```
#!/usr/bin/env python
# coding: utf-8

# In[14]:


import math
import re, sys
from collections import Counter
import nltk
from nltk import word_tokenize
from nltk.util import ngrams


def calculate(words):

    unigrams = dict(Counter(zip(words)))
    bigrams = dict(Counter(zip(words,words[1:])))
    word1 = {}
    for key, value in unigrams.items():
        word1[key[0]] = 0
    word2 = word1
    for key, value in bigrams.items():
        word1[key[0]] = word1[key[0]] + 1
        word2[key[0]] = word2[key[0]] + 1
    results = []
    word1_sum = sum(word1.values())
    word2_sum = sum(word2.values())
    bigram_sum = sum(bigrams.values())
    for key, value in bigrams.items():
        oneone = value
        onetwo = word2[key[1]] - value
        twoone = word1[key[0]] - value
        twotwo = bigram_sum - value - onetwo - twoone
        total = oneone + onetwo + twoone + twotwo

        item1 = ((oneone - (((oneone + twoone)/total)*((oneone+onetwo)/total)*(total)))**2)/(((oneone + twoone)/total)*((oneone+onetwo)/total)*(total))
        item2 = ((onetwo - (((onetwo + twotwo)/total)*((oneone + onetwo)/total)*(total)))**2)/(((onetwo + twotwo)/total)*((oneone + onetwo)/total)*(total))
        item3 = ((twoone - (((oneone + twoone)/total)*((twoone + twotwo)/total)*(total)))**2)/(((oneone + twoone)/total)*((twoone + twotwo)/total)*(total))
        item4 = ((twotwo - (((onetwo + twotwo)/total)*((twoone + twotwo)/total)*(total)))**2)/(((onetwo + twotwo)/total)*((twoone + twotwo)/total)*(total))
        chi_square = item1+item2+item3+item4
        #print(chi_square)
        pmi = math.log((((value)/(bigram_sum))/((word1[key[0]]/bigram_sum)*(word2[key[1]]/bigram_sum))),2)
        entry = (key, chi_square, pmi)
        results.append(entry)
    return results



regex = re.compile('[^a-zA-Z.]')
myfile = regex.sub('',sys.argv[1])
words = re.findall(r"(?:(?<=^)[A-Za-z.]+|(?<= )[A-Za-z.]+(?= )|(?<= )[A-Za-z.]+$)", open(myfile).read())
chi_pmi = regex.sub('', sys.argv[2])
results =  calculate(words)
if chi_pmi == 'pmi':
    pmi_index = 2
    results.sort(reverse=True if pmi_index == 2 else True, key=lambda k: k[pmi_index])
    count = 0
    template = "{0:20}{1:20}{2:10}"
    for item in results:
        if count < 20:
            p = (item[0][0], item[0][1],str(round(item[pmi_index],4)))
            print(template.format(*p))
            count = count + 1

elif chi_pmi == 'chi':
    chi_index = 1
    results.sort(reverse=True if chi_index == 2 else True, key=lambda k: k[chi_index])
    count = 0
    template = "{0:20}{1:20}{2:10}"
    for item in results:
        if count < 20:
            c = (item[0][0], item[0][1],str(round(item[chi_index],4)))
            print(template.format(*c))
            count = count + 1



# In[ ]:
```

```
# In[ ]:
```