MIST.3050
Programming Assignment: Basis Text Analysis

This assignment will give you the opportunity to develop a program that analyzes text by recognizing and counting the words of different sentiments. The program can be useful in several application scenarios. For example, in your cover letter for job applications, you may want to avoid using negative words – you can use the program to identify the negative words, then revise your letter to avoid them. You may also use the program to count positive and negative words in online reviews, then use these counts as an alternative to star ratings in a predictive model that you need to build.

1. Background

Words such as "able" and "joy" often have a positive sentiment. And other words such as "novice" and "revoke" often have a negative sentiment. Linguists and text analysis researchers have categorized words and documented their results as dictionaries. Several dictionaries are used in practice. The following are two popular ones:

(1) General Inquirer Harvard IV-4 Dictionary (**IV-4**). Originally developed by Professor Philip Stone at Harvard University for applications in psychology and sociology, the dictionary has been widely used in many areas. The dictionary has 11,888 entries and more than180 categories, including different categories developed by different researchers. Additional information about this dictionary can be found at http://www.wjh.harvard.edu/~inquirer/. Spreadsheet versions can be found at http://www.wjh.harvard.edu/~inquirer/spreadsheet_guide.htm. A plain text version can be found at wjh.harvard.edu/~inquirer/inqdict.txt.

(2) Master Dictionary for Business (**MD**). Developed by professors Tim Loughran and Bill McDonald at the University of Notre Dame, this dictionary is specific to financial reports and can be used for business related texts. One of the motivations of developing this business specific dictionary is that frequently used words in business documents such as "liability" and "vice" have no negative implications, but they are categorized negative in the Harvard IV-4 dictionary. The master dictionary has more than 80,000 entries. Different forms of the same base word are included as separate entries. These different forms, also known as inflections in natural language processing (NLP), can be singular vs plural for nouns and different tenses for verbs. Information about this Master Dictionary can be found at https://sraf.nd.edu/textual-analysis/resources/.

With either dictionary, you can look up a word to see if it has been categorized as positive or negative. Words are in uppercase in both dictionaries. The main difference between the two dictionaries is different classifications of word sentiment. Recall the example given earlier, "liability" is negative according to IV-4, but it is not negative according to MD. Likewise, "joy" is positive in IV-4 but not in MD.

Another notable difference is: IV-4 captures different word meanings, while MD captures different word forms. A word can have different meanings, also known as word senses in NLP. In this case, the word has multiple entries in IV-4, each is listed as the word followed by "#" and

a number, e.g., “ABOVE#1”, “ABOVE#2”, etc. In contrast, MD has only one entry for the word “above”. In terms of word forms, IV-4 is quite limited, but MD includes all forms of a word. For example, IV-4 has two entries for the word “book”: book and booking. In contrast, MD has book, books, booking, bookings, booked, and many book-prefixed words such as bookends and bookkeeper – a total of 53 entries. As a results, MD ends up having more entries.

2. Tasks

Your main task of the assignment is to identify and count the occurrences of positive words and negative words in any given text. This is conceptually simple: for each word in the text, look it up in a dictionary (or in both dictionaries) to see if it is positive or negative. For this assignment, you may choose either dictionary or use both.

For example, given the text (which is from an online review):

> I thought they were pretty nice at first. Sound quality is good. They charged quick in the case. But transparency mode amplified sounds like crazy. At the gym someone was moving weights, the cling of them hitting was so loud it hurt my ears. When I took the airpods out it was much quieter in real life. Even in noise cancelling mode they pick up voices and then they end up sounding robotic. There's a white noise in the background too. It's like a little static, annoying as hell. Just now I started hearing a popping noise in the right bud. It kept happening even when the music was off. When I turned to transparency mode that popping became a buzz.

using MD, your program should be able to output something like below:

```
Positive words: {'transparency': 2, 'good': 1}

Negative words: {'hurt': 1, 'cancelling': 1, 'annoying': 1}

Positive occurrences: 3; Negative occurrences: 3
```

In addition, your program should produce decorated version of the text where positive words are bolded and negative words are underlined:

> I thought they were pretty nice at first. Sound quality is **good**. They charged quick in the case. But **transparency** mode amplified sounds like crazy. At the gym someone was moving weights, the cling of them hitting was so loud it <u>hurt</u> my ears. When I took the airpods out it was much quieter in real life. Even in noise <u>cancelling</u> mode they pick up voices and then they end up sounding robotic. There's a white noise in the background too. It's like a little static, <u>annoying</u> as hell. Just now I started hearing a popping noise in the right bud. It kept happening even when the music was off. When I turned to **transparency** mode that popping became a buzz.

