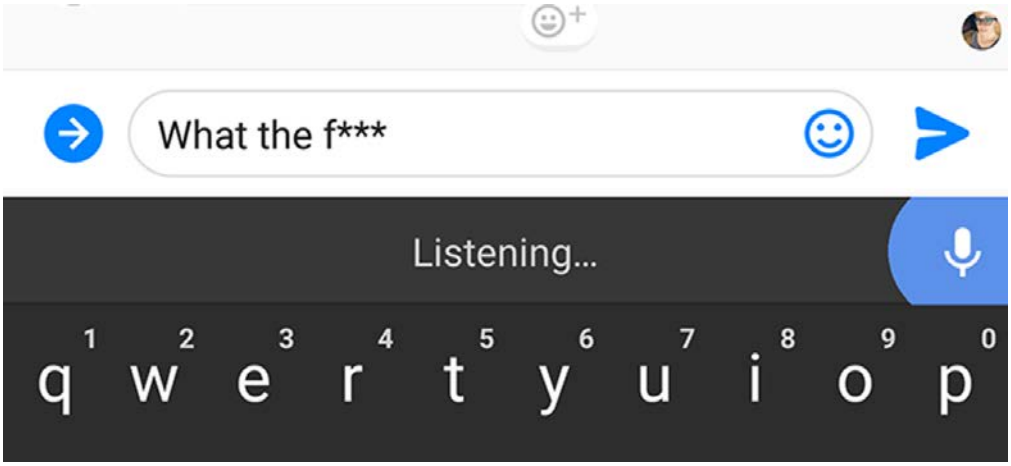
Laboratory Experiment for Practical Exam
AI & Application Lab (14B17CI671)

Names / Roll NO:
1. Yash Raj Pandey (191B295) **Dept: CSE**
2. Samarth Dubey (191B304)
3. Aryman Tripathi (191B309)
Hrithik Bansal (191B298)

Expt. Title: **"Profanity Chat Filter"** **Date: 22/03/21**

Description:



A Python script to detect language of text and filter out the OFFENSIVE Words.
Profanity Checker is a Python script that uses the Natural Language Toolkit (NLTK) library to solve the usual problem of toxicity in society.
The problem was divided into below three phases:
1. Detect the language of the text
2. Fetch offensive words for the language detected
3. Output the bad words in each line of the text

Language detection can be achieved by using the 'stopwords' function as provided by the Python's NLTK library. Datasets containing the most common bad words in each of the 4 languages - English, Spanish, French and German are attached as the CSV files. User can be asked to test out in any of the 4 languages mentioned above.

Approach / Algorithms

- Importing the sys module of python
- Importing the time module
- Importing the NLTK Library for symbolic and statistical natural language processing for English written in the code

- Calculating the language probability based on the ratio probability of the stopwords

- We load the bad words for the specific language after the language detection

- Iterating over the words in each line we check for whether it is offensive or not

- Finally, printing over the line number containing the offensive word and the number of offensive words in the specific line

Test Procedures

- Importing all the libraries required for the code

```
import sys
import time
```

- Using stopwords from NLTK Library

```
try:
    from nltk import wordpunct_tokenize
    from nltk.corpus import stopwords
    print("nltk is installed. \n")
except ImportError:
    print("You need to install nltk (http://nltk.org/index.html)")
```

- Starting off with the calculate_language_ratios function, we compute per language included in nltk number of unique stopwords appearing in analyzed text

```
def calculate_languages_ratios(text):
    languages_ratios = {}
    tokens = wordpunct_tokenize(text)
    words = [word.lower() for word in tokens]

    # Compute per language included in nltk number of unique stopwords appearing in analyzed text
    for language in stopwords.fileids():
        stopwords_set = set(stopwords.words(language))
        words_set = set(words)
        common_elements = words_set.intersection(stopwords_set)

        languages_ratios[language] = len(common_elements)  # language "score"

    return languages_ratios
```

- Moving on, we use the load_bad_words function to utilise the included offensive dataset for the specific language from the .csv files

```
def load_bad_words(language):
    if language.upper() in ['ENGLISH','FRENCH','SPANISH','GERMAN']:
        badwords_list=[]
        lang_file = open('datasets/'+language.lower()+'.csv','r')
        for word in lang_file:
            badwords_list.append(word.lower().strip('\n'))
    return badwords_list
```

- Coming on to the detect_language function, we calculate the ratios of the language detected from the calculate_language_ratios function and then maxing out on the ratio for a specific language

```
def detect_language(text):
    ratios = calculate_languages_ratios(text)
    most_rated_language = max(ratios, key=ratios.get)
    return most_rated_language
```

- Now, we take input from the user on the whereabouts of the file. We input the whole text file as a paragraph and then read it line by line

```
print ("Language can be in the form of: english, french, german and spanish.")
print ('\n')
testlang = input ("Language to test: ")
print('\n')
filename = 'test-files/'+testlang.lower()+'.txt'
file = open(filename,'r')
text = ''
line_count=1
for i in file:
        text+=str(line_count)+'| '+i
        line_count+=1
time.sleep(2)
```

- Once we get the input, we read the file and show the user his written paragraph with line number and then move on to show the language detected. Continuing we move on to check for offensive words in the file.

```
print ('----------------------Input Text----------------------')
print ('\n')
print (text)
print ('\n')
print ('-----------------------Text Read-----------------------')
print ('\n')
language = detect_language(text)
print ('\n')
time.sleep(1)
print ('----------------------------')
print ('Language Detected: ',language.upper())
print ('----------------------------')
print ('\n')
time.sleep(1)
print ('----------------------------------------')
print ('Checking for offensive words in '+language.upper()+'.')
print ('----------------------------------------')
print ('\n')
badwords = load_bad_words(language)
badwords = set(badwords)
text_list = text.split('\n')
```

- Iterating over the words in each line, we remove the unnecessary punctuations and convert all the words into lowercase so as make it easy to read.

