Doc 1

Overview of the issue to be investigated

In 2019 when NLP entered UK, numerous retailers straightforwardly reprimanded that buyers could at any point take to purchasing design clothes through on the web. Online clothes deal in UK have raised the eyeballs of numerous high road retailers, breaking the deep-rooted legend these days, they will sell their clothes through internet finding that their business rate have significantly expanded year on year contrasted with off- store deals. E-trade is a meriting idea in the further of clothes industry. Almost 52% of clothes are sold in UK through on the web, the rate has neither expanded yet not diminished. UK being a moderate country for quite a long time constrained numerous retailers to receive our conventional looks. In couple of years, the Internet has shown a gigantic development as far as business exchange volume, which cleared approach to new definition in all angles. The passage of late data and correspondence innovations have empowered firms to offer high types of assistance quality, lower costs for clients, and increment the overall revenues for organizations.

Doc 2

Web crawler

In the fast growing of digital technologies, crawlers and search engines face unpredictable challenges. Focused web-crawlers are essential for mining the boundless data available on the internet. Web-Crawlers face indeterminate latency problem due to differences in their response time. It is the process of indexing data on web pages by using a program or automated script.

The concept of extracting information from the internet is not novel. Search engines, Data Miners, and surveyors have used automated retrieval information since beginning of twentieth century. The automated software browses the web and extracts data from the internet. Such software is referred to as web crawler or web spider. The web crawler here-in-after simply referred as crawler. The crawler is central concept behind all search engines. Search engines use crawler to index mass data, but they can also be used as a tool for data extraction. There is always a specific purpose for data mining. The analysis of data is performed as to prove or to discover facts that were earlier unknown. When the analysis is specific, the information extracted must also be relevant.

Code:

import pandas as pd

import re

import requests

from bs4 import BeautifulSoup

from time import sleep

from .requester import Requester

class Crawler:

"""

"""

def \_\_init\_\_(self, url, sarcasm, as\_archived=False):

self.\_\_url = url

self.\_\_sarcasm = sarcasm

self.\_\_as\_archived = as\_archived

self.\_\_data = list()

self.\_\_requests = list()

def set\_requests(self, html\_class, regex, remove, element="a", shorter=0):

find\_pages = re.compile(regex)

urls = list()

html = Requester.get\_one\_request(self.\_\_url + ("1", "")[self.\_\_as\_archived], force=True)

bs = BeautifulSoup(html.text, "html.parser")

element = str(bs.find\_all(element, class\_=html\_class))

if self.\_\_as\_archived == True:

for page in find\_pages.finditer(element):

urls.append(str(self.\_\_url[:-shorter]) + element[page.start()+remove[0]:page.end()-remove[1]])

else:

for page in find\_pages.finditer(element):

pages = int(element[page.start()+remove[0]:page.end()-remove[1]]) + 1

for page in range(1, pages):

urls.append(self.\_\_url + str(page))

self.\_\_requests = Requester(urls, num\_threads=24).get\_requests()

def get\_raw\_data(self, html\_class, regex, element="div"):

find\_element = re.compile(regex)

raw\_data = list()

for request in self.\_\_requests:

bs = BeautifulSoup(request.text, "html.parser")

raw = str(bs.find\_all(element, class\_=html\_class))

for text in find\_element.finditer(raw):

raw\_data.append(raw[text.start():text.end()])

print("[+] Total of {0:04d} raw data collected".format(len(raw\_data)))

return raw\_data

def set\_data(self, raw\_args, regex, remove, html\_options, url\_prefix=0):

find\_link = re.compile(regex[0])

find\_title = re.compile(regex[1])

find\_text = re.compile(r"<.\*?>")

