CS5344 Lab 3

AY2018/2019 Semester 2

This lab requires you to work with a large set of documents and search for relevant documents. You will need to understand the concepts of RDD, transformation and action in Apache Spark, and implement an algorithm at the RDD level. **This is an individual lab assignment.**

Write a Spark program that finds that top-k relevant documents given a query comprising of set of keywords.

A *document* can be modelled as a vector of words (or terms). Each entry in the vector is a **TF-IDF** value that reflects how important a word is to a document in a collection, computed as **TF-IDF = (1 + log (TF)) * log (N/DF)** where N is total number of documents, TF is the count of the word in a document, and DF is the count of documents having the word. Figure 1 shows a simple example.

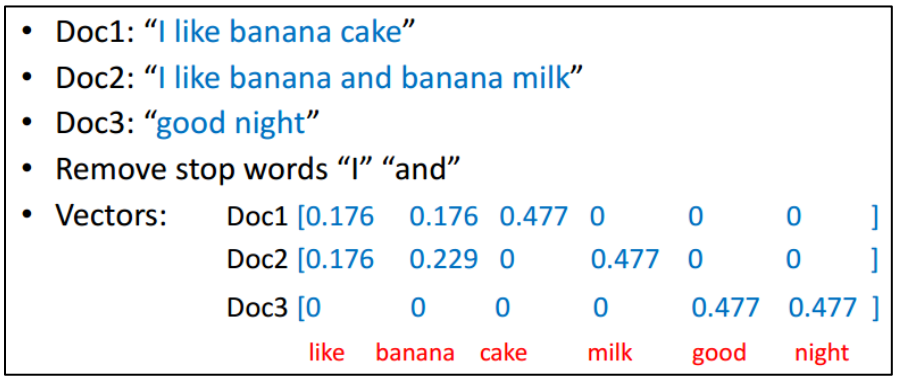

Figure 1. Example of representing documents as vectors.

A *query* can also be represented as a vector where each entry represents a word with a value 1 if the word is in the query, and 0 otherwise. We can compute a *relevance score* for each document *d* to a query *q* based on the based on the cosine similarity of their corresponding vectors V_1 and V_2 and rank the documents with respect to a query:

$$\textbf{relevance (q, d)} = cosine(\overrightarrow{V_1}, \overrightarrow{V_2}) = \frac{\overrightarrow{V_1} \cdot \overrightarrow{V_2}}{||\overrightarrow{V_1}|| \times ||\overrightarrow{V_2}||}$$

Algorithm:

Step 1. Compute term frequency (TF) of every word in a document.
This is similar to the Word Count program in Lab 1.

Step 2. Compute TF-IDF of every word w.r.t a document.
You can use key-value pair RDD and the groupByKey() API.

Step 3. Compute normalized TF-IDF of every word w.r.t. a document.
If the TF-IDF value of *word1* in *doc1* is t_1 and the sum of squares of the TF-IDF of all the words in *doc1* is S, then the normalized TF-IDF value of *word1* is $\frac{t_1}{\sqrt{S}}$.

Step 4. Compute the relevance of each document w.r.t a query.

Step 5. Sort and get top-k documents.

Input: (a) set of documents (in "datafiles" folder),
(b) set of keywords for a query (in *query.txt*),
(c) stopwords to remove (in *stopwords.txt*).

Output: One line per review in the following format: <*docID*> <*relevance score*>
The output should be sorted in descending order of the relevance of the documents to the query.

Deliverables: Upload your executable Spark program (with documentation in the code) to the Lab3 folder in IVLE.

Important Notes:

- Specify the configuration of your program clearly.
  - For Python program, specify the python version along with the supporting packages used.
  - For Java program, provide the pom.xml configuration file and specify the directory structure of your source code files.
- Your code should be executable either on the virtual machine configuration given in Lab 1 or on stand-alone Spark configuration.

References:

- https://spark.apache.org/docs/2.2.0/rdd-programming-guide.html#transformations
- https://spark.apache.org/examples.html