Data Visualization

Jacob Moose
De Katholieke Universiteit Leuven (KUL)
Introduction to Digital Humanities
Master's of Digital Humanities

I hereby declare that, in line with the Faculty of Science's code of conduct for research integrity, the work submitted here is my own original work and that any additional sources of information have been duly cited.

January 8, 2024

1 Introduction

In Assignment 1, my group was tasked with cleaning data for a series of periodicals published predominately (though not exclusively) in Belgium between the years 1924 and 2013. Using OpenRefine, we organized this data by removing bibliographic metadata and splitting multi-valued cells. Our goal was to prevent an excessive number of columns from forming within our dataset while simultaneously allow key groupings of information to remain together. In terms of the information we highlighted, we focused on the most fundamental periodical figures such as ID numbers, titles, places of publication, publisher names, dates of publication, frequency of publication, and more.

In this report, I discuss my efforts to visualize this data. In particular, I focus on network analysis and four tools I employed in order to showcase various relationships between each periodical's place of publication and publisher. In "Results and Discussion of the Data Analysis" (Section 3), I also reflect on the following three conclusions: 1.) When data visualization is the priority, approaches to OpenRefine can drastically change, 2.) Python is the most efficient way to label large numbers of Nodes/Edges for Gephi, and 3.) A comprehensive list of Nodes/Edges in Gephi may lead to a robust network, but it can potentially muddle the results.

2 Description of the Tools Used

The four tools I used to develop my data visualizations were OpenRefine, Python, Excel, and Gephi. Gephi was the most important of these, as it actively produced visualizations for several networks. Nevertheless, the full benefits of Gephi cannot be fully realized without the help of the other data-organization tools that, for instance, help prepare the lists of Nodes and Edges. In this section, I describe each of these four tools and demonstrate why and how each of them were relevant to my work. Since OpenRefine was also used in Assignment 1, I only briefly reflect on it.

2.1 OpenRefine

OpenRefine is a data cleaning tool that is designed to help users organize messy and unstructured data. In general, the application allows for a substantial amount of superfluous data to be eliminated without compromising the integrity of relevant information; this importantly relieves users from the need to sift through cells individually. Additionally, the tool includes a "cluster and edit" feature that scans through each column to identify data that may pertain to the same object but differ due to potential errors (such as a misspelled title) or syntactic discrepancies (such as different date formats). During my second-round of data cleaning, this feature turned out to be very important, though it did not instinctively recognize some clusters that were necessary for accurate visualizations (see Section 4).

2.2 Python

Python is a programming language that allows users to quickly analysis extensive amounts of data. By importing clean data and performing different queries, users can learn much more about their datasets in quantitative ways. After working in OpenRefine again, I exported my newly cleaned data to a .csv file, which was then opened and analyzed in Python. The different queries I coded were then used to quickly assign Node and Edge IDs to pertinent information. Considering there was a large amount of periodicals (over 1,100), cities (over 350), and publishers (over 600), Python saved me from having to spend a tremendous amount of effort and time preparing the data for Gephi. Of course, this means I did not directly create data visualizations through Python, something I could have utilized if I approached this assignment differently. For this project, the tool's benefits were more organizational than anything else.

2.3 Excel

Excel is a basic spreadsheet program developed by Microsoft. It works to organize data through different tables and charts. After creating a series of Nodes and Edges in Python, this information was then saved as a .csv file, opened in Excel, and further separated into a series of worksheets. Each of these worksheets included different combinations of Nodes and Edges which could be uploaded into Gephi and actively used for creating the visualizations (See Section 3 for more on this). Similar to my use of Python, this work was primarily employed for data organization than performing the visualizations itself.

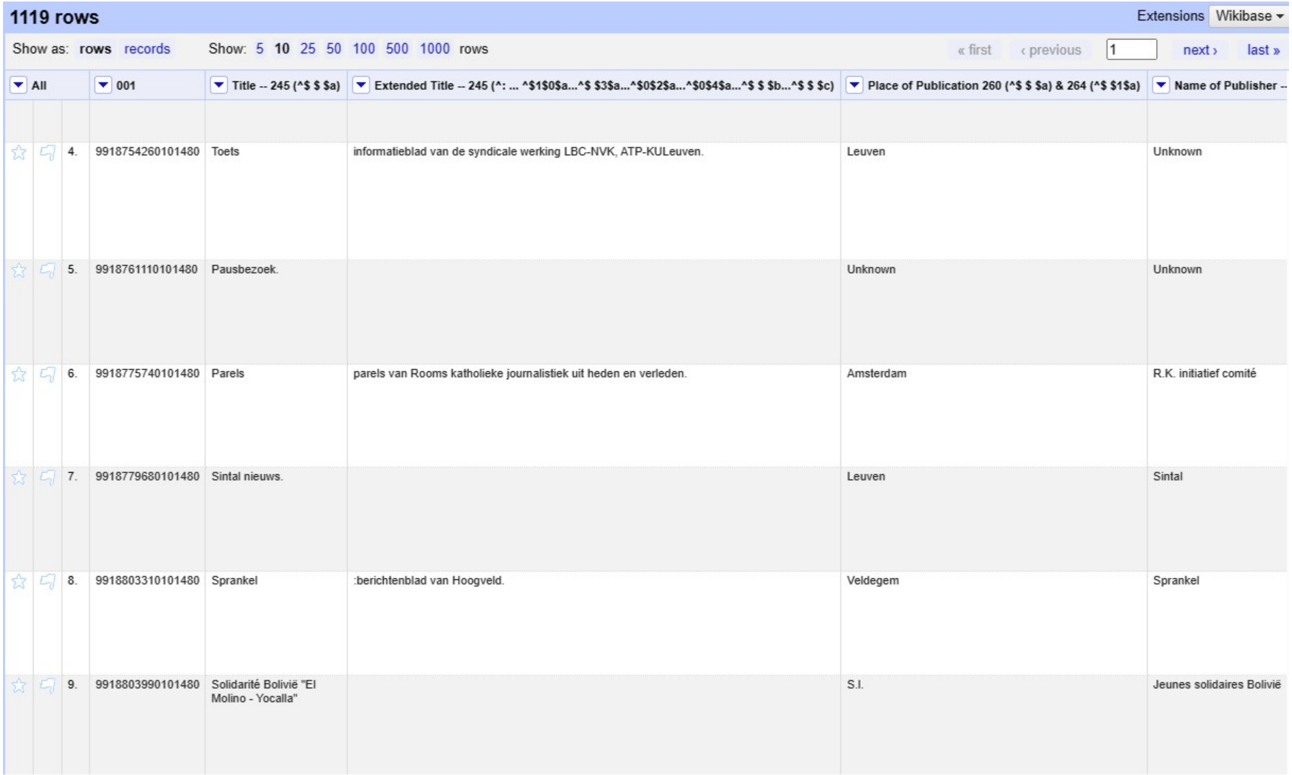
2.4 Gephi

Gephi is a tool for network analysis that allows users to actively visualize the structures of different networks. The tool takes a clean list of Nodes (individual data entities) and Edges (the relationship between the Nodes) and turns that information into a visualized network. Importantly, Gephi includes a number of different layouts the Nodes/Edges can be displayed under - for this project, I utilized *Fruchterman Reingold,* a force-directed layout that highlights the attraction between connected nodes and the repulsion between disconnected ones.

Additionally, Gephi allows users to add different weights to the Edges or customize the size of the Nodes based on key statistics, such as centrality (i.e., the number of Edges each Node relates to). This is beneficial for the user as it helps reveal unique entities or relationships in the dataset. For instance, changing the Node size in this project was especially effective in illustrating the places and publishers with the highest number of publications.

3 Results and Discussion of the Data Analysis

In Section 1, I introduced three key conclusions derived from my analysis: 1.) When data visualization is the priority, approaches to OpenRefine can drastically change, 2.) Python is the most efficient way to label large



1119 rows

Extensions Wikibase

Show as: rows records   Show: 5 10 25 50 100 500 1000 rows   « first ‹ previous 1 next › last »

All	001	Title -- 245 (^$ $ $a)	Extended Title -- 245 (^: ... ^$1$0$a...^$ 3a...^$0$2$a...^$0$4$a...^$ $ $b...^$ $ $c)	Place of Publication 260 (^$ $ $a) & 264 (^$ 1a)	Name of Publisher --
4.	9918754260101480	Toets	informatieblad van de syndicale werking LBC-NVK, ATP-KULeuven.	Leuven	Unknown
5.	9918761110101480	Pausbezoek.		Unknown	Unknown
6.	9918775740101480	Parels	parels van Rooms katholieke journalistiek uit heden en verleden.	Amsterdam	R.K. initiatief comité
7.	9918779680101480	Sintal nieuws.		Leuven	Sintal
8.	9918803310101480	Sprankel	:berichtenblad van Hoogveld.	Veldegem	Sprankel
9.	9918803990101480	Solidarité Bolivië "El Molino - Yocalla"		S.l.	Jeunes solidaires Bolivië

