Elise Michon 2014742018 CMPE 561

Assignment 1 - Authorship Recognition using Naive Bayes
https://github.com/hajoki/EliseMichon_HW1.git

I - Pre-processing

Pre-processing is done in the script ***1_set_creation.py***. This script takes as an argument the directory path to the dataset. We assumed it is the path of a folder that contains the directory "69yazar" with a subdirectory "raw_texts", which means we considered that the files would be in the following locations: *indicated path/69yazar/raw_texts/author1/* etc.

In this architecture, the script creates two folders:
indicated path/69yazar/training/ and *indicated path/69yazar/test/*,
and in each of them a subfolder for each author:
indicated path/69yazar/training/author1/ etc. and *indicated path/69yazar/test/author1/* etc.

Then, the script randomly selects 60% of the articles of each author and copies them in the training subdirectory for this author, and the remaining articles in the test subdirectory for this author. We chose to copy the files to be able to test several such partitions.

The number of documents in each class in the training and test sets are reported below:

```
abbasGuclu
Total: 15, training: 9, test: 6
abdullahAymaz
Total: 10, training: 6, test: 4
ahmetAltan
Total: 10, training: 6, test: 4
ahmetHakan
Total: 15, training: 9, test: 6
aliBulac
Total: 10, training: 6, test: 4
atillaDorsay
Total: 15, training: 9, test: 6
ayseArman
Total: 15, training: 9, test: 6
balcicekPamir
Total: 30, training: 18, test: 12
bekirCoskun
Total: 10, training: 6, test: 4
bulentKorucu
Total: 10, training: 6, test: 4
canDundar
Total: 10, training: 6, test: 4
cemilErtem
Total: 10, training: 6, test: 4
cemSuer
Total: 10, training: 6, test: 4
cengizCandar
Total: 10, training: 6, test: 4
cetinAltan
Total: 30, training: 18, test: 12
deryaSazak
Total: 10, training: 6, test: 4
doganHizlan
Total: 15, training: 9, test: 6
eceTemelkuran
Total: 10, training: 6, test: 4
ekremDumanli
Total: 30, training: 18, test: 12
elifSafak
Total: 10, training: 6, test: 4
emreAkoz
Total: 30, training: 18, test: 12
emreKongar
Total: 15, training: 9, test: 6
ergunBabahan
Total: 10, training: 6, test: 4
ertugrulOzkok
Total: 10, training: 6, test: 4
fatihAltayli
Total: 30, training: 18, test: 12
fehmiKoru
Total: 10, training: 6, test: 4
fikretBila
Total: 10, training: 6, test: 4
fundaOzkan
Total: 10, training: 6, test: 4
gulseBirsel
Total: 10, training: 6, test: 4
guneriCivaoglu
Total: 10, training: 6, test: 4
hakkiDevrim
Total: 15, training: 9, test: 6
hasanCemal
Total: 10, training: 6, test: 4
hasanPulur
Total: 10, training: 6, test: 4
hasmetBabaoglu
Total: 10, training: 6, test: 4
hekimogluIsmail
Total: 10, training: 6, test: 4
```

```
hincalUluc
Total: 10, training: 6, test: 4
huseyinGulerce
Total: 10, training: 6, test: 4
ismetBerkan
Total: 10, training: 6, test: 4
mahfiEgilmez
Total: 15, training: 9, test: 6
mehmetaliBirand
Total: 10, training: 6, test: 4
mehmetBarlas
Total: 30, training: 18, test: 12
mehmetOz
Total: 15, training: 9, test: 6
mehmetTezkan
Total: 10, training: 6, test: 4
melihAsik
Total: 10, training: 6, test: 4
mumtazerTurkone
Total: 10, training: 6, test: 4
muratBardakci
Total: 10, training: 6, test: 4
muratBelge
Total: 10, training: 6, test: 4
muratYetkin
Total: 10, training: 6, test: 4
nazliIlicak
Total: 10, training: 6, test: 4
nihalKaraca
Total: 10, training: 6, test: 4
nurayMert
Total: 10, training: 6, test: 4
omerUrundul
Total: 10, training: 6, test: 4
oralCalislar
Total: 10, training: 6, test: 4
raufTamer
Total: 10, training: 6, test: 4
rehaMuhtar
Total: 10, training: 6, test: 4
ridvanDilmen
Total: 15, training: 9, test: 6
ruhatMengi
Total: 15, training: 9, test: 6
samikohen
Total: 30, training: 18, test: 12
savasAy
Total: 10, training: 6, test: 4
serpilYilmaz
Total: 10, training: 6, test: 4
tahaAkyol
Total: 30, training: 18, test: 12
tamerKorkmaz
Total: 10, training: 6, test: 4
tarhanErdem
Total: 10, training: 6, test: 4
umurTalu
Total: 10, training: 6, test: 4
yaseminCongar
Total: 10, training: 6, test: 4
yigitBulut
Total: 10, training: 6, test: 4
yilmazOzdil
Total: 10, training: 6, test: 4
yukselAytug
Total: 15, training: 9, test: 6
zekiCol
Total: 10, training: 6, test: 4
```

