```python
from bs4 import BeautifulSoup
import requests
import difflib
import re
import urllib.request
# from urllib.request import urlopen
# import wikipedia

# ------------------------------------------------------------------------------------------------
# function to calculate similarity between two strings to remove rendundancy in storing
strings.
def calc_similarity(string1, string2):
    """
        Calculates the similarity between two strings using the difflib Library.
        Used to reduce redundancy in storing strings by comparing the similarity of URLs.

        Args:
            string1 (str): The first input string.
            string2 (str): The second input string.

        Returns:
            float: A value representing the percentage similarity between the two input
strings.
    """

    #ndiff returns a list of strings containing the differences between the two input strings.
    difference = difflib.ndiff(string1, string2)
    difference_count = 0
    for line in difference:

        # a "-", indicating that it is a deleted character from the input string.
        if line.startswith("-"):
            difference_count += 1

    # calculates the similarity by subtracting the ratio of the number of deleted characters to
the length of the input string from 1
    return 1 - (difference_count / len(string1))

# ------------------------------------------------------------------------------------------------

# Function to get urls related to the starter url
def get_urls(starter_url):
    """Based on first url, it parses the website and collects the list of urls present in it
and appends to a local variable.

    Args:
        starter_url (string): The url to collect links from.

    Returns:
        list: returns list of urls.
    """
    r = requests.get(starter_url)
    if r:
        data = r.text
        soup = BeautifulSoup(data, features="html.parser")
        url_list = []
        limit = 0
        # ban_list = ['donate', '.pdf', 'youtube', '#', '%', ]
        url_list.append(starter_url)

        for link in soup.find_all('a'):
            # print(link.get("href"))
            new_link = link.get("href")
            if new_link == None:
                continue
```

```
            if new_link.startswith("https") and 'wiki' not in new_link and 'donate' not in
new_link and '.pdf' not in new_link and 'youtube' not in new_link and '#' not in new_link and
'%' not in new_link:

                if len(url_list) >= 1:
                        if calc_similarity(new_link, str(url_list[-1:][0])) < 0.6:
                            url_list.append(new_link)


            elif limit > 60:
                break
            else:
                continue
            limit += 1
    else:
        return False
    return url_list

# ----------------------------------------------------------------------------------------------
----------------------------------------

# function to determine if an element is visible
def visible(element):
    """Function to filter out elements that are not visible in the website.

    Args:
        element (character or string): The html tag name.

    Returns:
        bool: Returns true if the element is visible and false if element is not visible.
    """
    if element.parent.name in ['table', 'ol', 'style', 'script', '[document]', 'head',
'title', 'id', 'class', 'nav', 'footer', 'header', 'figure' ]:
        return False
    elif re.match('<!--.*-->', str(element.encode('utf-8'))):
        return False
    return True

# ----------------------------------------------------------------------------------------------
----------------------------------------

def datascraper(my_url, counter):
    """Parses through the url's and gets the text data from the website.

    Args:
        my_url (string): The url to get the data from.
        counter (int): The counter number to store file in that corpus : ex. corpus1.txt,
corpus2.txt etc.

    Returns:
        bool: True if data scraping was successful, false if data scraping failed.
    """

    text_string = './Corpuses/corpus'+str(counter)+'.txt'

    if 'filmography' in my_url:
        my_url = 'https://en.wikipedia.org/wiki/Keanu_Reeves_filmography'
        res = requests.get(my_url).text
        soup = BeautifulSoup(res,'lxml')
        input_text = ""
        # input_text += ("Movies:\n\n")
        for items in soup.find_all('table')[0].find_all('tr')[1::1]:
            data = items.find_all(['th','td'])
            j = 0
            for i in data[:2]:
                input_text +=i.text
```

```
    input_text+="===\n"

    for item in soup.find_all('table')[2].find_all('tr')[1::1]:
        data = item.find_all(['th','td'])
        for i in data[:2]:
            input_text +=  i.text

    with open("./Corpuses/corpus0.txt", 'w', encoding='utf-8') as file:
            print("Scraping filmography to corpus success")
            file.write(input_text)

    return True

elif (calc_similarity(my_url, 'https://en.wikipedia.org/wiki/Keanu_Reeves') > 0.8):

    url = 'https://en.wikipedia.org/wiki/Keanu_Reeves'
    # Sending a GET request to the URL
    response = requests.get(url)

    # Parsing the HTML content
    soup = BeautifulSoup(response.content, 'html.parser')

    # Finding all tables and removing them
    for table in soup.find_all('table'):
        table.extract()

    # Finding the references section
    references_tag = soup.find('span', {'id': 'References'})

    # Removing content after references
    if references_tag:
        for sibling in references_tag.find_next_siblings():
            sibling.extract()

    # Finding the bibliography section
    bibliography_tag = soup.find('span', {'id': 'Bibliography'})

    # Removing content after bibliography
    if bibliography_tag:
        for sibling in bibliography_tag.find_next_siblings():
            sibling.extract()

    # Extracting visible text
    visible_text = soup.get_text()

    # Printing the visible text
    # print(visible_text)

    visible_text = re.sub("\n", "", visible_text)
    visible_text = re.sub(r'\[\d+\]', '', visible_text)

    # with open('wikipedia.txt', 'w', encoding='utf-8') as f:
    #     f.write(final_text)

    visible_text = visible_text.split(".")
