xsl:value-of select="@xml:id"/>
        <xsl:value-of select="$quote"/>
        <xsl:value-of select="$delim"/>
        <xsl:apply-templates select="persName[1]"/>
        <xsl:value-of select="$delim"/>
        <xsl:value-of select="$quote"/>
        <xsl:value-of select="@role"/>
        <xsl:value-of select="$quote"/>
        <xsl:value-of select="$delim"/>
        <xsl:value-of select="$quote"/>
        <xsl:if test="@xml:id!='CLERE'">
            <xsl:apply-templates select="persName[1]/ref[@type='DNB']"/>
        </xsl:if>
        <xsl:value-of select="$quote"/>
        <xsl:value-of select="$delim"/>
        <xsl:value-of select="$quote"/>
        <xsl:value-of select="replace(persName[1]/ref[@type='LOC']/text(), ' ', '')"/>
        <xsl:value-of select="$quote"/>
        <xsl:value-of select="$newline"/>
    </xsl:template>
</xsl:template>
```

Only for historical people template

Get the first name given

As we saw, this a is not actually
Clere's DNB entry

Get a string without spaces



2.1. The Devonshire Manuscript– TEI TO CSV : XSLT file – PART III

```
<xsl:template match="persName[1]/ref[@type='DNB']">  
    <xsl:param name="txt" select=" tokenize(text(), '/') " />  
    <xsl:value-of select="$txt[last()]" /></xsl:value-of>  
</xsl:template>  
  
