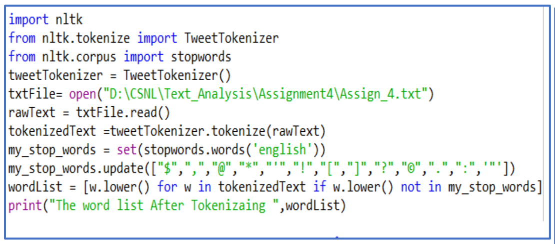
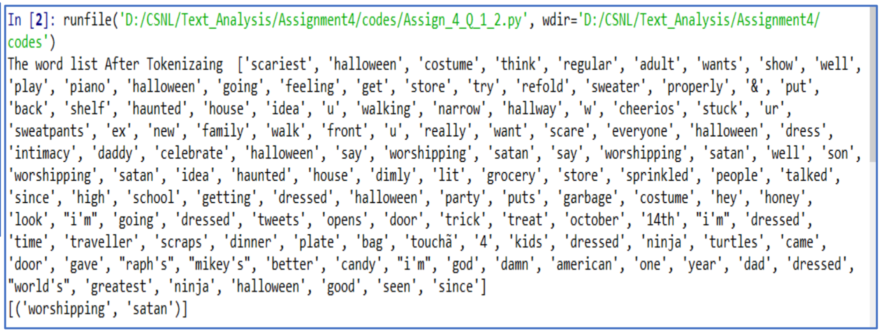
Q1) Short Texts on related Topic : Tweets on Halloween.

a) **Removal of Stop words** :
1)The code snippet, reads a file of tweets and uses Tweet Tokenizer to tokenize words.
2)I have made use of **Tweet Tokenizer** because it dosen't split on words having apostrophes like Mikey's, Raph's in my collected tweets.
3) The tokenized text is further processed for removing stop words .By default nltk has specified set of rules for removing stop words but we can **further update the list to improve the accuracy.**



```
import nltk
from nltk.tokenize import TweetTokenizer
from nltk.corpus import stopwords
tweetTokenizer = TweetTokenizer()
txtFile= open("D:\CSNL\Text_Analysis\Assignment4\Assign_4.txt")
rawText = txtFile.read()
tokenizedText =tweetTokenizer.tokenize(rawText)
my_stop_words = set(stopwords.words('english'))
my_stop_words.update(["$",",","@","*","'","!","[","]","?","©",".",":",'"'])
wordList = [w.lower() for w in tokenizedText if w.lower() not in my_stop_words]
print("The word list After Tokenizaing ",wordList)
```

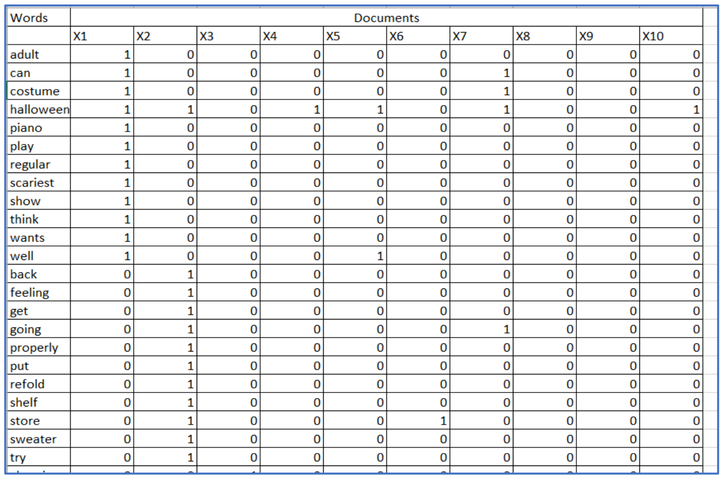


```
In [2]: runfile('D:/CSNL/Text_Analysis/Assignment4/codes/Assign_4_Q_1_2.py', wdir='D:/CSNL/Text_Analysis/Assignment4/
codes')
The word list After Tokenizaing  ['scariest', 'halloween', 'costume', 'think', 'regular', 'adult', 'wants', 'show', 'well',
'play', 'piano', 'halloween', 'going', 'feeling', 'get', 'store', 'try', 'refold', 'sweater', 'properly', '&', 'put',
'back', 'shelf', 'haunted', 'house', 'idea', 'u', 'walking', 'narrow', 'hallway', 'w', 'cheerios', 'stuck', 'ur',
'sweatpants', 'ex', 'new', 'family', 'walk', 'front', 'u', 'really', 'want', 'scare', 'everyone', 'halloween', 'dress',
'intimacy', 'daddy', 'celebrate', 'halloween', 'say', 'worshipping', 'satan', 'say', 'worshipping', 'satan', 'well', 'son',
'worshipping', 'satan', 'idea', 'haunted', 'house', 'dimly', 'lit', 'grocery', 'store', 'sprinkled', 'people', 'talked',
'since', 'high', 'school', 'getting', 'dressed', 'halloween', 'party', 'puts', 'garbage', 'costume', 'hey', 'honey',
'look', "i'm", 'going', 'dressed', 'tweets', 'opens', 'door', 'trick', 'treat', 'october', '14th', "i'm", 'dressed',
'time', 'traveller', 'scraps', 'dinner', 'plate', 'bag', 'touchã', '4', 'kids', 'dressed', 'ninja', 'turtles', 'came',
'door', 'gave', "raph's", "mikey's", 'better', 'candy', "i'm", 'god', 'damn', 'american', 'one', 'year', 'dad', 'dressed',
"world's", 'greatest', 'ninja', 'halloween', 'good', 'seen', 'since']
[('worshipping', 'satan')]
```

