COMP 202 - Foundations of Programming

Assignment 4

McGill University, Fall 2023

Due: Friday, December 1st, 11:59 pm on Ed Lessons

Late Penalty: 10% per day and up to 2 late days

Important notice

Please carefully read the following before starting the assignment:

- Make sure that all function names are spelled exactly as described in this document. Otherwise, a 50% penalty per question will be applied.
- You may make as many submissions as you like prior to the deadline, but we will only grade your final submission (all prior ones are automatically deleted).
- You must do this assignment individually.
- This assignment consists of two **independent** parts; Sentiment Analysis and Job Database. Both parts should be submitted in the **same** python file.
- We have created a "**FAQ for Assignment 4**" thread on Ed Discussion Board. The thread will have a collection of questions (with their answers) you all have asked us in the discussion board or office hours. If you have a question about assignment 4, check this thread, someone may have already asked it! While doing your assignment, don't forget to check this thread when you have a question.
- On Ed lessons, you will notice that you have three files: One named `'assignment4.py'` this is where you are going to submit your code (for both part A and part B) and a text file named `'posts.txt'` and a pickle file named `'sentiment_dictionary.pkl'` that the autograder will use to test your solution for part A. No need to import the text file or the pickle file to Ed lessons, just focus on writing the functions.

Part A: Sentiment Analysis

Sentiment analysis is one of the challenges related to natural language processing. It is the task of identifying if the sentiment behind a text, a social media message or voice message is either positive, negative or neutral. It is widely used in different areas such as marketing, entertainment or healthcare to evaluate the subjective information given by users, customers or patients.
The basic idea is to analyze a given text (for example, a social media post) and identify the sentiment that is behind it. For example:

- `I am so happy the weather is amazing!` ⇒ Positive sentiment
- `This movie was the worst movie ever.` ⇒ Negative sentiment
- `This is food.` ⇒ Neutral sentiment

There are many natural language processing libraries and a variety of algorithms in the literature that are used for sentiment analysis using machine learning algorithms or a rule based approach. The objective of part A is to build a rule-based approach to identify the sentiment for a given text. **Please note that**:

- Part A includes the use two files: a pickle file `sentiment_dictionary.pkl` and a text file named `posts.txt`. `sentiment_dictionary.pkl` consists of a sentiment dictionary wherein words are assigned to sentiments (ex: positive, negative, neutral). `posts.txt` includes texts to perform sentiment analysis on.
- For **all functions** you should provide type contract, small description and 3 examples in the docstrings (make sure also to have an example for functions that raise an exception).

Question A1 `read_text(text_path)` [10 points]

Input Parameter(s)

- `text_path`: A string having the file name.

Output Parameter(s)

- `text_list`: A list of strings.

Description

The function will open and read the text file located in the given path. The function will catch any FileNotFoundError by printing *'File does not exist'*. It will read the text line by line. Each line in the text follows the structure:

`user pseudonym, comment`

separated by ','. The function will ignore the user pseudonym and add only the `comment` into the `text_list`. Each element in the list is one comment from one line of the text and

the end of each line will be indicated with \n as the last character of the string. You should **not** use the `readline` or `readlines` functions. Assume that the content of the file is valid.

Examples

If we have a text file named `'text.txt'` with the following content:

```
user1,Hello
user2,How are you today?
user3,Good I hope!
user4,Ok take care!
```

Then the function will return the list of strings:

```
['Hello \n', 'How are you today?  \n', 'Good I hope!  \n',
'Ok take care!  \n']
```

Question A2 `read_pickle(path_to_pkl)` [5 points]

Input Parameter(s)

- `path_to_pkl`: A string containing the path to a pickle file.

Output Parameter(s)

- `word_dict`: A dictionary consisting of content of the pickle file.

Description

The function will load the object within the pickle file at `path_to_pkl` and return the object. **Assume** that the object within the pickle file will be a dictionary.

Examples

- If the function reads the provided pickle file `"sentiment_dictionary.pkl"`, it will return the following dictionary:

  ```
  {'POSITIVE': ['great', 'love', 'recommend', 'laugh', 'happy',
  'brilliant'], 'NEGATIVE': ['terrible', 'awful', 'hideous', 'sad',
  'cry', 'bad'], 'NEUTRAL': ['meh', 'indifferent', 'ignore']}
  ```

Question A3 `sentiment_frequencies(text,dictionary_word)` [15 points]

Input Parameter(s)

- `text`: A string consisting of words or expressions to be analyzed.