<xsl:template match="persName[1]">  
    <xsl:value-of select="$quote"/>  
    <xsl:choose>  
        <xsl:when test="forename and surname">  
            <xsl:value-of select="forename" separator="," />  
            <xsl:text> </xsl:text>  
            <xsl:value-of select="surname" separator="," />  
        </xsl:when>  
        <xsl:when test="(forename or surname) and roleName">  
            <xsl:value-of select="roleName"/>  
            <xsl:text>, </xsl:text>  
            <xsl:if test="forename">  
                <xsl:value-of select="forename"/>  
            </xsl:if>  
            <xsl:if test="surname">  
                <xsl:value-of select="surname"/>  
            </xsl:if>  
        </xsl:when>  
    </xsl:choose>  
    <xsl:value-of select="$quote"/>  
</xsl:template>  
</xsl:stylesheet>
```

...

}

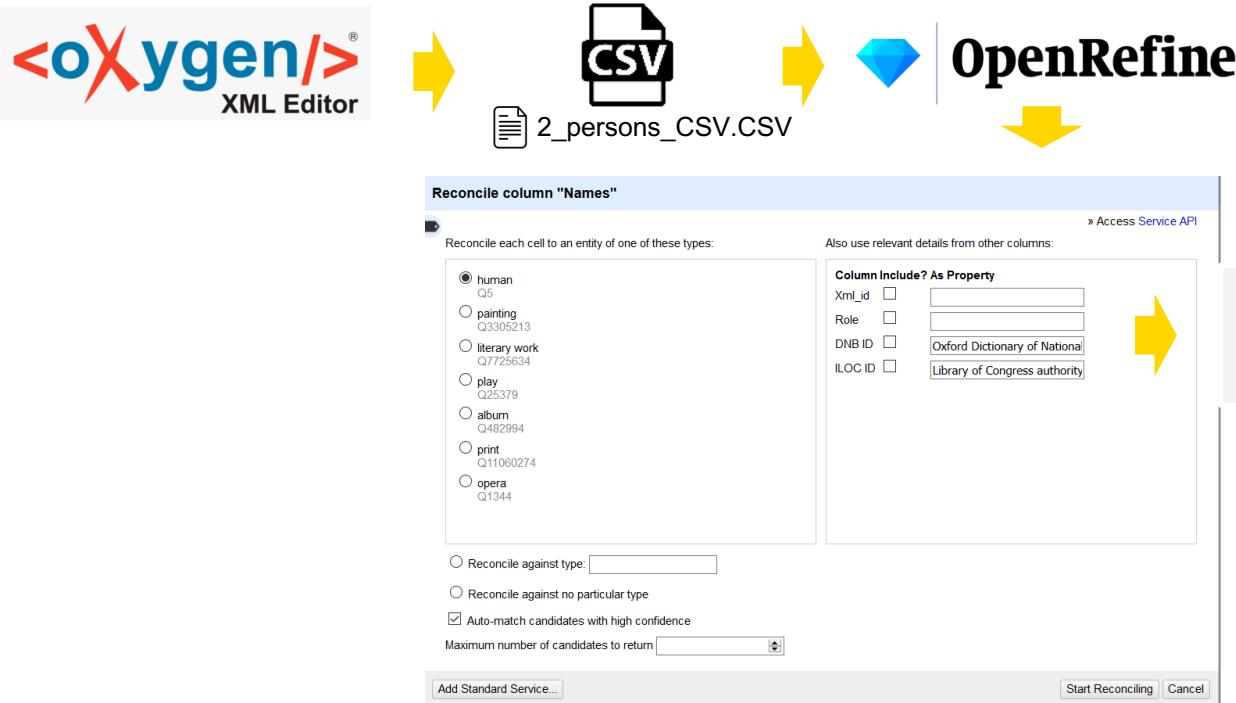
Tokenize DNB url and get only the ID

}

Some people have a noble title



2.2. The Devonshire Manuscript- Reconciling data with OpenRefine



Oxford Dictionary of National Biography ID
& Library of Congress authority ID

Match & Filter by Judgment



2.2. The Devonshire Manuscript – making links with OpenRefine

Get VIAF, Getty, and ISNI IDs by reconciliation

Do the same with the Oxford DNB and Library of Congress ID

(we had some errors in base file, we could correct the errors this way;)

22 matching rows (25 total)

Show as: rows records Show: 5 10 25 50 rows

All	Xml_ID	Names	Role	DNB ID
1.	WYATT	Facet	poet	30111
2.	LEE	Text filter	poet	16288
3.	KNYVET	Edit cells		
4.	CHAUCER	Edit column	Split into several columns...	
5.	SURREY	Transpose	Add column based on this column...	
6.	SHELTON	Sort...	Add column by fetching URLs...	
7.	QUEEN	View	Add columns from reconciled values...	
8.	ANNE BOLEYN	Reconcile	Rename this column	
9.	MARY_SCOTS	Choose new match	Remove this column	
10.	DOUGLAS	Mary, Queen of Scots		
11.	M_HOWARD	Choose new match		
		Margaret Douglas		
		Mary FitzRoy, Duchess of		
		Choose new match		



Add columns from reconciled column Names

Add Property Preview Reset

ULAN ID	Names	Oxford Dictionary of National Biography ID	Library of Congress authority ID	VIAF ID	ISNI
		remove config	remove config	remove config	remove config
Thomas Wyatt	30111	n50019425	44340301	0000 0001 08:8784	
Henry Lee	16288	no2010077487	121217313	0000 0000 79:105X	
Edmund Knyvet	15797				
Geoffrey Chaucer	5191	n79027228	100185203	0000 0001 15:454X	
				0000 0003 75:0787	
Henry Howard, Earl of Surrey	13905	n50083266	17378697	0000 0001 16:8072	
Mary Shelton	68085		88671425	0000 0000 62:2215	
Anne Boleyn	557	n79063685	29521340	0000 0000 96:1375	
Mary, Queen of	18248	n80044764	104722318	0000 0001 21:5913	

OK Cancel

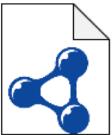
And then concatenate IDs with human readable urls as we saw slide 16 and 17

For Oxford DNB and LOD ID use :

« [http://www.oxforddnb.com/view/article/\\$ID_DNB](http://www.oxforddnb.com/view/article/$ID_DNB) »

« [http://id.loc.gov/authorities/names/\\$ID_LOD](http://id.loc.gov/authorities/names/$ID_LOD) »





2.2. The Devonshire Manuscript- exporting it into RDF

And then extract refined data into rdf (as in slide [18](#))

22 matching rows (28 total)

Show as: **rows** records Show: **5 10 25 50** rows

All	XML_ID	Names	Oxford Dictionary of N...	Library of Congress autho...	VIAF ID	ISNI
	1. WYATT	Thomas Wyatt Choose new match	https://doi.org/10.1093/etdmsm/2019/425	http://id.loc.gov/authorities/names/nm00000109425	https://viaf.org/viaf/44340301/rdf.xml	http://www.isni.org/0000000108928
	2. LEE	Henry Lee Choose new match	https://doi.org/10.1093/etdmsm/2019/428	http://id.loc.gov/authorities/names/nm00000109428	https://viaf.org/viaf/121217313/rdf.xml	http://www.isni.org/000000079301
