A computational approach to gender asymmetries and gender homophily in Wikipedia

Trung Nguyen
Aalto University, School of Science

Abstract
The aim of this project is to revise the gender bias in Wikipedia manifested by the lack of diversity of the editorial community, most of which are men editors. We also introduce a different approach to study the notability of men's and women's politicians as well as an analysis on the homophily of the articles' network. Two main results are observed: (i) the higher notability for women's profiles is evident in the network structure of Wikipedia, and (ii) strong homophily is observed between men's articles whereas female's articles are indicated to be non-homopholic.

Contents

1 Introduction

Gender inequality in academic contexts has been a central topic of several studies and research in recent years. One prominent and controversial example for this is Wikipedia: although this open-source platform is designed to be an objective and unbiased source of knowledge, several studies have pointed out indications of gender gaps on the platform, which stemmed from the narrow diversity of the male-dominated editor community. A thorough examination of this phenomenon is therefore necessary so as to better understand the mechanism of gender biases online, as well as to raise awareness of these issues on Wikipedia and the internet as a whole.

In this project, we examine gender gaps and bias in Wikipedia articles and biographies in particular, both qualitatively and quantitatively. Section 2 of this project includes a literature review on potential gender inequalities in Wikipedia articles across multiple dimensions, some of which are notability, topical focus and linguistic bias[1]. In the paper, several data analyses is carried out to confirm the introduced hypotheses regarding the glass ceiling[1]. Section 3 sets out to reapply some of the analyses mentioned earlier on a different dataset from Wikipedia. For this section, the analysis chosen for implementation is articles' network construction to compare notability difference. Furthermore, in this section we expand the scope of analysis to further study gender asymmetry from the dataset. In particular, one hypothesis that we are going to test is that there exists homophily in Wikipedia: articles and biographies sharing the same genders will be more likely to link to each other than to different genders. Section 4 will conclude what we have covered in this project and discuss possible further studies from the results obtained.

2 Review of Literature

An overview of the paper's primary goals, methodology, results and a discussion on possible follow-up analysis.

2.1 Primary goals

Regarding gender asymmetries in Wikipedia, one prominent study is *Women through the glass ceiling: gender asymmetries in Wikipedia*[1] published in 2016. In this study, the authors acknowledge the subtle gender biases on Wikipedia caused by the nature of the editorial community, which has been addressed earlier by several studies[2] [3] [4]. Using the literature as the backbone, the aim of this paper is to "assess potential gender inequalities in Wikipedia articles", which included several data analyses as confirmation for the proposed hypotheses. There are four main questions/hypotheses this paper is trying to address:

i. Is Wikipedia using the same thresholds and criteria to decide which men and women to be depicted on the site, or is the threshold higher for women in general?

ii. Are there any topical biases regarding articles about men or women, that is if there are any over-represented topics for articles about women?

iii. Is there linguistic bias in how men and women are depicted on Wikipedia?

iv. Is there any difference in the structural properties of articles about men and women?

These questions are assessed on a computational basis to provide empirical insights on the phenomenon. In particular, for each question a suitable method is chosen to process and model the data to verify if the hypothesized statements regarding gender bias are indeed significant. Finally, from the findings, potential actions are proposed to reduce gender bias in the Wikipedia editorial community.

2.2 Methodology

The key method to assess systematic gender asymmetry is to trace out the differences in the way that the two genders are treated and presented. The paper used a computational approach to compare biographies about men and women in Wikipedia across multiple dimensions. In particular, for question (i), global notability is assessed from an internal and external perspective. That is, public interests are used as a proxy measures to reflect the interests of Google users and editors, and from here the number of language editions and Google search volume are investigated in return to study the unconscious application of different thresholds. For question (ii) and (iii), the idea is that family-, gender-, and relationship-related topics are more prevalent in women biographies, and Linguistic Intergroup Bias (LIB) theory[5] suggests that language used to describe women's success tends to be more abstract (the use of more adjectives) while that used to describe failure tends to be more concrete (the use of more verbs). The lead section (also called the overview) of the biographies are compared, and for the topical bias, topics such as gender, relationship and family are observed. As for linguistic bias, a lexicon-based approach and syntactic annotations proposed by Otterbacher[6] is applied to detect abstract and subjective language. Finally, question (iv) is studied by constructing a hyperlink network between URLs of articles, and the results obtained are analyzed to study the visibility and reachability difference between articles about men and women.

