**Report on Corpus:**

As a part of the cousework, we were assigned the task of analyzing a corpus. The first part of the task was to access the corpus and to preprocess it.

At first we import the required packages like nltk, download a couple of packages, import json for reading the json corpus, import the wordnet lemmatizer for lemmatizing words, import bigrams from nltk to identify the bigrams from corpus and finally import the word\_tokenize package from the nltk.tokenize package to quickly and easily tokenize the words that we receive from the corpus.

1. Preprocessing, lemmatization, and computing number of tokens and vocabulary size

I have applied the following preprocessing to the corpus. At first, the entire content is converted into lower case. Further, the regex expression is applied to remove the various types of url's, non-alphanumeric characters except space, remove words with 3 characters or fewer, removing strings fully made up of digits. Then the pre-processed output was lemmatized using the wordnet lemmatizer from the NLTK. Further, all the tokens in a are lemmatized using a for loop and the output is assigned to the variable token. The preprocessed output obtained by filtering the contents of json file after applying the regular expression is stored into the list news\_variety. Then an if condition checks if counter is less than 16000 and if the condition is true, it lemmatizes all the tokens in assigning these tokens to list training. When the number of lines becomes greater than 16000, the lemmatized tokens obtained from a are assigned to the list testing. Finally, counter is incremented at the end of for loop. After that the total number of token and the total vocabulary size is printed.

1. Printing the top 25 bigrams

The next task was to print the top 25 bigrams. In this part I have made use of a dictionary to access and print the 25 bigrams that have the highest frequency. The reason that I have used dictionary instead of list is because it’s much quicker to fetch records from dictionary by specifying the element's hash keys. So, at first empty dictionary is declared here in order to store the bigrams into it and to access them quickly. Then, a for loop is used to loop through bigrams of token variable. Intially, using the else part the first bigram is inserted into the dictionary dict and then the for loop uses the if loop to update the members inside the dictionary dict. Further, we access the dictionary of bigrams ordered by highest frequency and then list the top 25 bigrams based on the number of occurrences in the entire corpus.

1. Computing the number of positive and negative word counts in the corpus

In this section I have printed the total number of positive and negative words. In order to do that,we first begin with opening the file negative-words.txt as a to read the contents by specifying the encoding. Then, we read the lines from the file a, after the line 35 into the variable file (as lines from 1-35 can be ignored). Further, a for loop is used to loop through each line in the file. Then, the list negative\_word here is appended after stripping each of the lines from the default whitespace characters. Same set of steps are repeated for the file positive-words.txt and the positive words are appended into the list positive\_word. Then the function find\_word() is called by passing token as an argument and the output(two values) returned is assigned to the variables pos i.e positive and negative. Finally, the total number of positive and negative words are printed by printing the values of the variables pos and neg.

As mentioned above, two custom functions have been used here find\_word() and search(). The find\_word() function’s purpose is to determine the total number of negative and positive words in the corpus and this function makes use of another function search() to search the positive and negative words in the corpus using binary search.

The details of how the **find\_word() function** works is as follows:

The purpose of this function is to determine the number of positive and negative words in the corpus provided. Then the variable negative which implies number of negative words is initialized to value 0 and variable positive which implies the number of positive words is initialized to value 0. For loop to use to loop through the lemmatized tokens stored in the token list. Then, search() function is called and positive\_word and ind (which holds each line of the token) variables are sent as arguments. The output returned by the search function is stored in the variable i. Moreover, an if loop is used to check if the value assigned to i variable is not -1 then, count of the variable positive is incremented by 1. However, when i has value -1 assigned to it.Search function is called and positive\_word and ind variables are sent as arguments. The output returned is stored in the variable i. Further, if the value returned is not -1 then the negative variable is incremented by 1. Finally, the function returns the value of the positive and the negative variables.

The details of the working of **search() function** is as follows:

The purpose of this function is to search positive and negative words in the corpus using binary search. Binary search is an efficient algorithm for finding an item from an ordered list of items. It works by repeatedly dividing in half the portion of the list that could contain the item, until you've narrowed down the possible locations to just one. Same logic has been implemented in this function to search the positive and negative words in the corpus. Initially the begin variable is initialized to 0. Its value is set to one less than the length of the entire "string". The string argument in our case will be positive\_word and negative\_words. Then, a while loop is applied with condition that value of begin variable is less than finish.We identify the central location(of positive words corpus) for search by considering the integral average of begin and finish. Then the string at the central location is assigned to the variable focus. We compare the values(strings) of focus and desired\_result. If length of focus string is greater than desired result & new value of the finish variable becomes center reduced by 1. However, if the length of focus variable is lesser than desired result then, begin variable becomes equivalent to centre+1. If none of the above condition holds i.e both variables are of same size then focus is returned. Once the condition begin <= finish no more holds true the while loop quits.

1. Compute the number of positive and negative word counts in the corpus

In this section I have printed the total number of positive and negative words. In order to do that, we first begin with opening the file negative-words.txt as a to read the contents by specifying the encoding. Then, we read the lines from the file a, after the line 35 into the variable file(as lines from 1-35 can be ignored). Further, a for loop is used to loop through each line in the file. Then, the list negative\_word here is appended after stripping each of the lines from the default whitespace characters. Same set of steps are repeated for the file positive-words.txt and the positive words are appended into the list positive\_word. Then the function find\_word() is called by passing token as an argument and the output(two values) returned is assigned to the variables pos i.e positive and negative. Finally, the total number of positive and negative words are printed by printing the values of the variables pos and neg.

1. In [ ]:
2. Compute the number of news stories with more positive and more negative words

In this section I have tried to compute the number of news stories with more positive than negative words, as well as the number of news stories with more negative than positive words. In order to do the same at first I have used a for loop to loop through each and every item i.e news in the news\_variety list. Further, I split every item into strings and assign it to the variable token. Then, the find\_word function is called by passing token variable as argument and output returned is assigned to the variables pos and neg. Then an if loop is ran to check, if pos variable is greater than neg. If the conditio is satisfied then the news\_positive variable is incremented by 1, which means news is considered more positive. However, in other situation news\_negative variable is incremented by 1, which means news is considered more negative. Finally, after an exhaustive search of each and every member in news\_variety, the total number of positive and total number of negative news stories is printed by printing the news\_positive and news\_negative.

1. Compute language models for bigrams based on the first 16,000 rows of the corpus.

In order to build this model we had seen that in the preprocessing section itself that we had sen that if counter is less than 16000 and if the condition is true, it lemmatizes all the tokens in a assigning these tokens to list training. When the number of lines becomes greater than 16000,the lemmatized tokens obtained from a are assigned to the list testing. In this section we have a training pair and a testing pair for building the language model that produce a sentence beginning with the word “They,” of 10 words by appending the most likely next word each time.