Boston University Metropolitan College

Assignment 4
Big Data Analytics

MET CS777

Faculty - Farshid Alizadeh-Shabdiz, PhD, MBA

1 Description

In this assignment you will implement k-nearest neighbor classifier (KNN classifier) to classify text documents. For example, given a search text ”How many goals did Vancouver score last year?” the algorithm searches all the documents corpus (corpus: large and structural text) and returns the top K similar documents.

The TF-IDF (Term Frequency - Inverse Document Frequency) is used as the similarity/distance measure between two document/texts.

In the first step, the top 20k English words of the corpus will be selected, then the TF-IDF matrix of a text corpus get computed, which is used to find similarity between the texts.

Wikipedia Dataset

In this assignment, the Wikipedia data set is used. The entire Wikipedia data set has been downloaded from (https://dumps.wikimedia.org) and stored in a large file.

Each Wikipedia Page is a document and have a unique document ID and a specific URL. For example,

- docID 418348
- URL https://en.wikipedia.org/wiki?curid=418348

Dataset and how to retrieve them

Small Data Set - Wikipedia Pages

You can find a small data set (Only 1000 Wikipedia pages) at Google Storage:

gs://metcs777-fa/WikipediaPagesOneDocPerLine1000LinesSmall.txt

Small data set Wikipedia Page Categories for the above small data set can be found here:

gs://metcs777-fa/wiki-categorylinks-small.csv.bz2

Large Data Set

Large data set consists of 1 million pages (2.2 GB) and can be found here:
gs://metcs777-fa/WikipediaPagesOneDocPerLine1m.txt

Categories of the large data of Wikipedia can be found here:

gs://metcs777/wiki-categorylinks.csv.bz2

Data Description:

Each line is a single document in a pseudo XML format.

```
<doc id="431953" url="https://en.wikipedia.org/wiki?curid=431953"
 title="Doug Harvey (ice hockey)">Doug Harvey (ice hockey)Douglas Norman Harvey
    ...
</doc>
```

Assignment Tasks

Task 1 Generate a 20K dictionary (10 points)

1.1 Using Wikipedia pages, find the top 20,000 English words, save them in an array, and sort them based on the frequency of the occurrence.

1.2 As a result, a dictionary has been generated that contains the top 20K most frequent words in the corpus. Next go over each Wikipedia document and check if the words appear in the Top 20K words. At the end, produce an **RDD** that includes the docID as key and a Numpy array for the position of each word in the top 20K dictionary.

(docID, [dictionaryPos1, dictionaryPos2, dictionaryPos3...])

Task 2 - Create the TF-IDF Array (20 Points)

After having the top 20K words we want to create a large array that its columns are the words of the dictionary with number of occurrences of each word and the rows are documents.

The first step is calculating the "Term Frequency", TF (x, w), vector for each document as follows:

$$TF(x,w) = \frac{x[w]}{\sum_{w'} x[w']}$$

"Term Frequency" is an indication of the number of times a term occurs in a document. Numerator is number of occurrences of a word, and the denominator is the sum of all the words of the document.

Next, calculate "Inverse Document Frequency" for all the documents and finally calculate TF-IDF(w) and create TF-IDF matrix of the corpus:

$$IDF(w) = log\left(\frac{size\ of\ corpuse}{\sum_{"x\ in\ corpus"} 1\ if\ \ (x[w]\geq 1),0\ otherwise}\right)$$

$$TF - IDF(w) = TF(x,w) \times IDF(w)$$

Note that the "size of corpus" is total number of documents (numerator).

To learn more about TF-IDF see the Wikipedia page:

https://en.wikipedia.org/wiki/Tf-idf

Task 3 - Implement the getPrediction function (30 Points)

Finally, implement the function getPrediction(textInput, k), which will predict the membership of the textInput to the top 20 closest documents, and the list of top categories.

You should use the cosine similarity to calculate the distances.

$$Cos(x_1, x_2) = \sum_{i=1}^{d}(x_1 \times x_2)$$

Task 4 – Implement the code using Dataframes (30 points)

Implement the complete code in Dataframe and print out the results of the task 3 using dataframes in pyspark. From the beginning of your code to the end of your kNN implementation you are allowed to use spark dataframe and python (including python libraries like numpy). You are not allowed to use RDDs.

Task 5 - Removing Stop Words and Do Stemming (10 points)

Task 5.1 - Remove Stop Words (5 point)

Describe if removing the English Stop words (most common words like ”a, the, is, are, i, you, ...”) would change the final kNN results.

Does your result change significantly after removing the stop words? Why? Provide reasons.

You do not need to implement this task.

Task 5.2 – Considering English word stemming (5 point)

We can stem the words [”game”,”gaming”,”gamed”,”games”] to their root word ”game”.

Does stemming change your result significantly? Why? Provide reasons.

You can learn more about stemming at:
https://en.wikipedia.org/wiki/Stemming

You do not need to implement this task.

Submission Format

Please zip all of your code and your document (use .zip only, please!), or else attach each piece of code as well as your document to your submission individually.

Academic Misconduct Regarding Programming

In a programming class like our class, there is sometimes a very fine line between ”cheating” and acceptable and beneficial interaction between peers. Thus, it is very important that you fully understand what is and what is not allowed in terms of collaboration with your classmates. We want to be 100% precise, so that there can be no confusion.

The rule on collaboration and communication with your classmates is very simple: you cannot transmit or receive code from or to anyone in the class in any way—visually (by showing someone your code), electronically (by emailing, posting, or otherwise sending someone your code), verbally (by reading code to someone) or in any other way we have not yet imagined. Any other collaboration is acceptable.

The rule on collaboration and communication with people who are not your classmates (or your TAs or instructor) is also very simple: it is not allowed in any way, period. This disallows (for example) posting any questions of any nature to programming forums such as StackOverflow. As far as going to the web and using Google, we will apply the ”two line rule”. Go to any web page you like and do any search that you like. But you cannot take more than two lines of code from an external resource and actually include it in your assignment in any form. Note that changing variable names or otherwise transforming or obfuscating code you found on the web does not render the ”two line rule” inapplicable. It is still a violation to obtain more than two lines of code from an external resource and turn it in, whatever you do to those two lines after you first obtain them.

Furthermore, you should cite your sources. Add a comment to your code that includes the URL(s) that you consulted when constructing your solution. This turns out to be very helpful when you're looking at something you wrote a while ago and you need to remind yourself what you were thinking.