Distributed Lab 6
Solution
25/May/2017

Preprocessing :

Preprocessed all txt files to a single txt file using a unique identifier of “ next file ”

```
target_file = target = open('1_2_3.txt', 'w')
for i in (range(0,len(sys.argv)-1)):
        file = sys.argv[i+1]
        print (file)
        with open(file,'rb') as f:
                #print (f.read())
                _string = f.read()
                _string = str(_string)
                print(_string)
                target.write(_string)
                target.write("\n\n\n\nnext_file\n\n\n\n")


target_file.close()
```

Exercise 1: Data cleaning and text tokenization

1. Given the text file (Anyfile) used mapper.py file for cleaning

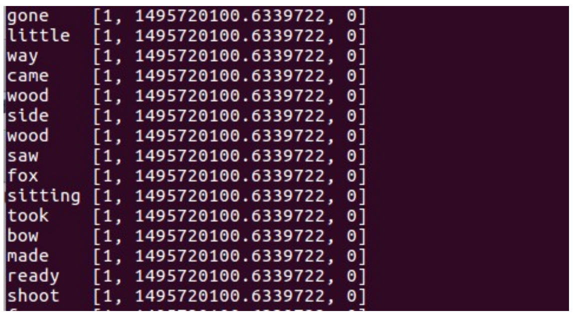
```
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize

if (line == "next_file"):
        file += 1

sentence = line.lower()
words = word_tokenize(sentence)
words =[word.lower() for word in words if word.isalpha()]
_stopwords = set(stopwords.words('english'))
filtered_sentence = []
if (len(words) != 0):
        for w in words:
                if w not in _stopwords :
                        filtered_sentence.append(w)
                        print ('%s\t%s' % (w,[1,start_time,file]))
```

2. Now in this algorithm

- First read the line from the file
- Convert the file to lower case letter’s
- Now removed all punctuations, and number using string.isalpha()
- Now removed all stop words using NLTK stopwords.words(‘english’)
- Now after filtration Tokenized the word and send it to Reducer
- Send these arguments (word, count , start_time and fileNumber)



```
gone    [1, 1495720100.6339722, 0]
little  [1, 1495720100.6339722, 0]
way     [1, 1495720100.6339722, 0]
came    [1, 1495720100.6339722, 0]
wood    [1, 1495720100.6339722, 0]
side    [1, 1495720100.6339722, 0]
wood    [1, 1495720100.6339722, 0]
saw     [1, 1495720100.6339722, 0]
fox     [1, 1495720100.6339722, 0]
sitting [1, 1495720100.6339722, 0]
took    [1, 1495720100.6339722, 0]
bow     [1, 1495720100.6339722, 0]
made    [1, 1495720100.6339722, 0]
ready   [1, 1495720100.6339722, 0]
shoot   [1, 1495720100.6339722, 0]
```

Exercise 2 : Calculate TFIDF scores of words/tokens

Now after receiving in Reducer function, I keep track of the following things at each receiving from mapper for TFIDF in an array

- The token/word counted in each document
- Total Number of words in each document

Algorithm

- Now as single line is received in reducer it get parsed as key is the word & value is an array of few items

```
word, array = line.strip().split('\t')

array = re.split('[|,|]',array)

doc = int ((array[2].strip().split(']'))[0])
tme = float(array[1].strip())

total_words_docs[doc] += 1
```

- Now we have each and every element separated all we have to do now to calculate the TFIDF for each word
- We have saved the words with their counts , and we have total words from each documents and their total count
- Now we take each word and do the following,

for TF

```
   tf = _tf(word_count_document[i][0],word_count_document[i][1],
total_words_docs[word_count_document[i][2]])

def _tf (wrd,cnt,ttl_doc_count):
    return cnt/ttl_doc_count
```

where word_count_document[I] [0] *is word*

where word_count_document[I] [1] *is its count*

where total_word_docs[word_count_document[I][2]] *is words in particular document*

Now IDF

```
idf = _idf(len(total_words_docs), _word_in_all_docs(word_count_document[i][0]))

def _idf(ttl_docs , word_number_of_docs):
    var = math.log(ttl_docs/word_number_of_docs)
    return var
```

