YelpTipExtraction- (Nltk, Selenium, BeautifulSoup)

Extraction Part :

Using selenium webdriver scrape all the restaurant reviews and their menus:
ScrapRestro function within **SeleniumScraper.py** extracts all the restaurant reviews for
Query: Type of restaurant
Location: location of the restaurant

```
def ScrapRestro(self, query, location, driver):

  #indian restro in nyc
  try:
      myElem = WebDriverWait(driver, 3).until(EC.presence_of_element_located((By.XPATH,
'//*[@id="header-search-submit"]')))

      search = driver.find_element_by_id("find_desc")
      search.send_keys(query)
      time.sleep(2)
      city = driver.find_element_by_id("dropperText_Mast")
      city.click()
       # Find the search box
      city.send_keys(Keys.DELETE + location)
      time.sleep(2)
      #find and click the search button
      searchBtn = driver.find_element_by_id("header-search-submit")
      time.sleep(2)
      searchBtn.click()

  except TimeoutException:
      print ("Loading took too much time!")


#    While condition is used to check if a given webpage has <Next> icon to go from one web page to other..
#    This is the outer while loop which runs through various restro in NYC....till have <Next> icon

  outer_condition = 1
  outer_loop = 0
  while outer_condition == 1:

      html = driver.page_source
```

```
        print(driver.current_url)
        main_url= driver.current_url
        time.sleep(1)
        soup = BeautifulSoup(html,'lxml')
        links = soup.find_all("h3",{"class":"search-result-title"}) #links to all the restro on a given page
        for link in links[1:]:
            res_name = ''.join(map(lambda x: x.strip(), link.strings)) # fetches all the restro names
            dir_name = self.MkResDir(res_name) # creating directory
            res_url = 'https://www.yelp.com/' + link.find('a').get('href') # create a URL's for the above fetched restro
            if not re.search('adredir',res_url): # the if condition is used to filter out all ads
                driver.get(res_url)
                time.sleep(1)

                inner_condition = 1
                inner_loop = 0
                while inner_condition == 1:
                    try:
                        continue_link = driver.find_element_by_partial_link_text('Next')
                        continue_link.click()
                        time.sleep(3)
                        page = driver.page_source
                        # code for putting the data into the file
                        inner_loop = inner_loop +1
                        self.WriteFiles(dir_name, page, inner_loop)
                        print('Review '+str(inner_loop)+' Created')

                    except(NoSuchElementException, StaleElementReferenceException) as e:
                        inner_condition = 0

                    print(driver.current_url)
                    print(inner_loop)


                print('done with restro %r'%(res_name))


        driver.get(main_url)
        time.sleep(3)
        try:
            continue_link = driver.find_element_by_partial_link_text('Next')
            continue_link.click()
            time.sleep(2)
            outer_loop = outer_loop +1

        except (NoSuchElementException, StaleElementReferenceException) as e:
            outer_condition = 0

        print(outer_loop)
```

ScrapMenu function within **SeleniumScraper.py** extracts all the restaurant reviews for
Query: Type of restaurant
Location: location of the restaurant

```
def ScrapMenu(self, query, location, driver):
  # example Indian Restaurant in NewYork,NY
  try:
      myElem = WebDriverWait(driver, 3).until(
          EC.presence_of_element_located((By.XPATH, '//*[@id="header-search-submit"]')))

      search = driver.find_element_by_id("find_desc")
      search.send_keys(query)
      time.sleep(2)
      city = driver.find_element_by_id("dropperText_Mast")
      city.click()
      # Find the search box
      city.send_keys(Keys.DELETE + location)
      time.sleep(2)
      # find and click the search button
      searchBtn = driver.find_element_by_id("header-search-submit")
      time.sleep(2)
      searchBtn.click()
  except TimeoutException:
      print ("Loading took too much time!")

  outer_condition = 1
  while outer_condition == 1:

      html = driver.page_source
      print(driver.current_url)
      main_url = driver.current_url
      time.sleep(1)
      soup = BeautifulSoup(html, 'lxml')
      links = soup.find_all("h3", {"class": "search-result-title"})  # links to all the restro on a given page

      for link in links[1:]:
          res_name = ''.join(map(lambda x: x.strip(), link.strings))  # fetches all the restro names
          res_url = 'https://www.yelp.com/' + link.find('a').get(
              'href')  # create a URL's for the above fetched restro
          if not re.search('adredir', res_url):  # the if condition is used to filter out all ads
              driver.get(res_url)
              time.sleep(1)
          file = codecs.open('/Users/deepanshparab/Desktop/Projects/Bia-660/Project/Menus/menus.txt', 'a',
encoding='UTF-8')
          try:
              menu = driver.find_element_by_partial_link_text('View the full menu')
              menu.click()
              print(driver.current_url)
              time.sleep(3)
              html = driver.page_source
              soup = BeautifulSoup(html, 'lxml')
              divs = soup.find_all("div", {
                  "class": "arrange_unit arrange_unit--fill menu-item-details"})  # links to all the restro on a given page
              for div in divs:
                  review_content = div.find('h4')
                  menus = str(review_content.text).replace('\n', '')
```

```
                menus = menus.replace('  ', '')
                file.write(menus)
                file.write('\n')

            print('menus added to set for restro: ', res_name)
            file.close()

        except(NoSuchElementException, StaleElementReferenceException) as e:
            print('menu link not found')

    driver.get(main_url)
    time.sleep(3)
    try:
        continue_link = driver.find_element_by_partial_link_text('Next')
        continue_link.click()
        time.sleep(2)

    except (NoSuchElementException, StaleElementReferenceException) as e:
        outer_condition = 0
```

