How to enrich TEI corpus with Wikidata

An OpenRefine and XSL transformation workflow

Celian RINGWALD

THE HUMANITIES DATA LAB

Two Case Studies:

1. The CraikSiteIndex

University of Calgary

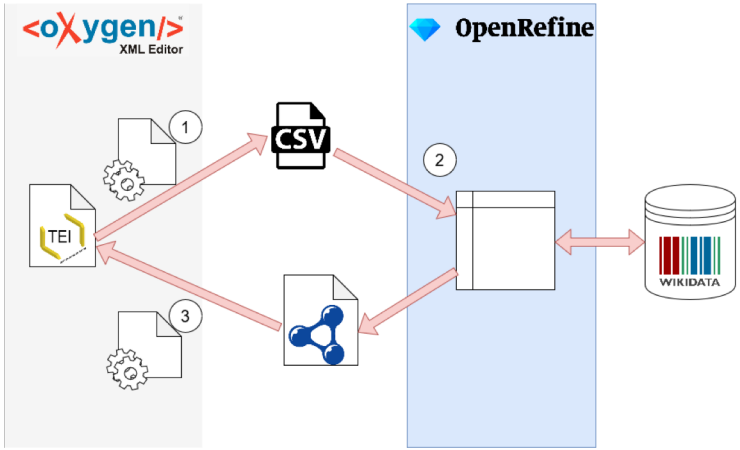
2. Devonshire Manuscript

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.0. 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 ?

<u>A Note About People</u>

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 Branxholme 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:schema="https://schema.org/"
    xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:tei="http://www.tei-c/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="'&quot;'" />
    <xsl:variable name="newline" select="'&#10;'" />

    <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 occumation 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:variable>
        <xsl:value-of select="$gender"/>
    </xsl:template>
    <xsl:template match="persName">
        <xsl:value-of select="$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

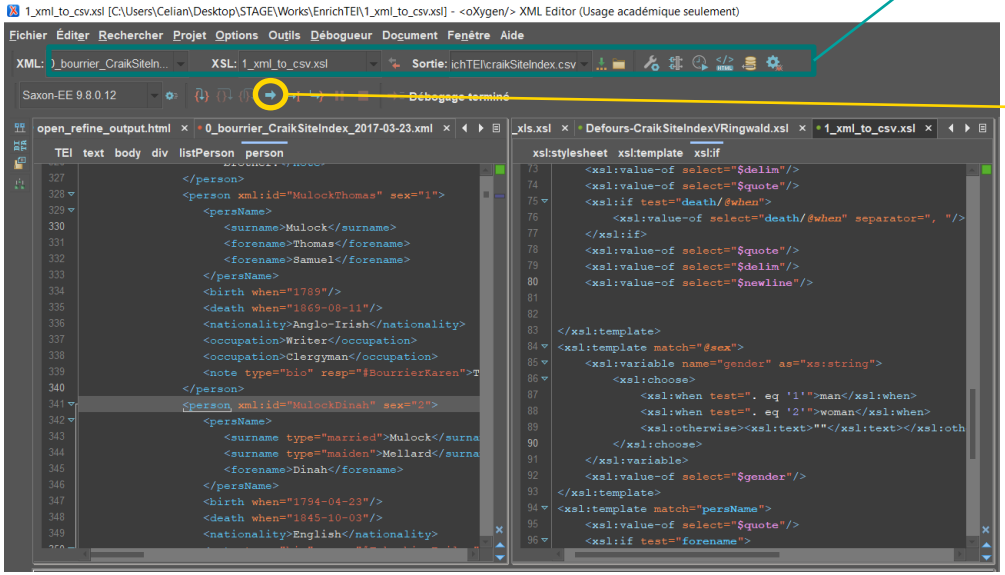
Template for the persName div

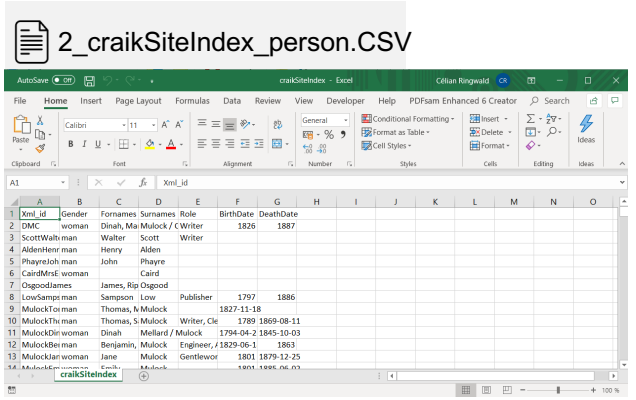
Specific process for the surname, in order to have each suurname @type appear in the output in the same order

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

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

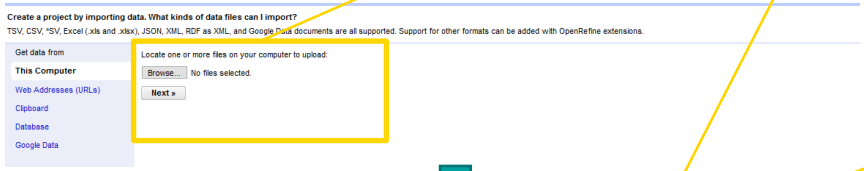
And go !

