Task1: Guided solution for PySpark Word Count

Description

This exercise provides hands-on experience with Spark, focusing on one of the foundational big data operations: word count. You'll be counting the frequency of each unique word in a given text.

Techniques Covered:

1. Initializing Spark Context: The prerequisite for any Spark-related operation.
2. Creating and Manipulating RDDs: Learn how to use Resilient Distributed Datasets (RDDs), Spark's core data structure.
3. Transformations and Actions: Master basic Spark operations.
4. Key-Value Pair Operations: Understand operations on paired RDD data.

Overview

This lab will guide you through the Spark setup to RDD creation, transformations, and actions, and finally data collection and visualization. By the end of this lab, you will be able to perform basic data manipulation tasks using PySpark and understand the flow of a Spark application.

Step 1: Initialize the Spark Context

- **Create folder named exercises_one.**
- **Create a Python script named my_first_app.py in a folder called exercises_one.**
- Start by importing the 'SparkContext' class and initializing a new Spark Context.

```
from pyspark import SparkContext
#importing spark context
sc = SparkContext('local', 'ex1_word_count')
```

Explanation:

- Importing SparkContext allows you to connect to a Spark cluster.
- Initializing it with 'local' means you run it on all available cores on your local machine.
- 'ex1_word_count' is simply the name of your Spark application and will appear when you monitor your application on Spark's web UI.

Step 2: Prepare the Input Data

```
text = """This is example of spark application in Python
Python is very common development language and it also one of Spark supported languages
The library of Spark in Python called PySpark
In this example you will implements word count application using PySpark
Good luck!!""".split("\n")
```

Explanation: define a multi-line string and then split it by newline characters to convert it into a list of strings (each line becomes a separate string in the list).

Step 3: Create the RDD

```
rdd = sc.parallelize(text)
```

Explanation: The parallelize function is used to create an RDD from the list of strings. This RDD is now distributed across your Spark cluster, and you can perform transformations and actions on it.

Step 4: Split Lines into Words

```
# splits each line into words and flattens the result
flattened_words = rdd.flatMap(lambda line: line.split(' '))
```

Explanation: flatMap is a transformation operation that applies the lambda function to each element of the RDD. This lambda function splits each line into words and flattens the result, so you get a single RDD consisting of all words.

Step 5: Map Words to Key-Value Pairs

```
words_with_rank = flattened_words.map(lambda word: (word, 1))
```

Explanation: The map function applies a lambda function to each element of the flattened_words RDD. This lambda function takes a word and turns it into a key-value pair **(**word, 1**)**. Each word is treated as a key with a value of 1.

Step 6: Count the Words

```
words_count = words_with_rank.reduceByKey(lambda a, b: a + b)
```

Explanation:

- reduceByKey is a transformation that performs a reduction on a per-key basis.
- For each unique word in the RDD, it sums up the values (which are all 1s), essentially counting the occurrences of each word.

Step 7: Collect and Display the Output

```
for word_count in words_count.collect():
  print(word_count)
```

Explanation:

- The collect function retrieves all elements of your RDD from the Spark cluster and brings them to the local machine, which could be your driver node.
- Then, a simple for-loop is used to print out each word and its corresponding count.

Step 8: Stop the Spark Context

```
sc.stop()
```

Explanation: It's a good practice to stop the Spark Context when you're done to free up resources.

After performing all these steps in sequence, you should have a working PySpark program that counts the frequency of each unique word in a given text.

Step 9 - Full Code Solution

```
# Import the SparkContext class from the pyspark package
from pyspark import SparkContext

# Initialize SparkContext
# 'local' implies that the Spark job will run locally using all available cores
# 'ex1_word_count' is the application name
sc = SparkContext('local[*]', 'ex1_word_count')

# Create an array of strings
text = """This is an example of spark application in Python
Python is a very common development language and it is also one of Spark's supported
languages
The library of Spark in Python called PySpark
In this example you will implement a word count application using PySpark
Good luck!!""".split("\n")

# Create an RDD from the array
rdd = sc.parallelize(text)

# Split each line into words using flatMap transformation
# flatMap flattens the output (as opposed to map which would keep nested structure)
flattened_words = rdd.flatMap(lambda line: line.split(' '))

# Map each word to a tuple (word, 1)
words_with_rank = flattened_words.map(lambda word: (word, 1))

# Reduce by key (word in this case)
# It sums up the values for each key (word), essentially counting occurrences
words_count = words_with_rank.reduceByKey(lambda a, b: a + b)

# Perform action to collect the results to the driver node
# Print each word and its count
for word_count in words_count.collect():
    print(word_count)

# Stop the SparkContext
sc.stop()
```

Run this Python script in your PySpark environment to see the word count output.