The **ExtractReviews.py** script extracts all the scraped reviews for all the restaurants present into the ScrappedData directory.

```
class ExtractReviews(object):


##########################################################################################################################################

##########################################################################################################################################
    '''
    In this function returns a dictionary which contains restro folder name as its key and the scrapped html pages as its values
    eg : {'FishmarketRestaurant.txt':
[reviewpage_1.txt,reviewpage_2.txt,reviewpage_3.txt,reviewpage_4.txt,reviewpage_5.txt,reviewpage_6.txt,reviewpage_7.txt,

reviewpage_8.txt,reviewpage_9.txt,reviewpage_12.txt,reviewpage_13.txt,reviewpage_14.txt,reviewpage_15.txt,reviewpage_16.txt,
    reviewpage_17.txt,reviewpage_18.txt,reviewpage_19.txt]}
    '''

    def __init__(self, inputdir, outputdir):
        self.inputdir = inputdir
        self.outputdir = outputdir


    def readRestaurant(self, inputdir):
        restro= {}
```

```
        for restroName in os.listdir(inputdir):
            if restroName =='.DS_Store':
                continue
            else:
                folder = os.path.join(inputdir,restroName)
                for reviews in os.listdir(folder):
                    restro.setdefault(restroName,[]).append((str(os.path.join(folder,reviews))))
        return restro


#########################################################################################################################################


#########################################################################################################################################
    '''
    The function accepts input and output dir paths as parameters and creates a text files of the restro withs all the extracted reviews for that restro

    '''
    def extractRestaurant(self,inputpath,outputpath):
        restro = self.readRestaurant(inputpath)
        reviews_counter= 0
        for k,v in restro.items():
            k = k.replace(" ","")
            filename = outputpath+'/'+str(k)+'.txt'
            try:
                restro_name = codecs.open(filename, 'a', encoding='UTF-8')
            except IOError:
                print ("Could not read file:", filename)
                sys.exit()
            for files in v:
                try:
                    html = open(files,'r')
                except IOError:
                    print ("Could not read file:", files)
                    sys.exit()
                soup = BeautifulSoup(html,'xml')

                reviews = soup.findAll('div', {'class':'review-content'})

                for review in reviews:
                    review_content = review.find('p',{'lang':'en'})
                    data = review_content.text
                    restro_name.write(data)
                    restro_name.write('\n')
                    reviews_counter = reviews_counter+1

            restro_name.close()
            print('Reviews successfully wrote in file '+k)
        print("total reviews extracted: "+str(reviews_counter))
```

Transformation Part:

The **UniqueReviews** function in **tipExtraction.py** generates unique sentences for all the reviews of a given restaurant.

Future, the **tokenize** function removes all the unwanted stop-words and other useless redundant words that affects the models accuracy.

```
def UniqueReviews(self, fpath):
  # read the input
  try:
     f = codecs.open(fpath,'r', encoding='UTF-8')
  except IOError:
     print ("Could not read file:", fpath)
     sys.exit()
  text = f.read().strip()
  f.close()

  # split sentences
  sentences = sent_tokenize(text)
  adj_sentence = set()

  # check for unique  sentences in the reviews
  counter = 0
  for sentence in sentences:
     counter += 1
     adj_sentence.add(sentence)

  #    print(counter)
  adj_sentence = list(adj_sentence)
  #    print(len(adj_sentence))
  print("The following tips are generated from {} unique sentences".format(len(adj_sentence)))
  return (adj_sentence)
```

```
def tokenize(self, string):
  filtered_words = [word for word in re.findall(r'\w+', string.lower()) if word not in stopwords.words('english')]
  return filtered_words
```

Final Model:

The final model uses n-grams(2,3,4) to generate tips based on most frequent (20) reviews given by the people who visited the restaurant.

```
def count_ngrams(self, lines, min_length=2, max_length=4):
  lengths = range(min_length, max_length + 1)
  ngrams = {length: collections.Counter() for length in lengths}
```

```
    queue = collections.deque(maxlen=max_length)

    # Helper function to add n-grams at start of current queue to dict
    def add_queue():
        current = tuple(queue)
        for length in lengths:
            if len(current) >= length:
                ngrams[length][current[:length]] += 1

    # Loop through all lines and words and add n-grams to dict
    for line in lines:
        for word in self.tokenize(line):
            queue.append(word)
            if len(queue) >= max_length:
                add_queue()

    # Make sure we get the n-grams at the tail end of the queue
    while len(queue) > min_length:
        queue.popleft()
        add_queue()

    return (ngrams)
```

```
def list_of_ngrams(self, ngrams, num=20):
    """Print num most common n-grams of each length in n-grams dict."""
    list_of_ngrams = set()
    for n in sorted(ngrams):
        for gram, count in ngrams[n].most_common(num):
            list_of_ngrams.add(' '.join(gram))
    return list(list_of_ngrams)
```

