**Lab Course: Distributed Data Analytics Exercise Sheet 6 Solutions**

**Excercise 1:**

**1.Code:**

**Dataset used:**

**http://www.gutenberg. org/files/2591/2591-0.txt**

**mapper_classic.py:**

**The text file is read line by line from the input stream and then punctuations, stopwords are removed to get accurate results.**

```
#reading input from stdin
for line in sys.stdin:
    #removing anything other than letters, line breaks, period
    line = re.sub('[^a-zA-Z \n\.]', ' ', line)
    #escaping punctuation from strings
    chars = re.escape(string.punctuation)
    line = re.sub(r'[' + chars + ']', ' ', line)
    #removing the leading and trailing white spaces
    line = line.strip()
    #tokenizing the line
    word_tokens = word_tokenize(line)
    #removing stop words
    line = [w for w in word_tokens if not w.lower() in stop_words]
    #printing words to the output
    for word in line:
        print('%s\t%s' % (word, 1))
```

**reducer_classic.py:**

**Calculating the word and count pairs in the reducer.**

```
    #inncrementing count as long as the same word is seen
    if current_word == word:
        current_count += count
    else:
        #printing a word and its count
        if current_word:
            print('%s\t%s' % (current_word, current_count))
        current_count = count
        current_word = word

#printing the word and the output of the last word.
if current_word == word:
    print('%s\t%s' % (current_word, current_count))
```

**2. Step by step application and output:**

**Outputs:**

**Executing the mapper and reducer:**

```
C:\Users\admin\Anaconda3\Scripts>hadoop jar C:\\Users\\admin\\hadoop-streaming-2.7.2.jar -mapper "p
ython.exe C:\\Users\\admin\\mapper_classic.py" -reducer "python.exe C:\\Users\\admin\\reducer_class
ic.py" -input /ex1gutenberg.txt -output /user/admin/op1038
20/06/22 11:30:20 INFO Configuration.deprecation: session.id is deprecated. Instead, use dfs.metric
s.session-id
20/06/22 11:30:20 INFO jvm.JvmMetrics: Initializing JVM Metrics with processName=JobTracker, sessio
nId=
```

**Successful execution of the mapper and the reducer:**

```
        File Output Format Counters
                Bytes Written=47295
20/06/22 11:30:58 INFO streaming.StreamJob: Output directory: /user/admin/op1038
```

**Sample Output of the mapper and reducer:**


**Excercise 2:**

**Code:**

**Datasets used:**

**1. Raw text 1: http://www.gutenberg.org/files/2591/2591-0.txt**

**2. Raw text 2: http://www.gutenberg.org/files/1400/1400-0.txt**

**3. Raw text 3: http://www.gutenberg.org/files/219/219-0.txt**

**4. Raw text 4: http://www.gutenberg.org/files/4300/4300-0.txt**

**5. Raw text 5: http://www.gutenberg.org/files/158/158-0.txt**

**1,2: mapper_clean.py:**

**The data set is cleaned by removing the punctuations and the numbers using python regex and string library.**

**Regex used for removing anything apart from alphabets, linebreaks and dots.**

**line = re.sub('[^a-zA-Z \n\.]', ' ', line)**

**Stopwords are removed using the nltk library in python.**

```
stop_words = set(stopwords.words('english'))

#reading input from stdin
for line in sys.stdin:
    #removing anything other than letters, line breaks, period
    line = re.sub('[^a-zA-Z \n\.]', ' ', line)
    #escaping punctuation from strings
    chars = re.escape(string.punctuation)
    line = re.sub(r'[' + chars + ']', ' ', line)
    #removing the leading and trailing white spaces
    line = line.strip()
    #tokenizing the line
    word_tokens = word_tokenize(line)
    #removing stop words
    line = [w for w in word_tokens if not w.lower() in stop_words]
    #printing words to the output
    for word in line:
        print('%s\t%s' % (word, 1))
```

**reducer_clean.py:**

**Prints the cleaned output from the mapper into the output stream:**

```
for line in sys.stdin:
    # remove leading and trailing whitespace
    line = line.strip()
    #getting the word from the mapper
    word, count = line.split('\t', 1)
    #string to float
    try:
        count = int(count)
    except ValueError:
        continue
    print('%s\t%s' % (word, count))
```

**Step by step application and output:**

**Output:**

**Executing the mapper and reducer programs:**

```
C:\Users\admin\Anaconda3\Scripts>hadoop jar C:\\Users\\admin\\hadoop-streaming-2.7.2.jar -mapper "p
ython.exe C:\\Users\\admin\\mapper_clean.py" -reducer "python.exe C:\\Users\\admin\\reducer_clean.p
y" -input /gutenbergtexts/* -output /user/admin/op1040
20/06/22 11:42:56 INFO Configuration.deprecation: session.id is deprecated. Instead, use dfs.metric
s.session-id
20/06/22 11:42:56 INFO jvm.JvmMetrics: Initializing JVM Metrics with processName=JobTracker, sessio
nId=
```

