Homework 10:
Supervised Learning: k-NN Classification

Robert Litschko*
Symbolische Programmiersprache

Due: Thursday January 19, 2022, 12:00 (noon)

In this exercise you will:

- Implement a K-Nearest Neighbours Classifier.

Exercise 1: K nearest neighbours [6 points]

Train a kNN classifier using the training set of newsgroups data and classify test documents (test set) into one of the 20 newsgroups. You can find the dataset *20news-bydate* in the `data/` folder of your project. After unzipping the file *20news-bydate.tar.gz* you can find train and test folders consisting of several newsgroups folders and their documents. Take a look at the data and the file `hw10_knn/classification.py`. The classes `TextDocument` and `DocumentCollection` should be familiar: they are almost identical with the classes we implemented in Homework 8 (but watch out for some minor differences). In this exercise you will have to complete some methods of the class `KNNClassifier` to make the classification work.

This homework will be graded using unit tests by running:

```
python3 -m unittest -v hw10_knn/test_knn.py
```

Implement the following methods (to get some help for the implementation, have a look in the unittest to see how the methods and their arguments are called):

1. `calculate_similarities(self,vecTestDoc,vectorsOfTrainDocs)`: calculate and return a list of (cosine) similarities between the vector of the test document and the vectors of the other train documents; do not forget to label them. Return something like this: [(similarity1, label1), (similarity2, label2), ...] **[1 point]**

2. `order_nearest_to_farthest(self,similarities)`: order the pairs of (similarity, label) from most similar to less similar. **[1 point]**

*Credit: Exercises are based on previous iterations from Katerina Kalouli.

3. `labels_k_closest(self,sorted_similarities)`: find k closest labels (return only the labels, not the similarities; k can be found as an attribute of the class: `self.n_neighbors`). **[1 point]**

4. `choose_one(self, labels)`. This method should return unique neighbor (label) from the given k nearest neighbors (labels). If there is a unique winner, return it, otherwise, reduce the number of k (the size of the labels list) and search again. Use recursion. **[1 point]**

5. `classify(self,test_file)`. This method should classify the given test document. Use the methods you have implemented before: convert the document to a vector, calculate the similarities of this document vector to the rest of the documents (`self.vectorsOfDoc_collection`), sort the similarities, get the k nearest neighbors and return the one of them being the final label. **[1 point]**

6. `get_accuracy(self,gold,predicted)`. This method should return the accuracy: proportion of correctly classified test documents over the whole test set of documents. **[1 point]**