Name: Delavin, Mark Nelson A. Lab No. 1

Git Repo/ Colab Link: https://colab.research.google.com/drive/1_05rlFd7lLWFVs2ZV2M0cn0VE9F42v9r?usp=sharing Date: 2025-01-26

Objective

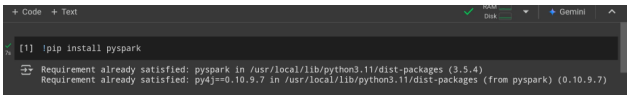
To understand and demonstrate preprocessing techniques using the rich features and functionalities of Apache Spark.

Introduction

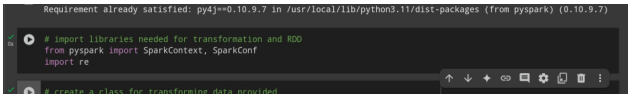
This lab activity will delve into the functionalities of Apache Spark, an open-source unified analytics engine for large-scale data processing. The subject of this lab is on data transformation methods by implementing pipelining on the given data, in preparation for data analysis.

Methodology:

1. Install the pyspark on google collab editor.



2. Import libraries from pyspark. The "SparkConf" is a configuration object in Apache Spark used to set various configuration parameters for a Spark application. It allows you to customize how your Spark application will run. "SparkContext" is the entry point for any Spark functionality. And "re" is Python's built-in regular expression module. It provides support for working with regular expressions, which are powerful tools for pattern matching and text manipulation.



3. Create a class mainly dedicated for the chosen process that we will take. For this code, I chose to provide an array of string in order to determine the number of each words that appeared throughout the array of string. The explanation of each method is in the picture.

```python
# create a class for transforming data provided
class DataPipeline:

    # upon executing the pipeline, this will execute
    # and will allow the other methods to use the spark context
    # within the class by storing the SparkContext
    def __init__(self, sc):
        self.spark_context = sc

    # this method will convert the provided data, e.g. ['Hello', 'wworld', '!']
    # into a data structure that the Spark can process
    def ingest_data(self, raw_data):
        return self.spark_context.parallelize(raw_data)

    # this method will remove the special characters and convert to lowercase
    def clean_data(self, input_rdd):
        return input_rdd.map(
            lambda x: re.sub(r'[^a-zA-Z0-9\s]', '', x).lower()
        )

    # this method will split strings into words and create a pair of values
    # pair: (word, wordcount)
    def transform_data(self, cleaned_rdd):
        """
        Stage 3: Data Transformation
        Split into words, create (word, 1) pairs
        """
```

```python
    def transform_data(self, cleaned_rdd):
        """
        Stage 3: Data Transformation
        Split into words, create (word, 1) pairs
        """
        return cleaned_rdd.flatMap(
            lambda x: x.split()
        ).map(
            lambda word: (word, 1)
        )

    # this method will remove very short and very long words to show simple words
    def filter_data(self, transformed_rdd):
        return transformed_rdd.filter(
            lambda x: 2 < len(x[0]) < 10
        )

    # this method will count the number of frequencies of the word
    def aggregate_data(self, filtered_rdd):
        return filtered_rdd.reduceByKey(
            lambda a, b: a + b
        ).sortBy(
            lambda x: x[1], ascending=False
        )

    # this will call all the method within the class
    def execute_pipeline(self, raw_data):
```

```python
    # this will call all the method within the class
    def execute_pipeline(self, raw_data):

        # This will implement pipelining process, transforming data
        # into valuable information for analysis
        ingested_data = self.ingest_data(raw_data)
        cleaned_data = self.clean_data(ingested_data)
        transformed_data = self.transform_data(cleaned_data)
        filtered_data = self.filter_data(transformed_data)
        final_result = self.aggregate_data(filtered_data)

        # return the processed value
        return final_result
```

