```
import  sys as os
import math
import nltk
from nltk import ngrams
from nltk import word_tokenize
from collections import Counter
from collections import OrderedDict
from collections import defaultdict

bicount =   []
uniscount =  []
def CollocationPMI(inputfile):
    data = inputfile.read()
    inputfile.close()
    tok = nltk.word_tokenize(data)
    newtok =   [w.lower() for w in tok if w.isalpha()]
    unis = ngrams(newtok,1)
    bis =  ngrams(newtok,2)
    uniscount = Counter(unis)
    bicount = Counter(bis)
    uniFreq = dict()
    biFreq = dict()
    biPMI = dict()
    unisumcount =0
    for u in uniscount:
        unisumcount =  unisumcount + uniscount[u]
        uniFreq[u[0]]  = uniscount[u]
    bisumcount=0
```

```
    for b in bicount:
        biFreq[b[0] + '_' + b[1]] = bicount[b]
        bisumcount = bisumcount + bicount[b]
    for b1 in bicount:
        biPMI[b1[0] + '_' + b1[1]] = pmi(b1[0],b1[1],uniFreq,biFreq,
unisumcount,bisumcount)
    revSortedPMI =  sorted(biPMI.items() ,reverse = True, key=lambda t : t[1])
    reversePMI = Counter(revSortedPMI).most_common(20)
    for key in reversePMI:
        print(key[0][0].replace('_',' ') + ' ' + str(key[0][1]) )
def CollocationCHI(inputfile):
    data =  inputfile.read()
    inputfile.close()
    tok = nltk.word_tokenize(data)
    newtok =   [w.lower() for w in tok if w.isalpha()]
    unis = ngrams(newtok,1)
    bis = ngrams(newtok,2)
    uniscount = Counter(unis)
    bicount = Counter(bis)
    uniFreq = dict()
    biFreq = dict()
    biWord1Freq =  dict()
    biWord2Freq = dict()
    biChiSquare = dict()
    unisumcount =0
    for u in uniscount:
        unisumcount = unisumcount + uniscount[u]
        uniFreq[u[0]]  = uniscount[u]
    bisumcount=0
```

```
    for  b in bicount:
        biFreq[b[0] + '_' + b[1]] = bicount[b]
        if b[1] in biWord2Freq:
            biWord2Freq[b[1]]  =  biWord2Freq[b[1]] + bicount[b]
        else:
            biWord2Freq[b[1]]  =  bicount[b]


        if b[0]  in  biWord1Freq:
            biWord1Freq[b[0]]  =  biWord1Freq[b[0]] + bicount[b]
        else:
            biWord1Freq[b[0]]   = bicount[b]
        bisumcount  =  bisumcount  +  bicount[b]
    for  b1  in  bicount:
        biChiSquare[b1[0] + '_' + b1[1]] =
 chisquare(biWord1Freq[b[0]],biWord2Freq[b[1]],biFreq[b1[0] + '_' + b1[1]],bisumcount)
     reversechisquare  =  sorted(biChiSquare.items() ,reverse  =  True, key=lambda t :
t[1])
    reversechi  =  Counter(reversechisquare).most_common(20)
    for  key  in  reversechi:
        print(key[0][0].replace('_',' ') + ' ' + str(key[0][1]) )
def  pmi(word1,   word2,   uni_freq,  bi_freq,  uniTotal,  biTotal):
    prob_word1  =  uni_freq[word1] /  uniTotal
    prob_word2  =  uni_freq[word2] /  uniTotal
    prob_word1_word2  =  bi_freq[word1 + '_' + word2] / biTotal
    return math.log(prob_word1_word2/float(prob_word1*prob_word2),2)
def chisquare(bi_word1,bi_word2,bi_freq,biTotal):
    obs_value =   bi_freq
    estim_value  =   (bi_word1/biTotal)* (bi_word2/biTotal)* biTotal
    return  ((obs_value-estim_value)**2)/estim_value
```

```
def main():
    measure  =  os.argv[2]
    filename  =  os.argv[1]
    inputfile  =  open(filename)
    if measure  ==   "PMI":
        CollocationPMI(inputfile)
    elif measure  ==  "chi-square":
        CollocationCHI(inputfile)
    else:
         print("enter correct measure: chi-square or PMI ")
if __name__ == "__main__":
    main()
```