Final Project - Big Data Technologies

Tel Aviv University

Coller school of management

May 26, 2017

In this project your task is to calculate the **TF-IDF** statistics for each document in a collection of documents, in a Spark environment, using the Python language.

1 Background:

TF-IDF is a simple but useful method frequently used in text classification.

For each word (term) in a document, the TF-IDF score reflects it's importance in the document, while accounting for the frequency of the word in the whole collection of documents.

1.1 TF:

A word's *term frequency* is defined to be the ratio between it's frequency in a certain document and it's frequency in all of the documents together:

$$\mathbf{TF(t)} = \frac{\textbf{Number of times word t appears in document}}{\textbf{Total words in document}} \tag{1}$$

1.2 IDF:

A word's *inverse document frequency* is defined to be the ratio between the number of documents in the collection, and the total number of documents including the word. The more documents contain the word, the closer this ratio is to one, and taking it's logarithm brings us closer to zero - so the score that indicates a word's importance across all documents diminishes as it appears in more and more documents.

$$\mathbf{IDF(t)} = \mathbf{ln}\frac{\textbf{Total number of documents}}{\textbf{Total documents with word t}} \tag{2}$$

Since words like "a", "is", "the" will have high TF ranking, but they will have high ranking across all documents - so ultimately they are not useful to classify the document.

The resulting TF-IDF statistic for a term t is a single score:

$$\mathbf{TF.IDF(t) = TF(t) * IDF(t)} \tag{3}$$

2 The project:

The end goal of the project is to compute both metrics for a collection of documents (corpus), that is:

1. Input: list of documents.
2. Output: $TF.IDF(t)$.

The following schema is a suggested structure for the project, however - **you can use your creativity** and implement the task in other ways.

2.1 Code structure:

1. TF function - takes as input a document and compute the frequency of the words (think how to use the word count example from class and aggregate function to compute the frequency of the words). The function returns a (key,value) pair where the key is a word and the value is the corresponding TF value,
2. IDF function - takes the corpus as input and for each word in the corpus returns a (key,value) pair - the key is the word, the value is the IDF ratio. Here you'll have to think about:
   a) Collecting all the words in the corpus.
   b) Removing duplicate words from that collection.
   c) Counting the number of documents in which a word appears.
3. External function - takes as input the corpus and returns the $TF.IDF$ value for each word as a list.

As mentioned before - you can use a different structure if you think it helps.

2.2 Project explanation:

In addition to the coding task you'll submit a detailed explanation of the structure: why you've used a given function (map vs flatmap for example), what are the (key,value) pairs in the process and **for one component in your code** a linage graph of the process.

In the end write a 2 paragraph section explaining the use of such statistics in text classification.

3 General guidance:

In the following link you'll find more details on TF.IDF:

- Wikipedia page
- Simple explanation of the algorithm

3.1 Grading

The grading will be based on the following:

1. The execution of the method (that is, your code should return correct results for the input)
2. Smart use of Pyspark functions
3. Correct coding - partitioning the code to functions.

You may share thought and ideas with class mates **but any copying** will be harshly handled, both groups will receive 0 and the group members will face Vaadat Mishmamt.