Figure 1: *(OpenRefine) Second-Round of Cleaning*

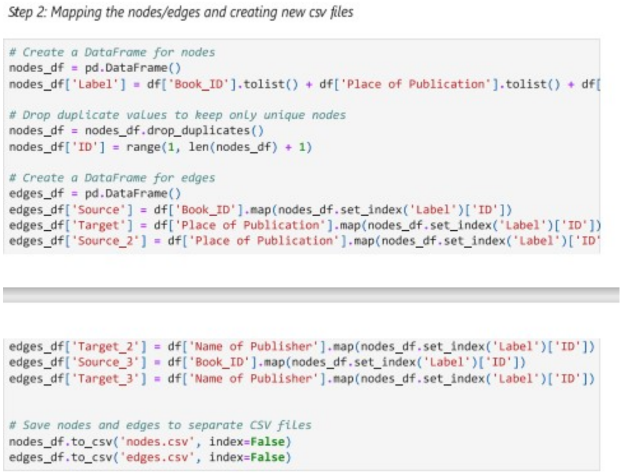
numbers of Nodes/Edges for Gephi, and 3.) A comprehensive list of Nodes/Edges in Gephi may lead to a robust network, but it can potentially muddle the results. In this section, I return to each of these insights by discussing them in further detail.

- *1.) When data visualization is the priority, approaches to OpenRefine can drastically change*

In Assignment 1, we prioritized organizing our data based on how it looked *within* OpenRefine and ignored some more practical approaches that would have been beneficial for data visualization. For instance, by splitting key information based on a multi-valued cell approach rather than column-splitting approach, our dataset was legible and easy to read, but also problematic and inconvenient for, say, creating Nodes and Edges. By returning to OpenRefine and separating the data according to a column-splitting approach (See Figure 1), this problem was solved. This may ultimately create more columns, but that may not be as big of a problem as we originally thought. By following this new approach, each column contains one, specific topic (such as Place of Publication), making individual information more analyzable.

- *2.) Python is the most efficient way to label large numbers of Nodes/Edges for Gephi*

Creating the Nodes and Edges from the second round of data cleaning would have been extremely difficult without the help of Python. While each of the periodical IDs are unique (there are no two periodicals that are exactly the same), places of publication and publishers are often repeated (i.e., two different periodicals could be published in the same city or by the same publisher). Moreover, given that Node IDs need to be remembered for each of the Edges, it is nearly impossible to go through the list of records and create the Edges by hand. Using Python (See Figure 2), I was able to create my Node/Edges list in 4 easy steps. First, I created a DataFrame for the Nodes; Second, I dropped the duplicate values to keep only unique Nodes; Third, I created a DataFrame for the Edges; and Fourth, I



Step 2: Mapping the nodes/edges and creating new csv files

```
# Create a DataFrame for nodes
nodes_df = pd.DataFrame()
nodes_df['Label'] = df['Book_ID'].tolist() + df['Place of Publication'].tolist() + df[

# Drop duplicate values to keep only unique nodes
nodes_df = nodes_df.drop_duplicates()
nodes_df['ID'] = range(1, len(nodes_df) + 1)

# Create a DataFrame for edges
edges_df = pd.DataFrame()
edges_df['Source'] = df['Book_ID'].map(nodes_df.set_index('Label')['ID'])
edges_df['Target'] = df['Place of Publication'].map(nodes_df.set_index('Label')['ID'])
edges_df['Source_2'] = df['Place of Publication'].map(nodes_df.set_index('Label')['ID'
```

```
edges_df['Target_2'] = df['Name of Publisher'].map(nodes_df.set_index('Label')['ID'])
edges_df['Source_3'] = df['Book_ID'].map(nodes_df.set_index('Label')['ID'])
edges_df['Target_3'] = df['Name of Publisher'].map(nodes_df.set_index('Label')['ID'])


# Save nodes and edges to separate CSV files
nodes_df.to_csv('nodes.csv', index=False)
edges_df.to_csv('edges.csv', index=False)
```

Figure 2: *(Python) Creating the Nodes/Edges*

saved this information to separate .csv files. I was then able to merge this data into a shared excel document, further preparing my data for Gephi.