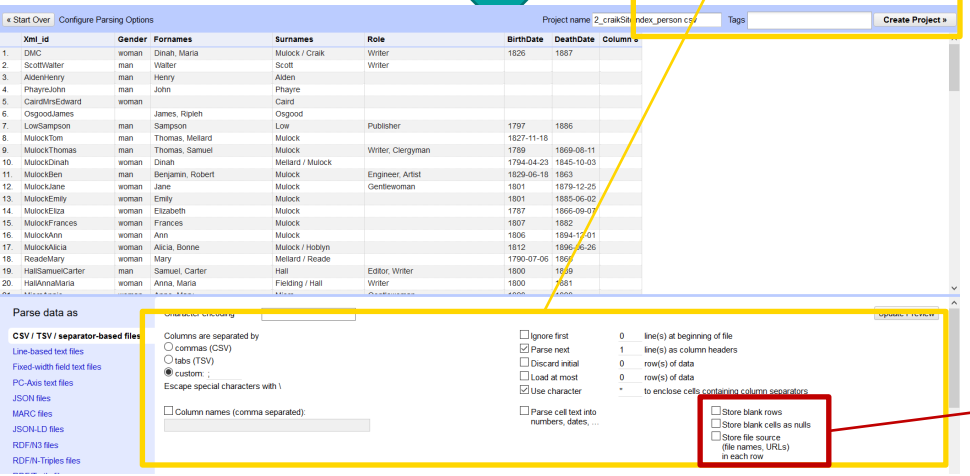


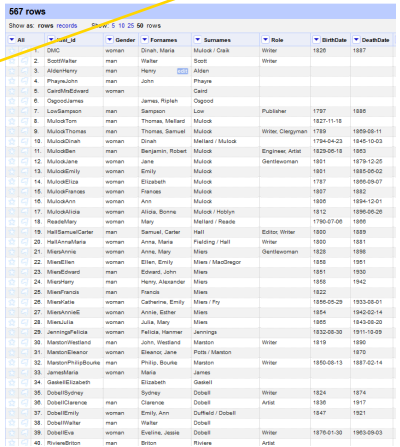


I.2. The Craik Site Index – OpenRefine - PART I : importing data

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





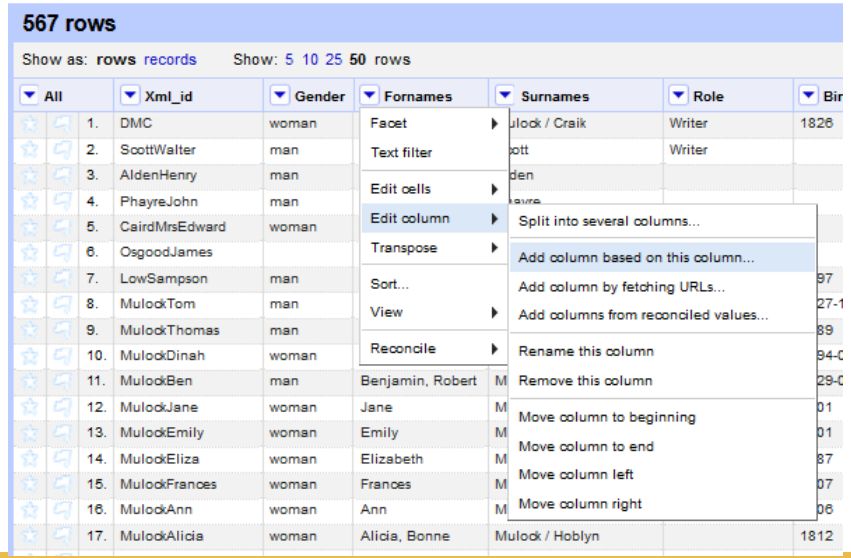


Important for next steps: unselect checkboxes here

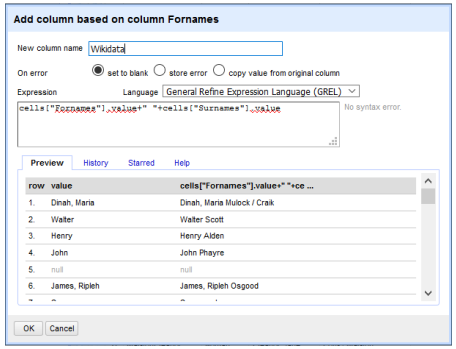
I.2. The Craik Site Index – OpenRefine - PART II : adding a column based on another

