Open the file that recorded clean wiki pages

In [40]:
```
from pyspark.sql import Row
from pyspark.sql import SparkSession
import csv
spark = SparkSession.builder.getOrCreate()
f = open("wiki_result.csv")
reader = csv.reader(f,delimiter='|')
ww = []
for w in reader:
    ww.append(w)
ww= map(lambda p: Row(label=int(p[0]), text=str(p[1])),ww)
wikies = spark.createDataFrame(ww)
wikies.show()
f.close
```

```
+-----+--------------------+
|label|                text|
+-----+--------------------+
|    0|asia ( ) earth 's...|
|    0|unilev ( ) dutch-...|
|    0|eurasia combin co...|
|    0|eric hoffer ( jul...|
|    0|shaman ( shah-men...|
|    0|the asian giant h...|
|    0|list asian pornog...|
|    0|georgia ( ; georg...|
|    0|calligraphi ( gre...|
|    0|hornet ( insect g...|
|    0|asia argento ( it...|
|    0|time american wee...|
|    0|the pacif war , s...|
|    0|the boundari cont...|
|    0|the demograph rus...|
|    0|thi list common s...|
|    0|A humid subtrop c...|
|    0|aed albopictu ( s...|
|    0|buddhism ( ) reli...|
|    0|thi list list wor...|
+-----+--------------------+
only showing top 20 rows
```

Out[40]: <function close>

Do tokenize

In [41]:
```
from pyspark.ml.feature import HashingTF, IDF, Tokenizer
tokenizer = Tokenizer(inputCol="text", outputCol="words")
wordsData = tokenizer.transform(wikies)
wordsData.show()
```

```
+-----+--------------------+--------------------+
|label|                text|               words|
+-----+--------------------+--------------------+
|    0|asia ( ) earth 's...|[asia, (, ), eart...|
|    0|unilev ( ) dutch-...|[unilev, (, ), du...|
|    0|eurasia combin co...|[eurasia, combin,...|
|    0|eric hoffer ( jul...|[eric, hoffer, (,...|
|    0|shaman ( shah-men...|[shaman, (, shah-...|
|    0|the asian giant h...|[the, asian, gian...|
|    0|list asian pornog...|[list, asian, por...|
|    0|georgia ( ; georg...|[georgia, (, ;, g...|
|    0|calligraphi ( gre...|[calligraphi, (, ...|
|    0|hornet ( insect g...|[hornet, (, insec...|
|    0|asia argento ( it...|[asia, argento, (...|
|    0|time american wee...|[time, american, ...|
|    0|the pacif war , s...|[the, pacif, war,...|
|    0|the boundari cont...|[the, boundari, c...|
|    0|the demograph rus...|[the, demograph, ...|
|    0|thi list common s...|[thi, list, commo...|
|    0|A humid subtrop c...|[a, humid, subtro...|
|    0|aed albopictu ( s...|[aed, albopictu, ...|
|    0|buddhism ( ) reli...|[buddhism, (, ), ...|
|    0|thi list list wor...|[thi, list, list,...|
+-----+--------------------+--------------------+
only showing top 20 rows
```