- `dictionary_word`: A dictionary that has a list of words in each sentiment category

Output Parameter(s)

- `dict_frequency`: A dictionary that contains the frequency of each sentiment category appearing in the text.

Description

`sentiment_frequencies` function will count the occurrence of each sentiment category in the input `text`, and will output the sentiment occurrence counts as frequency. First, the function will traverse the input string and determine the sentiment of each word in `text` by comparing it with the words in `dictionary_word`. Afterwards, the function will increment the value corresponding to the selected sentiment by 1. Last, the function will convert counts to frequency by dividing the values of each sentiment category by the total number of words in the input `text` and **rounded to 2 decimals**. All the words in input string should be in lower case. Assume that the text does not include any special characters such as '.' or '!' or '?'.

Examples

If we have the following:

```
text = 'i love this movie it is great and the adventure scenes are fun
i highly recommend it but the theatre was terrible and there was an
awful smell'
```

and the dictionary is: {'POSITIVE'['great','love','recommend','laugh','happy','brilliant'],
'NEGATIVE'['terrible','awful','hideous','sad','cry','bad'],
'NEUTRAL':'meh',indifferent','ignore']}

The function will compute 28 words in total with 3 positive words ('great','love','recommend') and 2 negative and 0 neutral words. Divided by the total number of words, the function will return the following dictionary:

{'POSITIVE': 0.11, 'NEGATIVE': 0.07, 'NEUTRAL': 0.0}

Question A4 `compute_polarity(dict_frequency)` [15 points]

Input Parameter(s)

- `dict_frequency`: The dictionary of frequencies as computed by `term_frequencies` function.

Output parameter(s)

- `polarity`: A string indicating the most prominent sentiment observed with the given frequencies.

Description

The function will traverse all the dictionary keys and will return the one that has the highest frequencies. If two or more keys have the same highest frequency, then the function will return the first highest key in the dictionary (see the last example). You should not use the built-in `max()` function.

Examples

- If we have the following dictionary {`'POSITIVE': 0.11, 'NEGATIVE': 0.07, 'NEUTRAL': 0.0`}, then the function will return the string `'POSITIVE'` since it has the maximum frequency.
- If we have the following dictionary {`'POSITIVE': 0.11, 'NEGATIVE': 0.07, 'NEUTRAL': 0.5`}, then the function will return the string `'NEUTRAL'`.
- If we have the following dictionary {`'POSITIVE': 0.07, 'NEGATIVE': 0.07, 'NEUTRAL': 0.01`}, then the function will return the string `'POSITIVE'`.

Question A5 `analyse_text(text_path, dict_path)` [15 points]

Input Parameter(s)

- `text_path`: A string containing the path to a text file.
- `dict_path`: A string containing the path to the sentiment dictionary saved in a pickle file.

Output Parameter(s)

- `list_polarity` a list of computed polarity for each line in the text file.

Description

The function will read the text file and pickle file. For each line in the text, the function will: put the text in lower case, remove leading and trailing whitespaces, convert it into a list of words and remove the following stop words from the text using this list of stop words ['!','.','?',';','\n'], compute the sentiment frequencies and polarity, and add the computed polarity value to `list_polarity`.

Notes:

- To remove the stop words within the text, one possibility is to use the `replace()` method so you will replace each stop word you find in the string by an empty string
- On Ed lessons, you will notice that you have three files: One named `'assignment4.py'` this is where you are going to submit your code (for both part A and part B) and you already have a file named `'posts.txt'` and a pickle file named `'sentiment_dictionary.pkl'`

that the autograder will use to test your solution for part A. No need to import the text file to Ed lessons, just focus on writing the functions.

Examples

- If we test the function with the provided text and pickle file it will return the following list:

```
['POSITIVE', 'NEGATIVE', 'NEUTRAL', 'POSITIVE']
```

Part B: Job Database

In part B of this assignment, we are going to create a Job Database for the talent recruitment company, CompLink. This database will keep track of the job opportunities CompLink clients can access. To do so, we will rely on using object oriented programming where we will define two main classes: Company and JobOffer.

IMPORTANT: Define all these classes in the same .py file you used for part A.

Question B1 [13 points]

Define Class `Company` that has two instance attributes:

- `name`: String containing the name of the Company offering the job.
- `location`: String containing the Company's location.