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…

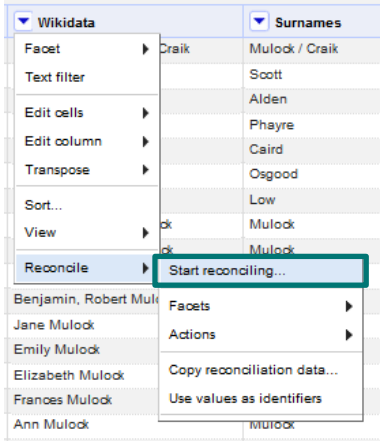


Thanks to the GREL langage :



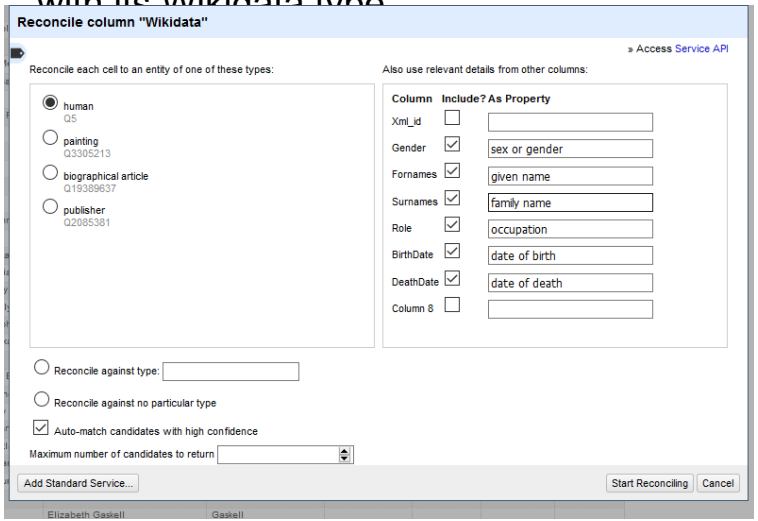
cells["Fornames"].value+" "+cells["Surnames"].value

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

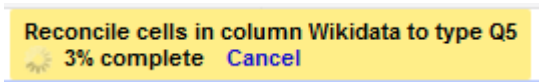


Then select Wikidata service and select « human » as reconcilation type…

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…

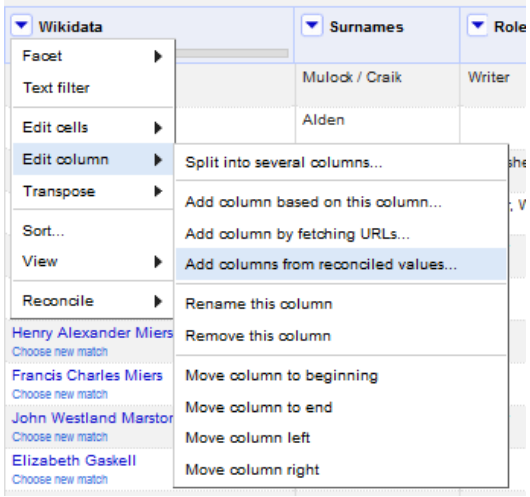


I.2. The Craik Site Index – OpenRefine - PART IV: getting more links !

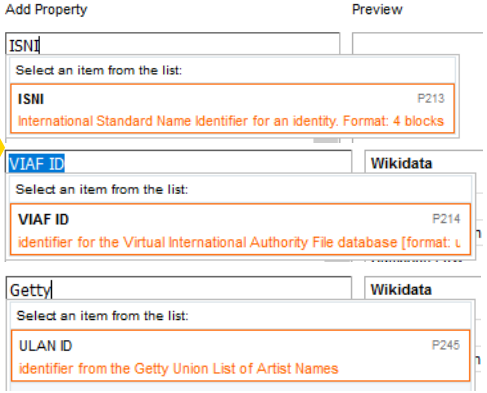
For this tutorial just filter the already matched data

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.