4. Create a main method that will create an instance of configuration of the SparkContext and the SparkContext itself. And also will create an instance of the DataPipeline class for processing the data provided by the main method.

```python
# this is the main method of the python program
def main():
    # configure and create spark context instance
    conf = SparkConf().setAppName("Data Processing Pipeline")
    sc = SparkContext(conf=conf)

    # create sample data
    sample_documents = [
        "Hello, world! This is a sample text.",
        "Another example of text processing pipeline.",
        "Data pipelines are powerful for big data processing."
    ]

    try:
        # create instance of DataPipeline and pass the instance of SparkContext
        pipeline = DataPipeline(sc)

        # execute pipeline
        results = pipeline.execute_pipeline(sample_documents)

        # output results
        print("Word Frequencies:")
        for word, count in results.take(5):
            print(f"{word}: {count}")

    finally:
        # stop the SparkContext
        sc.stop()
```

```python
    finally:
        # stop the SparkContext
        sc.stop()

# this will run the main only upon method call
if __name__ == "__main__":
    main()
```

Results and Analysis

For the sake of this acivity, a modified the provided data, this is the modified data:

```
sample_documents = [
    "Machine learning is transforming the way we analyze data and make decisions.",
    "Python programming has become essential in data science and artificial intelligence.",
    "Big data analytics provides insights into complex business challenges.",
    "Deep learning algorithms can recognize patterns in massive datasets.",
    "Cloud computing enables scalable and flexible data processing solutions.",
    "Artificial intelligence is revolutionizing healthcare and medical diagnostics.",
    "Data visualization helps in understanding complex information quickly.",
    "Cybersecurity is crucial in protecting sensitive information from threats.",
    "Blockchain technology offers transparent and secure transaction methods.",
    "Internet of Things connects various devices to create smart ecosystems.",
    "Quantum computing promises to solve complex computational problems.",
    "Renewable energy technologies are key to sustainable environmental solutions.",
    "Augmented reality is changing user experiences in various industries.",
    "Natural language processing enables machines to understand human communication.",
    "Robotics is advancing rapidly in manufacturing and service industries.",
    "Space exploration continues to push the boundaries of human knowledge.",
    "Genetic engineering offers potential breakthroughs in medical treatments.",
    "Autonomous vehicles are transforming transportation and mobility.",
    "Cryptocurrency is reshaping traditional financial systems.",
    "Climate change mitigation requires innovative technological solutions.",
    "5G networks are enabling faster and more reliable communication.",
    "Biotechnology is creating new possibilities in medicine and agriculture.",
    "Virtual reality provides immersive experiences in education and entertainment.",
    "Nanotechnology is opening new frontiers in materials science.",
    "Sustainable development is crucial for future global prosperity."
]
```

And this is the output:

```
Word Frequencies:
and: 10
data: 5
are: 3
complex: 3
solutions: 3
learning: 2
science: 2
artificial: 2
computing: 2
enables: 2
medical: 2
information: 2
reality: 2
human: 2
new: 2
transforming: 2
the: 2
intelligence: 2
provides: 2
processing: 2
crucial: 2
offers: 2
various: 2
sustainable: 2
experiences: 2
industries: 2
communication: 2
machine: 1
analyze: 1
```

```
analyze: 1
decisions: 1
python: 1
become: 1
essential: 1
big: 1
analytics: 1
insights: 1
business: 1
algorithms: 1
patterns: 1
massive: 1
datasets: 1
cloud: 1
revolutionizing: 1
diagnostics: 1
helps: 1
cybersecurity: 1
from: 1
technology: 1
transparent: 1
transaction: 1
```

The output shows the count of words sorted from the most to the least frequencies. This shows that the "and" and "data" word has the most count of them all. Of course the words that has 2 letters are filtered out.

Challenges and Solutions

Challenge: To be honest, while setting up the apache spark like how it shown in the youtube videos. I followed them exactly but there appears an error like a certain files cannot be found at the directory Appdata/local/temp/* and a log of access denied, I tried suggestions from youtube, google, chatgpt, claude, perplexity but it didn't work. At most, the scala works and I can perform complex processing in it using python in the terminal but the integration in vscode can't be perform due to some files that are not found and denial of access. I tried changing the permissions to allow all to the directory that the console is pointing out but it's still the same.

Solution: Becuse of the time I wasted on finding a solution, I jumped to Google Collab, and decided to code there and perform simple pipeline processing.

Conclusion

This lab activity shows the importance of the Pipelining in order for the data to be in a proper format to be used for data analytics. By transforming the data into something else, it is possible to get a valuable information or insights that can be used for other industries such as business, healthcare, technology, etc.