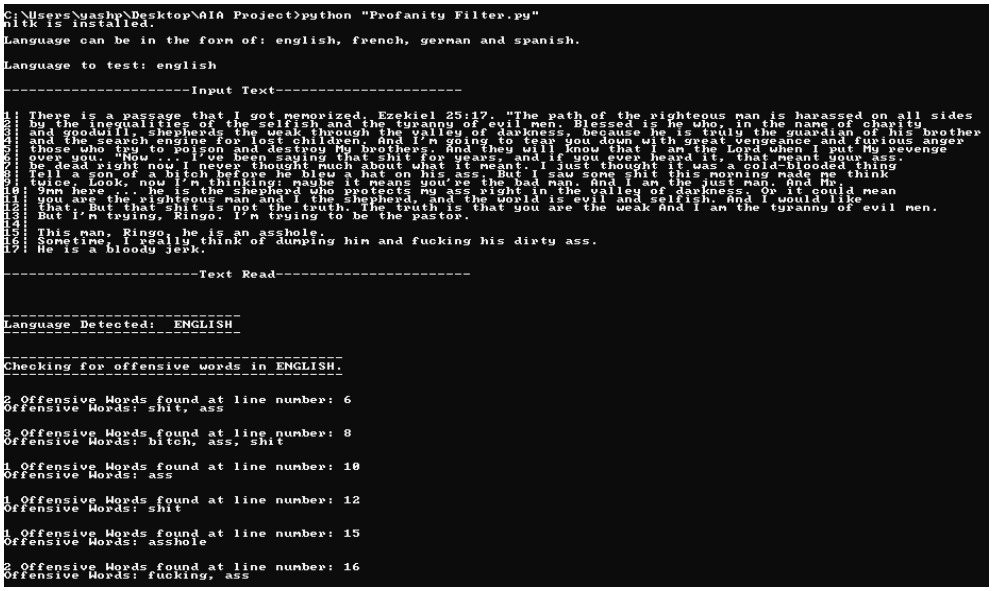
```
for sentence in text_list:
        line_number = str(text_list.index(sentence)+1)
        for key in ['.', ',', '"', "'", '?', '!', ':', ';', '(', ')', '[', ']', '{', '}']:
                sentence = sentence.replace(key,'')
        abuses=[i for i in sentence.lower().split() if i in badwords]
```

- Lastly, checking for abusive words in the specific language we go through each and every line to identify and print the offensive word

```
        if abuses == []:
                continue
        else:
                time.sleep(0.5)
                print (str(len(abuses))+' Offensive Words found at line number: '+line_number)
                x_words=''
                for i in abuses:
                        x_words+=i+', '
                print ('Offensive Words: '+x_words[:-2])
                print ('\n')
```

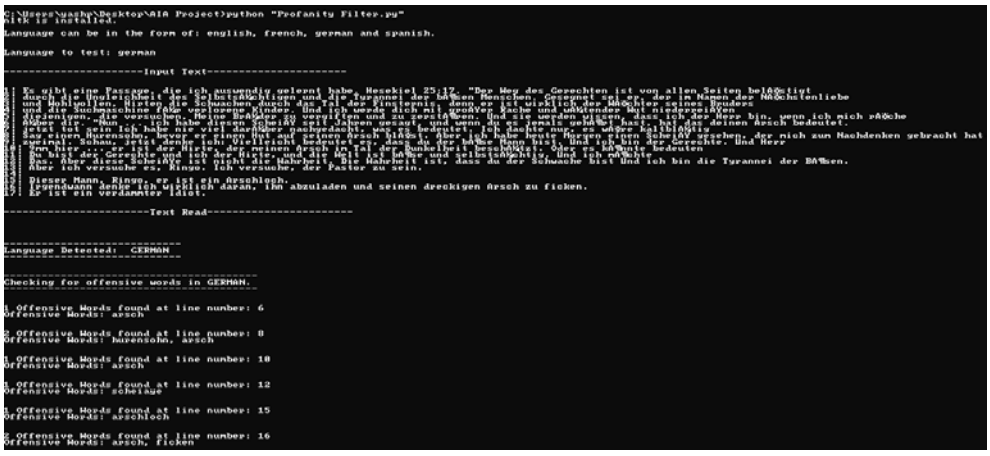
Results:

- Running the code for “English”, we run the script in Terminal and get the offensive words as the output on our screen



- Running the code for “French”

- Running the code for “German”



- Running the code for “Spanish”

Conclusion / Remarks

Completing this Project, Profanity Checker was a lot overwhelming for us (Yash, Samarth and Aryman).

- Learned over NLTK Library and usage in language detection
- Learned the usage of stopwords and the removal of articles despite the meaning remaining the same
- Added bad-words CSV datasets for 4 languages so as the data set can be used while reading the input
- Facility for user to give in the info for which language he'd like to be tested by our code

Evaluation / Marks