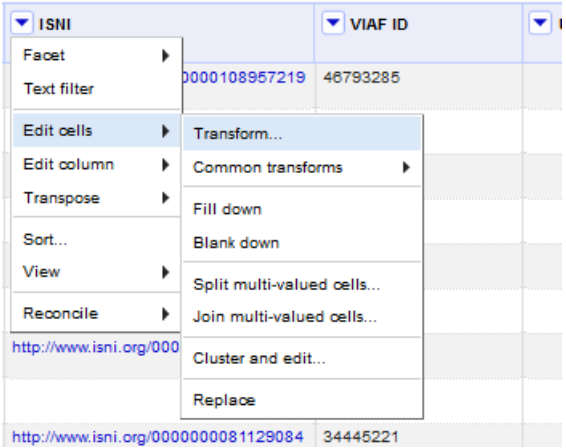
Then add column from renconciled values

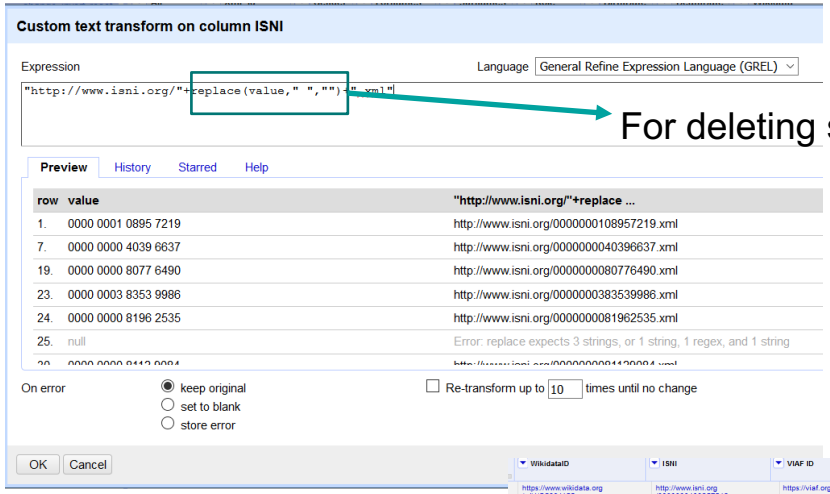


And add ISNI , VIAF and Getty IDs !



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

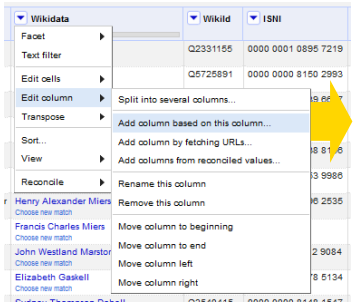


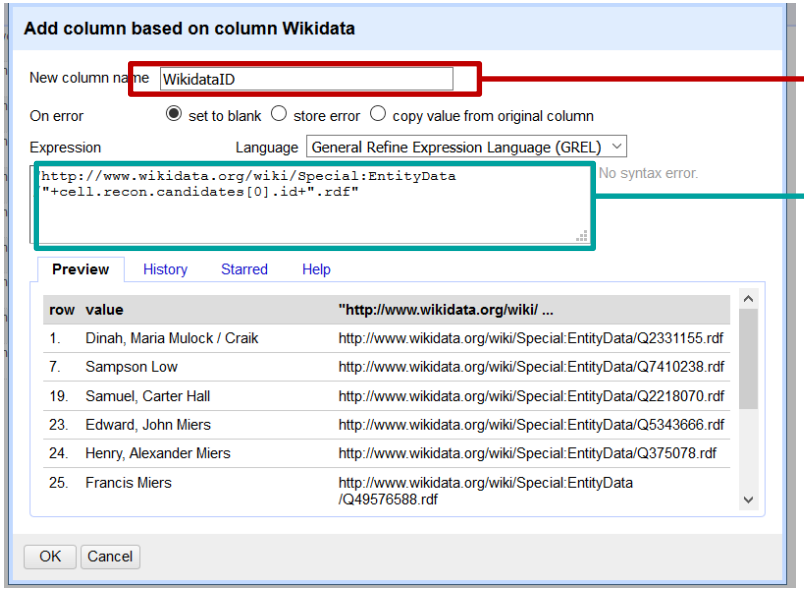


Machine readable urls :
ISNI : http://www.isni.org/$ID.xml
Getty : http://vocab.getty.edu/ulan/$ID.rdf
VIAF : https://viaf.org/viaf/$ID/rdf.xml

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?





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...

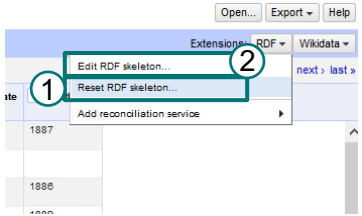
"http://www.Wikidata.org/wiki/Special:EntityData/"
+cell.recon.match.id+".rdf"

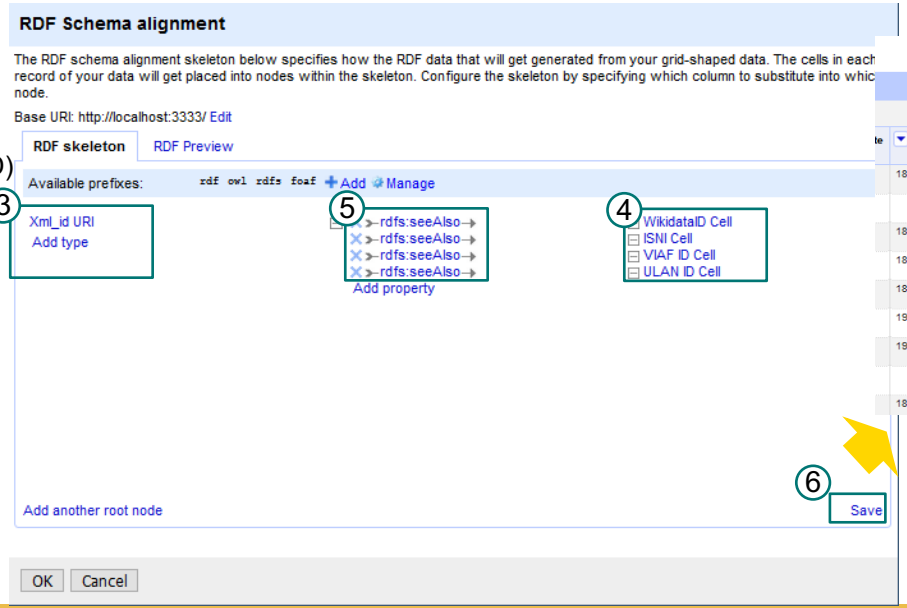
I.2. The Craik Site Index – OpenRefine - PART VII: Exporting results as a RDF file

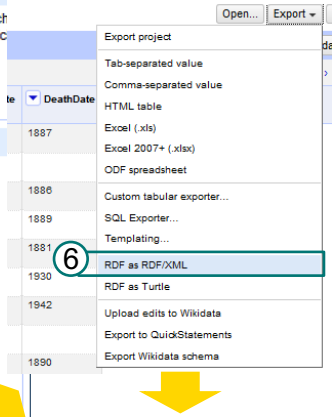
Why are using the RDF export schema?

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 interrest
   that we want to extract
   (WikidataID, INSI Cell, VIAF ID Cell, ULAN ID)
5- define each property
   as « rdfs:seeAlso »
6- Save
   & Export > RDF as RDF/XML







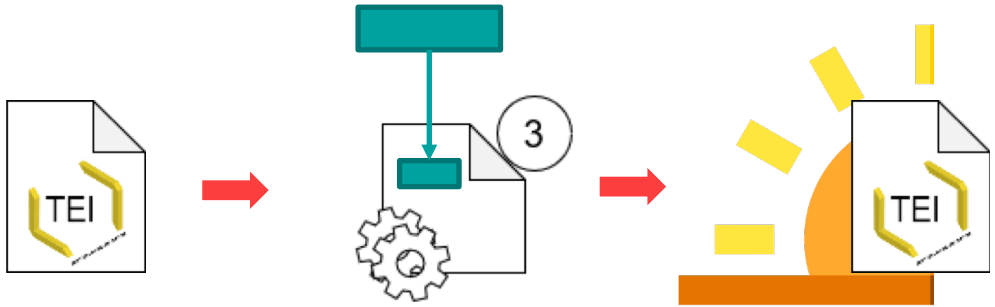
3_craikSiteIndex_rdf_person.rdf

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

The last part!

In this sectio we will present a first xsl file for mapping the data we've found in Wikidata as anxsl 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 orginial 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>
```

...

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 availablity 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>
```

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.

...

