Assignment 2: Word Counting and Analysis of text

Install and initialize

```
!pip install pyspark
from pyspark.sql import SparkSession

# Create a Spark Context
sc = SparkSession.builder.master("local[*]").appName("Test").getOrCreate().sparkContext
```

```
Requirement already satisfied: pyspark in /usr/local/lib/python3.11/dist-packages (3.5.5)
Requirement already satisfied: py4j==0.10.9.7 in /usr/local/lib/python3.11/dist-packages (from pyspark) (0.10.9.7)
```

Word Counting Task

Finish the following tasks

1. Start from the provided text, create an RDD named `words` that contains all words in lower case for future use
2. Use `words` RDD to generate a word counts list in the form of pairs like `[('apple', 32), ('banana', 21), ... ]` and store it in an RDD named `wordCounts`
   - Refer to the example notebook in this module for demonstration
   - You will be using `map`, `flatMap`, `reduceByKey` methods to achieve the goal

Hint: Use the `take` method to show the first 10 records in both `words` and `wordCounts` RDD at the end of each task to show what you get

```
text = '''word count from Wikipedia the free encyclopedia
the word count is the number of words in a document or passage of text Word counting may be needed when a text
is required to stay within certain numbers of words This may particularly be the case in academia legal
proceedings journalism and advertising Word count is commonly used by translators to determine the price for
the translation job Word counts may also be used to calculate measures of readability and to measure typing
and reading speeds usually in words per minute When converting character counts to words a measure of five or
six characters to a word is generally used Contents Details and variations of definition Software In fiction
In non fiction See also References Sources External links Details and variations of definition
This section does not cite any references or sources Please help improve this section by adding citations to
reliable sources Unsourced material may be challenged and removed
Variations in the operational definitions of how to count the words can occur namely what counts as a word and
which words don't count toward the total However especially since the advent of widespread word processing there
is a broad consensus on these operational definitions and hence the bottom line integer result
The consensus is to accept the text segmentation rules generally found in most word processing software including how
word boundaries are determined which depends on how word dividers are defined The first trait of that definition is that
characters such as a regular word space an em space or a tab character is a word divider Usually a hyphen or a slash is
Different word counting programs may give varying results depending on the text segmentation rule
details and on whether words outside the main text such as footnotes endnotes or hidden text) are counted But the behavi
of most major word processing applications is broadly similar However during the era when school assignments were done i
handwriting or with typewriters the rules for these definitions often differed from todays consensus
Most importantly many students were drilled on the rule that certain words don't count usually articles namely a an the
sometimes also others such as conjunctions for example and or but and some prepositions usually to of Hyphenated permane
compounds such as follow up noun or long term adjective were counted as one word To save the time and effort of counting
word by word often a rule of thumb for the average number of words per line was used such as 10 words per line These rul
have fallen by the wayside in the word processing era the word count feature of such software which follows the text
segmentation rules mentioned earlier is now the standard arbiter because it is largely consistent across documents and
applications and because it is fast effortless and costless already included with the application As for which sections
a document count toward the total such as footnotes endnotes abstracts reference lists and bibliographies tables figure
captions hidden text the person in charge teacher client can define their choice and users students workers can simply
select or exclude the elements accordingly and watch the word count automatically update Software Modern web browsers
support word counting via extensions via a JavaScript bookmarklet or a script that is hosted in a website Most word
processors can also count words Unix like systems include a program wc specifically for word counting
As explained earlier different word counting programs may give varying results depending on the text segmentation rule
details The exact number of words often is not a strict requirement thus the variation is acceptable
In fiction Novelist Jane Smiley suggests that length is an important quality of the novel However novels can vary
tremendously in length Smiley lists novels as typically being between and words while National Novel Writing Month
requires its novels to be at least words There are no firm rules for example the boundary between a novella and a novel
is arbitrary and a literary work may be difficult to categorise But while the length of a novel is to a large extent up
```

```
to its writer lengths may also vary by subgenre many chapter books for children start at a length of about words and a
typical mystery novel might be in the to word range while a thriller could be over words
The Science Fiction and Fantasy Writers of America specifies word lengths for each category of its Nebula award categori
Classification  Word count Novel over words Novella to words Novelette to words Short story under words
In non fiction The acceptable length of an academic dissertation varies greatly dependent predominantly on the subject
Numerous American universities limit Ph.D. dissertations to at most words barring special permission for exceeding this
'''
```

```
# Write your code here for task 1
# Example output: ['word', 'count', 'from', 'wikipedia', 'the', 'free', 'encyclopedia', 'the', 'word', 'count']
lines = sc.parallelize(text.split('\n'))
words = lines.flatMap(lambda line: line.split()) \
    .map(lambda word: word.lower())
print("Words:")
print(words.take(10))
```

```
Words:
['word', 'count', 'from', 'wikipedia', 'the', 'free', 'encyclopedia', 'the', 'word', 'count']
```

```
# Write your code here for task 2
# Example output: [('count', 11), ('wikipedia', 1), ('free', 1), ('is', 19), ('of', 25), ('in', 15), ('counting', 6), ('
wordCounts = words.map(lambda word: (word, 1))
wordCounts = wordCounts.reduceByKey(lambda a, b: a + b)
print("\nWord counts generated using reduceByKey, pair like results:")
print(wordCounts.take(10))
```

```

Word counts generated using reduceByKey, pair like results:
[('word', 28), ('from', 2), ('wikipedia', 1), ('encyclopedia', 1), ('number', 3), ('of', 25), ('words', 21), ('text'
```

3. Use the `words` and `wordCounts` RDD

   A. Calculate the average length of words in the `words` RDD

   B. Find the top 5 longest words

   C. Find the top 5 words with the highest counts

Sample run:

```
Avg length: 5.076543209876541
Top longest words: ['bibliographies', 'classification', 'automatically', 'predominantly', 'dissertations']
Top frequent words: [('the', 43), ('word', 28), ('a', 28), ('of', 25), ('and', 23)]
```

```
# Write your code here for task 3
# Average length of words in the words RDD
word_length = words.map(lambda word: len(word))
average_length = word_lengths.mean()
print(f"A. Average length: {average_length}")
# Find the top 5 longest words
top_longest_words = words.top(5, key=len)
print("\nB. Top longest words:")
print(top_longest_words)
# Find the top 5 words with the highest counts
top_5_counts = wordCounts.top(5, key=lambda x: x[1])
print("\nC. Top frequent words:")
print(top_5_counts)
```

```
A. Average length: 5.076543209876541

B. Top longest words:
['bibliographies', 'classification', 'automatically', 'predominantly', 'dissertations']

C. Top frequent words:
[('the', 43), ('word', 28), ('a', 28), ('of', 25), ('and', 23)]
```

Submission

1. After you have finished you code, run and get the correct output.
2. Print the page as PDF and upload the PDF file to Canvas