Antonello Mori
Salvetti's Network 1640-1642

Goals

The following project aims at analysing a restricted corpus of XML files containing documentation from the Medici Archive Project, concerning notices sent by the Florentine resident in London to the Grand Duke of Tuscany between 1641 and 1642. These files, thanks to their encoding, make it possible to carry out quantitative examinations of the places, characters and topics that appear in this documentation. The proposed analysis has been divided into two macro-phases:

1. The user will be provided with a restricted sample of XML to be inserted into the algorithm and from which the names of the characters present, the places and the topics will be extrapolated for each news item in the document. Using the extracted characters a social network is constructed, so as to illustrate how they are connected to each other within the document. The network is then subjected to a series of analyses, while the topics and places become the protagonists of very elementary quantitative analyses.
2. In the second phase the user is provided with two XLSX files, which have been generated by the internal platform of the Euronews Project, in which all the characters of the documents between 1641 and 1642 are present. These, in a first file, are related to each other according to the number of times a given character appears with another, while in the second file, the characters are related to topics, showing us how many times a given character is found within the entire corpus in news marked by a given topic. The first file allows us to generate a social network based on the interactions that these characters have with each other, accompanied by a series of inherent analyses, while the second file allows us to make simple quantitative analyses on the relationship between the protagonists of the documents and topics.

Documents

Once the aims of the project have been established, it is necessary to briefly illustrate the XML structure of this documentation in order to better understand how it will be implemented.

```
<newsFrom>
    <from>Palace of Westminster</from>
    <transc>Siando Hoggi il Venerdì santo poco si può scrivere. Si dirà sola
        , Thomas (Earl of Stafford)] per via di Decreto per Criminale di les
        Titolari, et in ultimo havuto il pacet del Re, altramente non porle
        diffidenza che la Camera Inferiore ha havuto di loro nel havere fatt
        credere che non sia mai per prestarvi il suo consenso se non tirator
        dalla fazione de puritani il più grande del Regnopiaccia a Dio che t
        fra medesimi Parlamentarij accomoderebbe tutto, perchè all'hora il R
        ripigliare la sua autorità.</transc>
    <newsPeople>Thomas Wentworth</newsPeople>
    <newsPeople>English Parliament</newsPeople>
    <newsTopic>Politics</newsTopic>
    <wordCount>count</wordCount>
    <position>1</position>
</newsFrom>
<newsFrom>
    <from date="30/04/1641">London</from>
    <transc>Arrivò martedì passato in questa Città il giovane Principe di Or
        direttamente alla Corte a fare riverenza a lor Maestà [Charles I Stu
        della Regina madre [Maria de' Medici-de Bourbon], con la quale passò
        della Corte di [propose reading :Arundox] Gran maresciallo, et tratt
    <newsPeople>William II van Oranje-Nassau</newsPeople>
```

The figure shows the structure of a document being analysed (doc_2.xml), from which it is possible to highlight the tags we are interested in:

- **<from>London</from>:** allows us to identify the place where the news comes from.

- **<newsTopic>Military</newsTopic>**: allows us to understand what the news event is about. It is possible to have more than one topic per news item.
- **<people>Charles I Stuart</people>**: allows us to identify the character mentioned in the story. There may be more than one.

The document as a whole is composed of a series of news elements delimited by the <newsFrom> tag, so by exploiting this structure, it is possible to analyse the protagonists, places and topics of each news element in the file.

First macro-phase development

The creation of this algorithm required the use of specific libraries for the type of analysis proposed, among which it is worth mentioning:

- Pandas, which allows the management and creation of DataFrames, i.e. tables.
- Networkx, which provides us with all the necessary functions for the creation and analysis of social networks.
- Xml.etree.ElementTree, which allows you to manage operations on uploaded xml files.