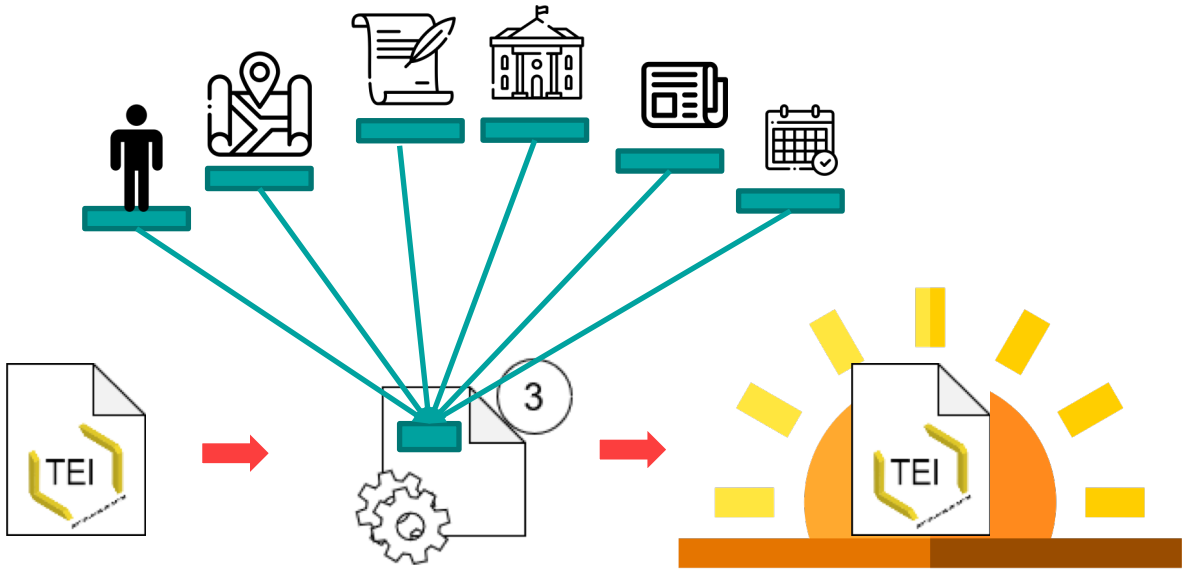
I.3. The Craik Site Index – going further

- 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 avaible in ‘going further’ directory)

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



6_tei_to_enrichedtei _all.xsl

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:xi="http://www.w3.org/2001/XInclude" 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>


            <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 spyede</title>,
                        attributed t <name key="WYATT">Sir Thomas Wyatt</name> in <title
                            type="book">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 teihHeader/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>
```

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 :

LIBRARY LIBRARY OF CONGRESS

Oxford DNB Oxford Dictionary of National Biography

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

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

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="'&#10;'" />

    <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>
...
```