    final_text = ""
    # j=0
    flag = 0
    for i in visible_text:

        if 'Keanu Charles Reeves (' in i:
            flag = 1

        if 'References^' in i:
            flag = 0
            break
```

```
            if flag == 1 and i!="" and 'Göttingen: Steidl Publishers.' not in i and 'Alexandra
(photographs by) (2014)' not in i and ' Ron Garney, 12-issue mini-series,' not in i and 'Steidl
Publishers. ISBN 9783869308272' not in i:
                final_text+= i + '.'

        final_text += " Keanu's favorite colour is Green."
        with open(text_string, 'w', encoding='utf-8') as file:
                file.write(final_text)

        print("Scraping wikipedia to corpus", counter, "success!")

        return True

    else:

        temp_str=""

        headers = {'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/58.0.3029.110 Safari/537.3'}
        req = urllib.request.Request(my_url, headers=headers)

        try:
            html = urllib.request.urlopen(req)
            soup = BeautifulSoup(html, features="html.parser")
            data = soup.findAll(string=True)
            result = filter(visible, data)
            temp_list = list(result)
            temp_str = ' '.join(temp_list)



            with open(text_string, 'w', encoding='utf-8') as file:
                file.write(temp_str)
                print("For Link:", counter, "Scraping to corpus success!")

            return True
        except urllib.error.HTTPError as e:
            print(f"Error accessing {my_url}: {e}")
            return False

# ------------------------------------------------------------------------------------------
----------------------------------------

def clean_text(text_string):
    """Function to remove undesirable characters from the corpus text.

    Args:
        text_string (string): The corpus string to basic clean.
    """
    if open(text_string, 'r') :
        with open(text_string, "r", encoding='utf-8') as file:
            try:
                text = file.read()
                text = text.strip()

                # Remove references like '[29]'
                text = re.sub(r'\[\d+\]', '', text)
                text = re.sub(r'[]', '', text)
                text = re.sub("\'", "'", text)
                text = re.sub("  ", " ", text)
                # Remove non-alphanumeric characters
                # text = re.sub("[^-9A-Za-z ]", "", text)

                # Remove non-word characters
                # text = re.sub(r"[^\w.',]", ' ', text)
```

```
                # Remove extra whitespaces
                text = ' '.join(text.split())

                # print(text)
            except Exception as e:
                print(f"Error: {e}")

        with open(text_string, "w", encoding='utf-8') as write_file:
            write_file.write(text)
            print("Write to ", text_string, " successful")
    else:
        print("File not found")

# ----------------------------------------------------------------------------------------
----------------------------------------

def should_append_line(i, flag):
    """Function containing specific hardcoded ban list of words and phrases to filter out.

    Args:
        i (string): _description_
        flag (bool): Another flag to determine if line should be included or not.

    Returns:
        bool: Returns true if the phrase can be included, false otherwise.