**Figure 2 : Output of wordlist after tokenizing and stop word removal.**

**Q1 b) Term frequency Score Matrix after stop word Removal and corresponding Word Cloud.**

**Term Frequencies** : TF frequency tells us the importance of word in a document or how frequently it occurs in the document.
The R program file Q_1.R for the below computed TF frequencies and word cloud is attached in the zip file .
The code first removes the stop words and then computes the Tf-Score Matrix.



| Words | Documents | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | X1 | X2 | X3 | X4 | X5 | X6 | X7 | X8 | X9 | X10 |
| adult | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| can | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| costume | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| halloween | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 |
| piano | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| play | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| regular | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| scariest | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| show | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| think | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| wants | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| well | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| back | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| feeling | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| get | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| going | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| properly | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| put | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| refold | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| shelf | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| store | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| sweater | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| try | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

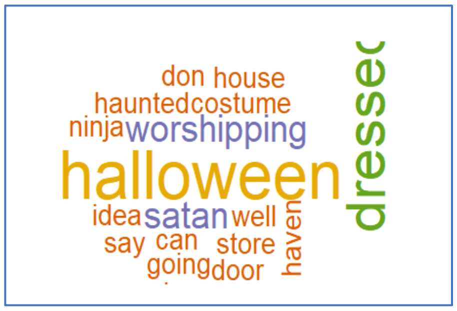


Figure 4 : Word Cloud after Stop words Removal.

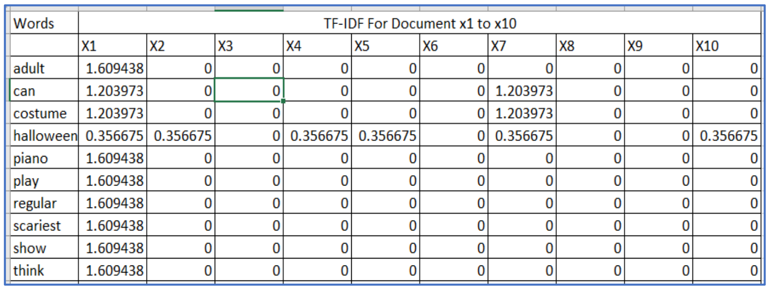
The TF-Matrix is extracted to a excel file using R and has been attached in the zip with name TF-Matrix.xlsx.

Q1c) **TF-IDF Matrix and corresponding word Cloud**

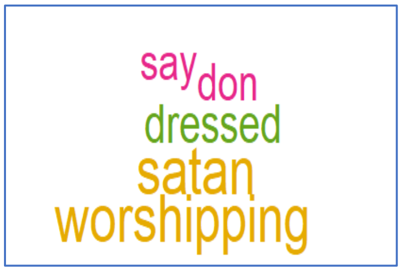
**Tf -IDF Frequency** : 1) The inverse document frequency **decreases the weight of most commonly used** words and vise versa for words in the collection of documents.

2)The combination of Tf and Idf i.e. TF-IDF can be used **to tell the importance of a particular word for a particular document in the collection of all documents.**

3)Likewise, if we see in the TF-IDF frequencies computed below the TF-IDF frequency of the word Halloween has been decreased whereas that of other words has increased seen in figure below. This is evidently seen from the word cloud in Q1.B and Q1.C where in B Halloween is highlighted but this is not the case in word cloud for Q1 C.
The code for the below computation is in Q_1.R file which is included in my zip folder.
The below table contains only the TF-IDF for few words other has been exported to a file TFIDF_Matrix.xlsx