**Tokenized and cleaned output of mapper and reducer:**

```
youthful        1
youthfully      1
youthfulness    1
youths  1
youths  1
yu      1
yum     1
yum     1
yumyum  1
yung    1
ywimpled        1
zeal    1
zeal    1
zeal    1
zeal    1
zeal    1
zeal    1
zealous 1
zealous 1
zealous 1
zealous 1
zebra   1
zenith  1
zephyrs 1
zest    1
zest    1
```

**Excercise 3:**

**Calculate TFIDF scores of words/tokens:**

**Datasets used:**

**1. Raw text 1: http://www.gutenberg.org/files/2591/2591-0.txt**

**2. Raw text 2: http://www.gutenberg.org/files/1400/1400-0.txt**

**3. Raw text 3: http://www.gutenberg.org/files/219/219-0.txt**

**4. Raw text 4: http://www.gutenberg.org/files/4300/4300-0.txt**

**5. Raw text 5: http://www.gutenberg.org/files/158/158-0.txt**

**mapper_tfidf.py:**

**Cleaning the input and sending it to the reducer along with the document name in which the word was found**

```
ading input from input stream
line in sys.stdin:
#removing leading and trailing white spaces
line = re.sub('[^a-zA-Z \n\.]', ' ', line)
#removing punctuations
chars = re.escape(string.punctuation)
line = re.sub(r'[' + chars + ']', ' ', line)
line = line.strip()
#tokenizing the text from the text files
word_tokens = word_tokenize(line)
line = [w for w in word_tokens if not w.lower() in stop_words]
# passing the cleaned text to the reducer with the name of the input text file
for word in line:
    filename = os.environ["map_input_file"]

    print(word,'\t',filename)
```

**reducer_tfidf.py:**

**docdict – dictionary that maintains document wise words and counts:**

**idfdict – dictionary that contains words and the number of document in which the word was present:**

```
    #calculating the word and its count for a single file and maintaining a diction
    if document not in docdict:
        docdict[document] = {}
    else:
        if word not in docdict[document]:
            docdict[document][word]  = 1
            if word not in idfdict:
                idfdict[word]=1
            else:
                idfdict[word]+=1
        else:
            docdict[document][word]+=1
```

**Calculating the tf scores for each word in a document and multiplying it with the IDF value of the word in the idfdict dictionary:**

$$TFIDF = TF * IDF\,,$$

$$\mathrm{TF}(t,d) = \frac{n^d(t)}{|d|}\,,$$

$$\mathrm{IDF}(t) = \log\frac{|C|}{n^C(t)}\,,$$

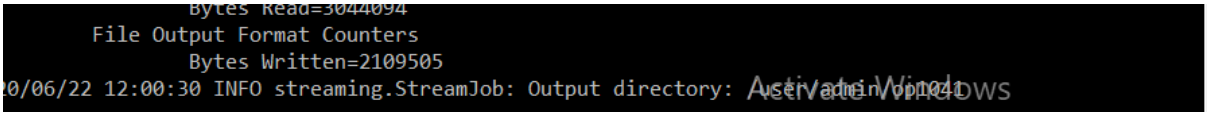
```
#calculating tfidf scores
for doc,dict in docdict.items():
    #iterating - word in document level
    print('current document',doc,len(docdict[doc]))
    for key,value in dict.items():
        tf = value/sum(docdict[doc].values())
        idf = np.log(5/idfdict[key])
        print('tfidf for',key,':',tf*idf)
```

**Output:**

**Executing the mapper and reducer programs:**

```
C:\Users\admin\Anaconda3\Scripts>hadoop jar C:\\Users\\admin\\hadoop-streaming-2.7.2.jar -mapper "p
ython.exe C:\\Users\\admin\\mapper_tfidf.py" -reducer "python.exe C:\\Users\\admin\\reducer_tfidf.p
y" -input /gutenbergtexts/* -output /user/admin/op1041
20/06/22 11:59:10 INFO Configuration.deprecation: session.id is deprecated. Instead, use dfs.metric
s.session-id
20/06/22 11:59:10 INFO jvm.JvmMetrics: Initializing JVM Metrics with processName=JobTracker, sessio
nId=
```

## Successful execution of the mapper and reducer programs:




## Sample output containing the TFIDF scores for the tokens: