Report - Task 4

Oweys Momenzada, Sönke Sobott

Tools

- Pandas, Sklearn, Numpy
- Keras, Tensorflow, Torch
- Matplotlib, Seaborn

Steps

For every subtask we use an own notebook. For instance, the classification for b&t, e&m, etc. all have their own .ipynb-file. Therefore, the steps of all notebooks are identical.

Data import and preparation

- Import of the above libraries which are required for the data processing.
- Read in all data from the uci-news-aggregator.csv file and remove all redundant columns.
- Conversion of the labels into integer values using the *LabelEncoder()* of the sklearn library.
- Split the input data into training and test data with the sklearn *train_test_split()* method which automatically balances the data

Define tokens based on Keras

- We define tokens based on *Keras.preprocessing.text*
- Furthermore, tokens are getting mapped to integer sequences based on Keras. For this we need to define a fixed number of used words and a maximum number of words per sequence.
- We define the batches similar to the one of the exercise.
- The function *pad_sequence(texts)* is converting strings to tokens and then to integers. This will be used for predictions.
- We also define an ANN model with Pytorch. The model has two dense layers.
- The model gets trained with the Adam optimizer and is evaluated with the *CrossEntropyLoss* error function. After the training we save the model for the FLASK API
- We need to convert the testset to tensor for predictions. Since we work on a huge dimension, converting the whole testset into tensor would cause a memoryerror. Therefore we use a divide and conquer approach to avoid that problem. We then just save the prediction of each row of the testset and evaluate them in the next step.

Results

- Based on the labels, we calculate the accuracy for the multi classification and all binary classification pairs.
- As a final step, we calculate a confusion matrix (CM) as well as the **precision**, **recall** and **f1score** for each classification.

Bonus
We implement an API based on FLASK. The predictions will be calculated via POST requests. We simply start the API over the local maschine without any frontend.

Results: Multi Classification

- Model accuracy is **93.9527%**
- precision: [0.90712035 0.97638686 0.9596603 0.91659513]
- recall: [0.92226873 0.96721906 0.90159574 0.93518647]
- f1score: [0.91463182 0.97178133 0.92972232 0.92579747]

Results: Binary Classification

B = Business
T = Science and Technology
E = Entertainment
M = Health

- B&E:
    - Model accuracy is **98.2771%**
    - precision: [0.97614796 0.98793423]
    - recall: [0.98439309 0.98152478]
    - f1score: [0.98025319 0.98471908]
- B&M:
    - Model accuracy is **97.6982%**
    - precision: [0.98238714 0.96282195]
    - recall: [0.98575841 0.95427303]
    - f1score: [0.98406989 0.95852843]
- B&T:
    - Model accuracy is **94.1734%**
    - precision: [0.9390951 0.94466943]
    - recall: [0.9497068 0.93306977]
    - f1score: [0.94437114 0.93883377]
- E&M:
    - Model accuracy is **98.7406%**
    - precision: [0.9390951 0.94466943]
    - recall: [0.9497068 0.93306977]
    - f1score: [0.94437114 0.93883377]
- T&E:
    - Model accuracy is **98.1136%**
    - precision: [0.986029   0.97436718]
    - recall: [0.98155515 0.98054994]
    - f1score: [0.98378699 0.97744878]
- T&M:
    - Model accuracy is **98.2628%**
    - precision: [0.97699251 0.98498595]
    - recall: [0.96456907 0.99032228]
    - f1score: [0.97074104 0.98764691]