AMOGH GAIKWAD
CS657
ASSIGNMENT -1

TASK 1:

For task 1 I have removed all the punctuations from the text and ran the wordcount program multiple times for getting the required running time vs size of file graph.

The pseudo-code for the tasks are given below:

```
#--- get all lines from stdin ---
for line in sys.stdin:
   #--- remove leading and trailing whitespace---
   line = line.strip()

   #--- remove the punctuation
   lines_p=re.sub('[^*a-z\s]',' ',line)

   #---remove all the HTML tags
   lines_f = re.sub("<.*?>"," ",lines_p)

   #--- split the line into words ---
   words = lines_p.split()

   #--- output tuples [word, 1] in tab-delimited format---
   for word in words:
      print '%s\t%s' % (word, "1")
```

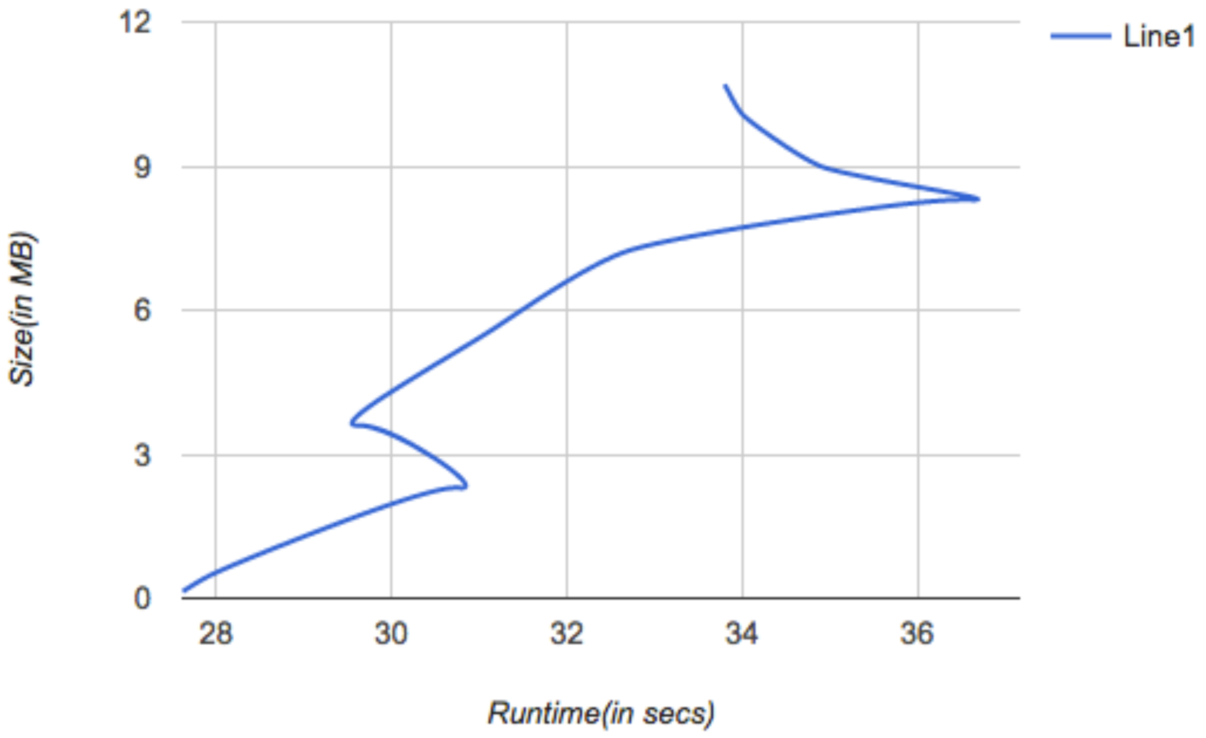
INPUT: The data files in incremental size order.

OUTPUT: The specific word counts for each file (Attached -Sample output- Task 1.txt)

The reducer.py was kept unchanged for this task. The sample output file is attached in the zip file of this submission.

AMOGH GAIKWAD
CS657
ASSIGNMENT -1

GRAPH:



The following can be interpreted from the graph above:

The hadoop map reduce program runs more efficiently for significantly larger file sizes.

AMOGH GAIKWAD
CS657
ASSIGNMENT -1

TASK 2:

Calculating the Average use of every word (all Addresses):
I have made modifications to the reducer to divide the count of each word to the no of years to obtain the average of each word per year (for all the addresses)

Pesudo-code :

REDUCER.py

```
#!/usr/bin/env python
import sys
word2count = {}

total_no_of_years = 0     -→ This gives the total no of years
 for line in sys.stdin:
   # remove leading and trailing whitespace
   line = line.strip()

   # parse the input we got from mapper.py
   word, count = line.split('\t', 1)
   # convert count (currently a string) to int
   try:
      count = float(count)
   except ValueError:
      continue

   try:
      word2count[word] = word2count[word]+count
   except:
      word2count[word] = count

   if(word == "***"):
      total_no_of_years += 1
for word in word2count.keys():
   print '%s\t%s'% ( word, float(word2count[word])/total_no_of_years ) → this gives
the required average of each word
```

AMOGH GAIKWAD
CS657
ASSIGNMENT -1

The average of every word is computed from 1790-2016. The sample output file is also attached in the submission (Sample_output_avg_all_adresses.txt)

TASK 3:
Starting in 1984, compute the average and standard deviation of times a word appears in a window of five years:

PART A: Computing Average:
REDUCER.py

```
#!/usr/bin/env python
import sys
# maps words to their counts
word2count = {}
total_no_of_years = 0     -→ This gives the total no of years

# input comes from STDIN
for line in sys.stdin:
   # remove leading and trailing whitespace
   line = line.strip()

   word, count = line.split('\t', 1)
   try:
      count = float(count)
   except ValueError:
      continue
   try:
      word2count[word] = word2count[word]+count
   except:
      word2count[word] = count

   if(word == "***"):
      total_no_of_years += 1
for word in word2count.keys():
   print '%s\t%s'% ( word, float(word2count[word])/total_no_of_years ) → this gives
the required average of each word
```

AMOGH GAIKWAD
CS657
ASSIGNMENT -1

PART B: Computing Standard Deviation:

Pseudo-code:

```
REDUCER.py
#!/usr/bin/env python
import sys

# maps words to their counts
word2count = {}

total_no_of_years = 0

# input comes from STDIN
for line in sys.stdin:
   # remove leading and trailing whitespace
   line = line.strip()

   # parse the input we got from mapper.py
   word, count = line.split('\t', 1)
   # convert count (currently a string) to int
   try:
      count = float(count)
   except ValueError:
      continue

   try:
      word2count[word] = word2count[word]+count
   except:
      word2count[word] = count

   if(word == "***"):
      total_no_of_years += 1

# write the tuples to stdout
# Note: they are unsorted
for word in word2count.keys():
```

AMOGH GAIKWAD
CS657
ASSIGNMENT -1

print '%s\t%s'% (word, ((float(word2count[word])-float(word2count[word]/total_no_of_years))**2)/len(word2count))

I have Include the computation of for the standard deviation :
word, ((float(word2count[word])-float(word2count[word]/total_no_of_years))**2)/len(word2count))

This line would calculate the average by dividing the count of every word with the total number of years and taking a square it and then dividing the whole value obtained with the total no of words in the document. This is the same as the standard deviation formula.