UNIVERSITÄT BONN INFORMATIK

CAISA Lab
https://caisa-lab.github.io/

Qualification Test

Natural Language Processing
(Python Programming Task)

The objective of this test is to assess your proficiency in Natural Language Processing (NLP) and related skills required for conducting research and completing a thesis in the CAISA Lab.

In the ZIP file (**CAISA-Thesis.zip**), you have been given a dataset of text documents as well as a Python file (**classifier.py**). The training data is given in **'train.tsv'** and the test data in **'test.tsv'**. Both files contain already tokenized data. The two classes are 's' and 'l'.

You are required to develop a Python program that classifies documents using two different models: logistic regression and multi-layer perceptron from scikit-learn.

The current version of the code uses the following features:

- TF-IDF
- Average number of characters per word
- Average number of words per sentence
- Number of sentences per document

However, the code still contains several errors and flaws. Your task is to fix and clean up the code and implement things that are missing (marked by TODO).

The code should follow a good and readable style including comments. Also, feel free to add additional files and organize the code better.

Feel free to use any additional libraries; however, implement the computation of the TF-IDF feature by yourself (e.g., don't use the TfidfVectorizer from scikit-learn). You may use collections.

For computing TF-IDF, please also provide the formulas you used for computation in your report and explain why you chose to compute it as you did.

There is no need to add further functions. However, feel free to remove any functions you deem unnecessary.

Your submission should be a ZIP file containing:

- A runnable Python file (NOT IPYNB!)
- Provide comments for the changes in the code you make and explain why.
- The predictions of your models
- A report including different evaluation metrics of your models, interpret the results, and analyze which feature you think was the most helpful and why.
- Can you think of a feature that achieves 100% accuracy?