# Problem-3

# ECE – 20875

Using Step 1 and Step 2, we obtain the plots and the results . The snap of the written program has been attached below.

## Step 1 program:

```
#Returns: a list of strings, each string is one line in the file
#hints: https://docs.python.org/3/tutorial/inputoutput.html#reading-and-writing-files
#       https://docs.python.org/3/library/stdtypes.html#str.splitlines
def getText(filename) :
    #fill in
    textlist = []
    currentline = 0
    infptr = open(filename, 'r')
    currentline = infptr.read()
    textlist = currentline.splitlines()
    infptr.close()
    return textlist
#Arguments:
#  line: a string of text
#Returns: a list of n-grams
#Notes: make sure to pad the beginning and end of the string with '_'
#       make sure to convert the string to lower-case
#       so "Hello" should be turned into "__hello__" before processing
def getNgrams(line):
    #fill in
    ngramlist = []
    newlowercasestring = line.lower()
    newlowercasestring = newlowercasestring.strip('\n')
    newlowercasestring = '__' + newlowercasestring + '__'
    for i in range (len(newlowercasestring)-2):
        ngramlist.append(newlowercasestring[i:i+3])
    return ngramlist

#Arguments:
#  filename: the filename to create an n-gram dictionary for
#Returns: a dictionary, with ngrams as keys, and frequency of that ngram as the value.
#Notes: Remember that getText gives you a list of lines, and you want the ngrams from
#       all the lines put together.
#       use 'map', use getText, and use getNgrams
#Hint: dict.fromkeys(l, 0) will initialize a dictionary with the keys in list l and an
#      initial value of 0
def getDict(filename):

    ##data structures
    listwithline = []
    gramslist = []
    newgramslist = []
    dictionary = {}
    ##counters
    i = 0
    j = 0
    k = 0
    listwithline = getText(filename)
    for i in range (len(listwithline)):
        gramslist += getNgrams(listwithline[i])
    dictionary = dict.fromkeys(gramslist,0)
    for gram in (gramslist):
        dictionary[gram] += 1
    #print(dictionary)
    return dictionary
```

## Step 2 program:

problem2.py

```
from helper import plotHisto
import operator

hw8_1 =__import__("hw8-1")
helper =__import__("helper")

englishfreqlist = []
frenchfreqlist = []
germanfreqlist = []
italianfreqlist = []
portuguesefreqlist =[]
spanishfreqlist = []
mysteryfreqlist = []

# 6 dictionaries
dictionary_final1 = hw8_1.getDict('ngrams/english.txt')
dictionary_final2 = hw8_1.getDict('ngrams/french.txt')
dictionary_final3 = hw8_1.getDict('ngrams/german.txt')
dictionary_final4 = hw8_1.getDict('ngrams/italian.txt')
dictionary_final5 = hw8_1.getDict('ngrams/portuguese.txt')
dictionary_final6 = hw8_1.getDict('ngrams/spanish.txt')
dictionary_final7 = hw8_1.getDict('ngrams/mystery.txt')

languagesorted = {}
languagesorted.update(dictionary_final1)
languagesorted.update(dictionary_final2)
languagesorted.update(dictionary_final3)
languagesorted.update(dictionary_final4)
languagesorted.update(dictionary_final5)
languagesorted.update(dictionary_final6)


#Converting all to dictionaries
languagesorted = sorted(languagesorted.items(), key = lambda x: x[0])
languagesorted = dict(languagesorted)

for word in languagesorted:
    if word in dictionary_final1:
        englishfreqlist.append(dictionary_final1[word])
    else:
        englishfreqlist.append(0)

for word in languagesorted:
    if word in dictionary_final2:
        frenchfreqlist.append(dictionary_final2[word])
    else:
        frenchfreqlist.append(0)

for word in languagesorted:
    if word in dictionary_final3:
        germanfreqlist.append(dictionary_final3[word])
    else:
        germanfreqlist.append(0)

for word in languagesorted:
    if word in dictionary_final4:
        italianfreqlist.append(dictionary_final4[word])
    else:
        italianfreqlist.append(0)

for word in languagesorted:
    if word in dictionary_final5:
        portuguesefreqlist.append(dictionary_final5[word])
    else:
        portuguesefreqlist.append(0)

for word in languagesorted:
    if word in dictionary_final6:
        spanishfreqlist.append(dictionary_final6[word])
    else:
        spanishfreqlist.append(0)

for word in languagesorted:
    if word in dictionary_final7:
        mysteryfreqlist.append(dictionary_final7[word])
    else:
        mysteryfreqlist.append(0)

plotHisto(englishfreqlist,'english.png',)
plotHisto(frenchfreqlist,'french.png')
plotHisto(germanfreqlist,'german.png')
plotHisto(italianfreqlist,'italian.png')
plotHisto(portuguesefreqlist,'portuguese.png')
plotHisto(spanishfreqlist,'spanish.png')
plotHisto(mysteryfreqlist,'mystery.png')
```