    """
    # Define regex patterns to match specific strings
    patterns = [
        r'No More 404|Disclaimer|Twitter  Site|see also|, Inc|SHARE THIS STORY|Go to
item|Scroll Up|#page|Ltd.|For your consideration|--|fslink|-Amazon|Write a review|Please create
a new|©2019|©2024|Already a Subscriber|Subscribe|You can|Something went wrong|Breaking News
Headlines|SIGNED IN|http:|\/|/Collider/|header|You already recently|CSS styles|mntl-sc-
block|Disclaimer:|Thank you for your|end div|END SIGNED OUT|Permalink|Privacy policy|Tap to
play|Support Us|support us|Support our mission|Sign up|View all stories|You can also
contribute|Will you help|check out our|Terms and Privacy|your email|sc block|Terms of
Service|porn|this web site|Disclaimer Content|Advertise|This web site|Please help us|Forgot
your password|please email|Taylor Swift ticket|display none|No repeatable ad|Repeatable debug
data|Footer|footer|advertisers|try again|article link|Up next|Read the original article|Your
account|ERROR TAB|Secure transaction|Cookies|Follow us|Powered by|log in|Log In|Israel|daily
newsletter|Copyright|signing up|can unsubscribe|All rights reserved|All Rights
Reserved|Added.*?by|View Related Entries|Uploaded by|External References|login|signup|privacy
policy|Like us|https|Contact Us|membership comments|Skip to main|parse',
        r'^[0-9a-z]|^[^\w\s]'
        r'\d{15,}|.{300,}'
    ]

    # Check if any pattern is found in the string
    for pattern in patterns:
        if re.search(pattern, i):
            return False

    # Check flag
    if flag != 1:
        return False

    return True

# ----------------------------------------------------------------------------------------
--------------------------------------

def filter_data(i):
    """Next level cleaning function to filter out undesirables specified by the user.

    Args:
        i (int): Counter value to open the files and write to it. File names are in incremental
fashion : corpus1.txt, corpus2.txt and so on.
    """
```

```
    filename = "./Corpuses/corpus" + str(i) + ".txt"
    filewrite = "./Corpuses/corpus" + str(i) + ".txt"

    if open(filename, 'r', encoding='utf-8'):
        with open(filename, 'r', encoding='utf-8') as file:
            read_file = file.read()

        lines = (read_file.split('.'))
        lines = [line for line in lines if line.strip()]
        new_lines = []
        flag = 0

        for i in lines:
            i = re.sub(r"\n", "", i)
            i = i.strip()
            # numbers = re.findall(r'\d+', i)
            if 'Keanu' in i or 'Reeves' in i :
            # or 'keanu' in i or 'reeves' in i:
                flag =1

            if 'ADVERTISEMENT' in i or 'end id' in i or 'Read more' in i or len(i) == 1:
                flag = 0
            if should_append_line(i, flag):
                i+='.'
                new_lines.append(i)

        text = ' '.join(new_lines)

        if open(filewrite, 'w', encoding='utf-8'):
            with open(filewrite, 'w', encoding='utf-8') as write_file:
                write_file.write(text)
            print('Filtering Data for', filewrite, "success")

        else:
            print("Writing filtered data unsuccessful for file", i)
    else:
        print("File not found")


#-------------------------------------------------------------------------------------------------
----------------------------------------

def main():
    """Driver function to execute all other functions in the necessary order.
    """

    start_url = "https://en.wikipedia.org/wiki/Keanu_Reeves"
    parenturls = get_urls(start_url)

    childurlx = []

    for urlx in parenturls[:5]:
        childurlx.extend(get_urls(urlx))

    # print(childurlx)
    childurly = []

    for urly in childurlx[:5]:
        childurly.extend(get_urls(urly))

    # print(childurly)

    totalurls = []
    totalurls = parenturls
    totalurls.extend(childurlx)
    totalurls.extend(childurly)
```

```
    # print(totalurls)

    final_urls= []
    for url in totalurls:
        if 'keanu' in url.lower() or 'reeves' in url.lower():
            final_urls.append(url)

    final_urls = list(set(final_urls))

    final_urls[:0] = ['https://en.wikipedia.org/wiki/Keanu_Reeves_filmography']

    if 'https://en.wikipedia.org/wiki/Keanu_Reeves' in final_urls:
        final_urls.remove('https://en.wikipedia.org/wiki/Keanu_Reeves')
        final_urls.insert(1, 'https://en.wikipedia.org/wiki/Keanu_Reeves')

    # for i in final_urls:
    #     print(i)

    with open("./Corpuses/url_list.txt", "w") as file:
        for i in final_urls:
            file.write(i + '\n')

    counter = 0
    with open('./Corpuses/url_list.txt', 'r') as read_file:
        for url in read_file:
            if datascraper(url, counter) != False:
                counter+=1
            if counter>20:
                break
    for i in range(1, 21):
        clean_text('./Corpuses/corpus'+str(i)+'.txt')

    for i in range(2, 21):
        filter_data(i)

# ------------------------------------------------------------------------------------------------
----------------------------------------

main()
```