Arsh misra

```
from google.colab import drive
import nltk
drive.mount('/content/drive')
nltk.download('punkt')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
True
```

I decided to use google drive on colab because spyder does not work for me. The content of the files is the same aside from the input text, which I will list in full in this notebook.

```
import os

# Path to the directory
path = '/content/drive/My Drive/NLPHWPOSNEG/'

# List files
files = os.listdir(path)
print(files)
```

```
['positive-words.txt', 'input-text2.txt', 'negative-words.txt', 'example-text.txt']
```

1. Create a function called gen_senti that tokenizes arbitrary text and compares each token with the positive and negative lexicons of each dictionary and ouputs the sentiment score S. Positive and negative words, pw and nw, count as a score of 1 and -1, respectively, for each word matched. The total count for pw and nw are pc and nc, respectively. Each message sentiment, S, is normalized between -1 and 1. Any text that does not have any positive or negative words would have been ignored, and not scored.

I used a different example since the question did not specify that we needed to use the original example. The input text is listed as:

"Karl Peterson has been living the cruel inverse of the American dream. His rent keeps getting higher, but his apartments keep getting smaller. Peterson left the Midwest nine years ago for the epicenter of an economic boom, only to gradually learn that endless sunshine and desert views are increasingly among the few bargains left in Arizona. Peterson married his wife and they struggled to save for a home, moving through four apartments as their rent nearly tripled from $625 to $1,800 a month"

This is an article taken from yahoo finance: https://finance.yahoo.com/news/american-despair-arizona-high-home-151927862.html

```
from nltk.tokenize import word_tokenize

with open('/content/drive/My Drive/NLPHWPOSNEG/input-text2.txt', 'r', encoding='ISO-8859-1') as f:
    input_text = f.read()

def gen_senti(input_text, path="/content/drive/My Drive/NLPHWPOSNEG/"):
    # Load positive words
    with open(path + 'positive-words.txt', 'r') as f:
        positive_words = set(f.read().splitlines())

    # Load negative words
    with open(path + 'negative-words.txt', 'r') as f:
        negative_words = set(f.read().splitlines())

    tokens = word_tokenize(input_text.lower())

    # counts and lists to store matched words
    pc = 0  # Positive count
    nc = 0  # Negative count
    matched_positive_words = []  # To store matched positive words
    matched_negative_words = []  # To store matched negative words

    # positive and negative tokens
    for token in tokens:
        if token in positive_words:
            pc += 1
            matched_positive_words.append(token)  # Add to matched positive words
        elif token in negative_words:
            nc += 1
            matched_negative_words.append(token)  # Add to matched negative words

    # sentiment score
    total_count = pc + nc
    if total_count > 0:
        S = (pc - nc) / total_count  # sentiment score between -1 and 1
    else:
        S = 0  # Neutral sentiment if no matched tokens

    return S, pc, nc, matched_positive_words, matched_negative_words

# function with the loaded input text
score, positive_count, negative_count, matched_pos, matched_neg = gen_senti(input_text)
print(f"Sentiment Score: {score}, Positive Count: {positive_count}, Negative Count: {negative_count}")
print(f"Matched Positive Words: {matched_pos}")
print(f"Matched Negative Words: {matched_neg}")

```

```
Sentiment Score: -0.5, Positive Count: 1, Negative Count: 3
Matched Positive Words: ['boom']
Matched Negative Words: ['cruel', 'desert', 'struggled']
```

2. Using the dataframe from the lecture, the_data, column body, apply this function to each corpus and add a column called simple_senti.

```
nltk.download('stopwords')
import pandas as pd

the_path = "/content/drive/My Drive/NLP Text Data/"
# Some preliminary cleaning functions
def clean_txt(var_in):
    import re
    # Modify the regex to preserve punctuation and capitalization
    tmp_t = re.sub("[^A-Za-z0-9!@#\$%\^&\*\(\)\-_\+=\.,;:!?'\"]+", " ", var_in).strip()

    return tmp_t

def read_file(full_path_in):
    with open(full_path_in, "r", encoding="UTF-8") as f_t:
        text_t = f_t.read() # Reads the entire file
        text_t = clean_txt(text_t)
    return text_t

def file_crawler(path_in):
    import os
    import pandas as pd
    my_pd_t = pd.DataFrame()
    for root, dirs, files in os.walk(path_in, topdown=False):
        for name in files:
            try:
                txt_t = read_file(root + "/" + name)
                if len(txt_t) > 0:
                    the_lab = root.split("/")[-1]
                    tmp_pd = pd.DataFrame({"body": txt_t, "label": the_lab}, index=[0])
                    my_pd_t = pd.concat([my_pd_t, tmp_pd], ignore_index=True)
            except Exception as e:
                print(f"Error with file {root}/{name}: {e}")
                pass
    return my_pd_t

df = file_crawler(the_path)
print("Sample of the DataFrame:")
print(df.head())  # Print the first few rows of the DataFrame

```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
Error with file /content/drive/My Drive/NLP Text Data/fishing/UK MongoBD report Nov122018.xls: 'utf-8' codec can't decode byte 0xd0 in p
Error with file /content/drive/My Drive/NLP Text Data/fishing/UK segment count Nov122018.xlsx: 'utf-8' codec can't decode bytes in posit
Error with file /content/drive/My Drive/NLP Text Data/fishing/UK vendor count Nov122108 .xlsx: 'utf-8' codec can't decode bytes in posit
Sample of the DataFrame:
                                               body           label
0  Machine Learning Total 239.99 Computer Science...  machinelearning
1  Rendezvous Server to the Rescue: Dealing with ...  machinelearning
2  The 10 Algorithms Machine Learning Engineers N...  machinelearning
3  Find a Job in Artificial Intelligence or Machi...  machinelearning
4  xkcd: Machine Learning Archive What If? Blag S...  machinelearning
```