Using the idea of step 1 and step 2, came up with the program above and printed out the following :

1) The most common 10 n-grams for given file mystery.txt
2) Plot of the n-gram histogram for mystery.txt

The results are attached below:

**Most-common 10 n-grams**

```
agarw184@ecegrid-thin1 ~/ECE20875/PA08
$ python3 problem1.py ngrams/mystery.txt
[(' de', 123), ('de ', 102), ('el ', 66), ('os ', 56), (' co', 56), ('as ', 53),
 (' la', 49), ('do ', 49), (' el', 45), ('ia ', 44)]
```

The snap above shows the 10 most common n-grams 'mystery.txt'. The idea was same as problem1 to solve this problem and obtain the desired result. The data obtained here, would serve to be one of the things that would be used to obtain the language of 'mystery.txt.

This obtained result is compared to result obtained in problem-1 for which I am attaching the results for all 6 languages in an ordered labelled sequence which are as follows:

1) **Language : English**

```
agarw184@ecegrid-thin1 ~/ECE20875/PA08
$ python3 problem1.py ngrams/english.txt
[('the', 149), (' th', 142), (' an', 129), ('he ', 121), ('nd ', 113), ('and', 111), ('ion', 102), (' of', 93), ('of ', 89), ('tio', 88)]
```

2) **Language : French**

```
$ python3 problem1.py ngrams/french.txt
[(' de', 204), ('es ', 183), ('de ', 136), (' et', 118), ('ion', 108), ('te ', 106), ('nt ', 100), ('e d', 98), ('et ', 93), (' a ', 92)]
```

3) **Language : German**

```
$ python3 problem1.py ngrams/german.txt
[('en ', 251), ('er ', 187), ('der', 152), (' un', 124), ('und', 122), ('nd ', 117), ('ein', 110), ('ung', 102), ('cht', 98), (' de', 94)]

agarw184@ecegrid-thin1 ~/ECE20875/PA08
```

4) **Language : Italian**

```
$ python3 problem1.py ngrams/italian.txt
[(' di', 177), ('to ', 124), (' de', 99), ('la ', 98), (' in', 97), ('ion', 95), (' e ', 93), ('e d', 89), ('di ', 89), ('a d', 81)]

agarw184@ecegrid-thin1 ~/ECE20875/PA08
```

5) **Language : Portuguese**

```
$ python3 problem1.py ngrams/portuguese.txt
[('os ', 138), ('de ', 134), (' de', 129), (' a ', 111), (' e ', 98), ('em ', 95), ('o d', 89), ('to ', 86), ('ao ', 78), (' di', 77)]

agarw184@ecegrid-thin1 ~/ECE20875/PA08
$
```

6) **Language : Spanish**

```
$ python3 problem1.py ngrams/spanish.txt
[(' de', 249), ('os ', 137), ('de ', 135), ('ion', 115), (' la', 104), ('cio', 101), ('la ', 97), (' y ', 92), (' a ', 84), ('rec', 75)]
```