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

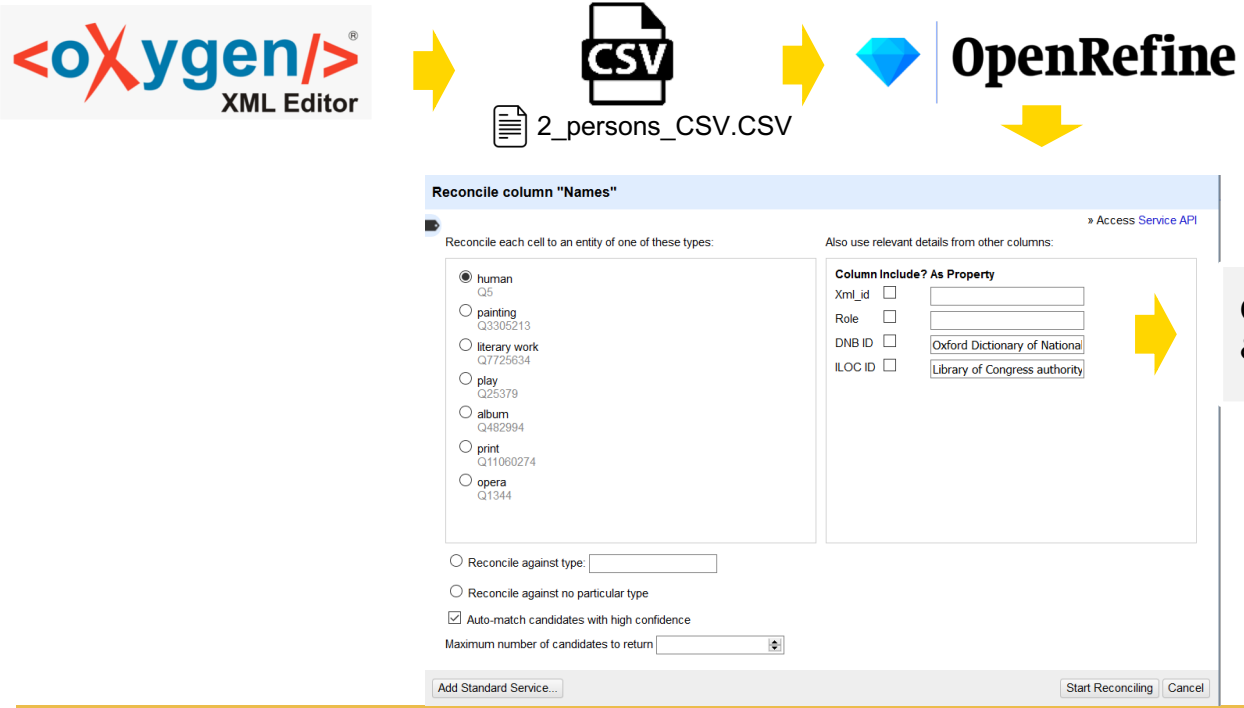
2.1. The Devonshire Manuscript– TEI TO CSV : XSLT file – PART III

```
...
    <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

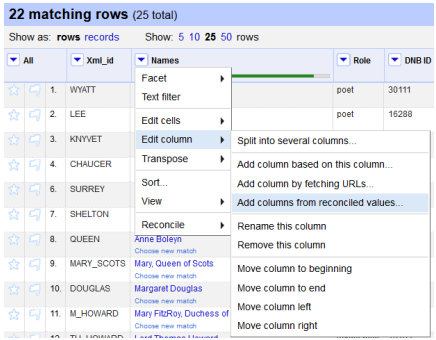


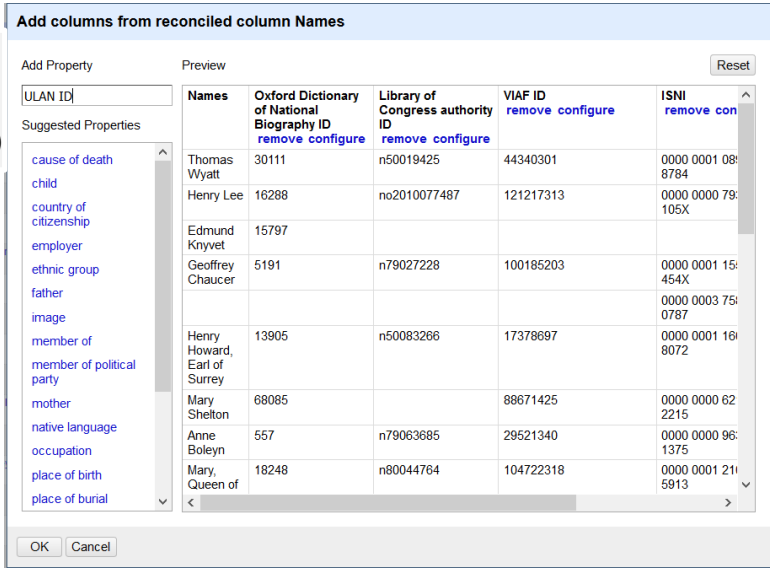
Oxford Dictionary of National Biography ID & Library of Congress authority ID

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;))

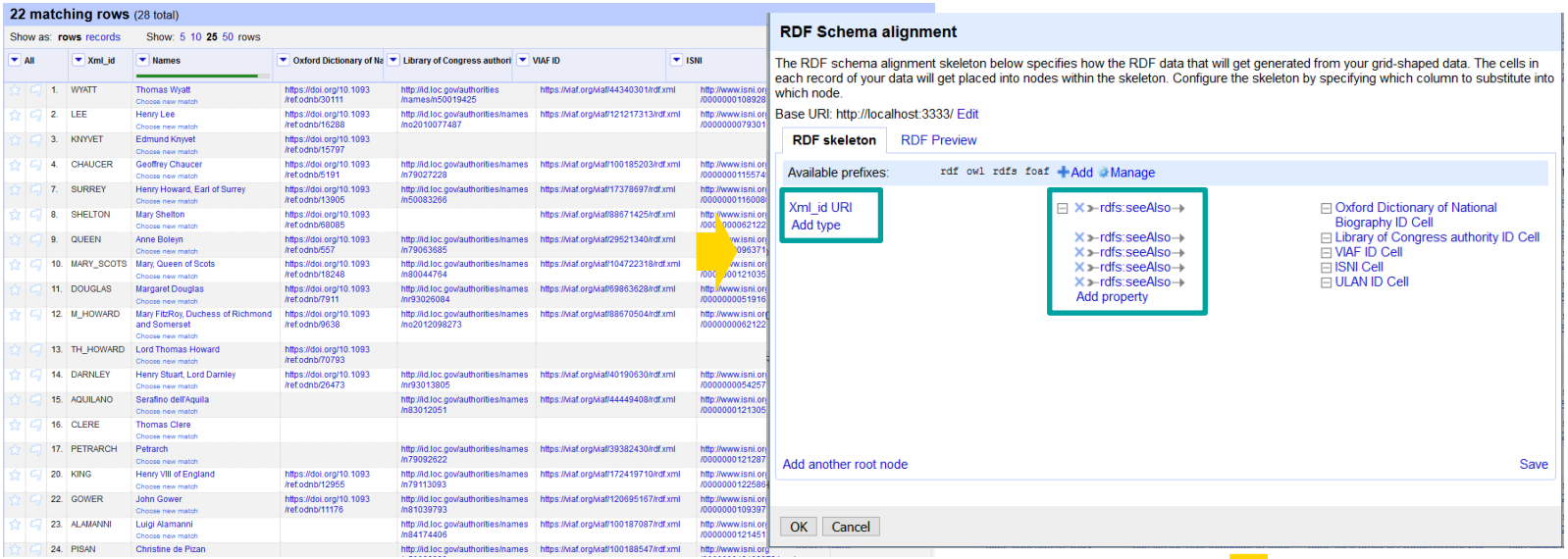




And then concatenate IDs with human readable urls as we saw slide 16 and 17
For Oxford DNB and LOD ID use :
« http://www.oxforddnb.com/view/article/$ID_DNB »
« http://id.loc.gov/authorities/names/$ID_LOD »

2.2. The Devonshire Manuscript- exporting it into RDF

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



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.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"
    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
            xmlns:xs="http://www.w3.org/2001/XMLSchema"
            version="3.0"
            >


        <axsl:variable name="viaf_pers_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_pers_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:variable name="isni_pers_ref" as="map(xs:string, xs:string)">
            <axsl:map>
                <xsl:for-each select="rdf:Description/child::*">
                    <xsl:if test="contains(text(),'isni')">

                        <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="getty_pers_ref" as="map(xs:string, xs:string)">
        <axsl:map>
            <xsl:for-each select="rdf:Description/child::*">
                <xsl:if test="contains(text(),'getty')">

                    <axsl:map-entry  key="'{replace(parent::node()/@rdf:about,'http://localhost:3333/','')}'" select="'{.}'"/>

                </xsl:if>
            </xsl:for-each>
        </axsl:map>
    </axsl:variable>
```

