Lab session 3 : tuning a K-NN classifier

1 Tuning

This is a follow-up of lab session 2. The objective is now to implement the "tuning" of certain hyperparameters of your K-NN.
The hyperparameters to test are:

- K : since your code shares a lot of computation across different K values, it is not computationally expensive to test a large range of K values (e.g. from 1 to 300 or more)
- cos_or_dist boolean : whether to use cosine similarity or distance (caution sorting directions are different between both)
- use_weight boolean : whether or not to weight the neighbors when performing the majority vote among classes of the K nearest neighbors
- **to go further** : use_idf boolean : whether or not to use **TF.IDF** values instead of **TF** in the BOW vectors
    - TF.IDF weighting was developed to weight tokens for information retrieval, in the 80's
    - TF means "term frequency" => TF(w,d) = the number of occurrences of a term w in document d, normalized or not
    - IDF means "inverse document frequency" :
        - computed within a set of documents. Suppose you have T documents, and we note nb_doc(w) the number of DOCUMENTs in which w occurs at least once
        - TF(w) = ln(T / nb_doc(w))

You will add options to your main K-NN program for the booleans above.
Then add an option to perform or not the **tuning** of hyperparameters, namely to launch the K-NN with each hyperparameter combination **on the dev set** (which is called doing a "**grid search**").

The combination working best on the dev set is supposed to be used to evaluate the accuracy on another set of examples, usually called the **test set**. This provides a more accurate evaluation of how the tuned classifier behaves on unseen examples.
A larger dataset is provided: reuters.train / reuters.mysplit.dev / reuters.mysplit.test[1]

[1] Only a train and test split was provided in the original reuters21578 dataset. I did a further split of the original test into dev and test.

2 To go further : Vizualization

In order to ease the study of how hyperparameters impact performance, implement plots showing the accuracy as a function of K, for all the various combinations of the other hyperparameters:

- with the first 2 booleans only => we get 4 accuracy curves as a function of K
- or 8 curves if you also implemented the use_idf boolean

Plotting is done with the matplotlib python library. It is even easier using pandas DataFrames, and using the pandas.DataFrame.plot method, cf. documentation and examples https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.plot.html