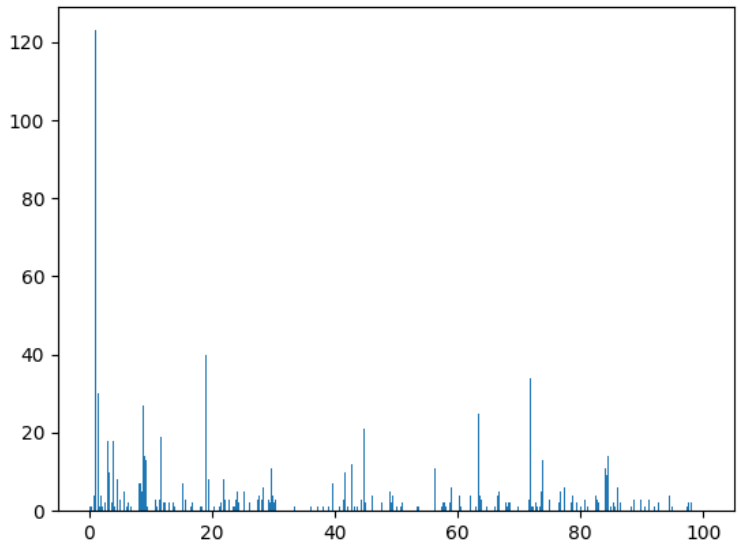
The snaps above correspond to the most n-grams for the various 6 languages. Using the results above, we compare these results with the result obtained for 'mystery.txt'

**Observation:**

On comparing the following things were observed :

1) Various common n-grams can be seen with different frequencies in each of the output. However, some of the n- grams are common in certain outputs with relative closeness in their occurrences (frequencies) as well.
2) On close observation, it was observed that mystery.txt shared relative closeness with Spanish.txt due to common n-grams and closeness in their frequencies.
3) The above is one of the major criteria to infer that the language of mystery.txt could probably be Spanish.

**Plot of the n-gram histogram for mystery.txt**

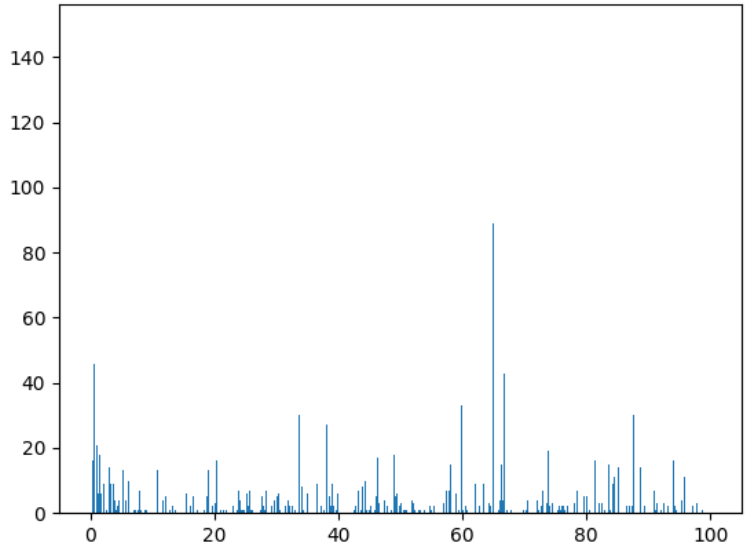



The snap above shows the n-gram histogram for 'mystery.txt'. The idea was same as problem2 to solve this problem and obtain the desired result. The data obtained here, would serve to be one of the things that would be used to obtain the language of 'mystery.txt.