The decoration can be done through HTML, additional information about which will be provided.

3. Implementation

As you know, there is usually more than one way to accomplish a certain task. The following suggestions intend to give you some ideas about implementation, and you may develop your own implementation different from the suggestions.

You may use either dictionary or use both dictionaries (you can earn up to 5% bonus points if you successfully use both dictionaries). In the following description, I assume you use only one dictionary. Think about how you want to represent the dictionary so that you can easily check if a word is categorized as positive or negative. You may want to implement functions or use OOP to support word lookup.

The process has two main steps:

1. represent dictionary to support word lookup
2. for each token in the text, obtain the word and determine if it is positive or negative.

3.1 Processing Dictionary File

For step 1, you need to process the dictionary file, find positive and negative words, and represent these words to support lookup in the second step. In IV-4, a positive word is marked by Positiv in the column named *Positiv*; in MD, a positive word is marked by a non-zero value in the column named *Positive*. Negative words are marked using the *Negativ* and *Negative* columns, respectively. See the Appendix for example entries in the two dictionary files.

The following example code gives you some idea about how to read the IV-4 dictionary file, find positive words, and remove characters that are not in the alphabet plus the space character (e.g., converting "ABOVE#1" to "ABOVE"):

```
import re, csv
print("Positive words in IV-4")
with open('iv4.csv' , 'r') as f:
  csvreader=csv.DictReader(f)
  for row in csvreader:
    if row['Positiv']:
      print(re.sub(r'[^a-zA-Z\s]', '', row['Entry']))
```

Note that the example uses two packages in the Python standard library (meaning that you need not install additional packages; and of course you could use the pandas package, if you prefer). It uses csv package's DictReader function, which understands that the first row is the headers (that define column names) and coverts each row of data as a dictionary (with the column name as the key). The sub function in the regular expression (re) package replaces anything that is not in the alphabet or a space with an empty string.

Similarly, the following code processes the MD files, finds positive words, cleans each word (not necessary because MD word entries uses characters only in the alphabet), and converts the word to upper case (also unnecessary because MD entries are in uppercase; included here to show how it is done because you will need case conversion for words in the text):

```
import re, csv
print("Positive words in MD")
with open('md2020.csv' , 'r') as f:
  csvreader=csv.DictReader(f)
  for row in csvreader:
    if int(row['Positive'])>0:
      print(re.sub(r'[^a-zA-Z\s]', '', row['Word']).upper())
```

Note that above examples only find the words. You need to decide how to represent the words to support lookup. I recommend Python dictionary (dict); you may also use list or set.

3.2 Processing Text

The text can be stored in a text file and read in by your program. If this becomes a challenge, you may hard code it (i.e., storing it in a variable in your code).

As noted earlier, in addition to space and the alphabet characters, there are often other characters in the text. For example, the following text includes quotation marks and a question mark:

It is a “good” practice?

You need to remove these punctuation marks for sentiment lookup. The same regular expression shown in the example code earlier can help you accomplish this task:

```
>>> re.sub(r'[^a-zA-Z\s]', '', 'It is a "good" practice?').upper()
' IT IS A GOOD PRACTICE'
```

The same regular expression also works on the single token.

You can use the split method of string object to convert the text into a list, with each token as a list element. This can be useful when you need to preserve the order of the tokens.