...

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

```
...

        <axsl:variable name="dnb_pers_ref" as="map(xs:string, xs:string)">
            <axsl:map>
                <xsl:for-each select="rdf:Description/child::*">
                    <xsl:if test="contains(text(),'ref:odnb')">

                        <axsl:map-entry  key="'{replace(parent::node()/@rdf:about,'http://localhost:3333/','')}'" select="'{replace(.,'https://doi.org/10.1093/ref:odnb/','')}'"/>

                    </xsl:if>
                </xsl:for-each>
            </axsl:map>
        </axsl:variable>
        <axsl:variable name="loc_pers_ref" as="map(xs:string, xs:string)">
            <axsl:map>
                <xsl:for-each select="rdf:Description/child::*">
                    <xsl:if test="contains(text(),'id.loc.gov')">

                        <axsl:map-entry  key="'{replace(parent::node()/@rdf:about,'http://localhost:3333/','')}'" select="'{replace(.,'http://id.loc.gov/authorities/names/','')}'"/>

                    </xsl:if>
                </xsl:for-each>
            </axsl:map>
        </axsl:variable>
        </axsl:stylesheet>
    </xsl:template>

</xsl:stylesheet>
```

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

```xml
<?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"
    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>
```

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:copy-of>
                            </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:copy-of>
                            </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

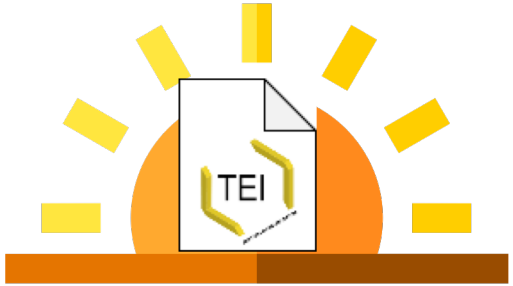
For each @ref we check if we have it in our maps, test to see if the @ref is already defined

We copy tags without ref inside

We list the persName

2.4. The Devonshire Manuscript - At the end

Transform it and get the final enriched TEI file :



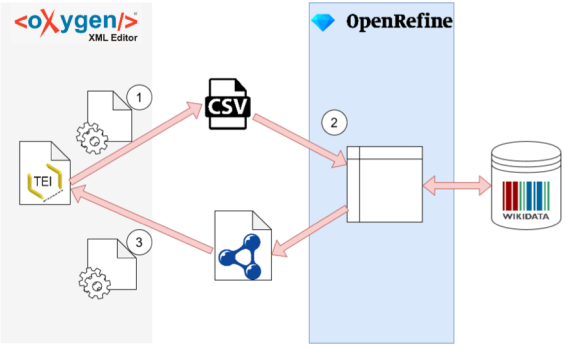
7_DMs_P5_2020_Iter.xml

```
<idno xmlns="http://www.tei-c.org/ns/1.0"
      type="WIKIDATA"
      ref="http://www.wikidata.org/wiki/Special:EntityData/Q314325.rdf"/>
<idno xmlns="http://www.tei-c.org/ns/1.0"
      type="ISNI"
      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 xmlns="http://www.tei-c.org/ns/1.0" type="LOC">n50019425.rdf</ref>
   <ref xmlns="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 authortity 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 maxmimal 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/