	3. KNYVET	Edward Knayvel Choose new match	https://doi.org/10.1093/etdmsm/2019/427	http://id.loc.gov/authorities/names/nm00000109427	https://viaf.org/viaf/12120077487/rdf.xml	http://www.isni.org/000000079301
	4. CHAUCER	Geoffrey Chaucer Choose new match	https://doi.org/10.1093/etdmsm/2019/5191	http://id.loc.gov/authorities/names/nm00000115574	https://viaf.org/viaf/100185203/rdf.xml	http://www.isni.org/0000000115574
	7. SURREY	Henry Howard, Earl of Surrey Choose new match	https://doi.org/10.1093/etdmsm/2019/13005	http://id.loc.gov/authorities/names/nm0000016008	https://viaf.org/viaf/17378697/rdf.xml	http://www.isni.org/000000016008
	8. SHELTON	Mary Shelton Choose new match	https://doi.org/10.1093/etdmsm/2019/69085	http://id.loc.gov/authorities/names/nm00000162122	https://viaf.org/viaf/8871425/rdf.xml	http://www.isni.org/0000000162122
	9. QUEEN	Anne Boleyn Choose new match	https://doi.org/10.1093/etdmsm/2019/69571	http://id.loc.gov/authorities/names/nm0000016008	https://viaf.org/viaf/29521340/rdf.xml	http://www.isni.org/000000016008
	10. MARY_SCOTS	Mary, Queen of Scots Choose new match	https://doi.org/10.1093/etdmsm/2019/69046	http://id.loc.gov/authorities/names/nm0000014764	https://viaf.org/viaf/104722318/rdf.xml	http://www.isni.org/000000014764
	11. DOUGLAS	Margaret Douglas Choose new match	https://doi.org/10.1093/etdmsm/2019/7911	http://id.loc.gov/authorities/names/nm00000129084	https://viaf.org/viaf/69883628/rdf.xml	http://www.isni.org/0000000151916
	12. M_HOWARD	Marj Fit/Ro. Duchess of Richmond and Somerset Choose new match	https://doi.org/10.1093/etdmsm/2019/9538	http://id.loc.gov/authorities/names/nm0000012098273	https://viaf.org/viaf/88670504/rdf.xml	http://www.isni.org/0000000162122
	13. TH_HOWARD	Lord Thomas Howard Choose new match	https://doi.org/10.1093/etdmsm/2019/70793	http://id.loc.gov/authorities/names/nm00000122051	https://viaf.org/viaf/40190630/rdf.xml	http://www.isni.org/000000054257
	14. DARNLEY	Henry Stuart, Lord Darnley Choose new match	https://doi.org/10.1093/etdmsm/2019/26473	http://id.loc.gov/authorities/names/nm0000013013805	https://viaf.org/viaf/104722318/rdf.xml	http://www.isni.org/0000000121305
	15. AQUILANO	Serafino dell'Aquila Choose new match				
	16. CLERE	Thomas de Cleve Choose new match				
	17. PETRARCH	Petrarch Choose new match				
	20. KING	Henry VII of England Choose new match	https://doi.org/10.1093/etdmsm/2019/12955	http://id.loc.gov/authorities/names/nm000001113093	https://viaf.org/viaf/172419710/rdf.xml	http://www.isni.org/0000000122586
	22. GOWER	John Gower Choose new match	https://doi.org/10.1093/etdmsm/2019/11176	http://id.loc.gov/authorities/names/nm000001039793	https://viaf.org/viaf/120695167/rdf.xml	http://www.isni.org/00000001039793
	23. ALAMANINI	Luigi Alamanni Choose new match				
	24. PISAN	Christine de Pizan Choose new match				

RDF Schema alignment

The RDF schema alignment skeleton below specifies how the RDF data that will get generated from your grid-shaped data. The cells in each record of your data will get placed into nodes within the skeleton. Configure the skeleton by specifying which column to substitute into which node.

Base URI: <http://localhost:3333/> Edit

RDF skeleton [RDF Preview](#)

Available prefixes: [rdf](#) [owl](#) [rdfs](#) [foaf](#) [+ Add](#) [Manage](#)

XML_ID [URI](#)
[Add type](#)

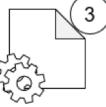
[X > rdfs:seeAlso >](#)
 [X > rdfs:seeAlso >](#)
 [X > rdfs:seeAlso >](#)
 [X > rdfs:seeAlso >](#)
 [X > rdfs:seeAlso >](#)
[Add property](#)

Add another root node

[OK](#) [Cancel](#)

[Save](#)

3_persons_rdf.rdf



2.3. The Devonshire Manuscript- Create a xslt map : part I