These were called up and two blocks of code were created to allow the upload of the files to be analysed and the "reading" of the chosen XML file.

```
#dichiarazione di tutte le librerie necessarie per l'esecuzione del codice
import pandas as pd
import networkx.algorithms as nxa
import networkx as nx
import matplotlib.pyplot as plt
import xml.etree.ElementTree as ET
import itertools

from google.colab import files #Istruzione che permette di caricare i file da analizzare
uploaded = files.upload()

tree=ET.parse('doc_2.xml')#Blocco che permette di leggere e creare una genialogia sul file XML
root= tree.getroot()
location=root.findall('newsFrom')
```

Once this had been done, a completely accessory block of code was added, allowing the user to view the structure of the individual document. We then moved on to create two functions necessary for the first macro-phase of the programme, namely the one allowing us to search the file for all the news from with their respective characters, and a function delegated to the creation of the edges of the social network, which makes use of the intertools library.

```
for i in root:#Blocco che permette di vedere nella
  print(i.tag, i.attrib)


def person_doc(var): #Funzione che viene richiamata
  doc_news=var
  frms=doc_news.findall('.//newsFrom')#element.find
  people_in_doc=[]
  for frm in frms:
     people_in_frm=[]
     x=frm.findall('.//newsPeople')
     for p_dc in x:
      people_in_frm.append(p_dc.text)
     people_in_doc.append(people_in_frm)
  return(people_in_doc)


def funz_edg(var2):#Funzione che richiamata ci perm
  f_network=var2
  f_network=itertools.combinations(f_network,2)
  for i in f_network:
    edg.append(i)
  return(edg)
```

This is followed by a block of code which we might call the main one, because this deals with creating the social network of the document in question using the characters present in the text, linking them together according to their presence in the individual news item. In this block the two previous functions are called, after which the list of characters is added as a node to the social network to which the previously found edges are then added. In the same block, in addition to all the instructions for the on-screen display of the said social network, we proceed to calculate the number of people present in the single document, the cliques, the density, the average clustering coefficient and the connected components. These operations are managed thanks to the Networkx library.

```
personaggi_net=person_doc(root)
prsng_tot=[]
for i in personaggi_net:
  prsng_tot.append(list(sorted(set(i))))
lista_pers = sorted(set(sum(prsng_tot,[])))
n_pers=len(lista_pers)#Calcolo delle persone nel documento
edg=[]
for i in prsng_tot:
  funz_edg(i)

network_documento=nx.Graph()#Blocco che crea il social network
network_documento.add_nodes_from(lista_pers)
network_documento.add_edges_from(edg)

comp=nx.connected_components(network_documento)#Istruzioni per il calcolo dei cliques
cliq=nx.find_cliques(network_documento)

print('Personaggi presenti nel documento',n_pers)#Serie di stampa a schermo di valori
print('Clique del network:', len(list(cliq)))
print('Densità del network:', nx.density(network_documento))
print('Coefficiente medio di clustering:', nx.average_clustering(network_documento))
print('Componenti connessi:', len(list(comp)))

plt.title('Social network according with documentation',fontsize = 20)#Istruzioni del
n_pers=nx.spring_layout(network_documento,k=2,scale=1, iterations=70)
nx.draw(network_documento,n_pers,node_color='r')
nx.draw_networkx_labels(network_documento,n_pers)
plt.show()
```

Next, with the help of the Networkx.algorithms library, the page rank, closeness, network diameter and degree were calculated with simple instructions.

The last point of this macro-phase consists of extracting places, people and topics from the document in order to make tables showing how many times they appear in a single document. In the figure below, we can see the block of code that allows the extraction and insertion into a list, followed by the creation of a table using the Panda library.

```
luoghi=[]
for item in root.iter('from'):#Istru
  luoghi.append(item.text)#Istruzion
  argomenti=[]
for item in root.iter('newsTopic'):#
  argomenti.append(item.text)#Istruz
personaggi1=[]
for item in root.iter('newsPeople'):
  personaggi1.append(item.text)#Istr
```

```
#Blocco per costruire una tabella che ci permette di vedere
df=pd.DataFrame(luoghi, columns=['Luoghi'])
df=df.groupby(['Luoghi']).size().reset_index(name="Volte")#
df
```

