```
{
 "cells": [
  {
   "cell_type": "markdown",
   "id": "08b5135e",
   "metadata": {},
   "source": [
    "# Assignment 6: Streaming Text Analysis using Spark"
   ]
  },
  {
   "cell_type": "markdown",
   "id": "376c238a",
   "metadata": {},
   "source": [
    "## Description:\n",
    "\n",
    "In this assignment, you will be designing and implementing streaming applications for performing basic analytics on textual data. The data stream to be analyzed will be coming from the Twitter stream. Streaming applications may seem complex but understanding how they operate is critical for a data scientist. To allow you to explore a more complex implementation in a short period of time, you are allowed to develop code based on already existing online code snippets and libraries with proper attribution. \n",
    "\n",
    "## Learning Outcome:\n",
    "\n",
    "In the scope of this assignment, you will learn:\n",
    "\n",
    "- How to capture real-time data\n",
    "- How to setup a stream processing pipeline\n",
    "- How to process and get basic insights\n",
    "- How to store the final processing results to a file"
   ]
  },
  {
   "cell_type": "markdown",
   "id": "78821619",
   "metadata": {},
   "source": [
    "## Important Notes:\n",
    "    \n",
    "- You must use the submit command to electronically submit your solution by the due date.\n",
    "- All programs are to be written using Python 3.\n",
```

```
    "- Your programs should be tested on the docker image that we provided before being submitted.\n",
    "- To get full marks, your code must be well-documented."
   ]
  },
  {
   "cell_type": "markdown",
   "id": "0b5d643a",
   "metadata": {},
   "source": [
    "# Part 1. Identifying Trends in Twitter (30%)"
   ]
  },
  {
   "cell_type": "markdown",
   "id": "07a12f3d",
   "metadata": {},
   "source": [
    "Twitter is one of the main online social networks where users post and interact with messages known as \"tweets\". Tweets allow for instant, short, and frequent communication and they have been proved an effective way to communicate news and other timely information. Therefore, a practical use for Twitter's functionality is to be used for identifying trends in real-time. Identifying trends is important for several industries and services, including marketing, customer service, and crisis response.\n",
    "\n",
    "Your task is to design and implement a Twitter streaming application that tracks specific hashtags and reports their popularity (# occurrences) in real-time. In particular, you need to:\n",
    "\n",
    "- Identify 5 related #hashtags (e.g., political parties, companies, product brands, stocks, etc.)\n",
    "- Collect tweets mentioning any of the 5 #hashtags in real-time\n",
    "- Compute the number of occurrences of each of the mentioned hashtags\n",
    "- Plot the results of your analysis in real-time. Alternatively, you can decide to store the results in a file, post-process them as a batch (offline) and create a plot based on the post-process analysis. The results are based on the time window that your application is running (from the time it begins, until it is killed or interrupted/stopped).\n",
    "\n",
    "For the needs of your assignment, you will need to stitch together a number of technologies that can enable the analysis to be performed, including:\n",
    "\n",
```

```
    "### A Twitter client:\n",
    "This is an application that connects to the Twitter service and obtains tweets as they become available. It requires to create your own credentials to access the Twitter APIs. See Appendix A.\n",
    "\n",
    "### Apache Spark Streaming:\n",
    "This is an apache spark streaming application that connects to your twitter client, receives the tweets as a stream, performs real-time processing of the incoming tweets, extracts useful information, and computes the quantities of interest (i.e., number of occurrences of a #hashtag).\n",
    "\n",
    "### Real-time reporting: \n",
