databricks Assignment

```
> from pyspark import  SparkContext
  import re
  import requests
```

```
> def removePunctuation_tokenize(text):
      """Removes punctuation, changes to lower case, and strips leading and
  trailing spaces, then split by white space

      Note:
          Only whitespace, letters, and numbers should be retained.  Other
  characters should should be
          eliminated (e.g. it's becomes its).  Leading and trailing spaces
  should be removed after
          punctuation is removed.

      Args:
          text (str): A string.

      Returns:
          str: The cleaned up string.
      """
      return re.sub(r'[^A-Za-z\s\d]',r'',text).strip().lower().split()
```

```
> removePunctuation_tokenize("   I lobe's tijasj  kkk!!")  ## test the
  removePunctuation_tokenize function to make sure it works

Out[4]: ['i', 'lobes', 'tijasj', 'kkk']
```

```
> ## read in three partial books file as well the query text from S3, load into
  rdd.  Query are texts taken from the book RiderHaggard_SheandAllan with
  alterations, including words skip and words typos.
  r1 = requests.get('https://s3-us-west-
  1.amazonaws.com/datasw/GeorgeJames_WhiteRace.txt')
  r2 = requests.get('https://s3-us-west-
  1.amazonaws.com/datasw/RiderHaggard_SheandAllan.txt')
  r3 = requests.get('https://s3-us-west-
  1.amazonaws.com/datasw/StanleyMatthews_DoubleTrouble.txt')
  q=requests.get('https://s3-us-west-1.amazonaws.com/datasw/Query.txt')
  Book_GJ=[line for line in r1.iter_lines()]
  Book_RH=[line for line in r2.iter_lines()]
  Book_SM=[line for line in r3.iter_lines()]
  query=[line for line in q.iter_lines()]
  BookGJ_RDD = sc.parallelize(Book_GJ)
  BookRH_RDD = sc.parallelize(Book_RH)
  BookSM_RDD = sc.parallelize(Book_SM)
  Q_RDD=sc.parallelize(query)
```

```
> BookGJ_RDD.take(5)
  BookRH_RDD.take(5)
  BookSM_RDD.take(5)
  Q_RDD.take(5)
```

```
Out[6]:
['And how did she know that we were coming? I could not guess and when I',
 'asked Robertson, he merely shrugged his shoulders and intimated that he',
 'took no interest in the matter. The truth is that nothing moved the man,',
 'whose whole soul was wrapped in one desire, namely to rescue, or avenge,',
 'the daughter against whom he knew he had so sorely sinned.']
```

```
> ## transform each book RDD using removePunctuation_tokenize function
  BookGJ_RDD_transformed=BookGJ_RDD.flatMap(removePunctuation_tokenize)
  BookRH_RDD_transformed=BookRH_RDD.flatMap(removePunctuation_tokenize)
  BookSM_RDD_transformed=BookSM_RDD.flatMap(removePunctuation_tokenize)
  Q_RDD_transformed=Q_RDD.flatMap(removePunctuation_tokenize)
```

```
> BookGJ_RDD_transformed.count()
  BookRH_RDD_transformed.count()
  BookSM_RDD_transformed.count()
  Q_RDD_transformed.count()
```

```
Out[8]: 247
```

```
>
  BookGJ_RDD_transformed.take(10)
  BookRH_RDD_transformed.take(10)
  BookSM_RDD_transformed.take(10)
  Q_RDD_transformed.take(10)

Out[9]: ['and', 'how', 'did', 'she', 'know', 'that', 'we', 'were', 'coming', 'i']
```

```
> ##  now we have both books RDD and query RDD ready,  we need build a ngram for
  each of them, in order to trace which book the query is taken from.  We do
  have noises in the query, so the idea here is to find the book sources that
  have the maxium count of ngram matching with the query.  I used trigram here.

  input_list = ['all', 'this', 'happened', 'more', 'or', 'less']

  def find_trigrams(input_list):
    return zip(input_list, input_list[1:],input_list[2:])

  find_trigrams(input_list)

Out[10]:
[('all', 'this', 'happened'),
 ('this', 'happened', 'more'),
 ('happened', 'more', 'or'),
 ('more', 'or', 'less')]
```

```
> BookGJ_trigram=find_trigrams([x for x in
  BookGJ_RDD_transformed.toLocalIterator()])
  BookRH_trigram=find_trigrams([x for x in
  BookRH_RDD_transformed.toLocalIterator()])
  BookSM_trigram=find_trigrams([x for x in
  BookSM_RDD_transformed.toLocalIterator()])
  Q_trigram=find_trigrams([x for x in Q_RDD_transformed.toLocalIterator()])
  #print BookGJ_trigram
  print Q_trigram
```

```
[('and', 'how', 'did'), ('how', 'did', 'she'), ('did', 'she', 'know'), ('she',
 'know', 'that'), ('know', 'that', 'we'), ('that', 'we', 'were'), ('we', 'wer
e', 'coming'), ('were', 'coming', 'i'), ('coming', 'i', 'could'), ('i', 'coul
d', 'not'), ('could', 'not', 'guess'), ('not', 'guess', 'and'), ('guess', 'an
d', 'when'), ('and', 'when', 'i'), ('when', 'i', 'asked'), ('i', 'asked', 'rob
ertson'), ('asked', 'robertson', 'he'), ('robertson', 'he', 'merely'), ('he',
 'merely', 'shrugged'), ('merely', 'shrugged', 'his'), ('shrugged', 'his', 'sh
oulders'), ('his', 'shoulders', 'and'), ('shoulders', 'and', 'intimated'), ('a
nd', 'intimated', 'that'), ('intimated', 'that', 'he'), ('that', 'he', 'too
k'), ('he', 'took', 'no'), ('took', 'no', 'interest'), ('no', 'interest', 'i
n'), ('interest', 'in', 'the'), ('in', 'the', 'matter'), ('the', 'matter', 'th
e'), ('matter', 'the', 'truth'), ('the', 'truth', 'is'), ('truth', 'is', 'tha
t'), ('is', 'that', 'nothing'), ('that', 'nothing', 'moved'), ('nothing', 'mov
ed', 'the'), ('moved', 'the', 'man'), ('the', 'man', 'whose'), ('man', 'whos
e', 'whole'), ('whose', 'whole', 'soul'), ('whole', 'soul', 'was'), ('soul',
 'was', 'wrapped'), ('was', 'wrapped', 'in'), ('wrapped', 'in', 'one'), ('in',
 'one', 'desire'), ('one', 'desire', 'namely'), ('desire', 'namely', 'to'),
 ('namely', 'to', 'rescue'), ('to', 'rescue', 'or'), ('rescue', 'or', 'aveng
e'), ('or', 'avenge', 'the'), ('avenge', 'the', 'daughter'), ('the', 'daughte
r', 'against'), ('daughter', 'against', 'whom'), ('against', 'whom', 'he'),
 ('whom', 'he', 'knew'), ('he', 'knew', 'he'), ('knew', 'he', 'had'), ('he',
```

```
> ### now find the overlap of the ngram between query and the book sources

  def Overlap_count(query,book):
    return(len([x for x in query if x in book]))
```

```
> print 'Query Trigrams found \n in Book_GJ is : %d \n in Book_SM is %d \n in
  Book_RH is %d'%
  (Overlap_count(Q_trigram,BookGJ_trigram),Overlap_count(Q_trigram,BookSM_trigra
  m),Overlap_count(Q_trigram,BookRH_trigram))


  ## we can see it makes correct prediction, the query is taken from book
  RH(RiderHaggard_SheandAllan) with alterations, including words skip and words
  typos
```

```
Query Trigrams found
 in Book_GJ is : 5
 in Book_SM is 1
 in Book_RH is 224
```