Assignment 1

Due Date: Mentioned in eLearning

Instructions

- This assignment will involve writing code using PySpark. You can code on your local computer but cannot use local paths. You have to read files from public URLs using wget command. You can submit your code files on eLearning.
- You should use a cover sheet, which can be downloaded from
  http://www.utdallas.edu/~axn112530/cs6350/CS6350_CoverPage.docx
- 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 WordCount for Named Entities

In this part, you will compute the word frequency for named entities in a large file. You are free to use any NLP library that works with Scala/PySpark. Some examples are:
NLTK library https://www.nltk.org/
John Snow Labs https://github.com/JohnSnowLabs/spark-nlp-workshop

The steps of the assignment would be as follows:

1. Select a large text file from the Gutenberg project: https://www.gutenberg.org and note its URL. Use the **wget** command within your PySpark code to read that file. **Do not use local paths.**
2. Use a NLP library to extract only the **named entities** from the text.
3. Write code for a mapreduce program that performs wordcount on the extracted named entities.
4. The output from the map task should be in the form of (key, Value) where key is the named entity, and value is its count (i.e. once every time it occurs)
5. The output from the reducer should be sorted in descending order of count. That is, the named entity that is most frequent should appear at the top.
6. You can display output of your program on the screen

2 Search Engine for Movie Plot Summaries

In this part, we will work with a dataset of movie plot summaries that is available from the Carnegie Movie Summary Corpus site. We are interested in building a search engine for the plot summaries that are available in the file "plot_summaries.txt" that is available under the Dataset link of the above page.

You will use the tf-idf technique studied in class to accomplish the above task. For more details on how to compute tf-idf using MapReduce, see the links below:

1. Good introduction from Coursera Distributed Programming course
2. Chapter 4 of the reference book Data-Intensive Text Processing using MapReduce.

This assignment has to be done using PySpark. As stated previously, do not use any local paths.

Below are the steps of the project:

1. Download the compressed movie summaries file from
   `http://www.cs.cmu.edu/~ark/personas/data/MovieSummaries.tar.gz`
   and then extract the file **plot_summaries.txt** to your current working directory. You also need a file for search terms that should be in the current working directory.
2. You will need to remove stopwords by a method of your choice.
3. You will create a tf-idf for every term and every document (represented by Wikipedia movie ID) using the MapReduce method. **Note that you cannot use a library for computing tf-idf.**
4. Create a search file with search terms. There should be at least five single term queries and 5 multiple term queries. Then read the search terms from the search file and output following:
   (a) **In case of a single term query**: You will output the top 10 documents with the highest tf-idf values for that term.
   (b) **In case of multiple term query**: An example could be "Funny movie with action scenes". In this case, you will need to evaluate *cosine similarity* between the query and all the documents and return top 10 documents having the highest cosine similarity values. **You are allowed to use any Spark based library for computing cosine similarity.**
   You can read more about cosine similarity at the following resources:
   - http://text2vec.org/similarity.html : *Read the cosine similarity section*
   - https://janav.wordpress.com/2013/10/27/tf-idf-and-cosine-similarity/
   - https://courses.cs.washington.edu/courses/cse573/12sp/lectures/17-ir.pdf

   For the search terms entered by the user, you will return the list of **movie names** sorted by their relevance values in descending order. Note again, that you have to return movie **names**, and not movie ID. You would need to use the movie.metadata.tsv file to lookup the movie names.
5. You can display output of your program on the screen

Remember that you have to write your code in PySpark. Do not reference any local paths. You are free to use the wget command to download the files to your current working directory.