    "This is a visualization component that reports through a plot the results computed by the apache spark streaming application in real-time. This can be implemented using AJAX (asynchronous HTTP calls); see the resources of the Appendix B for examples. Alternatively, results can be stored in a file in real-time, post-processed as a batch (offline), and presented as a plot.\n",
    "\n",
    "### Notes:\n",
    "\n",
    "- Several implementation approaches exist for this application. You are encouraged to make assumptions, make decisions and follow the technical path that you feel is more appropriate. You will have a chance to explain your approach during the marking session\n",
    "- Appendix A provides instructions on how to setup a Twitter application\n",
    "- Appendix B provides several online resources related to the assignment"
   ]
  },
  {
   "cell_type": "markdown",
   "id": "e2119425",
   "metadata": {},
   "source": [
    "# Part2: Real-time Sentiment Analysis of Twitter Topics (40%)"
   ]
  },
  {
   "cell_type": "markdown",
   "id": "ac5fe1ca",
   "metadata": {},
   "source": [
    "In the field of computational linguistics and natural language
```

```
processing, sentiment analysis aims to determine the attitude of a speaker, writer, or other subject with respect to some topic or the overall contextual polarity or emotional reaction to a document, interaction, or event. A basic task in sentiment analysis is classifying the polarity of a given text at the document, sentence, or feature/aspect level—whether the expressed opinion in a document, a sentence or an entity feature/aspect is positive, negative, or neutral. For example, consider the following three text inputs:\n",
    "\n",
    "“I love ice cream a lot”\n",
    "\n",
    "“I dislike ice cream a lot”\n",
    "\n",
    "“ice cream is made from milk”\n",
    "\n",
    "One would expect that the polarity of the first is (rather) positive, of the second is (rather) negative and of the third is (rather) neutral. The word “rather” is used here to express subjectivity, since humans not always agree about the polarity of a sentence. We rely on an out-of-the-shelf library to perform sentiment analysis. The analysis will be on the level of a document, where the document is a tweet (i.e., all the words in a single tweet)."
   ]
  },
  {
   "cell_type": "markdown",
   "id": "7efa0c98",
   "metadata": {},
   "source": [
    "Your task is to design and implement a Twitter streaming application that performs sentiment analysis of tweets related to competitive topics and provides a real-time monitoring of the polarity.\n",
    "\n",
    "- Identify 5 competitive topics (e.g., Cybersecurity, Generative AI, AWS services, Microsoft Azure, etc.)\n",
    "- Manually select a set of 10 hashtags that better describe each of the topic identified above\n",
    "- Collect tweets related to the 5 topics in real-time and perform sentiment analysis for each topic\n",
    "- Plot the results of your analysis in real-time. Alternatively, you can decide to store results in a file, post-process them and create a plot based on the post-process analysis\n",
    "\n",
    "### Notes:\n",
    "\n",
    "- The implementation approach for the streaming application should be
```

```
similar to the one followed in Part A\n",
    "- For the sentiment analysis you should employ Python’s Natural Language Toolkit (NLTK) library (similar to A0)\n",
    "- Appendix A provides instructions on how to setup a Twitter application\n",
    "- Appendix B provides several online resources related to the assignment"
   ]
  },
  {
   "cell_type": "markdown",
   "id": "02a2bed0",
   "metadata": {},
   "source": [
    "# Part3: Real-Time Analysis with Kafka and Spark Streaming (30%) "
   ]
  },
  {
   "cell_type": "markdown",
   "id": "f4b921b4",
   "metadata": {},
   "source": [
    "In this section, you need to develop and deploy a stream processing pipeline that captures real-time tweets from the Twitter API, ingests them into Kafka, and processes them using Spark Streaming for sentiment analysis."