2.3 Results overview

From the data analysis, the results have indicated the following evidences on gender asymmetries:

- There exists different thresholds for different genders in the decision on depiction of biographies on Wikipedia. In other words, fewer women' biographies are likely to be included on Wikipedia than men due to the higher thresholds imposed on them. This is present in several dimensions, including the high men-women ratio of editions in different languages and low Google search trends for women biographies, especially before the 1900s. One direct consequence of this is that the high thresholds for women means on average women are slightly more notable than men on this platform. This effect produced by Wikipedia's entry barriers is a subtle form of glass ceiling, and it is more profound considering profiles with less notability than that of globally notable people, where the entry difference has manifested gender bias and imply a non-objective approach from the editorial community.

- There are significant differences in primary topics covered in men's and women' s biographies, where the lead sections of women's articles are mostly associated with family, gender and relationship, whereas for men the topics are more diverse, such as politics and sports. The significance is even stronger for biographies with birth dates before the 1900s, and similar results hold for analysis in five other different languages.

- Semantic abstraction analysis has shown a clear difference between the language used in men's and women's biographies. In particular, by applying one-tailed chi-square tests, logit transformation and beta regression, the data indicates a stronger ratio of abstract language used for men than concrete language, while the opposite is true for women. It also means that for women, their failures tend to be generalized while their successes are described in concrete statements, and this aligns with the LIB theory proposed in the method.

- Structural inequality exists for both meta-data and the network structure of the articles. The investigation of meta-data has shown biases exist as several attributes are used more for men than women, which reflects both the editorial bias and the inequality context of historical documentation. Meanwhile, the investigation of hyperlink networks reveals a bias in ranking algorithms made by Wikipedia editors, leading to fewer women's biographies listed in the subset of top-ranked biographies, and this bias is more evident for biographies with birthdates before the 1900s.

2.4 Further possible approaches

Based on the findings of the paper and assuming an ideal scenario where unlimited access to data and compute resources are available, there are several possible approaches that we can take to further investigate the gender asymmetries on Wikipedia. In particular, one of the approaches is to correlate different causes for gender biases by studying the homophily and the dynamics of the network structure with a larger and more diverse dataset. Since inequality can be caused by both the subjective editorial processes and the respective manifestation of itself on Wikipedia, understanding the dynamics of the network will give us a better insight on the causes of gender biases, and with unlimited data and computational resources we can verify our hypothesis on a larger scale and better generalize the mechanism of gender asymmetry on online platforms. Later on in this project we will implement a small-scale preliminary analysis on this topic: we will analyze the homophily of politicians' articles to identify if there is gender clustering on Wikipedia, and from here we can reach some conclusions on gender homophily for this specific group, setting the grounds for more complex further analyses on the topic.

3 Computational analysis

A short revision of the methods discussed in the paper and a follow-up statistical analysis on gender homophily regarding gender bias in Wikipedia.

3.1 Tools

For this project, we limit our scope of analysis to articles of male and female politicians on Wikipedia. Our source of data is the database of politicians' biographies available from Wikipedia, where the list of politicians is first obtained from *GESIS - Leibniz-Institute for the Social Sciences*[7]. The genders of the politicians are then cross-validated with the gender database available at Carnegie Mellon University, and after the validation the data is fetched straight from Wikipedia website. A final data point for each politician would consist of several attributes, but one main attribute of interest for our analysis is the hyperlinks to other politicians for network analysis. The code is implemented in Python on Jupyter Notebook, where the construction of networks is aided by the open-source library NetworkX. The full implementation is included along with this report in the Appendix.

3.2 Revision of prior analysis

The first part of our analysis is an introduction to a new approach to the analysis of question (i) in Section 2.1. In the paper, the notability of women on Wikipedia is analyzed by studying the languages of editions and Google search trends. Here we will take a network analytical approach: a person is defined to be more notable if there is more links to other people in the hyperlink network, since notability is implied to be correlated to the influence to other people in the network. In order words, if the average degree of female nodes in our network is higher than those of male, then the conclusion is correct. A more formal construction is by letting d_i^M and d_i^F to be the degree of a male and female node, and N_M and N_F are the numbers of male and female nodes, then the above statement is equal to:

$$\frac{1}{N_F}\sum_i d_i^F \geq \frac{1}{N_M}\sum_i d_i^M \tag{1}$$

Figure 1 illustrates our network of politicians' Wikipedia articles. This network consists of 6880 nodes with 12094 edges (no self-edge is included), including $N_M = 2835$ male nodes and $N_F = 386$ female nodes. Applying equation (1) in our model allows us to calculate the average degree for male nodes is $\bar{d}^M \approx 4.088$, and the average degree for female nodes is $\bar{d}^F \approx 7.176$. Since $\bar{d}^F > \bar{d}^M$, we can conclude that the hypothesis given from the paper holds true for our dataset and method.

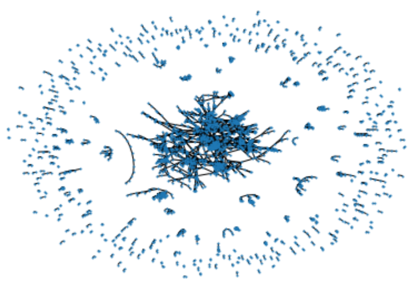

Figure 1: The hyperlink network of the politicians' Wikipedia article.

3.3 Gender homophily

3.3.1 Methodology

In our context, gender homophily is a principle describing the clustering of nodes (articles) of the same genders in our hyperlink network. In other words, gender homophily implies that biographies about men on Wikipedia will be more likely to link to those of other men, and the similar applies to women's biographies. To study the homophily of our Wikipedia articles, for simplicity we will first modify our hyperlink network such that only nodes with identified genders (male and female) and have at least one edge connected to it (smallest degree is 1) are kept. This reduces our original network down to 1573 nodes and 5736 edges.

Next we will construct a model to verify our hypothesis. The underlying idea is that if our network is homophilic then there would be fewer connections between nodes of different genders than when the network is randomized. Therefore by theorizing a threshold for homophily, we can compare our empirical value to check if the value is approximately close to the threshold (homophilic) or not. The theoretical threshold is constructed by first considering the total number of edges E as a sum of all possible types of edges:

$$E = e_{MM} + e_{FF} + e_{FM} \tag{2}$$

where e_{MM} is the number of edges connecting male-male articles, e_{FF} is the number of edges connecting female-female articles, and e_{FM} is the number of edges connecting female and male articles. If we consider female nodes separately, the random state can be thought of as a node has the same probability of connecting to a female node as to a male node. In other words:

$$e_{FF} \approx e_{FM}$$

Similarly, a male node would also has the same probability of connecting to a male node as to a female node, or $e_{MM} \approx e_{FM}$. In the case of absolute and perfect randomization, we have the relationship between the nodes is:

$$e_{MM} = e_{FF} = e_{FM} \tag{3}$$

Therefore, the number of edges connecting female and male nodes can be derived from the total number of edges by:

$$e_{FM} = \frac{E}{3} \tag{4}$$

For a network of $E = 5736$ edges, our threshold will be $e_{FM} = 1912$ female-male edges. Taking a 10% margin of error, we can testify that our network is homophilic if:

$$e_{FM} - \bar{e}_{FM} > 0.1 e_{FM} = 191.2 \tag{5}$$

where $\bar{e}_{FM}$ is the empirical value of female-male edges. The absolute value is omitted since $e_{FM} \leq \bar{e}_{FM}$ would indicate a stronger connection between different genders and therefore if $e_{FM} - \bar{e}_{FM} \leq 0$ then our network is not homophilic.

The threshold e_{FM} can be used to study the homophily of the whole network, but in order to study the homophily of each gender, we have to implement one extra step. The total number of connections made by male nodes would include both male-male and male-female connection, which is:

$$E_M = \bar{e}_{MM} + \bar{e}_{FM}$$

Hence, our network is male homophilic if the female-male edges $\bar{e}_{FM}$ is significantly smaller than the male-male edges, or in other words:

$$\frac{\bar{e}_{FM}}{E_M} = \frac{\bar{e}_{FM}}{\bar{e}_{MM} + \bar{e}_{FM}} \ll \frac{1}{2} \tag{6}$$

Using a 10% margin of error, the inequality $\bar{e}_{FM}/E_M < 0.45$ would satisfies our male homophily condition. Similarly, $\bar{e}_{FM}/E_F < 0.45$ would satisfies our female homophily condition, where $E_F = \bar{e}_{FF} + \bar{e}_{FM}$. The code implementation is available in the Appendix.

3.3.2 Results

From our new network, we have the empirical value of different types of edges are $e_{MM} = 3794$, $e_{FF} = 320$, and $e_{FM} = 1622$. The total number of connections made by male nodes is $E_M = \bar{e}_{MM} + \bar{e}_{FM} = 5416$, and for female nodes the number is $E_F = \bar{e}_{FF} + \bar{e}_{FM} = 1942$. First we can check that for this new network, equation (1) still holds for the notability of male and female in the network. The new average degree for male nodes is $\bar{d}^M \approx 6.669$, and the new average degree for female nodes is $\bar{d}^F \approx 11.781$. Since $\bar{d}^F > \bar{d}^M$, the same conclusion that female is more notable than male in general still holds for our new network.

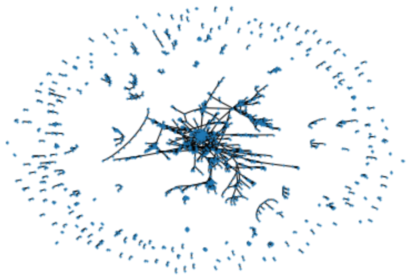

Figure 2: The new hyperlink network with only two genders and no zero-degree nodes.

