Fake-news Detection Project
Neuro-fuzzy Computing

Noli Chrysoula 2780
Panagiotis Boulogeorgos 2905

February 18, 2022

Abstract

In this project have been asked to implement a neural network which predicts fake news in an article. It has been trained and tested on same data. Given URL's we extract the text, edit it and train the neural network. In this report we describe in detail our project sharing all the modules and files we used.

Data

Training and testing data (same) are provided by the link : https://data.mendeley.com/datasets/p4c49m3pvr/3 from where we download the GermanFakeNC.json file.

GermanFakeNC.json file's structure:

- Date
- URL
- False Statement 1 Location
- False Statement 1 Index
- False Statement 2 Location
- False Statement 2 Index
- False Statement 3 Location
- False Statement 3 Index
- Ratio of Fake Statements
- Overall Rating

comment: The .json file that provided to us, have 491 URLs, but the 232 links have expired. So, we work with 259 URLs in total.

Training Algorithm

In dataExtract.py:

- Convert .json to .csv for more accessibility
- Iterate through the URLs, trying to open them and extract the text. If not succeed, drop this line from .csv
- Pre-proccess texts by converting to lowercase, removing punctuation and stopwords (both german and english)
- Calculate and keep length of each text and save it to GermanFakeNCRateLength.csv
- Normalize Overall Rating
- Keep only texts in GermanFakeNCTexts.cssv and rate and length in GermanFakeNCRateLength.csv.

In DataPreProccess.py:

- Tokenize text, convert the words, letters into counts or numbers
- Calculate vocabulary size
- Apply padding to make them even shaped
- Calculate embeddings matrix according to german vocabulary
- Prepare test and train data for split

In model.py:

- Choose Train or Predict
- If train model, compile model, fit the data in it, save it and evaluate it
- If predict, load model, fit test data and calculate accuracy

Neural network's architecture

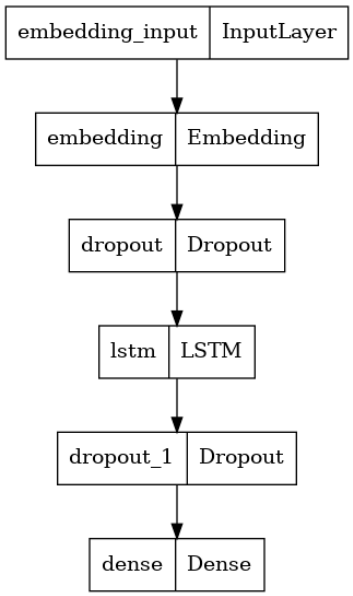

Figure 1: model's layers

Layer (type)	Output Shape	Param #
embedding (Embedding)	(None, 3843, 100)	2716200
dropout (Dropout)	(None, 3843, 100)	0
lstm (LSTM)	(None, 100)	80400
dropout_1 (Dropout)	(None, 100)	0
dense (Dense)	(None, 1)	101

Total params: 2,796,701
Trainable params: 2,796,701
Non-trainable params: 0

Figure 2: model's parameters

Some basic characteristics:

- loss function = binary crossentropy
- optimizer = adam
- epochs = 10
- batch size = 64
- validation split = 0.2

Performance

training:

```
3/3 [==============================] - 13s 4s/step - loss: 0.6902 - acc: 0.6485 - val_loss: 0.6768 - val_acc: 0.6905
Epoch 2/10
3/3 [==============================] - 11s 4s/step - loss: 0.6749 - acc: 0.6485 - val_loss: 0.6451 - val_acc: 0.6905
Epoch 3/10
3/3 [==============================] - 11s 4s/step - loss: 0.6374 - acc: 0.6485 - val_loss: 0.5740 - val_acc: 0.6905
Epoch 4/10
3/3 [==============================] - 11s 4s/step - loss: 0.5901 - acc: 0.6485 - val_loss: 0.5138 - val_acc: 0.6905
Epoch 5/10
3/3 [==============================] - 11s 4s/step - loss: 0.5291 - acc: 0.6606 - val_loss: 0.4890 - val_acc: 0.7143
Epoch 6/10
3/3 [==============================] - 11s 4s/step - loss: 0.5058 - acc: 0.7879 - val_loss: 0.4442 - val_acc: 0.8333
Epoch 7/10
3/3 [==============================] - 11s 4s/step - loss: 0.4555 - acc: 0.8182 - val_loss: 0.3872 - val_acc: 0.8571
Epoch 8/10
3/3 [==============================] - 12s 4s/step - loss: 0.4038 - acc: 0.8364 - val_loss: 0.3691 - val_acc: 0.8571
Epoch 9/10
3/3 [==============================] - 11s 4s/step - loss: 0.3932 - acc: 0.8424 - val_loss: 0.3562 - val_acc: 0.8571
Epoch 10/10
3/3 [==============================] - 11s 4s/step - loss: 0.3736 - acc: 0.8545 - val_loss: 0.3549 - val_acc: 0.8810
Saved  model to disk
Train: [0.3530312776565552, 0.8599033951759338] Test: [0.5371887683868408, 0.7115384340286255]
```