   ]
  },
  {
   "cell_type": "markdown",
   "id": "22d9bb8a",
   "metadata": {},
   "source": [
    "## 1. Create a Kafka Cluster on AWS MSK:\n",
    "\n",
    "- Set up an Amazon Managed Streaming for Apache Kafka (MSK) cluster on AWS.\n",
    "- Configure a Kafka topic for incoming tweets.\n",
    "\n",
    "## 2. Develop a Python Kafka Producer:\n",
    "\n",
    "In the first step, you need to write a simple Python script that uses Tweepy to capture tweets from the Twitter API. For simplicity, a sample code has been provided for you in the following: Then you wil use confluent_kafka to produce these tweets to the Kafka topic. To do so, install dependencies:"
```

```
   ]
  },
  {
   "cell_type": "raw",
   "id": "0cc0f809",
   "metadata": {},
   "source": [
    "pip install tweepy confluent_kafka"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "id": "3baf5372",
   "metadata": {},
   "outputs": [],
   "source": [
    "# Example Python Kafka Producer\n",
    "\n",
    "from tweepy import OAuthHandler, Stream, StreamListener\n",
    "from confluent_kafka import Producer\n",
    "import json\n",
    "\n",
    "# Twitter API credentials\n",
    "consumer_key = 'your_consumer_key'\n",
    "consumer_secret = 'your_consumer_secret'\n",
    "access_token = 'your_access_token'\n",
    "access_secret = 'your_access_secret'\n",
    "\n",
    "# Kafka configuration\n",
    "kafka_bootstrap_servers = 'your_kafka_bootstrap_servers'\n",
    "kafka_topic = 'twitter-topic'\n",
    "\n",
    "# Kafka producer setup\n",
    "producer = Producer({'bootstrap.servers': \nkafka_bootstrap_servers})\n",
    "\n",
    "class TwitterStreamListener(StreamListener):\n",
    "    def on_data(self, data):\n",
    "        tweet = json.loads(data)\n",
    "        producer.produce(kafka_topic, key='tweet', \nvalue=json.dumps(tweet))\n",
    "        return True\n",
    "\n",
    "    def on_error(self, status):\n",
    "        print(status)\n",
```

```
    "\n",
    "# Twitter authentication setup\n",
    "auth = OAuthHandler(consumer_key, consumer_secret)\n",
    "auth.set_access_token(access_token, access_secret)\n",
    "\n",
    "# Start capturing tweets and producing to Kafka\n",
    "twitter_stream_listener = TwitterStreamListener()\n",
    "twitter_stream = Stream(auth, twitter_stream_listener)\n",
    "twitter_stream.filter(track=['keywords'])\n"
   ]
  },
  {
   "cell_type": "markdown",
   "id": "af6e851f",
   "metadata": {},
   "source": [
    "## 3. Configure Spark Streaming for Sentiment Analysis:\n",
    "\n",
    "Here's a simple example of a Spark Streaming application in Python using PySpark that reads from Kafka and processes tweets. Please note that you'll need to install the pyspark and pyspark[sql] packages. \n"
   ]
  },
  {
   "cell_type": "raw",
   "id": "3cc5d87f",
   "metadata": {},
   "source": [
    "pip install pyspark"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "id": "213c14b7",
   "metadata": {},
   "outputs": [],
   "source": [
    "from pyspark import SparkContext\n",
    "from pyspark.streaming import StreamingContext\n",
    "from pyspark.streaming.kafka import KafkaUtils\n",
    "import json\n",
    "\n",
    "# Set up Spark context and streaming context\n",
    "sc = SparkContext(appName=\"TwitterStreamingApp\")\n",
    "ssc = StreamingContext(sc, 5)  # 5-second batch interval\n",
```

```
    "\n",
    "# Kafka configuration\n",
    "bootstrap_servers = \"your-kafka-broker-url:9092\"\n",
    "topics = {\"twitter-topic\": 1}  # Dictionary with topic and number of
partitions\n",
    "\n",
    "kafka_params = {\"bootstrap.servers\": bootstrap_servers}\n",
    "\n",
    "# Create a DStream that represents streaming data from Kafka\n",
    "kafka_stream = KafkaUtils.createStream(ssc, bootstrap_servers,
\"spark-streaming-consumer-group\", topics)\n",
    "\n",
    "# Process tweets\n",
    "def process_tweet(tweet):\n",
    "    # Your tweet processing logic here\n",
    "    # Implement sentiment analysis, aggregations, etc.\n",
    "    return tweet\n",