II - Tokenization

The tokenization have been placed in a function ***my_tokenizer(training_dir_path, test_dir_path)*** in the main script ***2_author_classification.py***. This might not have been the most judicious choice but it enabled this module to be called easily.

Despite an attempt to set the locale variable to Turkish, the text was displayed in Eclipse in the maternal language of the computer French, as is attested by the absence of problems with 'ç' or 'ü' which are possible in French and not English, but the presence of problems with 'ğ' or 'ş' which are possible in Turkish and not in French. To remedy this situation, we did simple replacements of the problematic characters.

The next step consisted in suppressing all upper comas (« sarayi'ni » become « sarayi ») and all the numbers, and to convert all words to lower case. We then found all the sequences of 1 or more alphabetical characters. We note that in this case \[a-z]+ would have not led to the accurate result because it does not take into account the specifically Turkish characters.

III – Naive Bayes classifier

The classifier is composed of two functions **learn(training_dir_path, test_dir_path)** and **classify_simple(**`training_dir_path`**,test_dir_path)**.

The function **learn(training_dir_path, test_dir_path)** first extracts the number of documents in each class, as well as the global vocabulary.
With a first loop, it stores the number of documents in each class, and by tokenizing each document, it stores the words in vocabulary. Then it concatenate all articles of one author in one megadocument and calculate the log of the probability of each class.
With a second loop, it learns the log of conditional probabilities for each word given each class, by tokenizing these megadocuments. We note that this second tokenization does not seem optimal, but is mainly due to the lack of experience of the programmer with the storage structures of Python, who used only lists.

The function **classify_simple**(training_dir_path, test_dir_path) learns the variables returned by the previous function learn(), and use them to go over all the test files, tokenize them and calculate the appropiate sum of logarithms to be maximized, using Laplace smoother. It assigns an author to each test file and fills the confusion matrix.

Unfortunately, an error must remain in the implementation since the result of the application of this function is the attribution of all test files to the same class, the one which the higher class probability, even when training_dir_path is given as test directory path.

The function **output(confusion_matrix, authors)** computes the micro and macroaveraged precison, recall and f1-scores.
This fucntion has proven robust with confusion matrices given by the programmer but gives an error in this case because all test files being attributed to only a class, the total numbers of files attributed to other classes are null and consequently cannot be used as dividers.

IV – Extra feature

The extra feature chosen was the number of words by article for an author. By analogy with the vocabulary, a first attempt has consisted in storing all the possible number of words by article in the whole training set,

As the Naive Bayes classifier with words as features did not yield observable results, last efforts were put in trying to make it function and this attempt of extra feature has not be entirely fixed. It is still proposed in comment with the functions **learn_extra(training_dir_path, test_dir_path)** and **classify_extra(**`training_dir_path`**,test_dir_path)**.

V – Conclusion

This homework was a good first try on Python for my part but feedback would be most appreciated because my implementation certainly suffers from lacks of Python culture, from the correct setting of the locale variable, to the best way to structure programs or to store and easily access data.
For example, I am quite familiar with R or Matlab where it is easy to have labels to call rows and columns of a matrix and I missed it several times in this program, so it might be that I don't know the implementation for it in Python, or that I need to change my reasoning to another programming logic.