Workshop 3: TEXT PROCESSING

In this workshop, we will use the tools demonstrated in 'Text-Processing-2019' (HLT 8) on examples of real texts. Firstly, in pairs within our group, we will go through Part 1 which looks at reading and determining the subject matter of an online news article. We may want to also try this out on news articles from today. Part 2 looks at reading raw text files and comparing lexical diversity of two famous novels

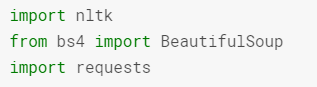
The different parts of this assignment are as follow:

## Part 1: Reading HTML from the web

* Part 2: Reading text files

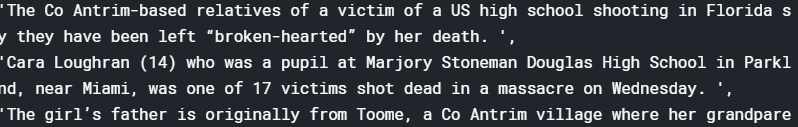
## Part 1: Reading HTML from the web:

The main goal of this part is **understand the subject of an online news article through content words,** A huge amount of text is available online. While it is a fantastic resource for NLP research, we will use this fantastic research to do my this assignment research of text processing, all the content available on the internet use HTML to structure most web pages means accessing the content of interest can be difficult; we often end up with more tags and links than the text of interest. We will use the Python packages 'requests' and 'BeautifulSoup' to first retrieve the data, then parse the HTML. And for the text processing we will use the most popular library of the natural language processing that is nltk (natural language processing tool kit).



Now the next step of the part 1 is getting data from a website for the analysis purpose using python as mentioned above. In our case we have use the URL of an article form Florida school shooting. The specified url of our interest is url = 'https://www.irishtimes.com/news/ireland/irish-news/florida-shooting-victim-had-been-due-to-visit-grandparents-in-ireland-this-summer-1.3396310’ , now we have used Python BeautifulSoup and requests to get the content form this url, but since using this modules of python will return the whole html and javascript content available of the website, but since we are not interested in all the content of the website ( because on an website there are a lots of repeated contents are there ), we only want the story of the website which is in the p tag with class = ‘no\_name’ of the html ( as identified by us after inspection ). So, we have used find\_all( ) method to locate the p class = ‘no\_name’.



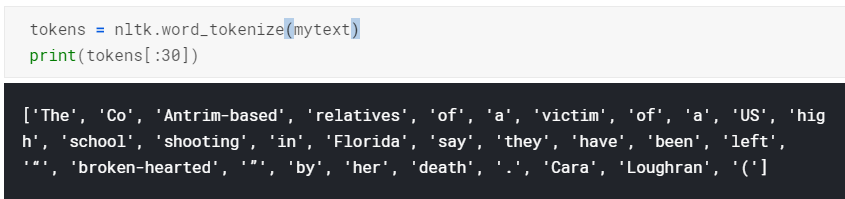
Now we have the story with the tags of the html. But we don’t want the tags, so to remove the tags we have used the get\_text ( ) method, this will return the text in the each tags. The example which we got after applying the get\_text( ) is as: 

This is just the few part of the story, now we will apply the get\_text ( ) method to all the tags and gather the output, the output will be stored in an list, now we have used the .join ( ) method to join all the text and form a whole sentence as it is present in on the website as a article.

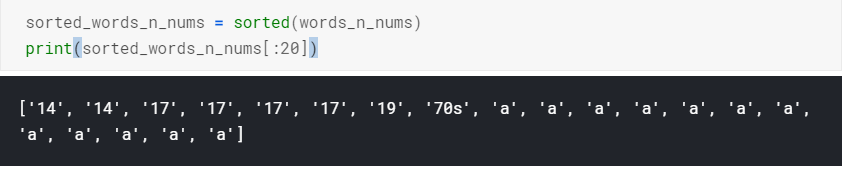
Our next task is to remove the unwanted words from the above result, There is a few words at the end which are not part of the story - ' - Additional reporting: PA'. These can be manually removed by slicing to the first index of the unwanted words. Here the last 50 characters of the remaining text are displayed.

Now we have the story as it is present on the website, i.e. we have the data for our research so next we have performed a lots of technical task using python to get the desired result.

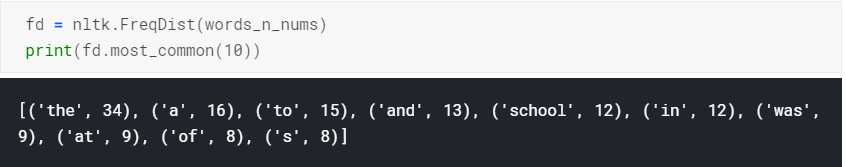
The first step is to do tokenization, tokenization is a process of breaking a string ( sentence ) into individual words , because we have to deal with the each word separately. And for tokenization we have used the popular natural language processing library nltk, we can perform tokenization without using nltk but since there are methods which are pre built in the nltk library so why bother, word\_tokenize( ) is the method which will take string as input and tokenize it.