    "\n",
    "# Extract tweets from the Kafka stream\n",
    "tweets = kafka_stream.map(lambda x: json.loads(x[1]))\n",
    "\n",
    "# Process and print each tweet\n",
    "processed_tweets = tweets.map(process_tweet)\n",
    "processed_tweets.pprint()\n",
    "\n",
    "# Start the streaming context\n",
    "ssc.start()\n",
    "ssc.awaitTermination()\n"
   ]
  },
  {
   "cell_type": "markdown",
   "id": "7350d942",
   "metadata": {},
   "source": [
    "In this Python code, we're using PySpark to create a Spark Streaming
application. The KafkaUtils.createStream method is used to create a DStream
representing streaming data from Kafka.\n",
    "\n",
    "Make sure to replace \"your-kafka-broker-url:9092\" with the actual
bootstrap servers of your Kafka cluster and adjust the topics and
processing logic based on your specific requirements.\n",
    "\n",
    "Remember to stop the streaming context when you're done:"
   ]
  },
```

```
  {
   "cell_type": "code",
   "execution_count": null,
   "id": "0f05fcae",
   "metadata": {},
   "outputs": [],
   "source": [
    "ssc.stop(stopSparkContext=True, stopGraceFully=True)\n"
   ]
  },
  {
   "cell_type": "markdown",
   "id": "b2eff7a9",
   "metadata": {},
   "source": [
    "## 4. Deploy on AWS <To be completed>:\n",
    "\n",
    "In this section, we want to deploy the entire application in the cloud. To do so, follow these steps:\n",
    "\n",
    "- Set up an EC2 instance for the Kafka producer. You need to launch an EC2 instance, copy the kafka producer script on it and execute the script. \n",
    "- Deploy the Spark Streaming application on an Amazon EMR cluster with the appropriate configurations. You have to choose how many nodes in the cluster would provide a good cost/performance trade off for you.\n",
    "- Ensure all necessary libraries are installed on the EC2 instance and EMR cluster.\n"
   ]
  },
  {
   "cell_type": "markdown",
   "id": "06445fe9",
   "metadata": {},
   "source": [
    "# Appendix A – Setting Up a Twitter Application & Installing Tweepy"
   ]
  },
  {
   "cell_type": "markdown",
   "id": "d45cc3b4",
   "metadata": {},
   "source": [
    "To start collecting tweets, you need to set up a Twitter application and get credentials that allow you to pull tweets out of the twitter streaming API. Then, you need to develop a Twitter client that connects to
```

```
Twitter and acquires Twitter data. You can do that using Tweepy."
   ]
  },
  {
   "cell_type": "markdown",
   "id": "bafb801d",
   "metadata": {},
   "source": [
    "## Create a Twitter Application and Obtain OAuth Access Keys"
   ]
  },
  {
   "cell_type": "markdown",
   "id": "ffd095ff",
   "metadata": {},
   "source": [
    "Briefly, you need to:\n",
    "- Create a Twitter developer account: 
https://developer.twitter.com/en/apply-for-access\n",
    "- Create a New Application\n",
    "- Fill in your Application Details\n",
    "\n",
    "  - Name: Your app name. It needs to be a unique name across all 
twitter applications\n",
    "  - Description: A short description for your app\n",
    "  - Website: The website address where the app will be hosted. Use a 
placeholder for now\n",
    "  - Callback URL: Ignore this field\n",
    "  \n",
    "- Create Your Access Token\n",
    "- Choose what Access Type You Need (choose 'Read only')\n",
    "- Make a note of your OAuth Settings, including Consumer Key, Consumer 
Secret, OAuth Access Token, OAuth Access Token Secret\n",
    "You should keep these secret, since anyone with the keys, could 
effectively access your Twitter account.\n",
    "Detailed information is provided in the links below (and other 
resources similar online).\n",
    "- Create your own Twitter App: 
https://smashballoon.com/doc/create-your-own-twitter-app/\n",
    "- Twitter Tutorial: Step-by-step guide to making your first request to 
the new Twitter API v2: 
https://developer.twitter.com/en/docs/tutorials/step-by-step-guide-to-makin
g-your-first-request-to-the-twitter-api-v2"
   ]
  },
  {
```

```
   "cell_type": "markdown",
