INDIVIDUAL PROJECT 2 – TEXT MINING

By Tanvi Kini

# Text Analysis of Last Words

The aim here is to perform some text mining analysis on the last words of people who were on death row in Texas to identify common themes in the things they say. This will help us understand what really matters to a person minutes before their imminent death and gives us a different perspective the way we look at our own lives.

## About the dataset

This dataset was sourced from Kaggle.com and can be found at the link below:

<https://www.kaggle.com/mykhe1097/last-words-of-death-row-inmates>

It consists of 546 rows and 20 columns. It was originally collected and distributed by the Texas Department of Criminal justice. Each row in the dataset represents a person. It has details like Name, Age, Gender, Race etc. and the column of interest here is one called Last Statement which is the last sentence they spoke.

# Finding the Most Commonly Spoken Words

We will be trying to parse all the sentences and finding the most frequently occurring words in the Last Statement column in our dataset.

The first step here is to import the .csv file into Python.

*#Importing the required packages*

import re

import pandas as pd

*#Reading the .csv file into a dataframe from the bath where it is stored using the pd.read() function*

df = pd.read\_csv("C:/Users/Tanvi/Desktop/DMBI/last-words-of-death-row-inmates/Texas Last Statement - CSV.csv", encoding="latin1")

first\_lastStatement = df.LastStatement[0]

## Data Preparation

The data first needs to be prepared before we can perform any kind of text analysis. Prepared here refers to how easily it can be identified by our function to capture the essence of what is being said.

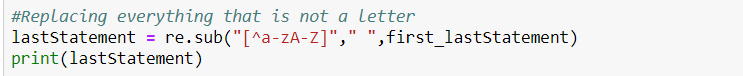
## Steps in Data Preparation

Let us see the effect of the different steps we will be performing to pre-process our last statements but applying it to the very first statement and printing It after each operation.

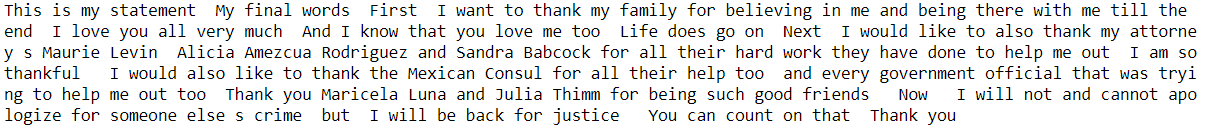
1. **Replace everything in the sentence that is not a letter.**

We do this by using the re.sub() function to substitute everything that is not a letter by nothing.

Code:



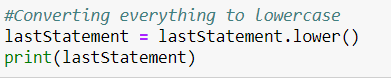
Output:



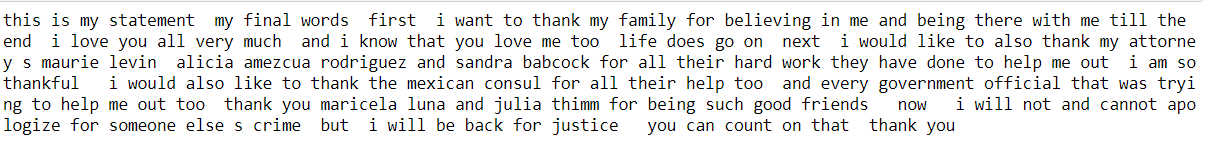
1. **Convert everything in the statement to lower case**

This is done to avoid detecting the same word in upper case and lower case as two separate words. We can achieve this using the lower() function.

Code:



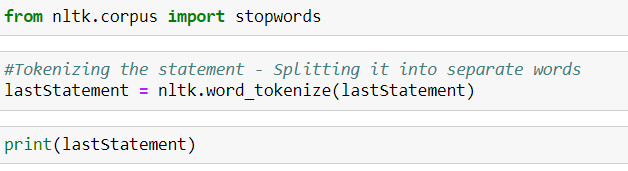
Output:



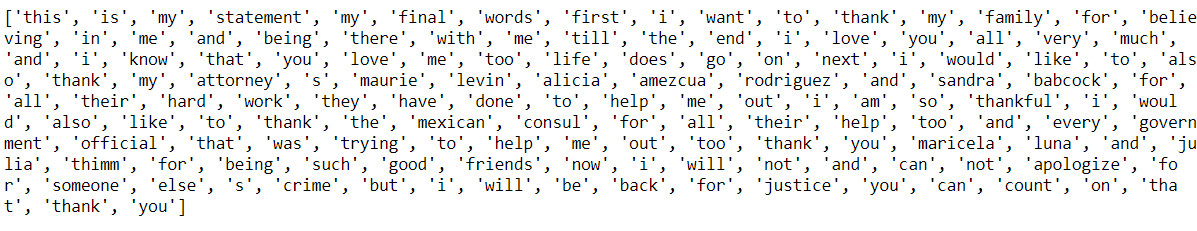
1. **Tokenizing the statement**

Tokenizing a statement means effectively splitting the sentence into separate words or tokens for easy parsing. We use the word.tokenize() function under the nltk package.

Code:



Output:



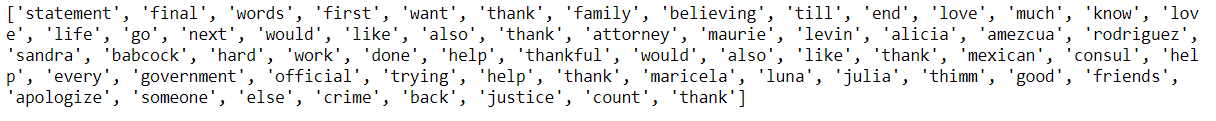
1. **Removing bridge words like conjunctions and prepositions**

Here, we are trying to remove all the words in the sentence that do not add any meaning to the sentence being spoken. They are only there to give the sentence grammatical structure. We do not require our sentences to have grammatical structure and these might show up as the most commonly used words because they will inevitably be used in every sentence.

Code:



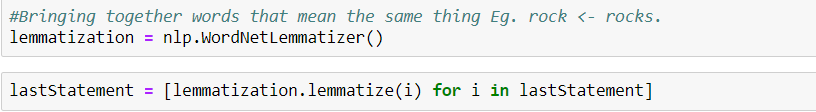
Output:



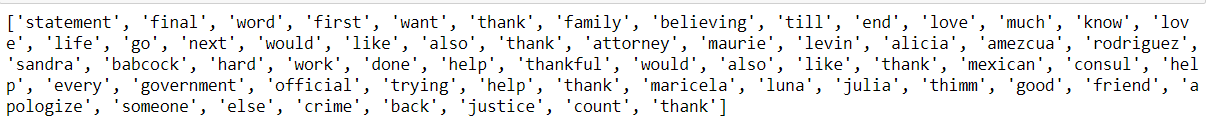
1. **Bring together words that mean the same thing**

The past and present tense of the same word or the singular and plural version of the same word must be identified as having the same meaning rather than being two different words. We can do this using the nlp.WordNetLemmatizer() function.

Code:



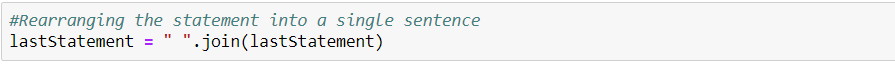
Output:



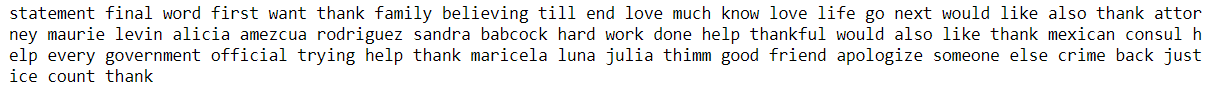
1. **Rearranging the statement back into a single sentence**

We need to now rearrange the sense back into a sentence format from the tokenized format.

Code:



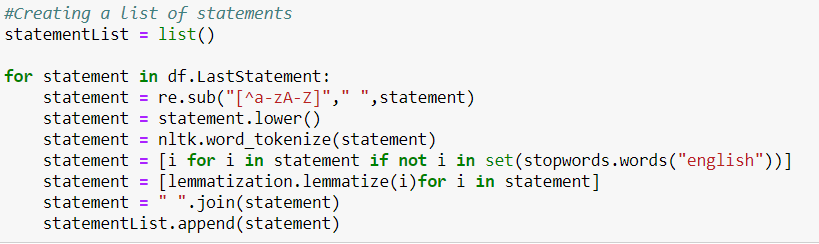
Output:



## Performing the same pre-processing for all the rows

The steps mentioned above will now be applied to the all the rows in the dataset. We do this by first creating a list with all the statements called statementList and apply the functions to all its items as shown below:

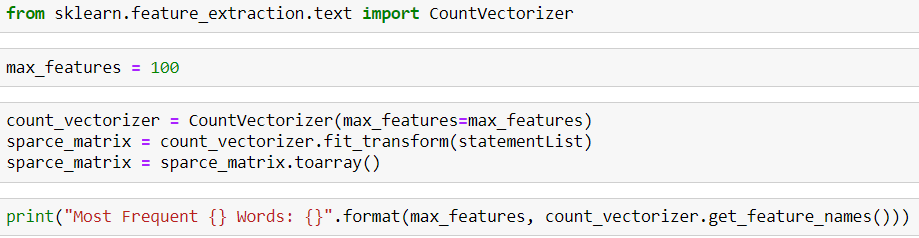
Code:



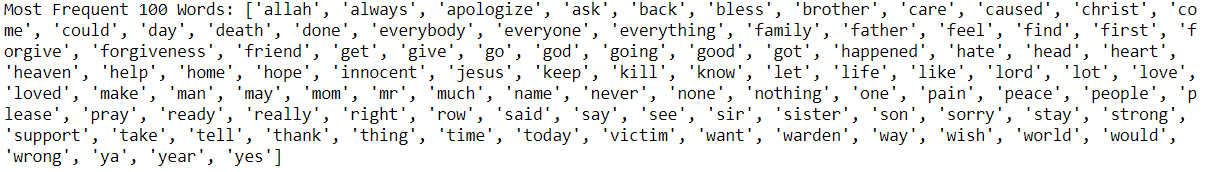
## Finding most frequently used words

First let us find the 100 most frequently used words

Code:

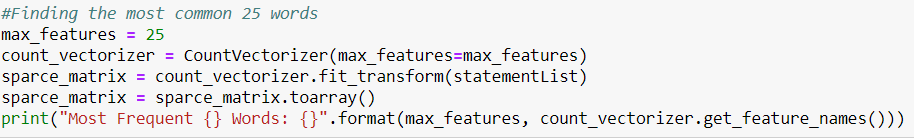


Output:



The results show many too many words for us to derive much insight. Let us check the same with setting the max\_features to 25.

Code:



Output:



We can see from the result above that some of the most commonly used words are referring to Love, Gratitude, Apology, God and Family.

# Creating a Word Cloud using R.

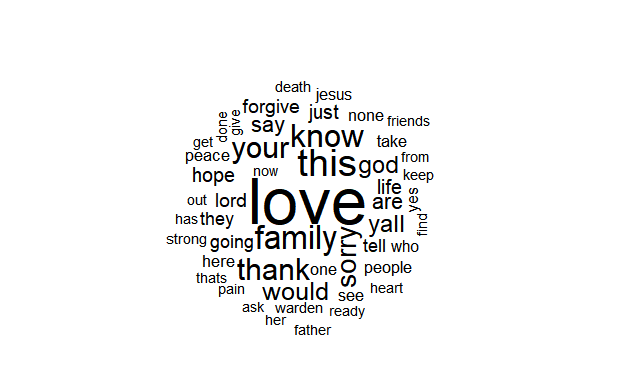
To create a word cloud, I used R instead of python because I found the libraries and functions to be more readily available. I performed a text preprocessing similar to the one detailed above but used R instead of python. The result at each stage of the preprocessing was stored in a corpora which is a collection of documents containing natural language in R.

The only difference between the preprocessing done in Python and that done in R is that to exclude bridge words, I created a bag of words to eliminate myself and iteratively kept refining the word cloud by adding words to the bag. I did this using the tm\_map() function under the library tm. I passed the function ‘removewords’, the corpora I created and a list of words I wanted to eliminate from my statements.

The first iteration of the word cloud had a lot of words like what, but, that etc. which I then eliminated to obtain the final word cloud as shown below.

I decided to stop at this iteration because it looked like we found the same common themes we discovered while finding the most frequently used words in Python.

## Wordcloud Obtained



# Conclusion

From the word cloud pictured above, we can tell that the most commonly used word is Love followed by family, thank, sorry and god. We do not see any words that would suggest anger or resentment which is surprising. This might suggest that people acquire a sense of self-realization at this point and decide that anger is not worth their time.