```

# Load positive and negative words
def load_word_lists(path):
    with open(path + 'positive-words.txt', 'r') as f:
        positive_words = set(f.read().splitlines())
    with open(path + 'negative-words.txt', 'r') as f:
        negative_words = set(f.read().splitlines())
    return positive_words, negative_words

# sentiment analysis function
def gen_senti(input_text, positive_words, negative_words):
    # Tokenize the input text
    tokens = word_tokenize(input_text.lower())

    # Initialize counts
    pc = 0  # Positive count
    nc = 0  # Negative count

    # Count positive and negative tokens
    for token in tokens:
        if token in positive_words:
            pc += 1
        elif token in negative_words:
            nc += 1

    # Calculate sentiment score
    total_count = pc + nc
    if total_count > 0:
        S = (pc - nc) / total_count  # Normalized sentiment score between -1 and 1
    else:
        S = 0  # Neutral sentiment if no matched tokens

    return S


path = "/content/drive/My Drive/NLPHWPOSNEG/"
positive_words, negative_words = load_word_lists(path)

the_path = "/content/drive/My Drive/NLP Text Data/"
df = file_crawler(the_path)

# Apply sentiment analysis to the 'body' column
df['simple_senti'] = df['body'].apply(lambda x: gen_senti(x, positive_words, negative_words))

# Display the updated DataFrame
print("Updated DataFrame with Sentiment Scores:")
print(df[['body', 'simple_senti']].head())  # Display the 'body' and 'simple_senti' columns

```

```
Error with file /content/drive/My Drive/NLP Text Data/fishing/UK MongoBD report Nov122018.xls: 'utf-8' codec can't decode byte 0xd0 in p
Error with file /content/drive/My Drive/NLP Text Data/fishing/UK segment count Nov122018.xlsx: 'utf-8' codec can't decode bytes in posit
Error with file /content/drive/My Drive/NLP Text Data/fishing/UK vendor count Nov122108 .xlsx: 'utf-8' codec can't decode bytes in posit
Updated DataFrame with Sentiment Scores:
                                               body  simple_senti
0  Machine Learning Total 239.99 Computer Science...      0.170732
1  Rendezvous Server to the Rescue: Dealing with ...      0.629630
2  The 10 Algorithms Machine Learning Engineers N...      0.411765
3  Find a Job in Artificial Intelligence or Machi...      0.730769
4  xkcd: Machine Learning Archive What If? Blag S...      0.000000
```

3. Using vaderSentiment, apply the compound value of sentiment for each corpus in column body on a new column of the_data called vader.

```python
# Install and import necessary libraries
import nltk
from nltk.sentiment.vader import SentimentIntensityAnalyzer
import pandas as pd

# VADER lexicon
nltk.download('vader_lexicon')

#VADER sentiment analyzer
sia = SentimentIntensityAnalyzer()


# Function to compute the compound score
def compute_vader_sentiment(text):
    score = sia.polarity_scores(text)
    return score['compound']  # Return the compound score

# Apply the function to the 'body' column and create a new 'vader' column
df['vader'] = df['body'].apply(compute_vader_sentiment)

# Display the updated DataFrame
print("Updated DataFrame with VADER Sentiment Scores:")
print(df[['body', 'vader']].head())  # Display the 'body' and 'vader' columns

```

```
[nltk_data] Downloading package vader_lexicon to /root/nltk_data...
Updated DataFrame with VADER Sentiment Scores:
                                                body   vader
0  Machine Learning Total 239.99 Computer Science...  0.9941
1  Rendezvous Server to the Rescue: Dealing with ...  0.9968
2  The 10 Algorithms Machine Learning Engineers N...  0.9473
3  Find a Job in Artificial Intelligence or Machi...  0.9992
4  xkcd: Machine Learning Archive What If? Blag S...  0.0000
```

4. Compute the mean, median, and standard deviations of both sentiment measures, simple_senti and vader

```python
import pandas as pd

# mean, median, and standard deviation for both sentiment measures
statistics = {
    'simple_senti': {
        'mean': df['simple_senti'].mean(),
        'median': df['simple_senti'].median(),
        'std_dev': df['simple_senti'].std()
    },
    'vader': {
        'mean': df['vader'].mean(),
        'median': df['vader'].median(),
        'std_dev': df['vader'].std()
    }
}

# results
print("Sentiment Statistics:")
for measure, stats in statistics.items():
    print(f"{measure} - Mean: {stats['mean']:.4f}, Median: {stats['median']:.4f}, Standard Deviation: {stats['std_dev']:.4f}")

```

```
Sentiment Statistics:
simple_senti - Mean: 0.3784, Median: 0.4286, Standard Deviation: 0.4338
vader - Mean: 0.8758, Median: 0.9851, Standard Deviation: 0.2641
```

we see that simple_senti and vader differ significantly across all measurements.This is mostly because vader is a lot more flexible and smarter than a simple matching function like simple_senti.

Homework 2 concluded

```
####################

####################
```