	Luoghi	Volte
0	London	3
1	London Tower	1
2	Palace of Westminster	4

Antonello Mori
Salvetti's Network 1640-1642

Second macro-phase development

As already mentioned in the introduction, the second macrophase makes use of XLSX documents containing the list of characters and the number of times they meet each other within the whole corpus under examination. First of all, this XLSX file has been assigned to a variable thanks to the Panda library which allows us to "read" it. The table, however, does not respond in order to the parameters that we need to work with the data contained, so through the instruction melt of the library Panda has been reformulated by placing in a column the character of reference, in another the character met and in the last the number of times.

```
norm_salvetti= pd.melt(table_salvetti, id_vars=['Personaggio di riferimento'], var_name='P. incontrato', value_name='Numero di volte')
norm_salvetti
```

	Personaggio di riferimento	P. incontrato	Numero di volte
0	Charles I Stuart	Charles I Stuart	0
1	Henriette Maria De Bourbon-Stuart	Charles I Stuart	71
2	Maria de' Medici De Bourbon-Stuart	Charles I Stuart	39
3	English Parliament	Charles I Stuart	102
4	Scottish Parliament	Charles I Stuart	26
...	...	...	...
479	James II Stuart	William Cecil	4
480	Herry Montagu	William Cecil	2
481	Ralph Hopton	William Cecil	7
482	Lucius Cary	William Cecil	2
483	William Cecil	William Cecil	0

484 rows × 3 columns

This new ordering made it possible to exploit the nx.from_pandas_edglist command to create the social network by assigning individual columns as nodes and edges, followed by a block of code to display the network.

```
G= nx.from_pandas_edgelist(norm_salvetti,source='Personaggio di riferimento',target='P. incontrato',edge_attr='Numero di volte',create_using=nx.DiGraph())

#Blocco per la stampa del network proveniente dal file XLSX
pers_doc=nx.spring_layout(G, k=8, scale=5, iterations=70)
nx.draw(G,pers_doc,node_color='r')
nx.draw_networkx_labels(G,pers_doc)
plt.title('Social network according with documentation',fontsize = 20)
plt.show()
```

Once again, network-related analyses followed, to which I refer you to the code for a more complete and orderly commentary on the whole.

The extracted data also allowed us to obtain tables with the mean and median of how many times a reference character appears on average with others within the corpus.

```
norm_salvetti.groupby("Personaggio di riferimento").mean() #B

norm_salvetti.groupby("Personaggio di riferimento").median()
```

The last phase of this project consists of assigning the XLSX file concerning the topics to a variable, as done previously, and from here reordering the table, in practically the same way as above, so as to have a column with the character, one with the topic and the last one with the number of times this character is present for a given topic.

```
#Blocco che ci consente di assegnare il file XLSX ad una variabile e visualizzarlo come tabella
upload="/content/topics.xlsx"
topics=pd.read_excel(upload)
topics
```

```
#Blocco che riorganizza la tabella in tre colonne ben distinte
norm_topics= pd.melt(topics, id_vars=['Personaggio'], var_name='Topic', value_name='Numero di volte')
norm_topics
```

In summary, we proceeded to repeat the same mean and median analyses described above on the topics, while also adding a block of code that prints a table on the screen showing how many times a single character appeared in a selected topic, in this case war.

```
#Blocco che stampa quante volte ogni singolo per
war=norm_topics[norm_topics['Topic']=='Guerra']
war
```

Conclusions

The work presented here is not intended to be a product of extreme complexity, but rather a simple exercise that allows one to search for elements within XML documents, with the possibility of also generating a social network. Regarding the second macro-phase, this one unfortunately suffers from the impossibility to access the whole XML corpus, falling back on already extracted data, but in the presence of the documentation it would have been sufficient to repeat, with appropriate modifications, what was done in the first macro-phase, to obtain a complete analysis of the corpus..