1. Answer the questions in the collaboration policy on page 1.

- Did you receive any help whatsoever from anyone in solving this assignment?

NO

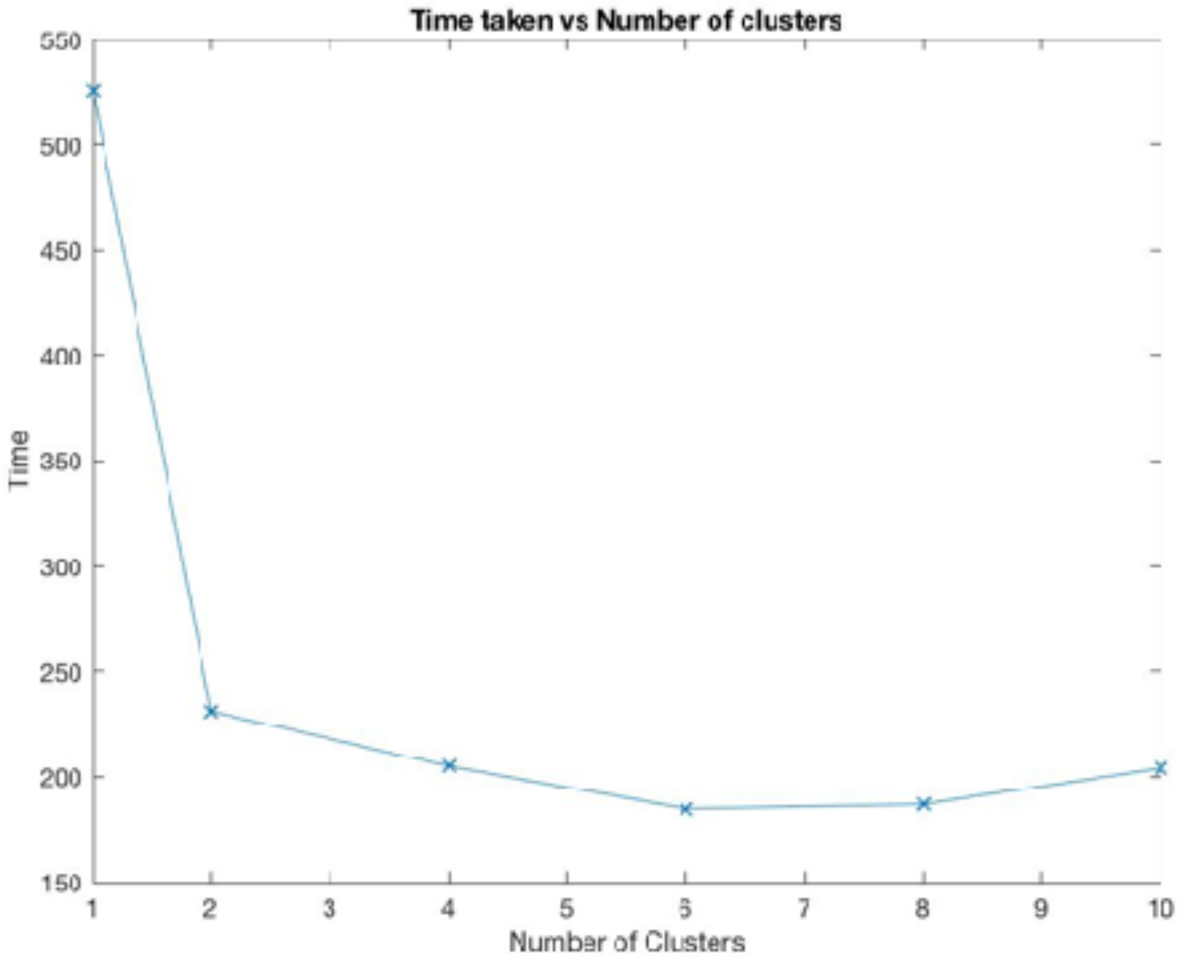
- Did you give any help whatsoever to anyone in solving this assignment?

