***Step 2:***

At this point, you must have successfully completed step 1. So, you know how to take care of counting specific words in a given sentence. In the previous step, you found out the sentence “this is a beautiful and sunny day” matches two of the given positive words. That is the first step in sentiment analysis and text mining. Text mining and sentiment analysis are two powerful tools when we want to analyze unstructured datasets. Yay! Lots of buzz words

Don’t you think that the program was not smart at all? After all, we wrote the program for a list of named constants and a given sentence.

WHAT IF YOUR CLIENT (the user of your program) WANTS TO TRY LONGER SENTENCE? OR A PARAGRAPH? WHAT IF THE USER WANTS TO CHECK FOR OTHER POSITIVE OR NEGATIVE WORDS?

In this case, you CANNOT use a hardcoding technique, like the one we used in step1. So, let’s move forward and fix our program and make it smarter, step by step.

In this step, you will learn how to find the valence of a list of words. 😉

Are you surprised? Don’t be.

By the end of this project, you will learn the basics of sentiment analysis. Sentiment analysis is a method to help us find and analyze a person's attitude through his or her texts. In short, the attitude of a person towards a concept can be positive, negative, or neutral. Usually, data scientists are interested in conducting sentiment analysis at group level rather than at individual level. Imagine there is a new article on FB about politics. Based on their political orientations, people may have a positive, negative or neutral attitude towards that piece of news. The overall attitude of comments (in other words, the group's overall attitude who commented) is what matters because it is a representative of a group of people who care for this topic (positively or negatively). Then based on the overall attitude, the politicians can adjust their policies and strategies.

Now, let’s learn how to analyze a text and see if the overall attitude of a text is positive or negative.

***Requirements:***

To achieve the goal in this step, we need to write a program to reads a given text and provide the following outputs:

A) the length of the words in the text,

B) the counts of positive words in the text

C) Positive-attitude-index, which is calculated as follows:

Positive-attitude-index = positive words in a text / the length of the words in the text.

D) the counts of negative words in the text

E) Negative-attitude-index, which is calculated as follows:

Negative-attitude-index = negative words in a text / the length of the words in the text.

F) Valence, which is calculated as follows:

Valence = Positive-attitude-index - Negative-attitude-index

DON’T WORRY! I’m going to help you. Here is the guideline (algorithm) on how to do it:

AS OF NOW, PLEASE FOLLOW THESE STEPS ACCORDINGLY. THE PROJECT WILL INVITE YOUR CREATIVITY IN FUTURE STEPS.

***Guideline:***

To achieve this goal

1. As s always start your program with defining a main function.
   1. As the first step inside your main function, you need to have a list of ALL the positive words. Relax 😊

I have created that list for you and named it positive\_words. You can find it in the script file attached (RequiredLists.txt). This is not a complete list of all the positive words in English. But it is a long list.

1-a- All you need to do is copy and paste positive\_words list from the script file into your own program.

1-b- Do not change the name of this list.

* 1. Second you need to have a list of ALL the negative words.

I have created a list of negative words too. I named it negative\_words. You can find it in the script file attached (RequiredLists.txt).

2-a- Like the previous step, copy and paste negative\_words list from the script file into your own program.

2-b- Do not change the name of this list.

* 1. Now, you need to copy and paste the text that I have also provided you in the script file. It is a paragraph in a format of a list data type. I have called it text. Don’t change the name.
  2. Call other functions appropriately.
  3. Calculate the valence of the text list and assign it to the variable with the name of valence, using the following formula:

valence = positive-attitude-index - negative-attitude-index

For instance, in the example below, valence = 0.021

Then your main function should display the pos\_attitude, neg-attitude, length, pos\_count , neg\_count ,and the valence of the text with enough explanation for the user.

1. Now you need to define a function to calculate the positive attitude of a sentence. Name this function get\_pos.
   1. You should pass two arguments into this function. (You should avoid using global variables. This function should get two arguments.) One argument should be your text list and the other argument should be positive\_words list. (Be careful that text list and positive\_words list have been created outside get\_pos function. You’re just passing them into this function)
   2. Then your get\_pos function should check for every element of positive\_words list and test if it exists inside text. We have lots of similar practices in module 7, specially practices 7-3, 7-4, and 7-5. 😉
      1. Your get\_pos function should keep track of every occurrence of each item of positive\_words list that also exists in text. Save the counts of these occurrences in a variable with the name of pos\_count.
      2. It should find the length of the words in text and save it in a variable with the name of length.
      3. It should calculate the positive-attitude-index (formula was given above) and save it as a variable with the name of pos\_attitude.
      4. And at last, returns these three values.

For instance, in the following list of words:

text = [“This” , “is” , “a” , “good” , “example” , “to” , “show” , “positive” , “attitude” , “in” , “a” , “text” , “Our” , “text” , “may” , “have” , “negative” , “words” , “too” , “Its” , “overall”, “valence” , “is” , “the”, “difference”, “between”, “positive”, “words”, “and”, “negative”, “words” , “The” , “word” , “positively” , “is” , “not” , “in” , “the” , “given” , “list” , “so” , “it” , “should” , “be” , “not” , “be” , “counted”]

Positive\_words = [“good”, “positive”, “shiny”, “well”]

Out of the 4 elements in the Positive\_words, 2 of them exist in the text. However, “good” has 1 occurrence and “positive” has 2, which gives the overall occurrence of 3 as the overall count of positive words. So, pos\_count = 3

The length of this text is 31. So, length = 47

This gives a positive-attitude-index of 3/47= 0.063. So, pos\_attitude = 0.063

So, the get\_pos function should return 3, 47, and 0.063

Do you think you’re done? Not yet.

You must do the same for negative words to find if the text has a negative valence.

1. Then you need to define a function to calculate the negative attitude of a sentence. Name this function get\_neg.
   1. You should pass two arguments into this function. One argument should be your text list and the other argument should be negative\_words list. (Be careful that text and negative\_words list both are created outside this get\_neg function.)
   2. Then your get\_neg function should access (take) every element of negative\_words list to check if it exists inside text.
      1. Your get\_neg function should keep track of every occurrence of each item of negative\_words list that also exists in text. Assign the count of these occurrences to neg\_count.
      2. It should find the length of the words in text and assign it to length
      3. It should calculate the negative-attitude-index (based on the given formula above) and assign it to neg\_attitude.
      4. And at last, return these three values.

For instance, in the following list of words:

text = [“This” , “is” , “a” , “good” , “example” , “to” , “show” , “positive” , “attitude” , “in” , “a” , “text” , “Our” , “text” , “may” , “have” , “negative” , “words” , “too” , “Its” , “overall”, “valence” , “is” , “the”, “difference”, “between”, “positive”, “words”, “and”, “negative”, “words” , “The” , “word” , “positively” , “is” , “not” , “in” , “the” , “given” , “list” , “so” , “it” , “should” , “be” , “not” , “be” , “counted”]

negative\_words = [“bad”, “negative”, “ugly”, “negatively”]

Out of the 4 elements in the negtaive\_words, 1 of them exist in the text with 1 occurrence. neg\_count = 2 and length = 47. Therefore, the neg-attitude = 0.042

So, the get\_neg function should return neg\_count, length, and neg\_attitude.

1. Don’t forget to call your main function to make sure everything works properly before submitting your program.

***Instruction:***

Once you are sure that your program is working, name your file as

*YourFirstName\_YourTeamMate’sFirstName\_step2.py*

then submitted as the second part of your project.

For every team, 1 submission is enough.

***Knowledge needed:***

To do this part of the project, you need to have enough knowledge about lists and strings. (Chapters 7 and 8).