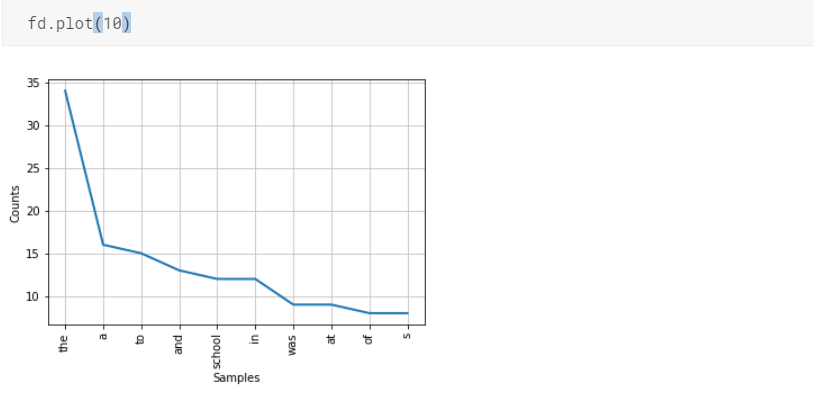
The above snapshot shows the 30 tokenize word from the list of 667 tokenize words. Now see the above snapshot you will find that there are some words which are in uppercase and some are in lowercase. But since python is an case sensitive language. So we to first convert it into one format either uppercase or lowercase. We have choosed lowercase and converted all into lowercase, Sorting the tokens can be useful for inspection, because when ever anything is placed in proper order is good to inspect and see (e.g. here showing first 20 and last 20)

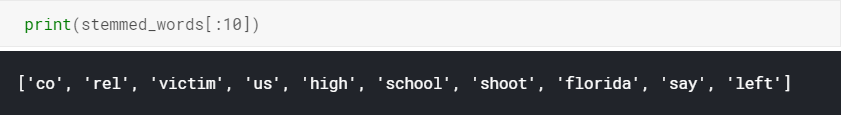


Now it’s time to check the word count of each of the word, word count means counting the ocurrance of each word in the sentence, i.e. is the frequency of each word in the sentence, Frequency distribution provides ordered counts of tokens (words ). For this we have used the FreqDist( ) function of the nltk library it returns a tupple of the word with the count. Below shows the 10 words with the count.

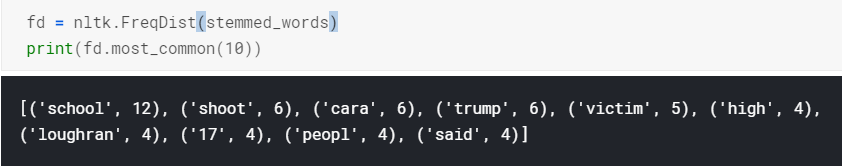


Let’s plot the 10 words with their count. The graph is as shown below:

 From these 10 most common words, would we be able to guess the content of this article? Probably not. The only word which gives us any useful information is \*'school'\*. There are a lot of stopwords (e.g. 'the, 'a', 'of') which don't provide much information on the content of the article. In addition, 'shoot', 'shot', 'shooting', 'shoots', 'shooter' will all have separate counts when it would be more appropriate to combine them into a single token. To remove stopwords, nltk provides an in-built list of stopwords. To combine counts for words with the same stem( stem is the form of the word with which affix can be attached example friendship contains the steam word friend similary there are a lots of word like this since for the research purpose for our case we need to treat the words with the same stem as a sinle word for this we have used a method of nltk to count stem words ), we can use nltk's Porter stemmer, 10 stemmed words are as shown below:

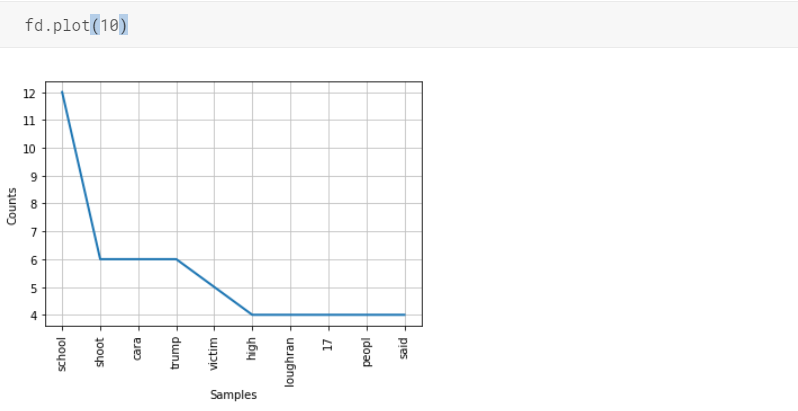


And the frequency( word count in the sentence) distribution of the stemmed word are as show below:



The above snapshot shows that the word school is used 10 times in the sentence. Similarly the count of other words is also shown.