- *3.) A comprehensive list of Nodes/Edges in Gephi may lead to a robust network, but it can potentially muddle the results.*

Having a clean set of Nodes and Edges in a saved Excel workbook made Gephi visualizations rather simple. The data for the Nodes/Edges was easily imported and subsequently transformed into visible networks. Nevertheless, choosing which information to highlight proved to be difficult. In the end, I created four different networks (Figures 3-6) that showcase different publication information.

My first visualization (Figure 3) highlights the direct relationship between Periodicals and Publishers and the undirected relationship between Publishers and Places of Publication. Of the latter, edges were weighted based on their total frequency - in other words, each edge is thicker based on the total number of times a publisher published in a select city. This approach is perhaps the most complete of the four visualizations, but also the most messy due to the large number of "1-to-1" relationships (i.e., the cities/publishers who only have one corresponding Edge).

To address the messy nature of Figure 3, Figure 4 is more selective and only high-

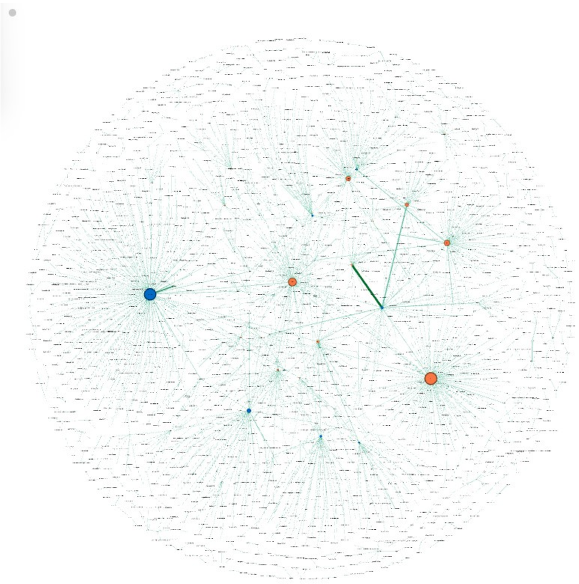


Figure 3: *(Gephi) Periodical, Publisher, Place*

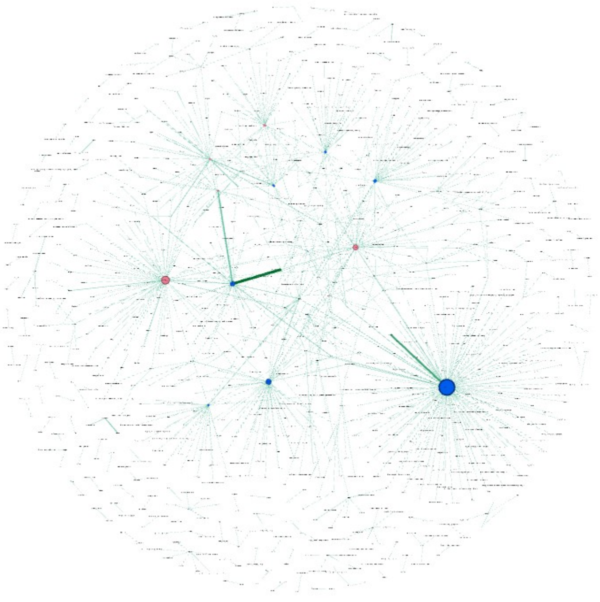


Figure 4: *(Gephi) Publisher, Place*

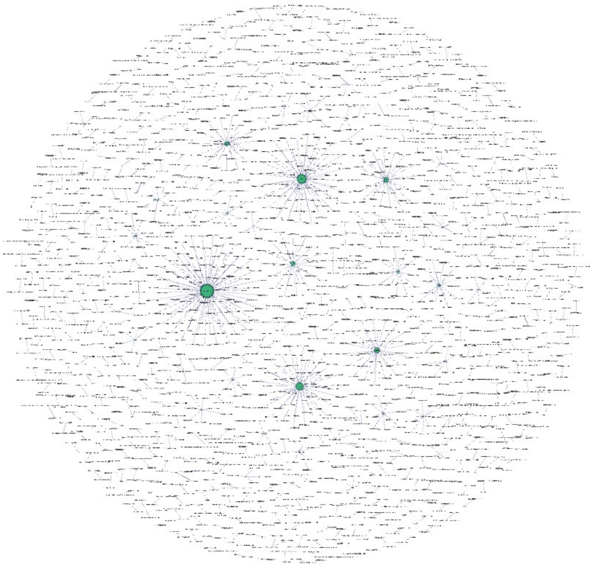


Figure 5: *(Gephi) Periodical, Publisher*

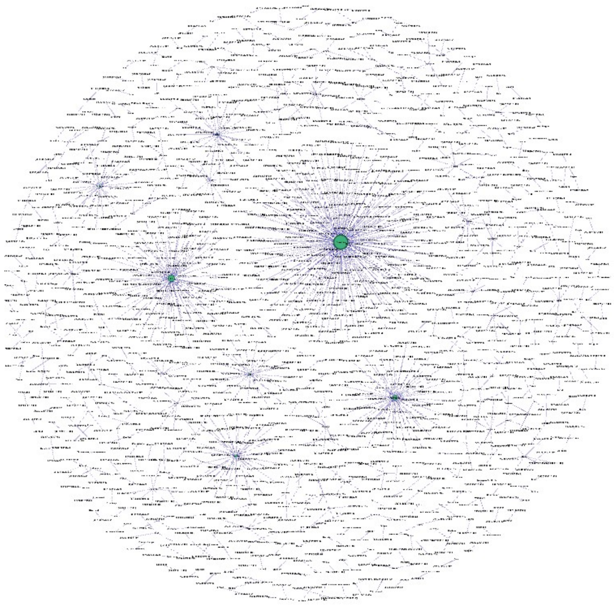


Figure 6: *(Gephi) Periodical, Place*

lights the relationship between Publishers and Places of Publication. In this visualization, it is easier to see the major publishing houses or key places for publishing. In particular, the thicker edge between CEPESS and Brussels demonstrates the frequent collaboration between these two entities. Nevertheless, the information only implicitly highlights periodical information, as they are no longer listed as individual Nodes in the network. If I want to know more about the individual periodicals relate to publishers/places of publication, new visualizations must be made.

In Figures 5 and 6, I highlight the relationship between each periodical and their publisher/place of publication. For all of these edges, the weight will be the same as there are no duplicates (i.e., each periodical has only one publisher and place of publication). Similar to Figure 4, these visualizations also highlight the major publishing houses/places of publication, but simultaneously include each of the periodical IDs. Looking at both the Node sizes and general centrality, We see Davidsfonds as one of the major publishing houses and Brussels as a key site for publication.

