Assignment 2

Spring 202

Due Date: Sunday March 22, 2020

Instructions

- This first two parts of this assignment involve writing code in the form of Scala classes in a project that can be packaged into a jar file and run on AWS EMR cluster. You should submit the project files and the built jar file. The third part involves writing code in Python that can run on a Spark cluster. You should submit a link to your AWS or Databricks notebook.
- All instructions for compiling and running your code must be placed in the README file.
- You should use a cover sheet, which can be downloaded from here
- You are allowed to work in pairs i.e. a group of two students is allowed. Please write the names of the group members on the cover page.
- **You have a total of 4 free late days for the entire semester. You can use at most 2 days for any one assignment. After four days have been used up, there will be a penalty of 10% for each late day. The submission for this assignment will be closed 2 days after the due date.**
- Please ask all questions on Piazza, not via email.

1 PageRank for Airports

In class, we studied the PageRank algorithm and how it can be used to rank nodes in a graph in order of their importance. Details about this algorithm and its implementation using MapReduce can be found in Chapter 5 of the reference book Data Intensive Text Processing using MapReduce. You can also look at the slides available at `http://lintool.github.io/UMD-courses/bigdata-2015-Spring/slides/session05.pdf`.

Note that you cannot use any external library that automatically computes PageRank, including Spark GraphX.

The dataset for this project will be a graph that shows connections between various airports. This data is available at:Bureau of Transportation website. You would need to do the following to download the data:

1. Go to the url `https://www.transtats.bts.gov/DL_SelectFields.asp?Table_ID=236` and get the data for the latest month available.
2. We are only interested in the following fields to create the graph
   - Origin (Origin Airport Code)
   - Dest (Destination Airport Code)
3. Download and unzip to get a csv file.

Below are the requirements of the project:

1. **Program Arguments**: Your class should have three parameters:
   1. Location of the csv file containing the fields mentioned above
   2. Maximum number of iterations to run
   3. Location of output file
2. You will compute the page rank of each node (airport) based on number of inlinks and outlinks. There may be multiple connections between two airports, and you should consider them independent of each other to compute the number of inlinks and outlinks. For example, if node A is connected to node B with an out-count of 10 and node C with an out-count of 10, then the total number of outlinks for node A would be 20.
3. You have to limit yourself to maximum number of iterations specified by the input parameter number 2.
4. You will use the following equation to compute PageRank:

$$\text{PR}(x) = \alpha \times \frac{1}{N} + (1-\alpha) \times \sum_{1}^{n} \frac{PR(t_i)}{C(t_i)}$$

   where $\alpha = 0.15$ and x is a page with inlinks from $t_1, t_2, \ldots, t_n$, $C(t)$ is the out-degree of t, and N is the total number of links in the graph.

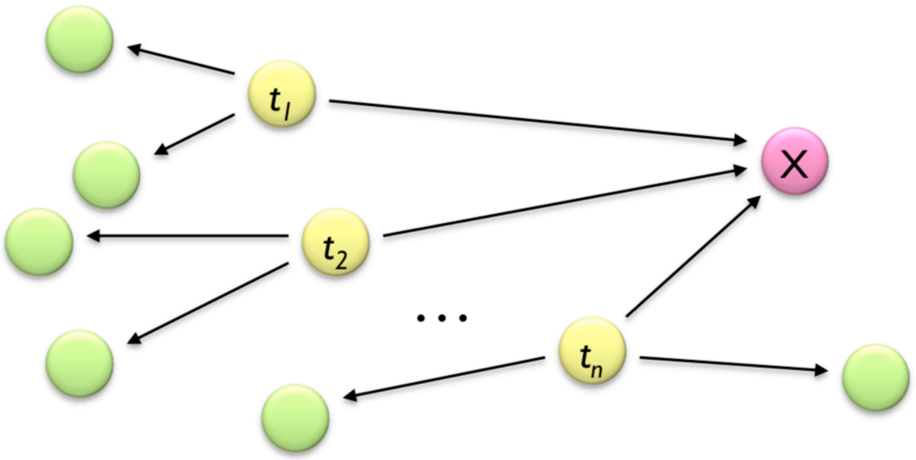

Figure 1: PageRank computation

You can initialize all the PageRank values to be 10.0.