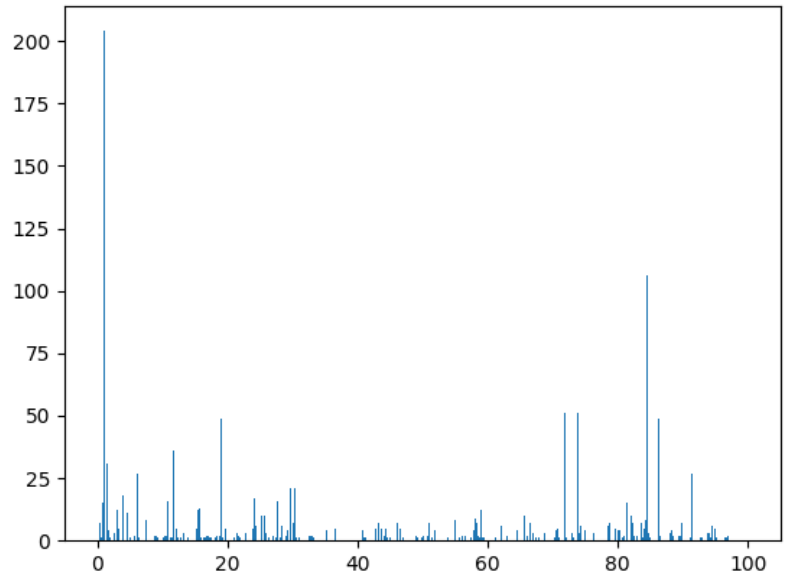
The obtained histogram is compared to the histogram plots for all 6 languages for which I am attaching the plots for all the 6 languages.

The plots are as follows:

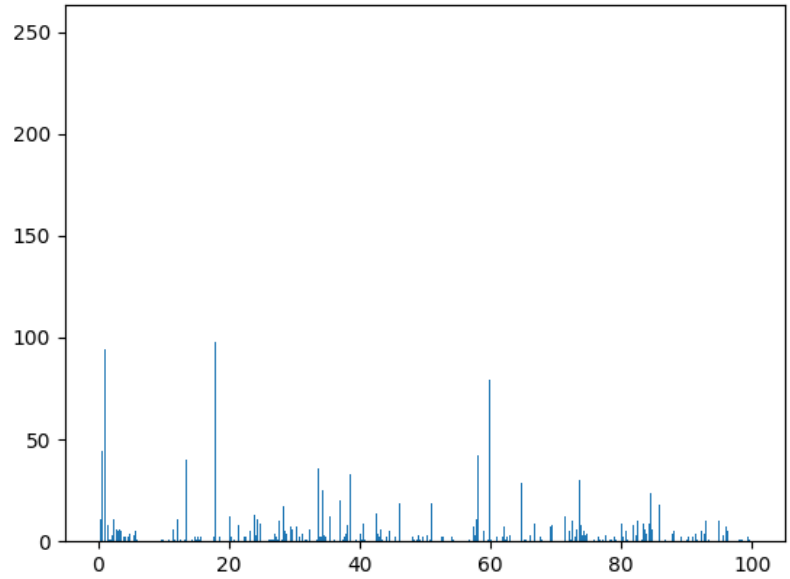
1) **Language : English**



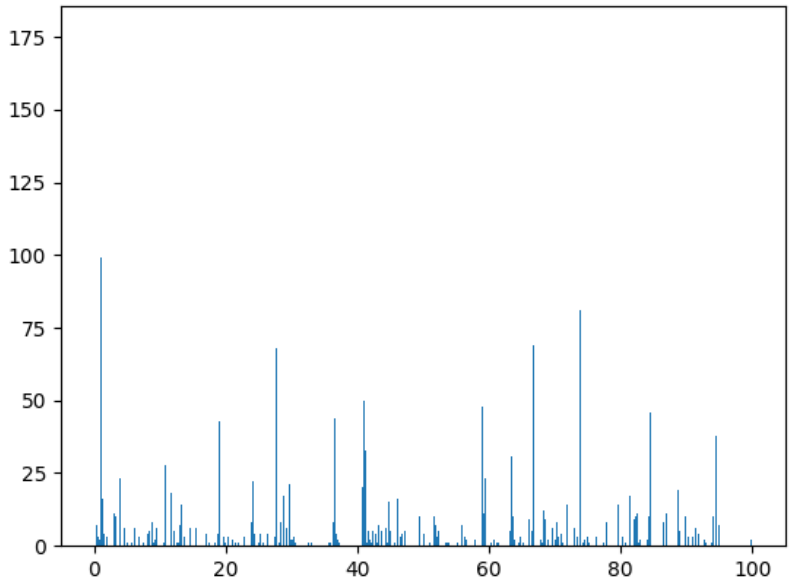
2) **Language : French**



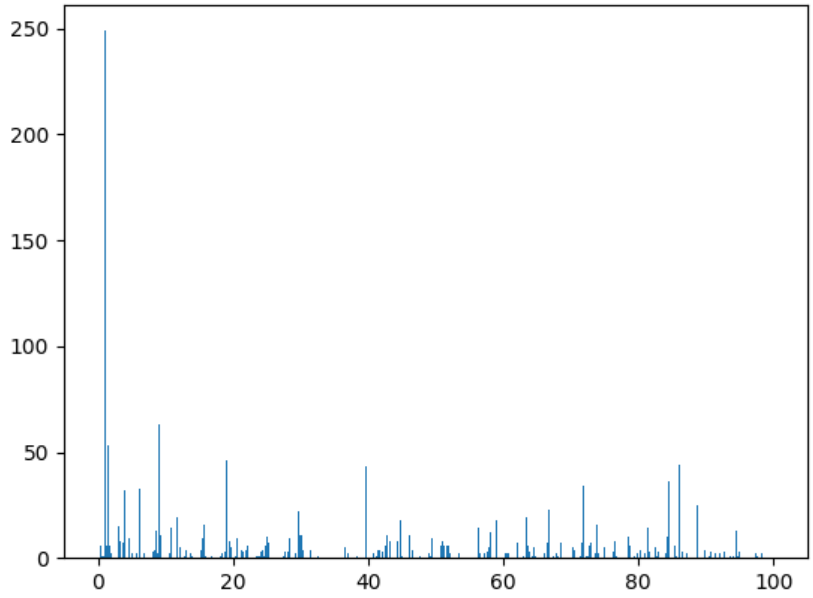

3) **Language : German**



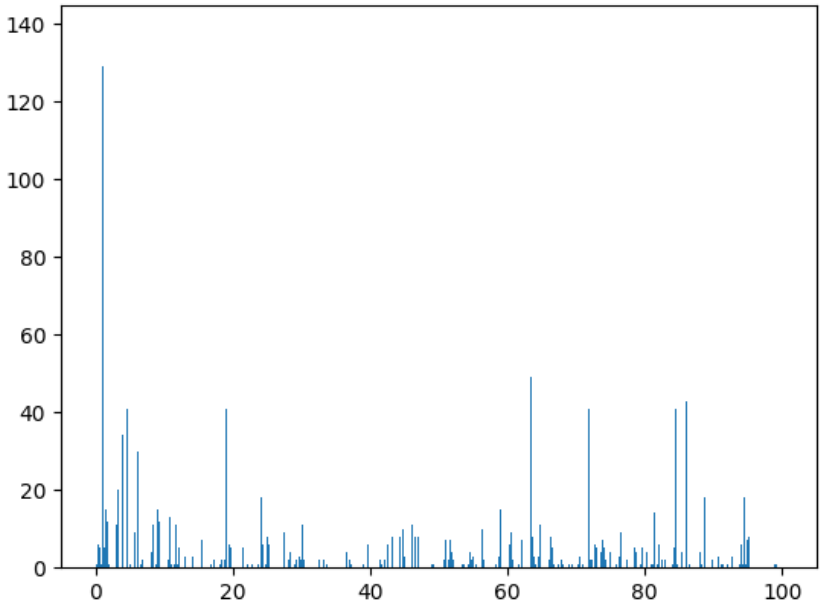
4) **Language : Italian**




5) **Language : Spanish**



## 6) Language : Portuguese




**Observation:**

On observing the plots, the following interpretations can be made:

1) Every plot is unique in its own way. No plot is identical to any of the plots above.
2) Most instances/histogram bars of mystery.txt coincide with the bars of Spanish.txt.
3) Relative closeness is observed in the frequency (y-axis) in mystery.txt and Spanish.txt.

**Result:**

Based on the observations ,plots and results shown above it could be inferred that the language of mystery.txt is Spanish. Therefore, we can say that NLP helps in language classification.