Do tokenize

In [42]:
```
hashingTF = HashingTF(inputCol="words", outputCol="rawFeatures",numFeatu
res=300)
featurizedData = hashingTF.transform(wordsData)
featurizedData.show()
```

```
+-----+--------------------+--------------------+--------------------+
|label|                text|               words|         rawFeatures|
+-----+--------------------+--------------------+--------------------+
|    0|asia ( ) earth 's...|[asia, (, ), eart...|(300,[0,1,2,3,4,5...|
|    0|unilev ( ) dutch-...|[unilev, (, ), du...|(300,[0,1,2,3,4,6...|
|    0|eurasia combin co...|[eurasia, combin,...|(300,[0,1,2,3,4,5...|
|    0|eric hoffer ( jul...|[eric, hoffer, (,...|(300,[0,1,2,3,4,5...|
|    0|shaman ( shah-men...|[shaman, (, shah-...|(300,[0,1,2,3,4,5...|
|    0|the asian giant h...|[the, asian, gian...|(300,[0,1,2,3,4,5...|
|    0|list asian pornog...|[list, asian, por...|(300,[1,4,8,10,12...|
|    0|georgia ( ; georg...|[georgia, (, ;, g...|(300,[0,1,2,3,4,5...|
|    0|calligraphi ( gre...|[calligraphi, (, ...|(300,[0,1,2,3,4,5...|
|    0|hornet ( insect g...|[hornet, (, insec...|(300,[0,1,3,4,5,6...|
|    0|asia argento ( it...|[asia, argento, (...|(300,[0,1,2,3,4,5...|
|    0|time american wee...|[time, american, ...|(300,[0,1,2,3,4,5...|
|    0|the pacif war , s...|[the, pacif, war,...|(300,[0,1,2,3,4,5...|
|    0|the boundari cont...|[the, boundari, c...|(300,[0,1,2,3,4,5...|
|    0|the demograph rus...|[the, demograph, ...|(300,[0,1,2,3,4,5...|
|    0|thi list common s...|[thi, list, commo...|(300,[1,4,5,7,9,1...|
|    0|A humid subtrop c...|[a, humid, subtro...|(300,[0,1,2,3,4,5...|
|    0|aed albopictu ( s...|[aed, albopictu, ...|(300,[0,1,2,3,4,5...|
|    0|buddhism ( ) reli...|[buddhism, (, ), ...|(300,[0,1,2,3,4,5...|
|    0|thi list list wor...|[thi, list, list,...|(300,[1,2,3,5,6,1...|
+-----+--------------------+--------------------+--------------------+
only showing top 20 rows
```

In [43]:
```
idf = IDF(inputCol="rawFeatures", outputCol="features")
idfModel = idf.fit(featurizedData)
rescaledData = idfModel.transform(featurizedData)

rescaledData.select("text", "features").show()
rescaledData = rescaledData.select("label","features")
```

```
+--------------------+--------------------+
|                text|            features|
+--------------------+--------------------+
|asia ( ) earth 's...|(300,[0,1,2,3,4,5...|
|unilev ( ) dutch-...|(300,[0,1,2,3,4,6...|
|eurasia combin co...|(300,[0,1,2,3,4,5...|
|eric hoffer ( jul...|(300,[0,1,2,3,4,5...|
|shaman ( shah-men...|(300,[0,1,2,3,4,5...|
|the asian giant h...|(300,[0,1,2,3,4,5...|
|list asian pornog...|(300,[1,4,8,10,12...|
|georgia ( ; georg...|(300,[0,1,2,3,4,5...|
|calligraphi ( gre...|(300,[0,1,2,3,4,5...|
|hornet ( insect g...|(300,[0,1,3,4,5,6...|
|asia argento ( it...|(300,[0,1,2,3,4,5...|
|time american wee...|(300,[0,1,2,3,4,5...|
|the pacif war , s...|(300,[0,1,2,3,4,5...|
|the boundari cont...|(300,[0,1,2,3,4,5...|
|the demograph rus...|(300,[0,1,2,3,4,5...|
|thi list common s...|(300,[1,4,5,7,9,1...|
|A humid subtrop c...|(300,[0,1,2,3,4,5...|
|aed albopictu ( s...|(300,[0,1,2,3,4,5...|
|buddhism ( ) reli...|(300,[0,1,2,3,4,5...|
|thi list list wor...|(300,[1,2,3,5,6,1...|
+--------------------+--------------------+
only showing top 20 rows
```