Next, we analyze the overall homophily of the whole network. Using equation (5) we can see that:

$$e_{FM} - \bar{e}_{FM} = 1912 - 1622 = 290 > 191.2$$

Since the difference between the empirical data and the threshold is larger than 10%, this network is confirmed to be homophilic by a moderate extent. However, when we take a closer look at the homophily for each gender there is a striking difference. First the analysis for male homophily is:

$$\frac{\bar{e}_{FM}}{\bar{e}_{MM} + \bar{e}_{FM}} \approx 0.299$$

We can see that homophily applies to male nodes since the calculation is smaller than the threshold of 0.45. However, the calculation for female homophily yeilds an opposite trait:

$$\frac{\bar{e}_{FM}}{\bar{e}_{FF} + \bar{e}_{FM}} \approx 0.835$$

This is well above the perfect non-homophily threshold of 0.5, meaning that there exists very weak homophily (or non-homophilic) in our female nodes, contradicting to our initial hypothesis. The existence of only male homophily therefore both reflect the historical contexts of social networks and verify the structural inequality created by the editorial community. Since women's biographies, especially before the 1900s, are less well-documented in general, it is more likely that a female relationship with another female is also less well-documented than a female-male relationship. It is also possible that the structural inequality, in particular the ranking algorithms, also plays a role in the imbalance of the female-male and female-female relationship. However, the extent of influence between these two factors and the extra objective factor of imbalance dataset is not obvious from the analysis, and to fully understand the underlying mechanism of the non-homophily of female nodes, more complex and intricate methods are needed for accurate conclusions.

4 Conclusions

In this project, we have reviewed on the mechanism of several methods for analyzing gender asymmetries in Wikipedia, as well as provided some further analyses to verify the conclusions from the paper. A calculation on the average degree of the articles' hyperlink network agrees to the statement that women on Wikipedia is in general more notable than men as a direct consequence of the glass ceiling effect. An analysis on gender homophily, on the other hand, has shown an interesting result: strong homophily is detected among articles about men whereas the opposite applies to articles about women. This can be a consequence of several attributing factors, including the nature of imbalance dataset, the manifested gender bias in the ranking algorithms, and the historical contexts of biographical documentation. However, it would take more studies of larger scale and higher complexity in order to accurately pinpoint the mechanism of this asymmetry.

The results of this project can be used as starting points for further studies regarding gender homophily and gender bias on online platforms in general. Possible next steps may include an in-depth investigation on the gender homophily in general, where the dataset is not limited to politicians and the analysis is extended to non-binary genders. We hope that by further the research on these phenomena, the mechanism of gender inequality can be better understood and better guidelines can be applied to mitigate the negative effect of unconscious gender bias in Wikipedia and other online platforms.

References

[1] Claudia Wagner, Eduardo Graells-Garrido, David Garcia, and Filippo Menczer. Women through the glass ceiling: gender asymmetries in wikipedia. *EPJ Data Science*, 5(1), March 2016.

[2] Shyong K. Lam, Anuradha Uduwage, Zhenhua Dong, Shilad Sen, David R. Musicant, Loren G Terveen, and John Riedl. Wp:clubhouse? an exploration of wikipedia's gender imbalance. pages 1–10, 2011. 7th Annual International Symposium on Wikis and Open Collaboration, WikiSym 2011 ; Conference date: 03-10-2011 Through 05-10-2011.

[3] Benjamin Collier and Julia Bear. Conflict, criticism, or confidence: An empirical examination of the gender gap in wikipedia contributions. In *Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work*, CSCW '12, page 383–392, New York, NY, USA, 2012. Association for Computing Machinery.

[4] Benjamin Mako Hill and Aaron Shaw. The wikipedia gender gap revisited: Characterizing survey response bias with propensity score estimation. *PLoS ONE*, 8(6):e65782, June 2013.

[5] Anne Maass, Daniela Salvi, Luciano Arcuri, and Gün R. Semin. Language use in intergroup contexts: The linguistic intergroup bias. *Journal of Personality and Social Psychology*, 57(6):981–993, 1989.

[6] Jahna Otterbacher. Linguistic bias in collaboratively produced biographies: Crowdsourcing social stereotypes? *Proceedings of the International AAAI Conference on Web and Social Media*, 9(1):298–307, August 2021.

[7] Claudia Wagner. Personal profile info from dbpedia and politician network from wikipedia, 2017.

Appendix

The code is implemented as part of the project. This notebook template is a material of course CS-E4730 Computational Social Science at Aalto University, which is available for download at A+ via plus.cs.aalto.fi. The notebook also contains functions to fetch data from wikipedia and parse through some of the source data.

