CS5344 Lab 1

AY2018/2019 Semester 1

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.**

Users of an online shopping website would like to find the relevant reviews of items they are interested in. **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 follows:

$$\textbf{TF-IDF} = (1 + \log(\text{TF})) * \log(\text{N/DF})$$

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.

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 as follows:

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

With this, we can rank the documents with respect to a query.

Algorithm: There are 5 main steps.

Step 1. Compute term frequency (TF) of every word in a document.
This is similar to the Word Count program in Lab0.

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

Step 3. Compute normalized TF-IDF of every word w.r.t. a document.
Suppose 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:

- Set of keywords for a query (in *query.txt*)
- Stopwords to remove (in *stopwords.txt*)
- Set of documents. We will use a subset of the Amazon product data[1] to construct the set of documents. Download the product reviews dataset from the link: https://www.dropbox.com/s/w9i0vfirn56sfct/bookreviews.json?dl=0

Construct a document from each review in *bookreviews.json* by combining the information in “reviewText” and “summary”. The document ID is obtained by combining the “reviewerID” and product ID “asin”. You should have as many documents as there are reviews in the file *reviews.json.* Note that stop-words should be removed from the documents constructed.

Output: The output should contain one line per relevant review in the following format:

<document ID> <relevance score>

[1] http://jmcauley.ucsd.edu/data/amazon/links.html

The output should be sorted in descending order of the relevance of the documents to the query. You need to only give the top-20 most relevant reviews.

Deliverables:

Upload your Spark program (with documentation within the code) to the Lab1 folder in IVLE

References:

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