In [44]:
```
from pyspark.ml.clustering import KMeans

kmeans = KMeans().setK(2).setSeed(1)
model = kmeans.fit(rescaledData)


wssse = model.computeCost(rescaledData)
print("Within Set Sum of Squared Errors = " + str(wssse))


centers = model.clusterCenters()
print("Cluster Centers: ")
for center in centers:
    print(center)
```

```
Within Set Sum of Squared Errors = 53447.3276304
Cluster Centers:
[  3.64005606   0.          6.9126512   2.73938205   3.9037373
   2.33138898   7.42197374   4.86614262   7.00360714   3.52862577   0.
   3.47475518   5.25497411   2.3896737   2.76852441   0.71731246
   4.4073333   1.77148904   4.51706615   5.45735621   6.33847393   0.
   6.18741141   2.22842045   2.76852441   2.78838379   2.8098329
   2.76852441   2.42906735   7.18551901   6.67360095   1.72702476
   4.41136294   5.95761387   6.32143761   4.320407   1.47998832
   3.52862577   3.04577306   4.95709856   3.20289532   6.69696294
   3.60291263   1.39883339   3.1250631   4.17147429   1.76914732
   3.32222929   3.06722217   0.58626499   6.32674656   5.42294066
   4.95709856   1.93145832   4.42006807   2.89562932   0.          3.4
9406762
   7.41290886   5.77570199   3.95477433   3.14214947   2.65968915
   4.07533015   5.25497411   0.          4.7132242   2.71023969
   7.55923641   3.00861776   1.63013206   2.25215614   2.56452788
   1.58723385   2.42374899   0.          3.98585034   1.79722902
   1.58661625   4.1172993   2.9343309   2.33138898   6.57438696
   5.25503099   3.68371544   2.9796245   3.98585034   3.46719941
   5.1844884   2.9254495   4.42872259   5.52584906   1.77315422
   6.98857382   2.46743116   2.31674054   4.43996496   2.97147433   0.
   1.51049451   2.3896737   2.62094938   7.25944877   7.03758856
   2.31650346   4.03243194   1.58661625   5.3486538   3.73807437
   2.6743269   2.18780882   4.36545513   3.23147834   1.30357746
   3.96863931   6.64308389   8.95156643   3.64279528   4.4004967
   5.82117996   3.90059934   2.35940167   2.76852441   4.19720749
   3.61365292   4.72970872   2.78008864   4.80832739   9.36358492
   3.49148234   4.90293265   1.80172491  10.0223733   1.73120604   0.
   3.78850709   6.16580924   3.50180357   2.09825008   2.85273111
   5.63926809   4.58596375   8.45129857   1.51641199   2.04158162
   6.06759897   1.89551498   4.4572115   2.38085077   0.          4.2
2199838
   3.25049945  11.96018419   0.          3.00861776   2.63968013
   3.32222929   4.01149035   2.63824005   3.16337328   1.73359971
   2.89718747   3.63823748   3.15719148   2.91058998   2.08782637   0.
   3.95477433   3.34606055   4.13967747   3.32222929   3.1959168
   3.15719148   5.82117996   2.23070703   4.42963906   2.50954541
   1.64277965   3.46794111   1.89612468   4.72970872   3.84679182
   3.15719148   3.03683835   4.83763213   3.81764945   3.8257732
  14.93461634   3.67193764   2.57389273   3.3198917   4.73176703
   3.08898738   3.12004805  10.24932943   5.05150637   5.75733937
   1.96860175   1.89261996   3.93082607   3.38005205   6.33847393
   4.7315499   2.27360524   2.50954541   4.60449323   5.11087003
  11.24895182   2.80817035   4.55525753   3.97280748   3.26026412
   2.36053134   1.01389358   6.39001403   8.14055635   2.01621597
   3.26394457   3.45404953   4.36545513   2.4143229   2.70258736
   0.92858573   3.19433491   2.6743269   6.63978339   5.02272221
   2.68109733   5.53227373   1.36196262   2.42374899   6.57981643
   1.65533645   3.46794111   4.63875278   3.8593154   4.31127032
   2.54676623   3.52862577   2.59396609   1.66223369   1.92339591
   4.60487422   4.38639081   0.92231156   2.13814292   2.19146232
   3.79604794   2.21845457   0.39314241   5.79125864   5.38168537
   2.95997664   4.38292464   2.91058998   8.23151229   5.07077103
   4.43690915   1.13114658   1.15825173   1.7776841   11.70448114
   7.83929544   0.          2.55543502   2.42374899   1.95253827
   2.04650858   2.87418021   3.15719148   5.50283418   4.61155781
```

```
   3.84679182   5.67785988   2.23249543   8.41457293   1.76914732