# find\_text = re.compile(r"<.\*?>|&([a-z0-9]+|#[0-9]{1,6}|#x[0-9a-f]{1,6});")

urls = list()

for raw in self.get\_raw\_data(\*\*raw\_args):

data = [None,None,None,None]

data[0] = self.\_\_sarcasm

for tmp in find\_link.finditer(raw):

data[1] = self.\_\_url[:url\_prefix] + raw[tmp.start()+remove[0][0]:tmp.end()-remove[0][1]]

urls.append(data[1])

for tmp in find\_title.finditer(raw):

data[2] = raw[tmp.start()+remove[1][0]:tmp.end()-remove[1][1]]

if data[2] == [""]:

continue

data[3] = ""

self.\_\_data.append(data)

print("[+] Dataframe completed")

def get\_data(self):

return self.\_\_data

from sys import version\_info

from .crawler.sites import Sites

if version\_info[0] != 3:

raise Exception("PLNCrawler requires at least python3.9")

if version\_info[0] != 3 and version\_info[1] != 9:

raise Exception("PLNCrawler requires at least python3.9")

def \_main():

Sites()

if \_\_name\_\_ == '\_\_main\_\_':

\_main()

Doc 3

Prototype NLP tasks

The huge infiltration of corporatized retail has changed the standpoint of style industry by presenting formal and methodical cycle in activities, appropriation, and acquirement. It has been normal that in simple future we would observer a more profound infiltration in corporatized retail in clothing industry which thusly would build the interest of all around the world marked items. The UK attire market has a board grouping of classifications beginning from dress, watches, embellishments, footwear, makeup, adornments and so on and goes on. The interest in clothes classifications fluctuates in various states.

The idea of web-based business has upset the style clothes industry somehow. Comfort was the motivation to incline toward e-shopping as well as one can guarantee that supported brands will discover a route to your closet. As indicated by an investigation the current assessed income of United Kingdom internet business, starting at 2020, is about EUROs 68 Billion and is projected to hit EUROs 190 Billion continuously 2020. This implies a yearly development pace of about a stunning 63%. It has been demonstrated that design clothes sold through online have taken off past assumptions. Particularly in United Kingdom impact of web overwhelm the buys just as deals channel. The exploration likewise uncovered that Fashion embellishments are one of the exceptionally favored classifications during on the web buys. Indeed, clothes and frill are one of the exceptionally favored classifications in Europeans and Asian nations. The determination that we make from these reports is that the attire enterprises in nations, for example, United Kingdom is seeing a blast because of the presence of web-based shopping entries.

As an online business stage, it ought to be exceptionally open furthermore, have the option to adjust to various gadgets. It is difficult to give guidelines appropriately. Albeit the program can know the gadget type, it is difficult to know which framework it is, or on the other hand to know whether the clients are utilizing a touchpad or a mouse. The communication configuration is muddled, so does how to give right guidelines.

Code:

import pandas as pd

from json import load as load\_json

from pathlib import Path

from os import sep

from .algorithms import Crawler

dir\_path = str(Path().absolute()) + sep

class Sites():

    datasets\_dir = dir\_path + sep + "datasets" + sep

    def \_\_init\_\_(self):

        self.\_\_sites = list()

        if not Path(self.datasets\_dir).is\_dir():

            Path(self.datasets\_dir).mkdir(exist\_ok=True)

        for site in self.get\_json\_file():

            self.\_\_sites.append(Site(site[0],site[1]))

    def get\_json\_file(self):

        with open(dir\_path + "sites.json", 'r') as file:

            json\_file = load\_json(file)

        sites = []

        for site in json\_file.items():

            sites.append([site[0],site[1]])

        return sites

class Site():

    def \_\_init\_\_(self, name, args):

        self.\_\_name = name

        self.\_\_args = args

        self.\_\_crawler = Crawler(\*\*self.\_\_args[0])

        self.process()

        self.\_\_dataframe = pd.DataFrame(self.\_\_crawler.get\_data(), columns=["is\_sarcastic","article\_link","headline","text"])

