UTD Summer Workshop on NLP with Python

Assignment 4

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 4
#           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

Submit a single file Python program that does the following:

1. Using Python, download the source HTML for one or more webpages of your choosing (A Wikipedia article may be easiest). You may assign the HTML source directly to a variable without first saving it to a file. Otherwise, you may save the source code to a local file, then open it 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. Once you have the plain text, tokenize it into words.

4. Create a list of bigram tuples from the word list using your own bigram function.
5. Convert the list of bigram tuples into a dictionary of tuples with the bigram counts, e.g. ((word1,word2), count).
6. Sort the list by count. Use a reverse sort so that the most frequent bigram is first.
7. Display the top 10 most frequent bigrams to the screen.