NO

2. As we increase number of clusters, time taken to complete the reduce process decreases upto some limit (6 clusters), after that it starts increasing. There can be many reasons behind this:

   A. Communication overhead : The output of sort operation has to be sent to reducers, as number of reducers increase this time increases. If data set is large enough this time won't have much effect on total time be because of computation overhead. But after certain limit, it will overweight and become more than computation time.

   B. If every processor in reduce step, does not have same speed or amount of work to do,(data is not evenly distributed across processes), the time taken for whole operation will be determined by processor that has smallest data.



3.

Mapper 1:

Pseudo-Code:

```
for line in sys.stdin:
    doc = line.split('\t')
    doc_id = doc[0]
    features = tokenizeDoc(doc[1])
    D = len(features)
    for feature in features:

        print "W=", feature, "D_ID=", doc_id, "D=", D, "\t", "1"

    print "W=*,D_ID=", doc_id, "\t", D       #Printing W* just to calculte avg
        #lenght in reducer, to avoid unnecessary parsing

    print "D*=", "\t",
```

Output:

```
W=alpha,D_ID=id1,D=560    1

W=beta,D_ID=id1,D=560     1

W=zebra,D_ID=id1,D=560    1

W=*,D_ID=id1    560

D=*   1

W=alpha,D_ID=id2,D=100    1

W=beta,D_ID=id2,D=100     1

W=zabra,D_ID=id2,D=100    1

W=*,D_ID=id2    100

D=*   1

.

.

.
```

```
W=zebra,D_ID=idn,D=500     1

W=*,D_ID=idn    500

D=*   1
```

Sort After Mapper 1:

Output:

```
D=*   1

D=*   1

D=*   1

W=*,D_ID=id1    560

W=*,D_ID=id2    100

W=*,D_ID=idn    500

W=alpha,D_ID=id1,D=560     1

W=alpha,D_ID=id2,D=100     1

W=beta,D_ID=id1,D=560       1

W=beta,D_ID=id2,D=100       1

W=zebra,D_ID=id1,D=560     1

W=zabra,D_ID=id2,D=100     1

W=zebra,D_ID=idn,D=500     1
```

Reducer 1:

Pseudo-Code:

```
#Calculatea avg doc length and reduce counts
total_doc_cnt = 0
total_word_cnt = 0
for line in sys.stdin:
    m = line.split("\t")
    event = m[0]
    cnt = int(m[1].replace("\n",""))
    if "W=*" in event:          #Using W* just to calculate avg document length
        total_doc_cnt += 1      #Not printing document length again,
        total_word_cnt += cnt   #as it is already included in word string
    else:
        if event == event_prev:
            cnt_prev += cnt
        else:
            if event_prev != "":
                print event_prev, "\t", cnt_prev, "\n"
                cnt_prev = cnt
                event_prev = event
        else:
            event_prev = event
            cnt_prev = cnt
#To print last event becuase it won't go in the if as there won't be any event
        after that
print event_prev, "\t", cnt_prev, "\n"
#Print avg_length of the document
print "avgdl", "\t", (total_word_cnt/total_doc_cnt)
```

Output:

```
D=*   777
W=*,D_ID=id1    560
W=*,D_ID=id2    100
W=*,D_ID=idn    500
W=alpha,D_ID=id1,D=560    50
W=alpha,D_ID=id2,D=100    60
W=beta,D_ID=id1,D=560     70
W=beta,D_ID=id2,D=100     80
W=zebra,D_ID=id1,D=560    90
W=zabra,D_ID=id2,D=100    95
W=zebra,D_ID=idn,D=500    11
```

avgdl 999

Mapper 2:

Pseudo-Code:

```
for line in sys.stdin:
    m = line.split("\t")
    event = m[0]
    cnt = int(m[1].replace("\n",""))
    if "W=" in event and "Y=" in event:
        if event == event_prev:            #Print as it is and accumulate count
            print event, "\t", cnt, "\n"    #W=alpha,D_ID=id1,D=560     50
            cnt_prev += 1
        else:
            if event_prev != "":
                print event_prev, "\t", cnt_prev, "\n"
                word = extract_word_from_event(event_prev) #extract alpha from
          W=alpha,D_ID=id1,D=560
                print "W=", word, "D_ID=*\t", cnt_prev  #Print number of
          documents containing word
                cnt_prev = 1
                event_prev = event
        else:

            event_prev = event

            cnt_prev = 1

#To print last event becuase it won't go in the if as there won't be any event
        after that

word = extract_word_from_event(event_prev) #extract alpha from
        W=alpha,D_ID=id1,D=560

print "W=", word, "D_ID=*\t", cnt_prev  #Print number of documents
        containing word
```

Output:

```
D=*   777

W=*,D_ID=id1    560

W=*,D_ID=id2    100

W=*,D_ID=idn    500

W=alpha,D_ID=id1,D=560     50

W=alpha,D_ID=id2,D=100     60
```

W=alpha,D_ID=*   60

```
W=beta,D_ID=id1,D=560      70

W=beta,D_ID=id2,D=100      80
```

W=beta,D_ID=* 80

```
W=zebra,D_ID=id1,D=560    90

W=zabra,D_ID=id2,D=100    95

W=zebra,D_ID=idn,D=500    11
```

W=zebra,D_ID=*11

```
avgdl        999
```

Reducer 2:

Pseudo-Code:

```
#Sort will just bring n(wi) term at the beginning of every
        W=word,D_ID=id,D=cnt

#so that it eases calculation in mapper 3 and we don't have to store n(wi) for
        every word

#W=alpha,D_ID=* will automatically come before
        "W=alpha,D_ID=id1,D=560" because of sort
```

Output:

```
avgdl        999

D=*   777
```

W=alpha,D_ID=*   60

```
W=alpha,D_ID=id1,D=560    50

W=alpha,D_ID=id2,D=100    60
```

W=beta,D_ID=* 80

```
W=beta,D_ID=id1,D=560     70
W=beta,D_ID=id2,D=100     80
```

W=zebra,D_ID=*11

```
W=zebra,D_ID=id1,D=560    90
W=zabra,D_ID=id2,D=100    95
W=zebra,D_ID=idn,D=500    11
```

Mapper 3:

Pseudo-Code:

```
#This will store query in memory and stream throught the conunts stored from
#reducer 2 and calculte score for every di and qi
#Assume counts from mapper 2 are coming from standard input
Q = file.read("Query")
for line in sys.stdin:
    m = line.split("\t")
    event = m[0]
    cnt = int(m[1].replace("\n", ""))
    if event == "avgdl":
        avgdl = cnt     #it will come first because of sorting
    if "D_ID=*" in line:
        n_wi = cnt
    else:
        word, did, D = extract_word_id_D_from_event(event) #Parsing based on
                                                          #commas and equal to sign
        if word in Q:
            IDF  = log(() N - n(qi)+0.5) / (n(qi) + 0.5))
            score = (IDF * cnt * (k1 + 1))/(cnt + k1*(1-b+(b*D/avgdl)))
            print did,word, "\t", score     #d1,q2 0.08
                                            #d2,q2 0.04
```

Output:

```
d1, q2       0.08
d2, q2       0.04
d3, q3       0.03
d4, q2       0.02
d1, q1       0.08
d2, q1       0.04
d3, q1       0.03
d4, q1       0.02
```

Reducer 3:

Pseudo-Code:

```
#As every statement in mapper output start with documet id,
# The input to reducer will be sorted based on document id, hence score for
# all queries for document will come sequentilly
#Reducer will just add the scores and print final score for each document
d_prev = ""
for line in sys.stdin:
    m = line.split("\t")
    event = m[0]
    d,q = extract_dq_from_event(event)
    score = int(m[1].replace("\n",""))
    if d == d_prev:
        score_prev += score
    else:
        if d_prev != "":
            print d, "score(", d ,",Q") =, score_prev
            cnt_prev = score
            d_prev = d
        else:
            d_prev = d
            score_prev = score
#To print last event becuase it won't go in the if as there won't be any event
     after that
print d, "score(", d ,",Q") =, score_prev
```

Output:

```
d1, score(d1, Q) = 0.08
d2, score(d2, Q) = 0.09
d3, score(d1, Q) = 0.08
d4, score(d2, Q) = 0.09
```