For Wikipedia API see documentation here: https://pypi.org/project/Wikipedia-API/

```
[1]: import wikipediaapi, time, json,requests,os
     #from tqdm import tqdm # you can import this for progress bar instead␣
      ↪if you are not using notebooks
     from tqdm.notebook import tqdm

     def ensure_person_data():
         """Ensures the existence of the person-data.tsv file.

         For downloading the file 'person-data.tsv', please go to https://
      ↪search.gesis.org/research_data/SDN-10.7802-1515

         Raises:
             Exception: If the person-data.tsv file is not found in the␣
      ↪current directory.
         """
         if not os.path.isfile("person-data.tsv"):
             raise Exception("For downloading the file 'person-data.tsv',␣
      ↪please go to https://search.gesis.org/research_data/SDN-10.
      ↪7802-1515")

     def ensure_gender_data():
         """Ensures the existence of the gender data file and downloads it␣
      ↪from a remote URL if it is not found.

         The file is downloaded from http://www.cs.cmu.edu/~ark/bio/data/
      ↪wiki.genders.txt
         """
         if not os.path.isfile("wiki.genders.txt"):
             print("Downloading the gender data file...")
             open('wiki.genders.txt', 'wb').write(requests.get("http://www.
      ↪cs.cmu.edu/~ark/bio/data/wiki.genders.txt", allow_redirects=True).
      ↪content)
```

```
def␣
↪filter_persons_by(occupation=None,birth_less=None,birth_more=None,nationality=None
↪
    """
    Filters persons from the person-data.tsv file based on specified␣
↪criteria.

    Args:
        occupation (str, optional): The occupation of the person.␣
↪Defaults to None.
        birth_less (int, optional): The upper bound of the birth year␣
↪of the person. Defaults to None.
        birth_more (int, optional): The lower bound of the birth year␣
↪of the person. Defaults to None.
        nationality (str, optional): The nationality of the person.␣
↪Defaults to None.

    Returns:
        dict: A dictionary of persons that match the specified criteria.
↪ The keys are the person names and the values are
        dictionaries containing the person's attributes.
    """
    ensure_person_data()
    pfile=open("person-data.tsv",'r')
    titles=pfile.readline().strip().split("\t")
    i=0
    persons={}
    for line in pfile:
        person=dict(zip(titles,line.strip().split("\t")))
        if person["birthDate"]=='NA':
            birthYear=None
        else:
            birthDate=person["birthDate"].strip("[]\t' ")
            birthYear=int(birthDate.strip("-").split("-")[0])
            if birthDate[0]=="-":
                birthYear=-birthYear

        occupation_ok=occupation==None or occupation in␣
↪person["occupation"]
        nationality_ok=nationality==None or nationality in␣
↪person["nationality"]
        birth_less_ok=birth_less==None or birthYear!=None and␣
↪birthYear<birth_less
        birth_more_ok=birth_more==None or birthYear!=None and␣
↪birthYear>birth_more
```

```python
        if occupation_ok and nationality_ok and birth_less_ok and␣
 ↪birth_more_ok:
            name=person["WikiURL"][len("http://en.wikipedia.org/wiki/"):
 ↪]
            persons[name]=person
    return persons

def get_genderdata():
    """Reads a tab-separated file containing Wikipedia article␣
 ↪information and returns a dictionary of gender data.

    The function reads a file named "wiki.genders.txt" and extracts the␣
 ↪gender data for each name in the file, using the first letter of the␣
 ↪gender field. The gender data is then stored in a dictionary with␣
 ↪the name as the key and the gender abbreviation as the value.

    Returns:
        A dictionary containing gender data for each name in the file.

    Raises:
        FileNotFoundError: If the input file cannot be found or opened.

    Example:
        >>> gender_data = get_genderdata()
        >>> gender_data['Albert_Einstein']
        'M'
    """
    ensure_gender_data()
    genderdata={}
    with open("wiki.genders.txt", "r") as inputfile:
        inputfile.readline()
        for line in inputfile:
            wid,gender,name=line.strip().split("\t")
            name=name.replace(" ","_")
            genderdata[name]=gender[:1]
    return genderdata

def fill_in_genders(persons):
    """
    Fills in the gender information of persons in a dictionary.

    Args:
        persons (dict): A dictionary containing information about␣
 ↪persons.

    Returns:
        None. The function modifies the input dictionary in place.
    Examples:
