UTD Summer Workshop on NLP with Python

Final Project
Assignment 5

Upload your completed Python program file to the MS Teams "Assignments" → Assignment 4" folder. Be sure to include your name in the comments at the top of the file in the following format:

```
###############################################################################
#
#          FILE:
#            filename.py
#       AUTHOR:
#            Your Name
#  DESCRIPTION:
#            Assignment 5
#            Description of your program, what it does
# DEPENDENCIES:
#            Created with Python 3.10.11 (Python version)
#            Any dependencies, i.e. extra libraries required to run (like
#            datetime, NLTK, or re)
#
###############################################################################
```

Extra help references:

- Python re reference – https://docs.python.org/3/library/re.html
- Requests
  - https://pypi.org/project/requests/
- Beautiful Soup
  - https://pypi.org/project/beautifulsoup4
- NLTK references
  - Tokenize – https://www.nltk.org/api/nltk.tokenize.html
  - WordNet –
    - Sample usage

This project utilizes most of the techniques you have learned so far.

Submit a single file Python program that does the following:

1. Create a text corpus. Using Python, download the source HTML for one or more webpages of your choosing (Wikipedia articles may be easiest). You may assign the HTML source directly to a variable without first saving it to a file. Otherwise, you may save all source code to local files, then open them using Python's built-in open() function.

2. Use the Beautiful Soup library to extract just the text without any HTML markup. Note that your resulting text may still have non-relevant phrases and words, such as text menus from the website. You may use techniques discussed in lecture to remove these if you desire. The body of Wikipedia articles is all within HTML <p> tags. Other web pages may use different tags.

3. In your text(s), locate the following instances:

    3.1. ‘operator.n.01’ ‘operate.v.01’ ‘vehicle.v.01’

    3.2. ‘event.n.01’ ‘occur.v.01’/‘act.v.01’ DATE

4. Display each located original sentence in it’s original form followed by the words that match the above pattern. Example:

```
"The game was played on February 12, 2023, at State Farm
Stadium in Glendale, Arizona."
      game.n.01
        => activity
          => act, deed, human action, human activity
            => event
      play.v.02
        => act
      February 12, 2023
        => date.n.01
```

5. Among the matches from #4, locate any ***trigrams*** that occur more than 3 times in your overall corpus.

6. You may find it useful to optionally preprocess your text with one or more of the following, although the details are up to you.

    6.1. Normalize your text into lowercase

    6.2. Stem your words

    6.3. Utilize a part-of-speech (POS) tagger to first narrow down word instances to noun or verb before searching for WordNet synsets.

7. Create a list of bigram tuples from the word list using your own bigram function.

8. Convert the list of bigram tuples into a dictionary of tuples with the bigram counts, e.g. ((word1,word2), count).

9. Sort the list by count. Use a reverse sort so that the most frequent bigram is first.

10. Display the top 10 most frequent bigrams to the screen.