In the end, even though Gephi is a fairly simple tool to use once Nodes/Edges have been cleanly defined, tough decisions must still be made. In the four visualizations I developed, each highlight different aspects of the cleaned data and can be used to tell different narratives. In particular, I found Figures 4, 5, and 6 the most important, as the large amount of data in Figure 3 was perhaps too messy.

4 Assessment and Evaluation of the Tools

Of the four tools I used to organize my data, Python was the most helpful. As I have already mentioned twice in this report, defining the Edges and Nodes lists would have been nearly impossible without it. Even though it still took some time to figure out the correct code, it was well worth it in the end. Similarly, Excel was also helpful in cleanly preparing the data for Gephi. By creating multiple worksheets, I could clearly define my Nodes and Edges for each of the four visualizations. (For a more detailed look at the work done in Excel, please see the attached document titled "Assignment 2 Cleaned Data".)

Once I had data the cleaned, Gephi was an easy and exciting program to use. Though it is rather limited in layout availability, there are still a number of interesting options that can be used to structure network visualizations. I would return to this program in the future and use it for other analyses.

To end, I would like to add a few comments on OpenRefine. While this program was ultimately helpful in organizing the data, there were a number shortcomings that made this assignment feel very tedious. For instance, while the cluster and edit feature could find some important spelling mistakes, it was unable to see the connection between abbreviated publishing houses and those that were fully spelled out. On top of this, many of the publishing houses were named according to a major publishing house *and* the city they were published in (i.e. CVP Roeselare). Even if this refers to an individual branch of CVP, leaving these as individual Nodes would drastically skew my network. To solve this, I had to comb through the Name of Publisher columns individually, a rather time-consuming task. While it may be more fair to criticize how the data was inputted rather than the application itself, it nonetheless reveals a significant limitation.