        >>> persons = {'Alice': {'age': 25}, 'Bob': {'age': 30}}
        >>> fill_in_genders(persons)
        >>> persons
        {'Alice': {'age': 25, 'gender': 'F'}, 'Bob': {'age': 30,␣
 ↪'gender': 'M'}}

    """
    genderdata=get_genderdata()
    for person in list(persons.keys()):
        if person in genderdata:
            gender=genderdata[person]
        else:
            gender="NA"
        persons[person]["gender"]=gender

def fetch_links(people,batch_size=None,lang='en'):
    """Uses the Wikipedia API to fetch Wikipedia links between the␣
 ↪given people.

    The links are filled into the people dictionary in place.

    Note that only links between the people are saved, and if you want␣
 ↪to inspect other links
    you should write your own fetching function.

    Args:
        people (dict): A dictionary containing names of people as keys␣
 ↪and attributes as values.
        batch_size (int, optional): The maximum number of people to␣
 ↪fetch links for in a single batch. Defaults to None, which means␣
 ↪there is no maximum.
        lang (str, optional): The language in which to fetch Wikipedia␣
 ↪links. Defaults to 'en'.

    Returns:
        bool: True if the links were not fetched for every person due␣
 ↪to the batch size, False otherwise.
    """
    wiki = wikipediaapi.Wikipedia(lang)
    i=0
    print('Fetching link data from Wikipedia')
    pbar=tqdm(total=len(people))
    for name,attributes in people.items():
        pbar.update(1)
        if "links" not in attributes:
            page=wiki.page(name)
            links=list(map(lambda x:x.replace(" ","_"),page.links.
 ↪keys()))
            plinks=list(filter(lambda x:x in people,links))
            #print(name,plinks)
            people[name]["links"]=plinks
            i+=1
            time.sleep(0.1)
        if i==batch_size:
            return True
    return False

def fetch_langs(people,batch_size=None,lang='en'):
    """Uses the Wikipedia API to fetch list of Wikipedia language␣
 ↪editions where each person in the people
    dictionary appears.

    The language editions are filled into the people dictionary in␣
 ↪place.

    Args:
        people (dict): A dictionary containing names of people as keys␣
 ↪and attributes as values.
        batch_size (int, optional): The maximum number of people to␣
 ↪fetch links for in a single batch. Defaults to None, which means␣
 ↪there is no maximum.
        lang (str, optional): The language in which to fetch Wikipedia␣
 ↪links. Defaults to 'en'.

    Returns:
        bool: True if the language editions were not fetched for every␣
 ↪person due to the batch size, False otherwise.
    """
    wiki = wikipediaapi.Wikipedia(lang)
    i=0
    print('Fetching language editions data from Wikipedia')
    pbar=tqdm(total=len(people))
    for name,attributes in people.items():
        pbar.update(1)
        if "langs" not in attributes:
            page=wiki.page(name)
            langs=list(page.langlinks.keys())
            #print(name, langs)
            people[name]["langs"]=langs
            i+=1
            time.sleep(0.1)
        if i==batch_size:
            return True
    return False

def fetch_summaries(people,batch_size=None,lang='en'):
    """Uses the Wikipedia API to fetch summary texts for each person in␣
↪the people dictionary.

    The summary texts are filled into the people dictionary in place.

    Args:
        people (dict): A dictionary containing names of people as keys␣
↪and attributes as values.
        batch_size (int, optional): The maximum number of people to␣
↪fetch links for in a single batch. Defaults to None, which means␣
↪there is no maximum.
        lang (str, optional): The language in which to fetch Wikipedia␣
↪links. Defaults to 'en'.

    Returns:
        bool: True if the summaries were not fetched for every person␣
↪due to the batch size, False otherwise.
    """

    wiki = wikipediaapi.Wikipedia(lang)
    i=0
    print('Fetching summary text data from Wikipedia')
    pbar=tqdm(total=len(people))
    for name,attributes in people.items():
        pbar.update(1)
        if "summary" not in attributes:
            page=wiki.page(name)
            summary=page.summary
            #print(name,summary)
            people[name]["summary"]=summary
            i+=1
            time.sleep(0.1)
        if i==batch_size:
            return True
    return False

def save_people_json(people,filename):
    with open(filename, "w") as pfile: json.dump(people,pfile)

def load_people_json(filename):
    with open(filename, "r") as pfile:
        return json.load(pfile)
```