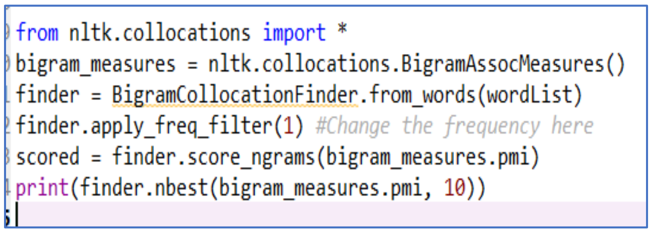


| Words | TF-IDF For Document x1 to x10 | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | X1 | X2 | X3 | X4 | X5 | X6 | X7 | X8 | X9 | X10 |
| adult | 1.609438 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| can | 1.203973 | 0 | 0 | 0 | 0 | 0 | 1.203973 | 0 | 0 | 0 |
| costume | 1.203973 | 0 | 0 | 0 | 0 | 0 | 1.203973 | 0 | 0 | 0 |
| halloween | 0.356675 | 0.356675 | 0 | 0.356675 | 0.356675 | 0 | 0.356675 | 0 | 0 | 0.356675 |
| piano | 1.609438 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| play | 1.609438 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| regular | 1.609438 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| scariest | 1.609438 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| show | 1.609438 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| think | 1.609438 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |



References : https://cran.r-project.org/web/packages/tidytext/vignettes/tf_idf.html

**Q2 ) PMI Scores for all Adjacent pairs in the collection**



```
from nltk.collocations import *
bigram_measures = nltk.collocations.BigramAssocMeasures()
finder = BigramCollocationFinder.from_words(wordList)
finder.apply_freq_filter(1) #Change the frequency here
scored = finder.score_ngrams(bigram_measures.pmi)
print(finder.nbest(bigram_measures.pmi, 10))
```

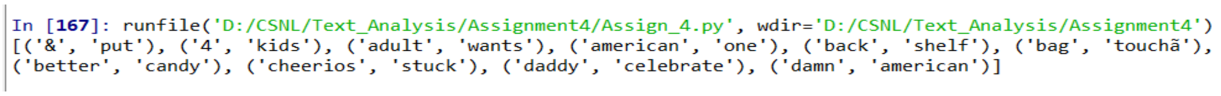
*Code -snippet for calculating PMI Score*

**PMI Score : The PMI Scores helps us to find the association between the words in the corpus.**

A **higher PMI Score signifies** that **the frequency of occurrence of the pair is only slightly low than the frequency of them occurring individually.**
The code snippet helps us find the top 10 Association pairs.

***<u>Frequency Filter = 1</u>***

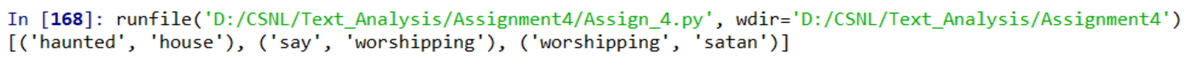
When the frequency filter is 1 ,the following are the output pairs :



```
In [167]: runfile('D:/CSNL/Text_Analysis/Assignment4/Assign_4.py', wdir='D:/CSNL/Text_Analysis/Assignment4')
[('&', 'put'), ('4', 'kids'), ('adult', 'wants'), ('american', 'one'), ('back', 'shelf'), ('bag', 'touchã'),
('better', 'candy'), ('cheerios', 'stuck'), ('daddy', 'celebrate'), ('damn', 'american')]
```

The above output pair of words don't actually make any sense , although the words occur together frequently.

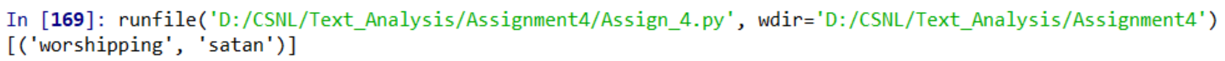
***<u>Frequency Filter = 2</u>***



```
In [168]: runfile('D:/CSNL/Text_Analysis/Assignment4/Assign_4.py', wdir='D:/CSNL/Text_Analysis/Assignment4')
[('haunted', 'house'), ('say', 'worshipping'), ('worshipping', 'satan')]
```

The above pair of words for frequency =2 actually make some sense as haunted house phrase is seen many times so is worshipping Saturn.

***<u>Frequency Filter = 3</u>***



```
In [169]: runfile('D:/CSNL/Text_Analysis/Assignment4/Assign_4.py', wdir='D:/CSNL/Text_Analysis/Assignment4')
[('worshipping', 'satan')]
```

The above pair (worshipping, satan) is only left because it relates to many people worshipping Saturn for thousands of years .
2)**The output at frequency filter 2 should be** considered because we get more set of associated pairs that make some sense rather than 1 where only reoccurrence is taken into consideration.
3) We can consider filter frequency 3 only if we want to extract exact accuracy and less association.
Reference : http://www.nltk.org/howto/collocations.html