   3.90006006   3.95477433   4.06312431   7.31362376   7.74702369
   2.8559515    3.96336126   3.72578218   3.00154592   2.23249543
   4.36545513   3.4563256    4.35416853   3.38005205   3.00861776
   0.67592905   2.08056329   8.98398012   3.64005606   3.00861776
   1.41812603   3.04702389   1.84462312]
[ 0.8589418   0.          0.86218648  0.53427664  0.92052414  0.7771296
6
  1.35437477  0.78638987  1.02325429  0.71965394  0.          0.6792216
9
  0.97063525  1.00784003  0.60713255  0.30175404  1.22569617  0.7776626
  0.54034797  2.42549165  1.04964045  0.          1.04640046  0.4547796
8
  0.38856483  0.63006749  0.33514228  0.78320099  0.97408409  1.1085254
8
  1.40854751  0.31591916  1.14642379  1.43066109  1.18432209  1.1180000
6
  0.40217074  0.97501502  0.49601058  0.99483056  1.44069628  0.8795829
9
  0.95180037  0.52213399  0.65105481  0.6207551   0.46510321  0.7042737
5
  0.74178158  0.17099396  1.27853003  1.17620859  0.84323733  1.9190771
7
  0.51846037  1.26907211  0.          1.28904404  1.07062717  1.0137797
1
  0.85777069  0.80133841  0.53622765  0.57197616  0.99320817  0.
  1.68168201  0.74070171  0.8943541   0.82025073  0.7149702   0.6345360
5
  0.55249062  0.49601058  0.67028456  0.          0.83038549  0.4124500
2
  0.38027307  1.09478627  0.46429286  0.6678458   1.06013537  0.9652097
7
  0.82428818  1.10607273  0.83038549  0.53046345  1.44013567  0.8351977
8
  0.79084332  2.79904119  1.16654883  1.70425492  0.94901198  0.5528585
4
  0.94286695  0.97501502  0.          0.43395105  0.65570315  0.5690219
1
  1.0609269   1.17293143  0.46026207  0.64347318  0.21646313  0.7660832
3
  0.59818219  0.50298394  0.69262738  0.84356621  0.79703608  0.3118125
1
  0.97910509  0.98855415  1.13751752  0.6192752   0.78320099  0.9758814
1
  0.58689573  0.55410191  0.93498412  0.76608323  0.97141207  0.8527119
1
  1.00333586  0.7776626   1.79257819  1.33871109  0.72739215  0.7641244
  0.82391132  0.2816371   0.          1.14140919  0.85120358  0.9095593
7
  0.74070171  0.77306153  0.75796614  2.51742719  1.21893729  0.4387766
2
  0.31324943  0.68847384  0.41830037  0.68096287  0.65241031  0.
  1.36487016  0.48531763  1.1297793   0.          0.5881043   1.1817716
9
  0.83177159  1.0214443   0.58091329  0.59313249  0.76747904  1.3387110
9
  0.85271191  0.61131894  0.57794918  0.67221682  0.          0.6207551
```

```
  0.75518728  0.78646722  1.05641063  0.57644473  0.42560179  1.4211865
1
  0.56303903  0.62534652  0.70156451  0.3890486   0.81962894  0.4966040
8
  0.91903395  0.73463038  1.0214443   0.93583126  1.86389692  0.5646332
7
  1.08335002  1.89293651  0.84998557  0.91605557  0.63479664  1.0384379
9
  0.84830146  1.45478431  1.04127704  1.00596787  0.68539754  0.4642928
6
  0.66726986  0.77342643  0.58036608  1.5010987   0.7582612   0.6881588
2
  0.50494771  1.14140919  0.6230628   0.6192752   0.62891315  0.9094698
2
  0.74432663  0.74178158  0.51606266  0.40952334  0.8962891   1.0327288
7
  0.51388483  0.49177736  0.92728125  1.29171076  0.65774822  0.5719761
6
  0.93632394  0.68096287  1.02918252  1.20327125  1.03123524  0.8317715
9
  0.66726986  0.2720415   0.54963334  0.71175899  0.21122783  1.1899797
9
  0.66322037  0.76448188  0.65241031  0.58742376  0.88215644  2.9788431
7
  0.33480329  0.54034797  0.82391132  0.69625251  0.37089079  0.5704588
3
  0.6422718   1.12036137  0.5060557   0.16237241  1.59527726  0.8969475
6
  1.27800924  1.15299395  0.72006783  1.07062717  0.78927231  1.2695270
1
  0.27876326  0.29939377  0.63748917  0.94901198  1.11712389  0.
  0.81904968  0.65687887  0.41892146  0.72006783  0.43345069  1.0137060
9
  1.37381363  0.91605557  0.80141496  1.09189613  0.44755215  1.0412770
4
  0.62013762  0.92084751  0.82391132  0.81262486  1.21893729  0.9254894
3
  0.57070459  1.01998268  0.68995966  0.85271191  0.26911632  0.9094698
2
  0.88113564  0.95627265  0.71965394  0.70417751  0.20116936  0.5004791
4
  1.1994457   1.11430287  1.16847038  0.33054505  0.73901699  0.7417815
8]
```