5. You should create a class in the same project as part 1, and name your class as **PageRank**. Your output should be stored in the location specified by the third parameter. The output should contain the airport code and its PageRank, and data should be sorted by the PageRank in a descending order.

Remember to write your code in the form of a Scala class with the above specified number of parameters and include the class in your Spark Scala project. Any files that you use should be hosted on AWS S3.

2 Tweet Processing & Classification using Pipelines

In this part, we will work with a set of Tweets about US airlines and examine their sentiment polarity. More details about the dataset is available at:
https://www.kaggle.com/crowdflower/twitter-airline-sentiment. It is part of the Kaggle competition on Twitter US Airline Sentiment. Our aim is to learn to classify Tweets as either "positive", "neutral", or "negative" by using logistic regression classifier and pipelines for pre-processing and model building.

All this has to be done in a Scala class, which has to be part of a Scala SBT or Maven project. Make sure you have all your dependencies and the class can be run on AWS.

The class will the following parameters -

1. Path of the input file
2. Path of the output file

You need to create a pipeline with the following steps. Again, you need to create a pipeline and not have to run these steps individually. Below are the steps of the project:

1. **Loading**: First step is to load the text file from the path specified in argument 1. After that, you will need to remove rows where the *text* field is *null*.
2. **Pre-Processing**: You will start by creating a pre-processing step with the following stages:
   - **Stop Word Remover**: Remove stop-words from the text column
     Hint: Use the org.apache.spark.ml.feature.StopWordsRemover class.
   - **Tokenizer**: Transform the *text* column into words by breaking down the sentence into words .
     Hint: Use the import org.apache.spark.ml.feature.Tokenizer class.
   - **Term Hashing**: Convert words to term-frequency vectors
     Hint: Use the import org.apache.spark.ml.feature.HashingTF class
   - **Label Conversion**: The label is a string e.g. "Positive", which you need to convert to numeric format
     Hint: Use the import org.apache.spark.ml.feature.StringIndexer class

   Remember that you need to create a pipeline of the above steps and then transform the raw input dataset to a pre-processed dataset.
3. **Model Creation** - You will need to create a logistic regression classification model. You will have to create a **ParameterGridBuilder** for parameter tuning and then use the **CrossValidator** object for finding the best model parameters. An example of this can be seen here: https://spark.apache.org/docs/2.2.0/api/scala/index.html#org.apache.spark.ml.tuning.CrossValidator
4. **Model Testing & Cross Validation**: Next, you will need to train and test your model on the given dataset and output classification evaluation metrics, such as accuracy, etc. You can see details of multi-class evaluation metrics at https://spark.apache.org/docs/2.2.0/mllib-evaluation-metrics.html.

5. **Output**: Finally, you have to write the output the classification metrics to a file whose location is specified by the second argument to the class.

Remember that you have to write your code in the form of a Scala class that should run on AWS. You can specify paths on AWS S3.

3 Image Processing using Spark Deep Learning Pipelines

In this part, you will perform image classification using Spark Deep Learning (DL) pipelines, which is a recent project. More details about it can be seen at the link below:
https://github.com/databricks/spark-deep-learning

Below are the requirements of the project:

1. Since DL pipelines are only available in Python as of now, you will need to use Python. You can use either Databricks or AWS EMR notebook.
2. You are free to use transfer learning
   https://github.com/databricks/spark-deep-learning#transfer-learning
3. You have to select an image classification dataset from Kaggle:
   https://www.kaggle.com/datasets?search=image
   and use that as your input.
4. Please host any input/output data on AWS S3.
5. You have to ensure that the necessary libraries are correctly referenced in your project.
6. This project needs to be done using Apache Spark. You cannot explicitly use other frameworks such as Tensorflow, etc.
7. You are free to use any reasonable assumptions.
8. Please submit a link to your Databricks or AWS notebook. Do not upload a Python project file.