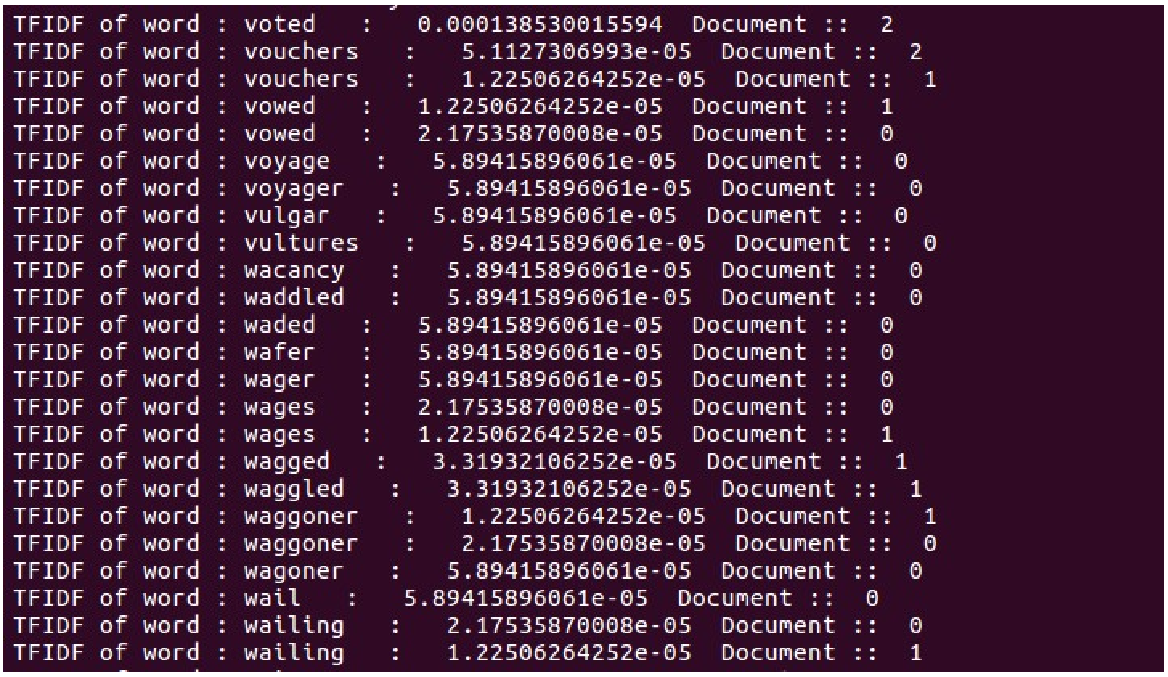
now here
len(total_words_docs) *is total number of documents*
_word_in_all_docs(word_count_document[I] [0]) *is the word that appears in different documents*

Now finally

```
print (" TFIDF of word :", word_count_document[i][0], " : ",tf*idf, " Document :: ",
total_words_docs[i][2])
```

Running the cat command to run the process

zfar@zfar-HP-ProBook-4530s:/media/zfar/media files/hadoop/Text Data/Data Cleaning $ cat p_files.txt | python mapper.py | sort | python reducer.py



```
TFIDF of word : voted   :   0.000138530015594  Document ::  2
TFIDF of word : vouchers   :   5.1127306993e-05  Document ::  2
TFIDF of word : vouchers   :   1.22506264252e-05  Document ::  1
TFIDF of word : vowed   :   1.22506264252e-05  Document ::  1
TFIDF of word : vowed   :   2.17535870008e-05  Document ::  0
TFIDF of word : voyage   :   5.89415896061e-05  Document ::  0
TFIDF of word : voyager   :   5.89415896061e-05  Document ::  0
TFIDF of word : vulgar   :   5.89415896061e-05  Document ::  0
TFIDF of word : vultures   :   5.89415896061e-05  Document ::  0
TFIDF of word : wacancy   :   5.89415896061e-05  Document ::  0
TFIDF of word : waddled   :   5.89415896061e-05  Document ::  0
TFIDF of word : waded   :   5.89415896061e-05  Document ::  0
TFIDF of word : wafer   :   5.89415896061e-05  Document ::  0
TFIDF of word : wager   :   5.89415896061e-05  Document ::  0
TFIDF of word : wages   :   2.17535870008e-05  Document ::  0
TFIDF of word : wages   :   1.22506264252e-05  Document ::  1
TFIDF of word : wagged   :   3.31932106252e-05  Document ::  1
TFIDF of word : waggled   :   3.31932106252e-05  Document ::  1
TFIDF of word : waggoner   :   1.22506264252e-05  Document ::  1
TFIDF of word : waggoner   :   2.17535870008e-05  Document ::  0
TFIDF of word : wagoner   :   5.89415896061e-05  Document ::  0
TFIDF of word : wail   :   5.89415896061e-05  Document ::  0
TFIDF of word : wailing   :   2.17535870008e-05  Document ::  0
TFIDF of word : wailing   :   1.22506264252e-05  Document ::  1
```

Now running the hadoop map-reduce

command :

zfar@zfar-HP-ProBook-4530s:~$ hadoop jar hadoop-2.7.2/share/hadoop/tools/lib/hadoop-streaming-2.7.2.jar -file hadoop-2.7.2/hadoop/Cleaning/mapper.py -mapper "python mapper.py" -file hadoop-2.7.2/hadoop/Cleaning/reducer.py -reducer "python reducer.py" -input /user/zfar/TextTFIDF/input -output /user/zfar/TextTFIDF/output

Test on Files :

text_File _ 1   , text_File_ 2 & text_File_3

Detail File is attached

Time : 311.117 seconds