   "id": "152ba474",
   "metadata": {},
   "source": [
    "## Install Tweepy"
   ]
  },
  {
   "cell_type": "markdown",
   "id": "60ee906f",
   "metadata": {},
   "source": [
    "Tweepy is a python library for accessing the Twitter API. You can
install Tweepy using pip:"
   ]
  },
  {
   "cell_type": "raw",
   "id": "b766eb4d",
   "metadata": {},
   "source": [
    "$pip install tweepy"
   ]
  },
  {
   "cell_type": "markdown",
   "id": "60ec829d",
   "metadata": {},
   "source": [
    "You may also use Git to clone the repository directly from Github and
install it manually:"
   ]
  },
  {
   "cell_type": "raw",
   "id": "cd61693e",
   "metadata": {},
   "source": [
    "$git clone https://github.com/tweepy/tweepy.git\n",
    "$cd tweepy\n",
    "$python setup.py install"
   ]
  },
  {
   "cell_type": "markdown",
   "id": "455f675f",
```

```
   "metadata": {},
   "source": [
    "The next step is to use Tweepy to create a Twitter application that
uses your Twitter credentials.\n",
    "More information: https://github.com/tweepy/tweepy"
   ]
  },
  {
   "cell_type": "markdown",
   "id": "9a45b733",
   "metadata": {},
   "source": [
    "# Appendix B – Useful Online Resources and Tutorials"
   ]
  },
  {
   "cell_type": "markdown",
   "id": "a45e2575",
   "metadata": {},
   "source": [
    "Spark Streaming Programming Guide\n",

"http://spark.apache.org/docs/latest/streaming-programming-guide.html\n",
    "\n",
    "Python Streaming Examples\n",

"https://github.com/apache/spark/tree/master/examples/src/main/python/strea
ming\n",
    "\n",
    "An easy-to-use Python library for accessing the Twitter API\n",
    "http://www.tweepy.org/\n",
    "\n",
    "Tutorial on Medium: Apache Spark at a Glance\n",

"https://medium.com/cloudnesil/apache-spark-at-a-glance-7088b9fe5ef5\n",
    "\n",
    "Apache Spark General Tutorial\n",
    "https://www.toptal.com/spark/introduction-to-apache-spark\n",
    "\n",
    "Apache Spark Streaming Tutorial: Identifying Trending Twitter
Hashtags\n",
    "https://www.toptal.com/apache/apache-spark-streaming-twitter\n",
    "\n",
    "Apache Spark Streaming with Twitter (and Python)\n",

"https://www.linkedin.com/pulse/apache-spark-streaming-twitter-python-laure
```

```
nt-weichberger/\n",
    "\n",
    "Twitter-Sentiment-Analysis-Using-Spark-Streaming-And-Kafka\n",

"https://github.com/sridharswamy/Twitter-Sentiment-Analysis-Using-Spark-Str
eaming-And-Kafka\n",
    "\n",
    "Twitter Sentiment Analysis using Python\n",

"https://www.geeksforgeeks.org/twitter-sentiment-analysis-using-python/\n",
    "\n",
    "Sentiment Analysis on Reddit News Headlines with Python’s Natural
Language Toolkit (NLTK)\n",

"https://www.learndatasci.com/tutorials/sentiment-analysis-reddit-headlines
-pythons-nltk"
   ]
  },
  {
   "cell_type": "markdown",
   "id": "55b5087b",
   "metadata": {},
   "source": [
    "# Deliverables"
   ]
  },
  {
   "cell_type": "markdown",
   "id": "722968c6",
   "metadata": {},
   "source": [
    "- A readme.txt file including step-by-step instructions on how to run
each application. \n",
    "- The python scripts (any *.py script) with representative names.\n",
    "- The output files (*.txt) that include example runs of your
application produce.\n",
    "- A well-documented and easily replicable deployment process for part
3 of the assignment.\n"
   ]
  },
  {
   "cell_type": "markdown",
   "id": "e7c3fd2d",
   "metadata": {},
   "source": [
    "# Good Luck!"
```

```
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "id": "f78acebb",
   "metadata": {},
   "outputs": [],
   "source": []
  }
 ],
 "metadata": {
  "kernelspec": {
   "display_name": "Python 3 (ipykernel)",
   "language": "python",
   "name": "python3"
  },
  "language_info": {
   "codemirror_mode": {
    "name": "ipython",
    "version": 3
   },
   "file_extension": ".py",
   "mimetype": "text/x-python",
   "name": "python",
   "nbconvert_exporter": "python",
   "pygments_lexer": "ipython3",
   "version": "3.9.13"
  }
 },
 "nbformat": 4,
 "nbformat_minor": 5
}
```