Summary

At first I took 10 pages for each category. The Error is around 2000. When I finally used 30 pages for each category. The Error increased to 50000. So the model is not effective to cluster.

next we perform the classification to the wikipages. Question3-wiki:

In [45]:
```
splits = rescaledData.select("label", "features").randomSplit([0.8,
0.2], 1234)
train = splits[1]
test = splits[0]
```

In [46]:
```
from pyspark.ml.classification import NaiveBayes
from pyspark.ml.evaluation import MulticlassClassificationEvaluator

nb = NaiveBayes()
model = nb.fit(train)
predictions = model.transform(test)

evaluator = MulticlassClassificationEvaluator(labelCol="label", predicti
onCol="prediction",
                                              metricName="accuracy")
accuracy = evaluator.evaluate(predictions)
print("Test set accuracy = " + str(accuracy))
```

```
Test set accuracy = 0.382978723404
```

In [47]:
```
from pyspark.ml.classification import DecisionTreeClassifier
from pyspark.ml.evaluation import MulticlassClassificationEvaluator

dt = DecisionTreeClassifier()

model = dt.fit(train)

predictions = model.transform(test)

evaluator = MulticlassClassificationEvaluator(labelCol="label", predicti
onCol="prediction",metricName="accuracy")
accuracy = evaluator.evaluate(predictions)
print("Test set accuracy = " + str(accuracy))
```

```
Test set accuracy = 0.382978723404
```

In [48]:
```
from pyspark.ml.classification import RandomForestClassifier
from pyspark.ml.evaluation import MulticlassClassificationEvaluator

rf = RandomForestClassifier()

model = rf.fit(train)

predictions = model.transform(test)

evaluator = MulticlassClassificationEvaluator(labelCol="label",predictio
nCol="prediction",metricName="accuracy")
accuracy = evaluator.evaluate(predictions)
print("Test set accuracy of RandomForest= " + str(accuracy))
```

```
Test set accuracy of RandomForest= 0.382978723404
```