3.3 Using HTML to Highlight Positive and Negative Words

HTML is used for webpages. It tells the browser how to render texts and other contents. In this assignment, we will use in-line style sheet to highlight positive and negative words. For any word that you need to decorate with a certain style, you simply tag it with the tag and specify the style for the tag. For example, with the following HTML:

```
<html>
</body>
<span style="font-weight:bold">"Boom"</span> and <span style="text-
decoration:underline">decline!</span>
```

```
</body>
</html>
```

the browser would render "**Boom**" and decline!

Note that when looking up a word to demine its sentiment, you need to make sure the word contains only the alphabet (no punctuation marks). When generating the decorated version of the original text, you need to preserve everything (e.g., upper- and lower-cases as well as all punctuation marks) and add necessary decorations. The following pseudo code may be helpful.

```
outlist=[]
For each original text token t
  Get the clean version tc
  If tc is positive
    Append <span style="font-weight:bold">t</span> to outlist
  If tc is negative
    Append <span style="text-decoration:underline">t</span> to outlist
  Else
        Append t to outlist
```

Note that the above is pseudo code. You need to think about how to convert it to Python syntax (pay attention to string concatenation). In the pseudo code, variables are bolded.

Then you can assemble the output HTML by joining the outlist and adding the required <html> and <body> tags. Recall the use of split and join methods of string (you need them when implementing the algorithm). For example:

```
>>> 'HTML is good!'.split()
['HTML', 'is', 'good!']
>>> ' '.join(['HTML', 'is', 'good!'])
'HTML is good!'
```

If you plan to use Python dictionary to keep track of positive and negative word occurrences, the above algorithm can be updated to include that:

```
outlist=[]
pos={}
neg={}
For each original text token t
```

```
Get the clean version tc
If tc is positive
    Append <span style="font-weight:bold">t</span> to outlist
    Update pos regarding tc
If tc is negative
    Append <span style="text-decoration:underline">t</span> to outlist
    Update neg regarding tc
Else
        Append t to outlist
```

You have seen updating dictionary in PPP3. The following code excerpt serves as a reminder about how it is usually done:

```
if tc in pos:
  pos[tc] += 1 //increase existing by 1
else:
  pos[tc] = 1 //add key tc and value 1 to pos
```

4. Comments

The two dictionaries have been used in numerous research projects of various domains. They are also used in text mining and NLP tools for applications such as sentiment analysis, opinion mining, and text classification. What you have done in this assignment is the first few steps usually done in these different applications. If you want to learn more, there are many online resources. Or if you plan to attend graduate school, text analysis techniques are covered in courses of the Big Data track of the M.S. in Business Analytics program at UMass Lowell.

5. Submission Requirement

Your submission should include the following files:

- Report
- All source code files (and other dependent files, e.g., files for test data, if any)

The report should include the following information:

- Cover page, with your name
- Documentation of your design and implementation (use diagrams and narratives effectively to let the reader know how you designed and implemented the solutions)
- Documentation of your testing
- Documentation of sources of help (including collaboration with other people)
- A summary to reflect on what you learned from the assignment

- Appendix: code listing to include source code for the assignment [Yes, you are including source code in the report and as separate files in your Blackboard submission]

Appendix: IV-4 and MD Dictionaries

A CSV file (iv4.csv) for the IV-4 dictionary is provided for your convenience. The first 24 rows and first 10 columns of the file are shown below:

```
Entry,Source,Positiv,Negativ,Pstv,Affil,Ngtv,Hostile,Strong,Power
A,H4Lvd,,,,,,,
ABANDON,H4Lvd,,Negativ,,,Ngtv,,,
ABANDONMENT,H4,,Negativ,,,,,,
ABATE,H4Lvd,,Negativ,,,,,,
ABATEMENT,Lvd,,,,,,,,
ABDICATE,H4,,Negativ,,,,,,
ABHOR,H4,,Negativ,,,,Hostile,,
ABIDE,H4,Positiv,,,Affil,,,,
ABILITY,H4Lvd,Positiv,,,,,,Strong,
ABJECT,H4,,Negativ,,,,,,
ABLE,H4Lvd,Positiv,,Pstv,,,,Strong,
ABNORMAL,H4Lvd,,Negativ,,,Ngtv,,,
ABOARD,H4Lvd,,,,,,,,
ABOLISH,H4Lvd,,Negativ,,,Ngtv,Hostile,Strong,Power
ABOLITION,Lvd,,,,,,,,
ABOMINABLE,H4,,Negativ,,,,,Strong,
ABORTIVE,Lvd,,,,,,,,
ABOUND,H4,Positiv,,,,,,,
ABOUT#1,H4Lvd,,,,,,,,
ABOUT#2,H4Lvd,,,,,,,,
ABOUT#3,H4Lvd,,,,,,,,
ABOUT#4,H4Lvd,,,,,,,,
ABOUT#5,H4Lvd,,,,,,,,
```