Furter the model checks the menu.txt file that contains list of restaurant's menu for to verify if the recommended dish is present in the restaurant's list of menu.

```
def model(self, path, filename):
    dishes = self.loadwords('/Users/deepanshparab/Desktop/Projects/Bia-660/Project/dishes.txt')
    recommender_list = self.loadwords('/Users/deepanshparab/Desktop/Projects/Bia-660/Project/List.txt')
    unique_sentences = self.UniqueReviews(path)
    time.sleep(1)
    print('\nExtracted from the restaurant: {}'.format(filename))
    ngrams = self.count_ngrams(unique_sentences)
    self.print_most_frequent(ngrams)
    time.sleep(1)
    # creates a list of dictionaries
    ngrams_list = self.list_of_ngrams(ngrams)
    ngrams_sent = dict()

    for grams in ngrams_list:
        for sentence in unique_sentences:
            if grams in sentence:
                ngrams_sent.setdefault(grams, []).append(str(sentence))
```

```
for grams, sentences in ngrams_sent.items():
    if grams in dishes:
        print("\nYou can try dish: {}".format(grams))
        print("-----------------------------------------")
        max_sentence = len(max(sentences))
        count = 0
        for sentence in sentences:
            count += 1
            words = sent_tokenize(sentence)
            for word in words:
                if (len(sentence) == max_sentence or word in recommender_list):
                    sentence = sentence.replace(grams, '**' + str(grams) + '**')
                    print("Sample review  : {}".format(sentence))
        print("Also the tip appears in another {} reviews.".format(count))
```

Finally the model recommends tips for the given restaurant.

```
The following tips are generated from 338 unique sentences

Extracted from the restaurant: 3.DilliJunction.txt

Logic behind our tips prediction:-
----- 20 most common 2-grams -----
chicken tikka: 26
kati roll: 18
dahi puri: 15
tikka masala: 13
indian food: 13
kati rolls: 9
dilli junction: 8
mango lassi: 7
masala platter: 6
food really: 5
roll chicken: 5
bhel puri: 4
back try: 4
street food: 4
shami kebab: 4
food great: 4
great place: 4
food definitely: 4
highly recommend: 4
vegetable kebab: 4

----- 20 most common 3-grams -----
```

chicken tikka masala: 12
kati roll chicken: 4
tikka masala platter: 4
ordered chicken tikka: 3
kabab kati roll: 3
dahi puri chicken: 3
indian street food: 3
masala platter potato: 2
come back try: 2
chicken tikka kati: 2
roll lamb kati: 2
good food really: 2
ordered dahi puri: 2
platter potato roti: 2
broken glass piece: 2
platter dahi puri: 2
typical indian restaurant: 2
roll chicken tikka: 2
worst indian food: 2
shami kebab roll: 2

----- 20 most common 4-grams -----
chicken tikka masala platter: 3
ordered chicken tikka masala: 3
chicken tikka kati roll: 2
masala platter potato roti: 2
kati roll lamb kati: 2
kati roll chicken tikka: 2
got chicken tikka masala: 2
roll lamb kati roll: 2
street style indian food: 2
bhel puri shami kebab: 2
dahi puri chicken tikka: 2
times try platters veg: 1
across restaurant days opened: 1
disappointed chicken tikka masala: 1
steven etc great go: 1
grandma aunt cooking highly: 1
restaurant days opened back: 1
value money small place: 1
expectations food really good: 1
puri favorites kati rolls: 1

You can try dish: mango lassi

Sample review : Ordered the chicken tikka masala platter and samosas, also treated myself

to a **mango lassi** because, how could I resist?!
Also the tip appears in another 5 reviews.

You can try dish: chicken tikka

Sample review : We ordered dahi puri and **chicken tikka** masala platters.
Also the tip appears in another 20 reviews.

You can try dish: tikka masala

Sample review : We ordered dahi puri and chicken **tikka masala** platters.
Also the tip appears in another 11 reviews.

You can try dish: chicken tikka masala

Sample review : We ordered dahi puri and **chicken tikka masala** platters.
Also the tip appears in another 10 reviews.

You can try dish: chicken tikka kati roll

Sample review : Loved their **chicken tikka kati roll**, mango lassi and dahi Puri chaat!!
Also the tip appears in another 2 reviews.

You can try dish: vegetable kebab

Sample review : We ordered golgappa , papari chat, **vegetable kebab** n panner shashilk..papari chat and panner shasilk were good (not that great though) **vegetable kebab** was so horrible, just took a bite and thrown rest of it.. please don ever order it.. never gonna come back here..
A great new addition.
Also the tip appears in another 2 reviews.

You can try dish: bhel puri

Sample review : Their dahi puri and **bhel puri** are some of our favorites, and kati rolls are great too.
Also the tip appears in another 4 reviews.