Part two of the project for the Big Data Architecture course (TDT4305 spring 2021)

Submission deadline: 11.30pm, March 21st, 2021

Introduction and context

This document describes the task related to the second part of the project for the course TDT4305. The main task in the second part is to implement a text summarisation algorithm in Spark. The main idea is to extract a few representative terms from a long text that can ideally represent the original text's information meaningfully.

The algorithm described here is similar to an algorithm called TextRank (see here: https://web.eecs.umich.edu/~mihalcea/papers/mihalcea.emnlp04.pdf). The algorithm consists of two major stages. Given a piece of textual information like a web-post, the algorithm constructs a graph of the terms in the document first. In this graph, terms in the input document are nodes, and the edges show the relationship between the terms. Then, it will rank nodes (or terms) based on their PageRank scores. Finally, it will return top-k terms with the most PageRank score as the input document's representative terms.

This part of the project's main goal is to implement the described algorithm to run in an Apache Spark cluster to benefit from Spark's parallelism features. This way, our algorithm will be highly scalable, and we can summerise millions of documents fast and with ease. It is expected that students will complete this part of the project either in teams of two or individually.

Description of the algorithm

This section describes the two stages of the algorithm.

Stage one—constructing the graph of terms

You should take the following steps to construct the term graph for the input document:

1. Turn all the characters to lower case
2. Remove all the punctuations (like '!' and '?') except 'DOT' characters
3. Remove all the symbols (like '$' and '>') and special characters (like 'TAB')
4. Tokenise the output of the previous step (the separator of tokens is the 'WHITESPACE' character); at this stage should have a sequence of tokens
5. Remove the tokens that are smaller than three characters long from the sequence of the tokens
6. Remove all the 'DOT' characters from the start or the end of each token
7. Remove the stopwords from the sequence of tokens (The list of stopwords is available at the end of this document.)

Now, you should have a sequence of terms extracted from the original document. First, create a graph where each unique term from the sequence of terms is a node. Then, use a sliding window of size 5 tokens over the sequence of tokens to find the relationship between two terms. The idea is that if two terms are seen within the same sliding window, there should be an edge between them. (Two terms are related if they co-occur in a window.) The final graph will be unweighted (assume the weight is 1 for each edge) and directed (for each edge (e1 -> e2) we also have an edge such as (e2 -> e1)).

Stage two—calculating the PageRank scores for nodes in the graph of terms

Assuming that we have the unweighted directed graph of terms for the input documents, we can now calculate each node's PageRank score. Run PageRank algorithm on the graph of terms with damping factor (=0.15) and tolerance for error (=0.0001) until it converges. (Normally the usual implementations of PageRank which are iterative should take about 20 to 30 iterations to converge.) You can use the PageRank implementation from GraphX Spark library, so you don't need to implement PageRank algorithm.

The task to implement for this part of the project

You should implement the mentioned algorithm using the API and libraries provided by Apache Spark. Your solution must return (by printing to the STDOUT) top 10 nodes (or terms) from the graph of terms for a given document (for example, document with id = 9) with the highest value of PageRank score. Your solution should work for all the post ids in the posts table (that is data in posts.csv.gz).

Additional information

Please use the list of stopwords that are accessible from this link:
https://gist.github.com/habedi/c7229ee5bd50bf49f5b2bc404366344d

Hard code the list of the stopwords in your code, so you do not need to read the list from a file.

Use Java, Python3, or Scala to develop and implement your solutions. Your team representative should submit a zip file before the deadline that includes these items:

1. A PDF file that briefly describes the logic behind your implementation of the task described in this document and how to run your code
2. Your code for the solution
3. Names of the people in the team

Your solution should be a runnable program or script that accepts a directory path and a post id as its input arguments. Your solution must read the posts.csv.gz file in the directory path it received and use the body of the post it received its post id from the command-line argument. After finishing its computations, your solution should write the results into the STDOUT (that usually is the screen of your monitor).

Your solution (assuming it is a Python3 script) should normally run by executing the command shown below and spit out the result to the screen.

```
spark-submit main.py --input_path /home/users/data --post_id ID_OF_POST
```

If you are using Java, please pack everything inside a runnable Jar file. The rest would be like the example command mentioned above. Please avoid using any special dependency that complicates the ability to run your code. Please use Spark 2.4.x or Spark 2.2.x to develop and run your solutions.

Example of the algorithm's input/output

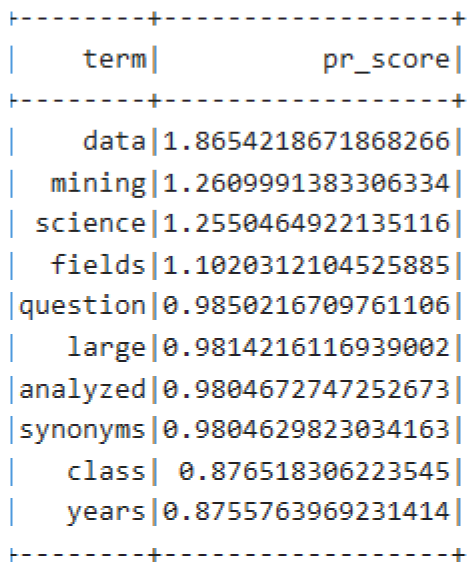
Imaging the input document is the post with id (=14). The text for the input document will be:

<p>I am sure data science as will be discussed in this forum has several synonyms or at least related fields where large data is analysed.</p>&#xA;&#xA;<p>My particular question is in regards to Data Mining. I took a graduate class in Data Mining a few years back. What are the differences between Data Science and Data Mining and in particular what more would I need to look at to become proficient in Data Mining?</p>&#xA;

The (final) sequence of tokens will be:

data, science, discussed, forum, synonyms, fields, large, data, analysed, question, data, mining, graduate, class, data, mining, years, differences, data, science, data, mining, proficient, data, mining

The results should look like this:



```
+--------+------------------+
|    term|          pr_score|
+--------+------------------+
|    data|1.8654218671868266|
|  mining|1.2609991383306334|
| science|1.2550464922135116|
|  fields|1.1020312104525885|
|question|0.9850216709761106|
|   large|0.9814216116939002|
|analyzed|0.9804672747252673|
|synonyms|0.9804629823034163|
|   class| 0.876518306223545|
|   years|0.8755763969231414|
+--------+------------------+
```

The first column (term) shows the top terms and the second one shows the PageRank score of these terms (pr_score).