Figure 3: evaluation of model and final weights

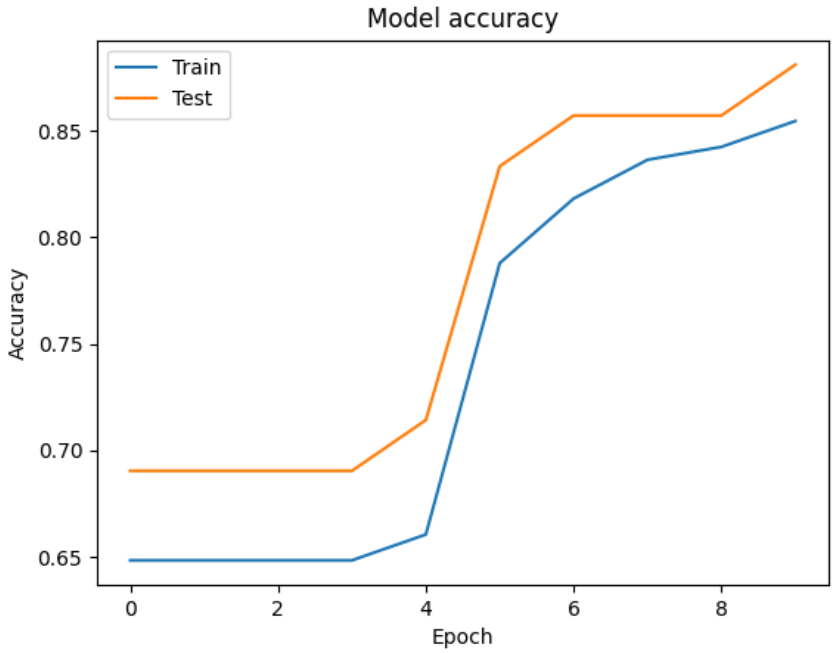

Figure 4: model's accuracy evaluation

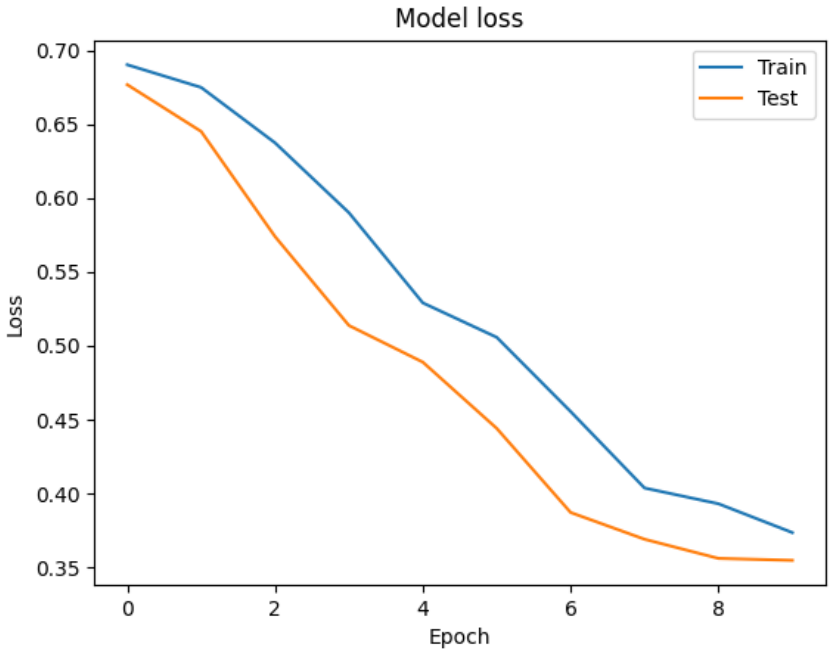

Figure 5: model's loss evaluation

predicting:

	precision	recall	f1-score	support
0.0	0.88	0.87	0.87	171
1.0	0.74	0.76	0.75	88
accuracy			0.83	259
macro avg	0.81	0.81	0.81	259
weighted avg	0.83	0.83	0.83	259

Figure 6: Our neural network predicts fake news with accuracy 83%.

