Description

The goal of this assignment is to implement a kNN classifier to classify text documents. The implementation will be in PySpark.

There are three subtasks: (1) data preparation, (2) classification, and (3) evaluation.

The data set is the widely-used "20 newsgroups" data set. A newsgroup post is like an old-school blog post, and this data set has 19,997 such posts from 20 different categories, according to where the blog post was made. The 20 categories are listed in the file `DataCategories.txt`. The format of the text data can be referred to `DataSample.txt`; the category name can be extracted from the name of the document. For example, the document with identifier `20_newsgroups/comp.graphics/37261` is from the `comp.graphics` category. The document with the identifier `20_newsgroups/sci.med/59082` is from the `sci.med` category. The data file has one line per document—39 MB of text.

1. Data Preparation

Write Spark code that builds a dictionary that includes the 20,000 most frequent words in the training corpus. The words in the dictionary should be ordered based upon the relative frequency of the word. For example, the 0th word should be the most frequent word, and the 19, 999th word should be the least frequent word in the dictionary. In the case of ties, order by the values of the words themselves, from lowest to greatest.

Then use this dictionary to create an RDD where each document is represented as one entry in the RDD. Specifically, the key is the document identifier (like `20_newsgroups/comp.graphics/37261`) and the value is a NumPy array with 20,000 entries where the ith entry in the array is the number of times that the ith word in the dictionary appears in the document.

2. Classification

It is often difficult to classify documents accurately using raw count vectors. Thus, the next task is to write some more Spark code that converts each of those 19,997 count vectors to TF-IDF vectors ("term frequency/inverse document frequency vectors"). The ith entry in a TF-IDF vector for document d is computed as:

$$TF(i,d) \times IDF(i)$$

Where $TF(i,d)$ is:

$$\frac{\text{Number of occurences of word } i \text{ in } d}{\text{Total number of words in } d}$$

Note that the "Total number of words" is not the number of distinct words. The "total number of words" in "Today is a great day today" is six. And the $IDF(i)$ is:

$$\log \frac{\text{Size of corpus (number of docs)}}{\text{Number of documents having word } i}$$

3. Evaluation

Next, build a kNN classifier, embodied by the Python function `predictLabel`. This function will take as input a text string and a number k, and then output the name of one of the 20 newsgroups. This name is the newsgroup that the classifier thinks that the text string is "closest" to. It is computed using the classical kNN algorithm. This algorithm first converts the input string into a TF-IDF vector (using the dictionary and count information computed over the original corpus). It then finds the k documents in the corpus that are "closest" to the query vector (where distance is computed using the L2 norm), and returns the newsgroup label that is most frequent in those top k. Ties go to the label with the closest corpus document.

Finally, test it using `test.py` (each is an except from a Wikipedia article, chosen to match one of the 20 newsgroups).