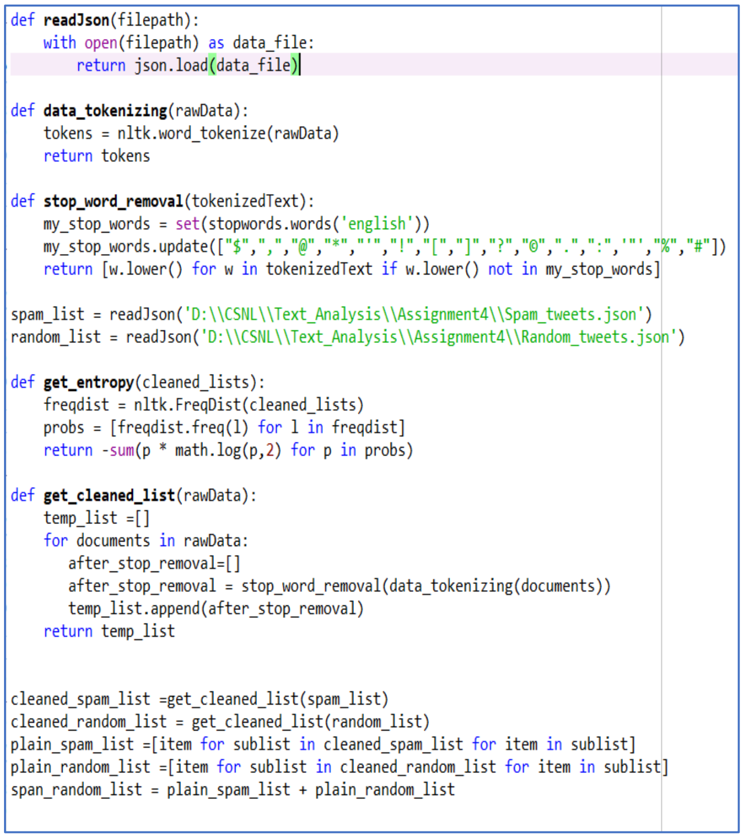
https://en.wikipedia.org/wiki/Pointwise_mutual_information

**Q3) Entropy Claculation for spam and random Tweets :**

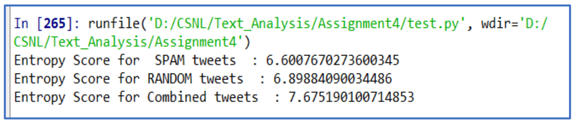
**Entropy : Entropy helps us to find the certainity or uncertainty in the data.**
**Entropy sore for Spam Tweets :6.60067(approx.):**
**The spam text have the lowest Entropy because they are related to a similar event(Black Friday).**

**2) Entropy for Random Tweets: 6.8998(approx.)**
**Entropy for random text is slightly higher than spam text because higher the entropy higher the variance in the data .Since all the data is selected at random from twitter.**

**3) Entropy for Combined tweets : 7.6571**
**Entropy of combined sample is seen to be much higher. This may be due to the skewness in Spam tweets and Random tweets .**
**Also as a whole the entire corpus is totally unassociated which justifies the higher entropy.**



```python
def readJson(filepath):
    with open(filepath) as data_file:
        return json.load(data_file)

def data_tokenizing(rawData):
    tokens = nltk.word_tokenize(rawData)
    return tokens

def stop_word_removal(tokenizedText):
    my_stop_words = set(stopwords.words('english'))
    my_stop_words.update(["$",",","@","*","'","!","[","]","?","©",".",":",'"',"%","#"])
    return [w.lower() for w in tokenizedText if w.lower() not in my_stop_words]

spam_list = readJson('D:\\CSNL\\Text_Analysis\\Assignment4\\Spam_tweets.json')
random_list = readJson('D:\\CSNL\\Text_Analysis\\Assignment4\\Random_tweets.json')

def get_entropy(cleaned_lists):
    freqdist = nltk.FreqDist(cleaned_lists)
    probs = [freqdist.freq(l) for l in freqdist]
    return -sum(p * math.log(p,2) for p in probs)

def get_cleaned_list(rawData):
    temp_list =[]
    for documents in rawData:
       after_stop_removal=[]
       after_stop_removal = stop_word_removal(data_tokenizing(documents))
       temp_list.append(after_stop_removal)
    return temp_list


cleaned_spam_list =get_cleaned_list(spam_list)
cleaned_random_list = get_cleaned_list(random_list)
plain_spam_list =[item for sublist in cleaned_spam_list for item in sublist]
plain_random_list =[item for sublist in cleaned_random_list for item in sublist]
span_random_list = plain_spam_list + plain_random_list
```



```
In [265]: runfile('D:/CSNL/Text_Analysis/Assignment4/test.py', wdir='D:/
CSNL/Text_Analysis/Assignment4')
Entropy Score for  SPAM tweets  : 6.6007670273600345
Entropy Score for RANDOM tweets  : 6.89884090034486
Entropy Score for Combined tweets  : 7.675190100714853
```

Note : All the code files for above in R-programming and python have been attached in the zip-file.
2) The spam Tweets and Random Tweets file have also been attached in the Tweets folder.

Refereces : nltk-frequency.