```

<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" xmlns:schema="https://schema.
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:xm="http://www.w3.org/2001/XMLSchema"
  xmlns:map="http://www.w3.org/2005/xpath-functions/map"
  xmlns:axsl="http://www.w3.org/1999/XSL/TransformAlias"

  version="3.0">
<xsl:output method="xml" encoding="UTF-8" indent="yes"/>

<xsl:namespace-alias stylesheet-prefix="axsl" result-prefix="xsl"/>

<xsl:template match="/rdf:RDF">
  <xsl:stylesheet
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    version="3.0"
  >

    <xsl:variable name="vif_pers_ref" as="map(xs:string, xs:string)">
      <xsl:map>
        <xsl:for-each select="rdf:description/child::*">
          <xsl:if test="contains(text(), 'vif')">
            <xsl:map-entry key=""(replace(parent::node()/@rdf:about, 'http://localhost:3333/',''))" select=".()"/>
          </xsl:if>
        </xsl:for-each>
      </xsl:map>
    </xsl:variable>

    <xsl:variable name="wikidata_pers_ref" as="map(xs:string, xs:string)">
      <xsl:map>
        <xsl:for-each select="rdf:description/child::*">
          <xsl:if test="contains(text(), 'wikidata')">
            <xsl:map-entry key=""(replace(parent::node()/@rdf:about, 'http://localhost:3333/',''))" select=".()"/>
          </xsl:if>
        </xsl:for-each>
      </xsl:map>
    </xsl:variable>

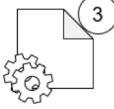
    <xsl:variable name="isni_pers_ref" as="map(xs:string, xs:string)">
      <xsl:map>
        <xsl:for-each select="rdf:description/child::*">
          <xsl:if test="contains(text(), 'isni')">
            <xsl:map-entry key=""(replace(parent::node()/@rdf:about, 'http://localhost:3333/',''))" select=".()"/>
          </xsl:if>
        </xsl:for-each>
      </xsl:map>
    </xsl:variable>

    <xsl:variable name="getty_pers_ref" as="map(xs:string, xs:string)">
      <xsl:map>
        <xsl:for-each select="rdf:description/child::*">
          <xsl:if test="contains(text(), 'getty')">
            <xsl:map-entry key=""(replace(parent::node()/@rdf:about, 'http://localhost:3333/',''))" select=".()"/>
          </xsl:if>
        </xsl:for-each>
      </xsl:map>
    </xsl:variable>
  </xsl:namespace-alias>

```

Time to use xslt to make an xslt file.
We create a map linking each authority
URI to the xml:id...

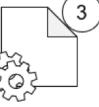




2.3. The Devonshire Manuscript- Create a xslt map : part II

And we create a map for the LOC and DNB id, delete url that we used to identify them with the contains function