        # data = self.\_\_crawler.get\_data()

        # file = Sites.datasets\_dir + "texto\_" + self.\_\_name + ".txt"

        # f = open(file, "w+", encoding="utf-8")

        # for d in data:

        #     if(d[3] == None):

        #         continue

        #     f.write(d[3])

        # f.close()

        # self.\_\_dataframe.to\_csv(

        #     Sites.datasets\_dir + self.\_\_name + ".csv",

        #     sep = '|',

        #     index = False,

        #     encoding = "utf-8-sig"

        # )

        self.\_\_dataframe.to\_json(

            Sites.datasets\_dir + self.\_\_name + ".json",

            orient = "records",

            lines = True,

            force\_ascii = False

        )

    def process(self):

        self.\_\_crawler.set\_requests(\*\*self.\_\_args[1])

        self.\_\_crawler.set\_data(\*\*self.\_\_args[2])

    def get\_name(self):

        return self.\_\_name

    def get\_args(self):

        return self.\_\_args

import requests

from time import sleep

from threading import Thread

class Requester:

    header = {'user-agent':"Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:74.0) Gecko/20100101 Firefox/74.0"}

    def \_\_init\_\_(self, url\_list, num\_threads = 4):

        self.\_\_len\_urls = len(url\_list)

        self.\_\_urls\_list = list()

        size = int(len(url\_list)/num\_threads) + 1

        for i in range(0, len(url\_list), size):

            self.\_\_urls\_list.append(url\_list[i:i+size])

        self.\_\_requests = list()

        self.\_\_progress = 0

        self.\_\_set\_requests()

    def \_\_set\_requests(self):

        indexer = 0

        threads = list()

        # Split the list of URLs to make the requests into multiple threads

        for urls in self.\_\_urls\_list:

            thread = Thread(

                    target=self.\_\_set\_request\_list,

                    args=(indexer, urls)

                    )

            thread.start()

            indexer += 1

            threads.append(thread)

        # Wait for the threads to complete

        for thread in threads:

            thread.join()

            sleep(0.01)

            del thread

        print()

        # Sort the requests in one list only in the same order they were requested

        requests\_sorted = sorted(self.\_\_requests, key=lambda x: x[0])

        self.\_\_requests.clear()

        for requests in requests\_sorted:

            for request in requests[1]:

                self.\_\_requests.append(request)

    def \_\_set\_request\_list(self, index, urls):

        requests = [index, list()]

        for url in urls:

            request = self.get\_one\_request(url)

            if request == None:

                continue

            requests[1].append(request)

            self.\_\_print\_progress()

        self.\_\_requests.append(requests)

    def \_\_print\_progress(self):

        self.\_\_progress += 1

        print("[+] {0:05d}/{1:05d} requests completed.".format(

                        self.\_\_progress,

                        self.\_\_len\_urls), end='\r')

    @staticmethod

    def get\_one\_request(url, force=False):

        timeout = None

        attempts = 10

        while timeout != False:

            try:

                request = requests.get(url, headers=Requester.header, timeout=5)

                timeout = False

            except (requests.ConnectionError,requests.ReadTimeout):

                timeout = True

                if not force:

                    attempts -= 1

                if attempts == 0:

                    return None

                sleep(5)

            except:

                if force:

                    continue

                return None

        return request

    def get\_requests(self):

        return self.\_\_requests

References:

Bozzi, C. and Mont’Alvão, C., 2020. The user experience (UX) on female apparel NLP websites in Brazil. *Work*, (Preprint), pp.1-18.

Dewanti, P. and Indrajit, R.E., 2018. The effect of XYZ generation characteristics to NLP C-to-C: A review. *IKRA-ITH INFORMATIKA: Jurnal Komputer dan Informatika*, *2*(2), pp.56-60.

Kulshrestha, S. and Saini, M.L., 2020, December. Study for the Prediction of NLP Business Market Growth using Machine Learning Algorithm. In *2020 5th IEEE International Conference on Recent Advances and Innovations in Engineering (ICRAIE)* (pp. 1-6). IEEE.