CS 432 Assignment #7
Larry Montgomery 01093132

Question 1:

When collecting the 98 blogs URLs I chose to use selenium python library and a search site called https://www.searchblogspot.com/ . I opened the chrome browser and adjusted the query parameters to search multiple topics. I have a list of a few random topics I picked myself. Once a search for the given topic has been obtained, I extract the innerHTML for the page. I sort through the links getting the href and checking that they have blogspot in their name. After that I have to find the element in charge of moving to the next page. For this site they have a div element called cursor-page, it is not a direct link but has the attribute role=”link”. This allows for me move the cursor to that element and then click it. Discovering this element took a bit of time but was worth it to discover how I can automate finding hidden menus and other hidden features.

```
def get_blogs():
    blogger_api_key = 'AIzaSyCIp4k2yRqpir-7w5FIN9GyNNdRHyCwTV0'
    browser = webdriver.Chrome("/usr/lib/chromium-browser/chromedriver")

    blog_links = ["http://f-measure.blogspot.com/atom.xml","http://ws-dl.blogspot.com/atom.xml"]

    query_terms = ["python", "data science", "Cars", "Java" , "Dreams", "Video Games", "Game of Thrones", "cats", "NBA", "baseball", "programming", "Fans", "Math"]
    query_id = 0
    while len(blog_links) != 100:
        selected_term = query_terms[query_id]
        query_id += 1
        url = f'https://www.searchblogspot.com/search?q={selected_term}'
        browser.get(url)
        for each in range(2,11):
            value = str(each)
            data = browser.execute_script("return document.body.innerHTML")

            soup_OBJ = BS(data, 'html.parser')

            for link in soup_OBJ.find_all('a'):
                if link.get('href') != None and "blogspot" in link.get('href'):
                    curr_Link = parse_urls(link.get('href'))
                    if curr_Link not in blog_links and len(blog_links) != 100:
                        blog_links.append(curr_Link)
                        print("Total blogs: ",len(blog_links))

            next_button = browser.find_element_by_xpath("//div[@class='gsc-cursor-page'and contains(text(),value)]")

            act = ActionChains(browser)
            act.move_to_element(next_button).click(next_button).perform()

    with open("blog_urls.txt",'w') as f:
        for each in blog_links:
            f.write(each + '\n')

    return blog_links
```

After I collected the URL’s I had get the RSS feed links for them. I discovered that the extension “atom.xml” would redirect me to a site for the blogs feed or give the raw file, both acceptable. I used feedparser to collect the information from the RSS feeds and imported the PCI code to help parse the data into the forms to generate the blog matrix. The Blog matrix exists in the blog_data.txt file in the repository.

Question 2:

For the rest of these questions I ran into a problem with the encoding of one of my blog entries. I was able to plot the dendrogram for the test data set, but when I ran mine one of the blogs titles conatin characters not in the “latin 1” codec and could not be printed by the PIL library used in the clustering library. With a little more time I could have removed the blog and replaced it making it a little easier to generate the dendrogram.