In the next cell, you will find the code for loading the politician data to a dictionary from the json file that you can download through A+. The commented out code was used to parse and fetch the data. You can inspect how the data was created using that code and the functions in the previous cell.

[2]:
```
filename="politicians.json"
if not os.path.isfile(filename):
    politicians=filter_persons_by(occupation="politician")
    fill_in_genders(politicians)
    save_people_json(politicians,filename)

politicians=load_people_json(filename)

## The code below fills in summaries, language editions and links from␣
 ↪wikipedia.
## The fetching takes place in batches of 1000 queries after which the␣
 ↪data is saved to disk.
#while fetch_summaries(politicians,batch_size=1000):␣
 ↪save_people_json(politicians,filename)
#while fetch_langs(politicians,batch_size=1000):␣
 ↪save_people_json(politicians,filename)
#while fetch_links(politicians,batch_size=1000):␣
 ↪save_people_json(politicians,filename)
#save_people_json(politicians,filename)
```

Use the next cell to inspect how the data looks like for a single politician.

[3]:
```
politicians['Benedict_Calvert,_4th_Baron_Baltimore']
```

[3]:
```
{'#DBpURL': 'http://dbpedia.org/resource/
 ↪Benedict_Calvert,_4th_Baron_Baltimore',
 'ID': '21',
 'WikiURL':
'http://en.wikipedia.org/wiki/Benedict_Calvert,_4th_Baron_Baltimore',
 'gender': 'M',
 'name': "[' (the right honourable) ', ' the lord baltimore ']",
 'birthDate': "[' 1679-03-21 ']",
 'deathDate': "[' 1715-04-16 ']",
 'occupation': "[' politician ']",
 'nationality': 'NA',
 'party': 'NA',
 'summary': "Benedict Leonard Calvert, 4th Baron Baltimore (21 March␣
 ↪1679 - 16
April 1715) was an English nobleman and politician. He was the second␣
 ↪son of
Charles Calvert, 3rd Baron Baltimore (1637-1715) by Jane Lowe, and␣
 ↪became his
father's heir upon the death of his elder brother Cecil in 1681. The␣
 ↪3rd Lord
Baltimore was a devout Roman Catholic, and had lost his title to the␣
 ↪Province of
Maryland shortly after the events of the Glorious Revolution in 1688,␣
 ↪when the
Protestant monarchs William III and Mary II acceded to the British␣
 ↪throne.
Benedict Calvert made strenuous attempts to have his family's title to␣
 ↪Maryland
restored by renouncing Roman Catholicism and joining the Church of␣
 ↪England.\nIn
February 1715 Benedict became the 4th Baron Baltimore upon the death of␣
 ↪his
father, and he immediately petitioned King George I for the restoration␣
 ↪of
Maryland to his control. However, before the King could rule on the␣
 ↪petition,
Baltimore died aged 36, outliving his father by just two months. Shortly
afterwards the King restored the title to Maryland to Calvert's young␣
 ↪son
Charles Calvert, 5th Baron Baltimore.",
 'langs': ['de', 'fr', 'ja', 'pl'],
 'links': ['Benedict_Leonard_Calvert',
  'Benedict_Swingate_Calvert',
  'Charles_Calvert,_5th_Baron_Baltimore',
  'Henry_Darnall',
  'Sir_Robert_Eden,_1st_Baronet,_of_Maryland']}
```

Revision of prior analysis

From the dictionary of politicians created above, we create a network by running a loop where for each policitians, we create edges from that politicians to the links.

[4]:
```python
import math
import networkx as nx

def construct_network():
    """
    This function constructs a social network from the data of␣
 ↪politicians.

    Args: filename (str) - The filename of the politicians' data file.
    Returns: net (nx.Graph) - A networkx graph object representing the␣
 ↪social network.
    """
    net = nx.Graph()

    for person, data in politicians.items():
        net.add_node(person)
        neighbors = data['links']
        for neighbor in neighbors:
            if neighbor != person:
                net.add_edge(person, neighbor)

    return net
```

Here we calculate the graph of the network, including the numbers of nodes, edges, average degree and the clustering coefficient. Zero-degree nodes are also included for the calculation of male and female nodes' degrees.

[5]:
```
from matplotlib import pyplot as plt
import numpy as np

net = construct_network()

# Print out some basic statistics of the network
print("The network has:")
print(len(net), "nodes")
print(net.number_of_edges(), "edges")
print(2*net.number_of_edges()/len(net), "average degree")
print(nx.average_clustering(net), "average clustering coefficient")
print(nx.average_shortest_path_length(net.subgraph(max(nx.
 ↪connected_components(net), key=len))), "average shortest path␣
 ↪length")

# Plot the network
# plt.figure()
# positions = nx.spring_layout(net)
# nx.draw(net, positions, node_size=1)
```

```
The network has:
6880 nodes
10294 edges
2.9924418604651164 average degree
0.14363897936984674 average clustering coefficient
7.290627875646906 average shortest path length
```