We mainly focus on columns 1, 3, and 4:

- Column 1 is word entry. It is in uppercase. Note when a word has multiple senses (e.g., ABOUT), each sense has its own entry where the word is followed by "#" and an integer.
- Column 3 indicates if the word is positive (where the value is "Postiv", empty otherwise)
- Column 4 indicates if the word is negative (where the value "Negativ", empty otherwise)

Observe that not all forms of a word appear in the dictionary. For example, "abandon" and "abandonment" are in it, but other forms such as "abandons" and "abandoned" are not.

A CSV file (md2020.csv) for the MD dictionary also provided to you. The first 20 rows and first 9 columns are shown below:

```
Word,Seq_num,Word Count,Word Proportion,Average Proportion,Std Dev,Doc Count,Negative,Positive
AARDVARK,1,312,1.4220500553431755e-08,1.3352006151127348e-08,3.7007466419293765e-06,96,0,0
AARDVARKS,2,3,1.367355822445361e-10,8.882162836473433e-12,9.36284874474273e-09,1,0,0
ABACI,3,9,4.102067467336083e-10,1.200533452194755e-10,5.359746954401173e-08,7,0,0
ABACK,4,15,6.836779112226805e-10,4.0805491679245766e-10,1.4069138859598925e-07,14,0,0
```

ABACUS,5,8009,3.6503842606549656e-07,3.7986978725328413e-07,3.523914003700375e-05,1058,0,0

ABACUSES,6,0,0.0,0.0,0.0,0,0,0

ABAFT,7,4,1.8231410965938147e-10,2.3635613858933468e-11,2.4914728723685797e-08,1,0,0

ABALONE,8,140,6.380993838078351e-09,5.055956353906653e-09,1.0806020090262814e-06,47,0,0

ABALONES,9,1,4.557852741484537e-11,8.502178008466845e-11,8.962299854182723e-08,1,0,0

ABANDON,10,118075,5.381684624507866e-06,4.661541031258797e-06,3.338304691637121e-05,62949,2009,0

ABANDONED,11,229309,1.0451566542970776e-05,1.0906749011813076e-05,7.769607144036667e-05,105201,2009,0

ABANDONING,12,20482,9.335393985108627e-07,8.512341538575199e-07,1.2073656043289562e-05,12810,2009,0

ABANDONMENT,13,276244,1.2590794727186544e-05,1.2509799855827664e-05,8.22414561339473e-05,91251,2009,0

ABANDONMENTS,14,14874,6.779350167684099e-07,9.385420076883375e-07,2.2776023279262917e-05,6309,2009,0

ABANDONS,15,7501,3.418845341387551e-07,1.7285579272106007e-07,3.683098291142433e-06,5364,2009,0

ABASE,16,47,2.142190788497732e-09,2.5491569372940634e-09,6.15972903410304e-07,41,0,0

ABASED,17,75,3.4183895561134025e-09,7.585516945650962e-09,1.849051702080402e-06,62,0,0

ABASEMENT,18,15,6.836779112226805e-10,2.6800446616231345e-10,1.0564373622804402e-07,11,0,0

ABASEMENTS,19,0,0.0,0.0,0.0,0,0,0

Note for the word “abandon”, all other forms are included; they are all identified as negative because the negative column (eighth column) has 2009, a non-zero value. MD uses the year when a word is identified to be negative (eighth column) or positive (ninth column).