Volume_17_SocialNetwork

June 20, 2021

1 Understanding the Social Networks of Emma B. Andrews

In this notebook, we will use Python to parse the Emma B. Andrews diaries TEI files. Our interest is to visualise the social networks in the life of Emma B. Andrews life on the nile. To understand these networks, we will use several text mining features to extract TEI elements (`<persName>`) as well as analyse the grammatical structure of the jounral entry to explore the social graph of Emma B. Andrews.

To accomplish our work, we will use several modules. The modules are as follows:

- csv
- Beautiful Soup 4
- lxml
- matplotlib
- nltk

The Beautiful Soup, lxml, NLTK, and Matplotlib modules need to be installed. If you are running Jupyter Notebook with the Python Virtual Environment, then these modules were installed when you created the Virtual Environment. The `csv` module comes preinstalled in the Python virtual environment.

1.1 Import Modules (Dependencies)

To import a module, we will use the Python import function.

[22]:
```
import csv # Python's Comma Separate Values Parser
from bs4 import BeautifulSoup # Beautiful Soup is for parsing HTML and XML files
import lxml # lxml is a secondary parser for beautiful soup
import nltk # Natural Langauge Toolkit
import re # Python's Regular Expression Module
from afinn import Afinn
```

1.2 Read Volume into Python to Parse with Beautiful Soup

The tagged TEI files of the Journals are located in the `/diary-volumes` directory. We need to tell Python the source of the file. We will want to use the Python OS module to make this work for either Windows or Mac. For now, it is hard coded.

[2]:
```
```

```
# Since the journal volume we want exists in the same directory as our Jupyter␣
 ↪Notebook, we can use the document name with extension.
journal = '../diary-volumes/volume17.xml'

# Now we want to create a Beautiful Soup object with our file. We will unpack␣
 ↪what this means in more detail below.
with open(journal) as xml:
    soup = BeautifulSoup(xml, 'lxml-xml')
```

1.3 Analyse Diary and Create Data Model

The Diaries are encoded according to the `TEI` standards. Thus, the `<text>…</text>` element encloses the contents of the dairy. We want to parse every day of the dairy and then further manipulate the data for Graph Analysis. As of now, a core interest is to understand the social network of Emma B. Andrews. Thus, we will process each volume, in order to create a data model. We can further use the data model in Gephi to visually represent the social network along our nodes of data. At this stage our nodes are:

- The date of writing. This is not necessarily the date Emma encountered the person, but the day she wrote about the person.
- The name of the person. Here we rely on the TEI tagging element of `persName` to aggregate each person as a node.
- Entry text. We strip the text of the TEI at this point and keep the person anchored to the writing of Emma B. Andrews. We will use the Entry text for further natural language processing to populate the relation field. (See discussion below)
- We analyse the sentiment of each entry for its overall score. We store the sentiment score in Entry_Sentiment.

1.3.1 Format Entries

Each child within the `<text>` root is an entry according to the day. Thus, we need to iterate over the `div` elements within root.

[29]:
```
# To extract the daily entries, we need to traverse the text root and gather␣
 ↪together all <div> elements with a type of entry. To do this, we will use
# the beautiful soup library. While we are working through each entry, we also␣
 ↪pass each entry through the sentiment analysis
# to score the entirety of the entry. We will improve on sentiment in␣
 ↪connection to people at another step.

# Set the Lexicon for Afinn Lexicon
afinn = Afinn(language='en')

with open('networks.csv', 'w', newline='') as f:
    writer = csv.writer(f)
    writer.writerow(["Date", "Name", "Relation", "Entry", "Entry_Sentiment"])
    for i in soup.find_all("div", {"type": "entry"}):
```

```
        #Extract the Date for the Graph Model
        match = re.search('EBA-([0-9--]+)', i.attrs['xml:id'])
        date = match.group(1) if match else None

        #Clean the Entry and Prepare for Post-Processing
        remove_newlines = re.sub("\n+", " ", i.text.strip())
        plain_text = re.sub(" +", " ", remove_newlines)

        #Extract all the PersName and Score Entry in which PersName appears
        people = i.find_all('persName')
        if not people:
            writer.writerow([date, "None", "None", "None", "None"])
        else:
            for p in people:
                if ' ' in p['ref']:
                    a = p['ref'].split(' ')
                    for b in a:
                        afinn_scr = afinn.score(plain_text)
                        writer.writerow([date, b, "", plain_text, afinn_scr])
                else:
                    afinn_scr = afinn.score(plain_text)
                    writer.writerow([date, p['ref'], "", plain_text, afinn_scr])
```

1.4 Relational Analysis

We are now in a position to analyse the entry so as to extract additional information about the relationship between Emma B. Andrews and her social networks. We are going to use natural language processing tools to analyse the language Emma B. Andrew's used apropos each

[ ]: