Assignment 1 - ANLP

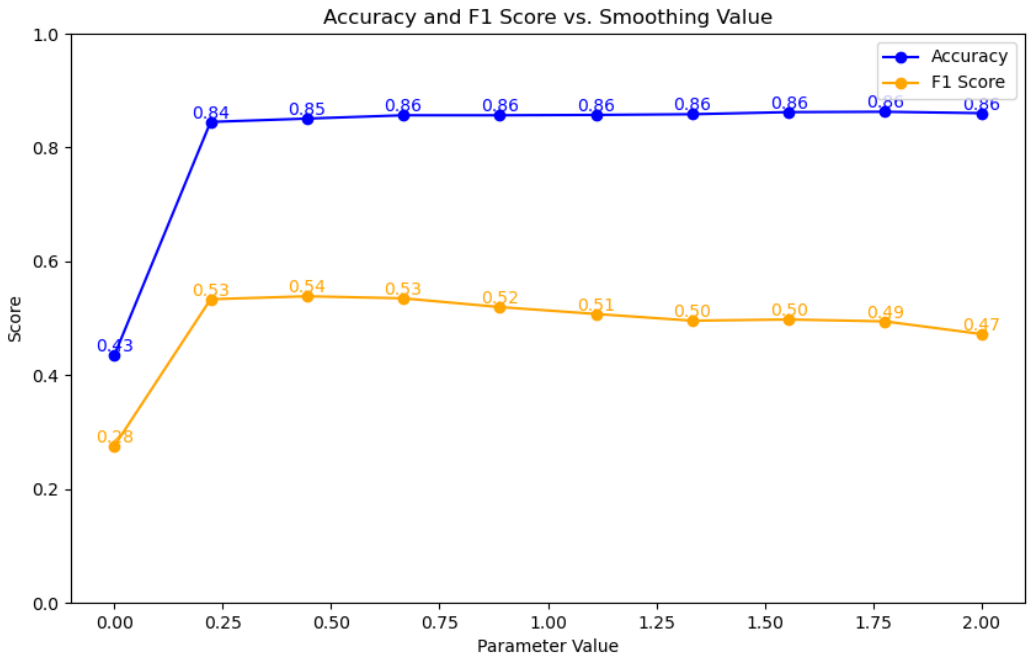
Alejandra Camelo Cruz - 800385

Universität Potsdam

1. Naive Bayes

- K smoothing Value

I tried different 10 evenly spaced values from 0 to 2 for the k smoothing constant. Without smoothing at all, the model works poorly, in this case with accuracy of 0,43 and 0.28 F1-score. As far as some smoothing is added. The model performs clearly better, although the tradeoff between Accuracy and F_1 is compromised with higher values: Accuracy goes higher and F1 goes lower. Probably, the best value was around k = 0,75 as accuracy is as high as with higher values and F1 starts decreasing from here onwards.



- Feature Engineering

For feature engineering I tried two strategies: **lemmatization** and **feature reduction** cutting off the most common words, as observed in the image. The model perfoms better when some feature engineering was done, lemmatization had a better accuracy than without common words, while the last one had a better F_1 score. Feature engineering happens to be an important step in the training of the model as the model can learn from most relevant features and it reduces irrelevant noise.

```
(base) alejandra@MacBook-Pro-von-ZAS tweet_classification_NLP % python assignment1.py --model naive-bayes --feature_eng
Training data: 12896
Test data: 3250
Training naive bayes classifier...
Accuracy:  0.8572307692307692
F_1:  0.5156576200417536
---- Training naive bayes classifier without common words
Accuracy:  0.8529230769230769
F_1:  0.5394999366088632
---- Training naive bayes classifier with lemmas
Accuracy:  0.8584615384615385
F_1:  0.5116772823779193
(base) alejandra@MacBook-Pro-von-ZAS tweet_classification_NLP %
```

2. Logistic regression

I trained logistic regression with 100 minibatched in 10 epochs on the whole data set. Results of training process can be observed in the following image. Loss was diminished until 0.328 and accuracy for training set was 0.866 while for test set was 0.828. Some feature engineering was also done inside the featurize function. Such as for Naive bayes, I lemmatized and cut off 1000 most common words, as it didn't change the performance of the model and it improved the training time.

```
(base) alejandra@MacBook-Pro-von-ZAS tweet_classification_NLP % python assignment1.py --model logreg
Training data: 12896
Test data: 3250
Training logistic regression classifier...
training logistic regression for 10 epochs with learning rate 0.01, regularization lambda 0.1 and mini-batch size 100

training epoch 1: 100%|████████████████████████████████████████████████████████████████████████████████████████████████| 128/128 [02:34<00:00,  1.21s/it]
epoch 1 evaluation:
accuracy = 0.6961848635235732 epoch_loss = 1.1593206206611328
training epoch 2: 100%|████████████████████████████████████████████████████████████████████████████████████████████████| 128/128 [02:45<00:00,  1.29s/it]
epoch 2 evaluation:
accuracy = 0.7411600496277916 epoch_loss = 0.858107149816763
training epoch 3: 100%|████████████████████████████████████████████████████████████████████████████████████████████████| 128/128 [02:53<00:00,  1.36s/it]
epoch 3 evaluation:
accuracy = 0.7686104218362283 epoch_loss = 0.6831516322298221
training epoch 4: 100%|████████████████████████████████████████████████████████████████████████████████████████████████| 128/128 [02:47<00:00,  1.31s/it]
epoch 4 evaluation:
accuracy = 0.7949751861042184 epoch_loss = 0.5687458776301033
training epoch 5: 100%|████████████████████████████████████████████████████████████████████████████████████████████████| 128/128 [02:40<00:00,  1.26s/it]
epoch 5 evaluation:
accuracy = 0.8131978908188585 epoch_loss = 0.4901056796669816
training epoch 6: 100%|████████████████████████████████████████████████████████████████████████████████████████████████| 128/128 [02:40<00:00,  1.25s/it]
epoch 6 evaluation:
accuracy = 0.8300248138957816 epoch_loss = 0.433889925527465
training epoch 7: 100%|████████████████████████████████████████████████████████████████████████████████████████████████| 128/128 [02:36<00:00,  1.22s/it]
epoch 7 evaluation:
accuracy = 0.8452233250620348 epoch_loss = 0.3918797679402089
training epoch 8: 100%|████████████████████████████████████████████████████████████████████████████████████████████████| 128/128 [07:11<00:00,  3.37s/it]
epoch 8 evaluation:
accuracy = 0.8570099255583127 epoch_loss = 0.3619853836649062
training epoch 9: 100%|████████████████████████████████████████████████████████████████████████████████████████████████| 128/128 [02:39<00:00,  1.25s/it]
epoch 9 evaluation:
accuracy = 0.8681761786600496 epoch_loss = 0.34071754470882
training epoch 10: 100%|███████████████████████████████████████████████████████████████████████████████████████████████| 128/128 [02:38<00:00,  1.24s/it]
epoch 10 evaluation:
accuracy = 0.8667803970223326 epoch_loss = 0.3282500907521551
--------logistic regression was trained--------
Evaluation on Test Data: accuracy = 0.828
(base) alejandra@MacBook-Pro-von-ZAS tweet_classification_NLP %
```