Include the following instance methods:

- **`__init__`:** The constructor of the `Company` class, takes two inputs to initiate the attributes of the class: the company name (string) and location (string). **Note that** the order of the input parameters has to be the same as they are listed above.
- **`update_location`:** This instance method of the `Company` class takes one input `new_location` (string), and updates the location of the company.

Example

If we create a `Company` object named `company1` with:

```
>>> company1 = Company('Cycle4Energy','Montreal')
>>> print(company1.name, company1.location)
    Cycle4Energy Montreal

>>> company1.update_location('San Francisco')
>>> print(company1.location)
    San Francisco
```

Question B2 [15 points]

Define `Class JobOffer` that has six instance attributes:

- `title`: String. The title of the job offer.
- `description`: String. The description of the job.
- `company`: Company. An object of the previously defined class Company.
- `contract`: String. The type of contract.
- `salary`: Float. The salary of the job offer.

Include the following instance methods:

- **`__init__`:** The constructor of the `JobOffer` class, takes five inputs to initiate the attributes of the class: title (string), company (Company), contract (string), salary (float), description (string). **Note that** the order of the input parameters has to be the same as they are listed above.
- **`update_description`:** The method takes as input a new job description (string) and updates the instance attribute `description` with the new description value.
- **`__str__`:** Returns a string for the object of the class `JobOffer`. The string should be configured so that each instance attribute will represent a single line, and include the attribute name and value. Example:

```
'Title: Fraud Analytics Manager\nCompany
Name: Harnham\nLocation: London\nContract: Permanent\n
Description: Design,implement and optimize fraud
software solutions Lead multiple workstreams across Fraud
Analytics and Data Science platforms
Develop business\nSalary: 120000'
```

Example

Creating a `JobOffer` object and using instance methods:

```
>>>cmp1 = Company('Harnham','London')
>>>about = 'Design, implement and optimize fraud software solutions
Lead multiple workstreams cross Fraud Analytics and Data Science Develop
business'
>>>job = JobOffer('Fraud Analytics Manager',cmp1,'Permanent',120000,about)

>>>str(job)
   'Title: Fraud Analytics Manager\nCompany Name: Harnham
   \nLocation: London\nContract: Permanent
   \nDescription: Design, implement and optimise fraud
   software solutions Lead multiple workstreams cross Fraud
   Analytics and Data Science Develop business\nSalary: 120000'
>>>print(job)
   Title: Fraud Analytics Manager
   Company Name: Harnham
   Location: London
   Contract: Permanent
   Description: Design, implement and optimise fraud
   software solutions Lead multiple workstreams cross Fraud
   Analytics and Data Science Develop business
   Salary: 120000

>>>new_about = 'Enjoy your job, that's it!'
>>>job.update_description(new_about)
>>>print(job)
   Title: Fraud Analytics Manager
   Company Name: Harnham
   Location: London
   Contract: Permanent
   Description: Enjoy your job, that's it!
   Salary: 120000
```

Question B3 [12 points]

`build_job_database()`

Input Parameter(s)

- `None`.

Output Parameter(s)

- No output

Description

The function creates two `JobOffer` objects, `offer1` and `offer2`, updates the job description of `offer1` and prints `offer1` object. For creating a job object, the function will ask user to input class attributes one by one. After creating two `JobOffer` objects, the function will ask user to input a new description for `offer1`.

IMPORTANT: To pass autograder tests, the text of your outputs MUST EXACTLY MATCH the text given in the example below. User-entered values may be modified.

Example

```
>>>build_job_database()
   Welcome to New Job Entry! Let's create our first entry.
   PLEASE ENTER REQUESTED DATA FOR OFFER 1
   Title: Researcher
   Company: Info4All
   Location: Toronto
   Contract: Permanent
   Description: Investigate how to create wisdom from information
   Salary: 100000
   PLEASE ENTER REQUESTED DATA FOR OFFER 2
   Title: Engineer
   Company: Air Carbon
   Location: CalgarY
   Contract: Project-based
   Description: Optimize carbon capture systems and lead installation
   Salary: 185000
   Employer modified OFFER 1 description!
   PLEASE ENTER THE UPDATED OFFER 1 DESCRIPTION
   Description: Investigate data storytelling
   Find updated OFFER 1 below:
   Title: Researcher
   Company: Info4All
```

```
Location: Toronto
Contract: Permanent
Description: Investigate data storytelling
```