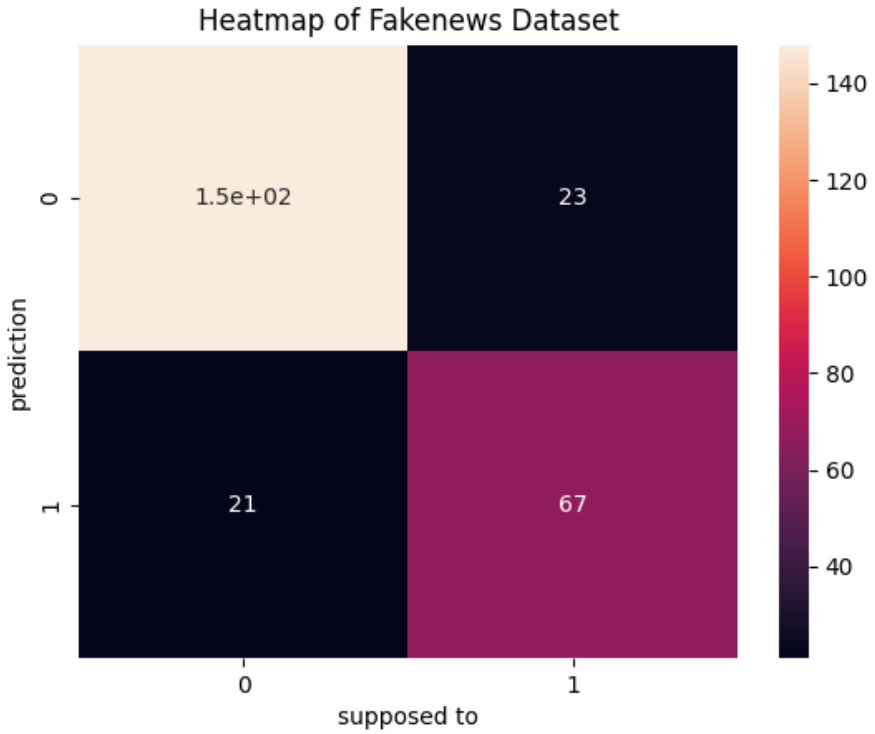

Figure 7: Heatmap of Fakenews dataset (259 texts)

While the data discrimination is:

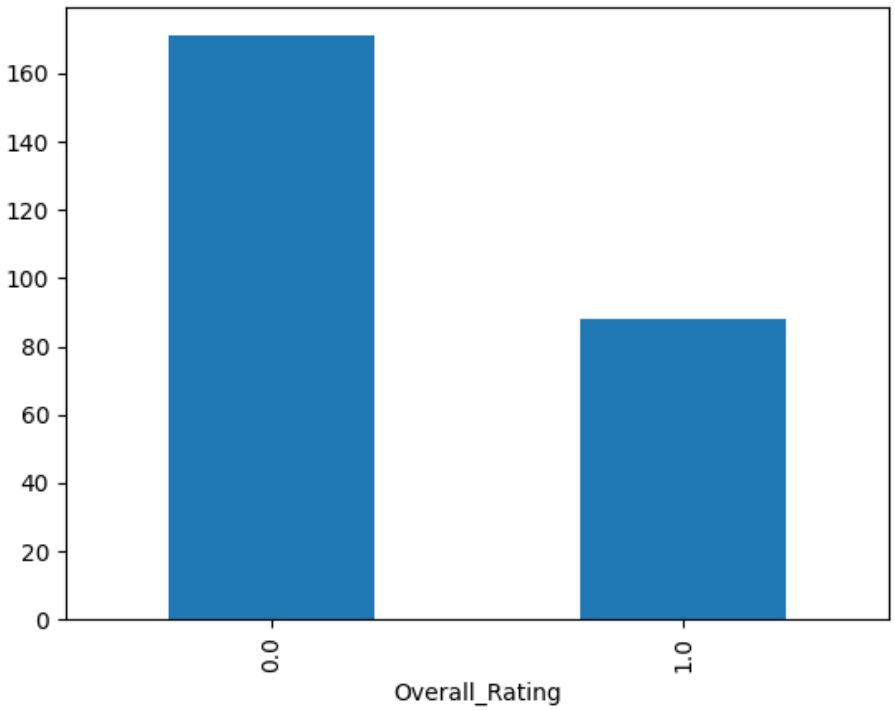

Figure 8: targets of data

Training time

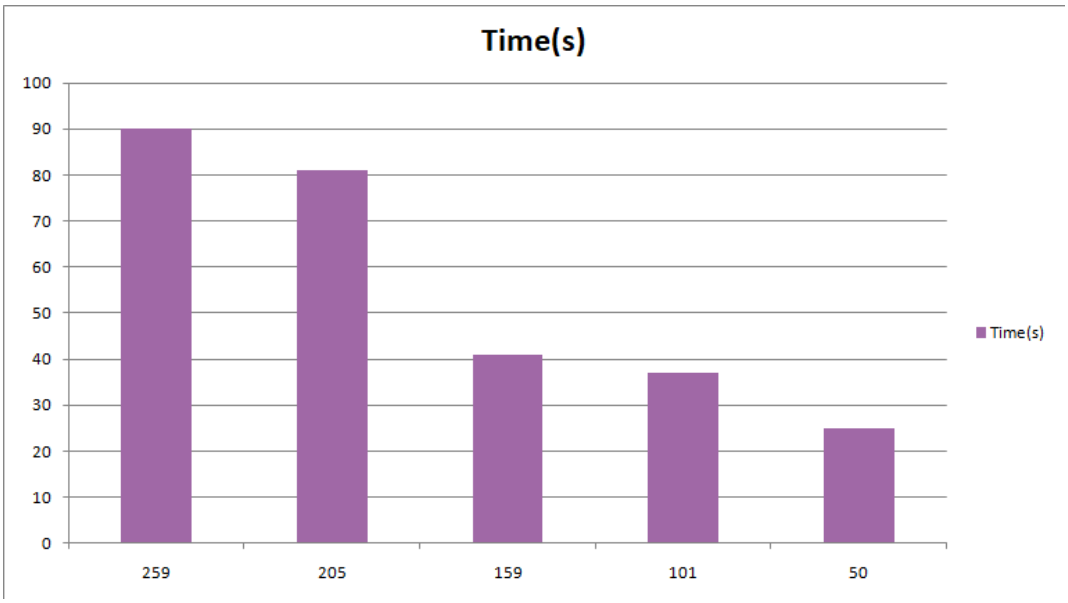

Figure 9: training time depending on input size (number of texts)

Notes

We created and tested our neural network on a device that has the following specs:

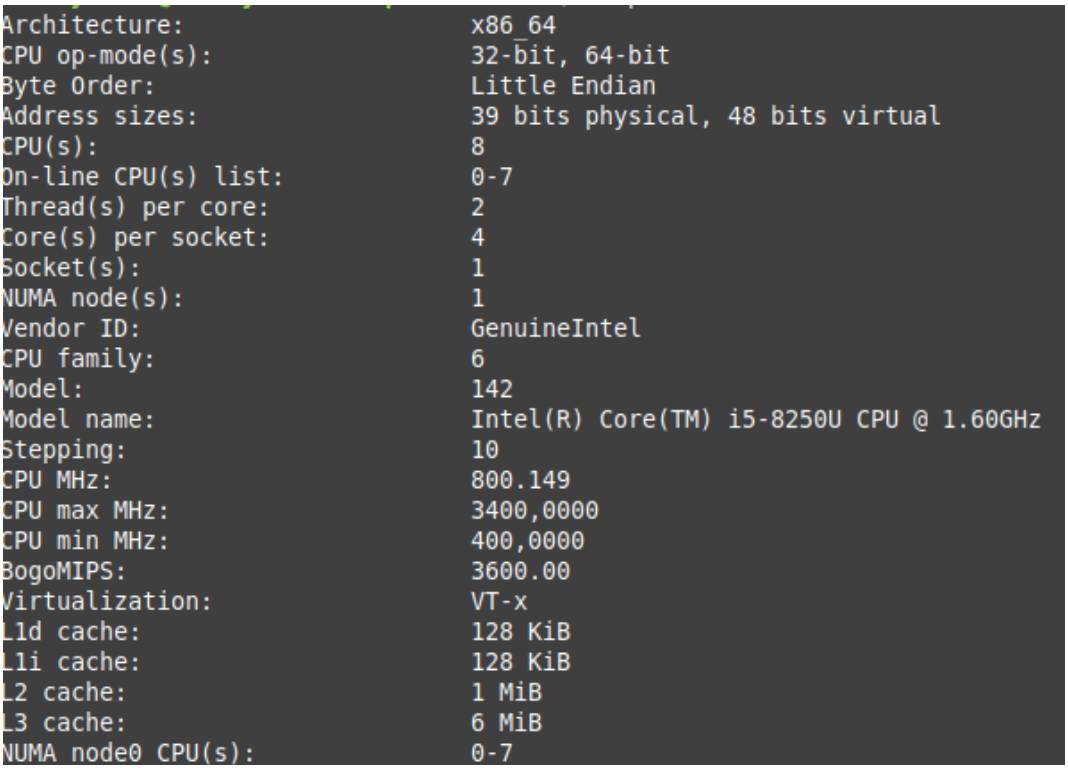

Figure 10: device's specs

And internet speed: 13 mbps.

references:
code for model's structure taken from https://github.com/wutonytt/Fake-News-Detection/blob/main/Balanced$_Data.ipynb$