```
<xsl:variable name="dnb_pers_ref" as="map(xs:string, xs:string)">
    <xsl:map>
        <xsl:for-each select="rdf:Description/child::*">
            <xsl:if test="contains(text(),'ref:odnb')">
                <xsl:map-entry key="{replace(parent::node()//@rdf:about,'http://localhost:3333/','')}" select="{replace(.,'https://doi.org/10.1093/ref:odnb/1','')}"/>
            </xsl:if>
        </xsl:for-each>
    </xsl:map>
</xsl:variable>
<xsl:variable name="loc_pers_ref" as="map(xs:string, xs:string)">
    <xsl:map>
        <xsl:for-each select="rdf:Description/child::*">
            <xsl:if test="contains(text(),'id.loc.gov')">
                <xsl:map-entry key="{replace(parent::node()//@rdf:about,'http://localhost:3333/','')}" select="{replace(.,'http://id.loc.gov/authorities/names/','')}"/>
            </xsl:if>
        </xsl:for-each>
    </xsl:map>
</xsl:variable>
<xsl:stylesheet>
    <xsl:template>
        </xsl:stylesheet>
```



2.3. The Devonshire Manuscript- Enriching our TEI file : part I

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet
    xmlns="http://www.tei-c.org/ns/1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:map="http://www.w3.org/2005/xpath-functions/map"
    xmlns:tei="http://www.tei-c.org/ns/1.0"
    exclude-result-prefixes="xsl tei xs"
    version="2.0">
    <xsl:strip-space elements="*"/>
    <xsl:output method="xml" indent="yes"/>
```

```
<xsl:include href="5_xsl_map_person.xslt"/>

<xsl:template match="node()|@*">
    <xsl:copy>
        <xsl:apply-templates select="node() | @*"/>
    </xsl:copy>
</xsl:template>
```

```
<xsl:template match="listPerson[@type='historical']">
    <xsl:apply-templates select="person[@xml:id]"/>
</xsl:template>
```

```
<xsl:template match="person[@xml:id]">
    <xsl:variable name="myId" select="@xml:id"/>

    <xsl:copy>
        <xsl:apply-templates select="@*"/>
        <xsl:if test="map:contains($wikidata_pers_ref, $myId)">
            <xsl:element name="idno">
                <xsl:attribute name="type">WIKIDATA</xsl:attribute>
                <xsl:attribute name="ref">
                    <xsl:value-of select="$wikidata_pers_ref($myId)"/>
                </xsl:attribute>
            </xsl:element>
        </xsl:if>
        <xsl:if test="map:contains($isni_pers_ref, $myId)">
            <xsl:element name="idno">
                <xsl:attribute name="type">INSI</xsl:attribute>
                <xsl:attribute name="ref">
                    <xsl:value-of select="$isni_pers_ref($myId)"/>
                </xsl:attribute>
            </xsl:element>
        </xsl:if>
    </xsl:copy>
</xsl:template>
```

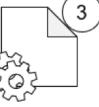
XSLT header, we include here the map that we made

As a default, we copy all the nodes

We define here the a special template for historical people

We get the mapping for adding idno tags





2.3. The Devonshire Manuscript- Enriching our TEI file : part II

```
<xsl:copy-of select="birth"></xsl:copy-of>
<xsl:copy-of select="death"></xsl:copy-of>

<xsl:for-each select="persName">
  <xsl:choose>
    <xsl:when test="(map:contains($loc_pers_ref, $myId) or map:contains($dnb_pers_ref, $myId)) and (ref[@type='LOC' or ref[@type='DNB']] ) ">
      <xsl:copy>
        <xsl:apply-templates select="@* | node() except ref"/>
        <xsl:choose>
          <xsl:when test="map:contains($loc_pers_ref, $myId)">
            <xsl:element name="ref" >
              <xsl:attribute name="type">LOC</xsl:attribute>
              <xsl:value-of select="$loc_pers_ref($myId)"/>
            </xsl:element>
          </xsl:when>
          <xsl:when test="ref[@type='LOC']">
            <xsl:copy-of select=". " />
          </xsl:when>
        </xsl:choose>
        <xsl:choose>
          <xsl:when test="map:contains($dnb_pers_ref, $myId)">
            <xsl:element name="ref" >
              <xsl:attribute name="type">DNB</xsl:attribute>
              <xsl:value-of select="$dnb_pers_ref($myId)"/>
            </xsl:element>
          </xsl:when>
          <xsl:when test="ref[@type='DNB']">
            <xsl:copy-of select=". " />
          </xsl:when>
        </xsl:choose>
      </xsl:copy>
    </xsl:when>
    <xsl:otherwise>
      <xsl:copy>
        <xsl:apply-templates select="node() | @*"/>
      </xsl:copy>
    </xsl:otherwise>
  </xsl:choose>
</xsl:for-each>
</xsl:copy>
</xsl:template>
</xsl:stylesheet>
```