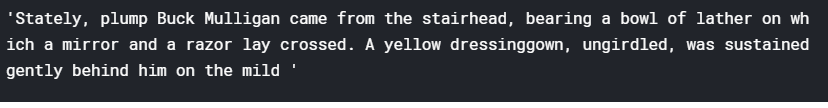
And the plot after calculating the stem of the word is as shown below:



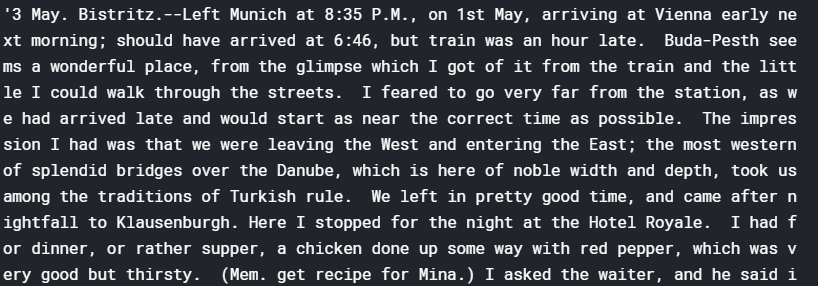
From the above plot, Now it is possible to quickly grasp the subject matter of the article - a serious school shooting. As it is clearly seen in the above plot ocurrence of the the word school ,shooting is highest.

Part 2: Reading text files

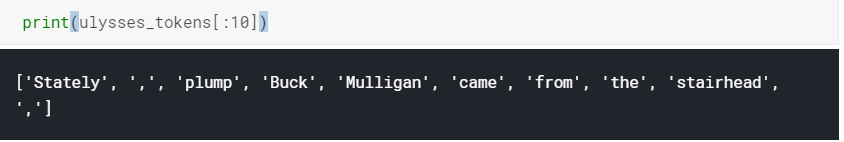
In this part our main task is to Compare the lexical diversity of two of Ireland's most famous novels, Ulysses and Dracula, Introduction and end include text we don't need. Because we are only concern with the main content of the article, we can check the first N characters by slicing the string just like a list, e.g. list\_name[:N], we will Find the location of the first word of interest - 'Stately' ( this is the start of the article main content). And we will next Find the location of the end - easily found by the string 'End of the Project Gutenberg', some part of the article which we got after performing above operations are:



We will Repeat the above task for Dracula article, some part of Dracula article is as:



Now we need to Tokenize the raw content of Dracula and Ulysses ass we done for our article which we scraped from the website. Some part of tokenize result which we got is as shown below:



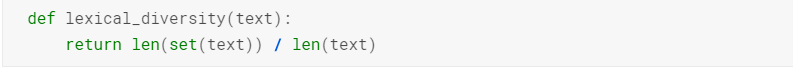
Now its time to perform the main task and perform the Lexical Diversity Calculation.

lexical diversity is calculated by dividing the number of unique words by the total number of words. For example:

"yes no yes no yes no yes no yes no" = 2/10 -> 0.2

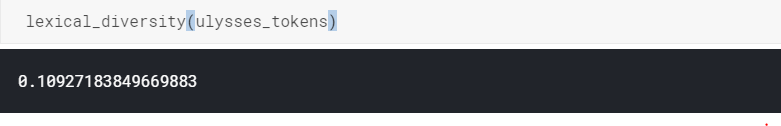
"I went to the cinema and watched a great film" = 10/10 -> 1.0

To do all of the task for the lexical diversity calculation we will design a function as shown below:

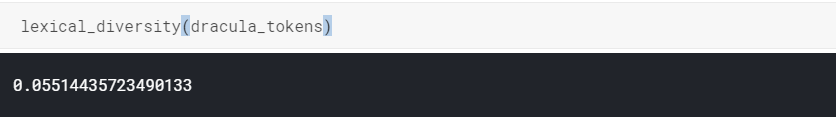


The output which we got is as:

Lexical diversity of the Ulysses article is = 0.109271



Lexical diversity of the dracula article is = 0.05514435



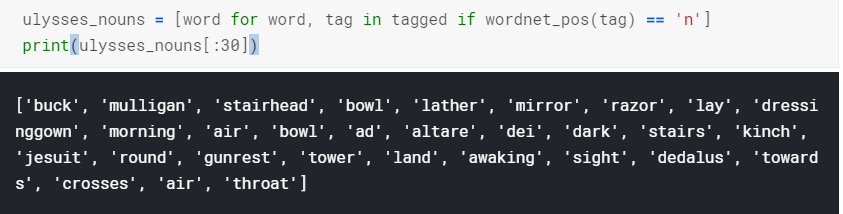
From the above result It appears that Ulysses has nearly twice the lexical diversity of Dracula. This states that the number of unique words in the Dracula article is less as compare to the Ulysses article.

## Part 3: An additional step beyond the workshop activities which may be useful:

Now look at the parts of speech of the words in the texts. Note that you can make the tags to the categories that you might be more familiar with, and group some of the tags.



Now we will use WordNetLemmatizer() of the nltk to lemitize and count for the nouns in ulysses article this is an good feature to know different type of tags in an article and nltk makes easy to do this.



The above snapshot shows some of the nouns in the Ulysses article, the total count of nouns which I got in the Ulysses article is 71871.