Homework 2

SE 6301.M02

Create a Python program that performs the tasks listed below. Upload your completed Python program file using the file name `hw02_⟨your_net_id⟩.py`

For example:

```
hw02_cid021000.py
```

Your Python file should have a header using the following format.

```
################################################################################
#
#         FILE:
#           filename.py
#       AUTHOR:
#           Your Name
#           Your NetID
#  DESCRIPTION:
#           Homework 2
#           Description of your program, what it does
# DEPENDENCIES:
#           Created with ActiveState Python 3.10.11 (Python version)
#           Any dependencies, i.e. extra libraries required to run (like
#           datetime, NLTK, re, etc.)
#
################################################################################
```

Extra help references:

- Python reference documenntation – https://docs.python.org/3/index.html
- Python regular expression module – https://docs.python.org/3/library/re.html#module-re
- Python regular expression operations – https://docs.python.org/3/library/re.html
- Python date class – https://docs.python.org/3/library/datetime.html

Submit a single file Python program that does the following:

1. Provided a plain text file (text_news.txt), your Python program should open it, and perform the following operations

2. Identity and take note of non-content "noise" to remove later, such as file numbers and indexing information.

3. First split on paragraphs "<p>", next on sentences.

4. For each sentence that contains a date, identify and extract dates based on on common date pattern using regular expressions. referenced in the text as "date" objects. If no year is mentioned, assume the current calendar year. For each instance, your output should return a 2-tuple in the form,
   `(string, [date list])`
   where the string "string" is the original sentence from which the date was extracted and [date list] is a list of dates in the sentence.

5. Display the 2-tuple to the screen with the two halves on separate lines

   “The celebration was held on July 4, 2023 at the park.”
   [2023-07-04]

6. Display a horizontal break between each displayed sentence that has the number of the sentence that contains the date:

   ```
   ----------------------- SENTENCE: 0034 -----------------------
   "The celebration was held on July 4, 2023 at the park."
   [2023-07-04]
   ----------------------- SENTENCE: 0051 -----------------------
   “Mary won’t arrive until Jan. 22.”
   [2024-01-22]
   ```

7. Your date matcher should recognize multiple common formats. Visually scan the text to identify possible formats.

   July 4, 2023
   2019 Sep 21
   Jan. 7, 2005
   etc.

   If no year is specified, assume the current year.

8. Use your word list of the entire text to count the number of instances of each word. You may do this using a Python dictionary data structure, (i.e. `<class 'dict'>`). Be careful to count instances of words ignoring case. That is, “the”, “The”, and “THE” should all be counted as instances of the same word.

9. Sort the dictionary by frequency, having the most frequent word token first.

10. For the top ten entries, display the following information to the screen. Exclude entries for punctuation.

    10.1. The word token, normalized to all lowercase.

    10.2. The word count

    10.3. The word frequency in the text as a floating point number in the range (0.0, 1.0). That is, the word count divided by the total number of words.

11. Then ***for each word***:

    11.1. Calculate the word count and frequency percentage of each word / unigram.

    11.2. Sort the words by frequency.

    11.3. For the top 20 most frequent words / unigrams, on each line display/print the word form (lexeme), its count (integer number of instances in the text), and frequency (float to exactly five decimal places).