We copy the birth and death tags

For each @ref we check if we have it in our maps, test to see if the @ref is already defined

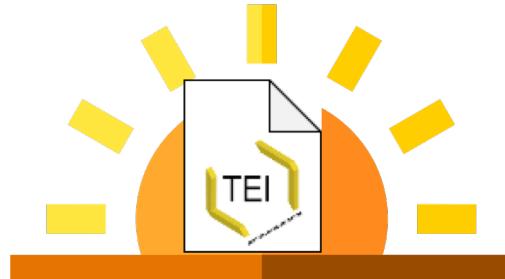
We list the persName

We copy tags without ref inside



2.4. The Devonshire Manuscript - At the end

Transform it and get the final enriched TEI file :



7_DM_P5_2020_Iter.xml

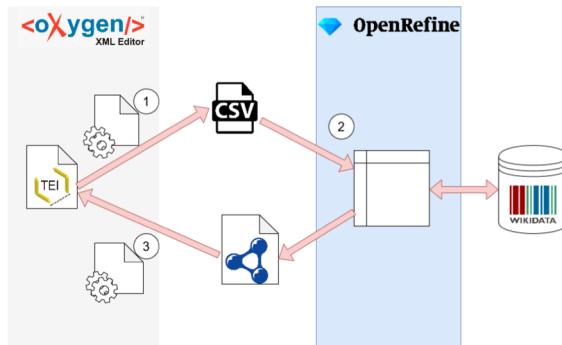
```
<idno xmlns="http://www.tei-c.org/ns/1.0"
      type="WIKIDATA"
      ref="http://www.wikidata.org/wiki/Special:EntityData/0314325.rdf"/>
<idno xmlns="http://www.tei-c.org/ns/1.0"
      type="INSI"
      ref="http://www.isni.org/0000000108928784.xml"/>
<idno xmlns="http://www.tei-c.org/ns/1.0"
      type="VIAF"
      ref="https://viaf.org/viaf/44340301/rdf.xml"/>
<birth notBefore="1502" notAfter="1504">1503</birth>
<death when="1542">1542</death>
<persName>
  <roleName type="honorific">Sir</roleName>
  <forename>Thomas</forename>
  <surname>Wyatt</surname>
  <addName type="epithet">the Elder</addName>
  <ref href="http://www.tei-c.org/ns/1.0" type="LOC">50019425.rdf</ref>
  <ref href="http://www.tei-c.org/ns/1.0" type="DNB">30111</ref>
</persName>
```

Note: Since the base Devonshire manuscript file was in a customized TEI format we couldn't write the « `xpath-default-namespace="http://www.tei-c.org/ns/1.0"` » command in the stylesheet header and it have as side effect to precise. As a result, In each tag that we had the namespace. if this is inconvenient for you, you can use « `Ctrl+H` » or « `replace all` » in your favorite text editor to delete it!

Conclusions

With this pipeline we could use Wikidata as a bridge for adding authority file URIs to the original TEI, but we still need to fight ambiguity, especially if we work on cultural named entities :

- we need to define if them are place/people/written works
- we need to get the maximal level of information that we could have from our TEI file to help with disambiguation



We also understand that every corpus is different : they were built in unique editorial contexts and they have different research purposes :
-> We still have to customise parts of our pipeline for get information to help with disabitguation and to inject the authroity URI that we get with our process back into the file/