Below is the calculation of the average degree of male nodes and female nodes. The result implies that in general female nodes have higher degrees than male nodes, indicating more nobility for the women included on Wikipedia. The average degree inequality is describe as:

$$\frac{1}{N_F}\sum_i d_i^F \geq \frac{1}{N_M}\sum_i d_i^M \tag{1}$$

[6]:
```
male_sum = 0        # Total degree of male nodes
male_count = 0      # Total degree of female nodes
```

```
female_sum = 0      # Number of male nodes
female_count = 0    # Number of female nodes

# Loop through the network to calculate the average degree for each␣
↪gender.
for node in net:
    value = politicians[node]
    if value['gender'] == 'M':
        male_sum += net.degree(node)
        male_count += 1
    elif value['gender'] == 'F':
        female_sum += net.degree(node)
        female_count += 1

print("The average degree of males in the network is:", male_sum /␣
↪male_count)
print("The average degree of females in the network is:", female_sum /␣
↪female_count)
```

```
The average degree of males in the network is: 4.087830687830688
The average degree of females in the network is: 7.176165803108808
```

Gender homophily

Next we will analyse the gender homophily of our data. For simplicity for our model, the function below creates a new network with only two genders (male and female) with no self-edge and no zero-degree nodes.

[7]:
```
def binary_network():
    """
    This function constructs a new social network from the data of␣
↪politicians, consisting of only two genders and no zero-degree nodes.

    Returns: net (nx.Graph) - A networkx graph object representing the␣
↪social network.
    """
    net = nx.Graph()

    for person, data in politicians.items():
        if data['gender'] != 'NA':          # Only consider politicians␣
↪that are either male or female
            neighbors = data['links']
            for neighbor in neighbors:
                nei_gender = politicians[neighbor]['gender']
                if neighbor != person and nei_gender != 'NA':
                    net.add_edge(person, neighbor)

    return net
```

The codes below create the new network defined above.

```
[8]: new_net = binary_network()

     print("The new network has:")
     print(len(new_net), "nodes")
     print(new_net.number_of_edges(), "edges")
     print(2*new_net.number_of_edges()/len(new_net), "average degree")
     print(nx.average_clustering(new_net), "average clustering coefficient")
     print(nx.average_shortest_path_length(new_net.subgraph(max(nx.
      ↪connected_components(new_net), key=len))), "average shortest path␣
      ↪length")

     # Plot the network
     # plt.figure()
     # positions = nx.spring_layout(new_net)
     # nx.draw(new_net, positions, node_size=1)
```

```
The new network has:
1573 nodes
5736 edges
7.293070565797839 average degree
0.339796557267688 average clustering coefficient
6.817727056393223 average shortest path length
```

First we can check that for our new network, notability difference between male and female still applies by the calculation of inequality (1).

```
[9]: new_male_sum = 0
     new_male_count = 0
     new_female_sum = 0
     new_female_count = 0

     for node in new_net:
         value = politicians[node]
         if value['gender'] == 'M':
             new_male_sum += new_net.degree(node)
             new_male_count += 1
         elif value['gender'] == 'F':
             new_female_sum += new_net.degree(node)
             new_female_count += 1

     print("The average degree of males in the new network is:",␣
      ↪new_male_sum / new_male_count)
     print("The average degree of females in the new network is:",␣
      ↪new_female_sum / new_female_count)
```

```
The average degree of males in the new network is: 6.66908037653874
The average degree of females in the new network is: 11.78125
```

Next we analyse the homophily of the whole network and for each gender. For the full explanation please refer to the report.

[10]:
```
fm_count = 0
mm_count = 0
ff_count = 0

# Empirical network data between men and women
for edge in list(new_net.edges):                       # Loop through every␣
 ↪edges (connections) between the politicians
    name_1, name_2 = edge                            # Get the name of the two␣
 ↪politicians in the edge
    gender_1, gender_2 = politicians[name_1]['gender'],␣
 ↪politicians[name_2]['gender'] # Get the gender of the politicians
    concat = gender_1 + gender_2                     # Create a concatenation␣
 ↪of the two genders
    if concat == 'FM' or concat == 'MF':             # A connection between␣
 ↪different genders can only be a man-woman connection and vice-versa
        fm_count += 1
    elif concat == 'MM':
        mm_count += 1
    elif concat == 'FF':
        ff_count += 1
```

The analysis of our network homophily, which is:

$$e_{FM} - \bar{e}_{FM} = \frac{E}{3} - \bar{e}_{FM} > 0.1e_{FM} = 191.2 \tag{2}$$

[11]:
```
new_net.number_of_edges()/3 - fm_count
```

[11]:
```
290.0
```

The analysis of male homophily, which is:

$$\frac{\bar{e}_{FM}}{\bar{e}_{MM} + \bar{e}_{FM}} < 0.45 \tag{3}$$

[12]:
```
fm_count/(fm_count + mm_count)
```

[12]:
```
0.2994830132939439
```

Similarly, the analysis for female homophily is:

[13]:
```
fm_count/(fm_count + ff_count)
```

[13]:
```
0.835221421215242
```