Assignment 1 - IN5550

Lotte Boerboom, Arthur Dujardin, Sigurd Hylin

1/31/2020

Introduction

This report concerns news article classification using Bag of Words. A feed forward neural network was trained to predict one of the 20 sources. A PyNews package implemented in python was used to explore the different possibilities and training hyper parameters, with the help of PyTorch.

We have trained and compared five different architectures on a training and development set. The main focus of this report is to analyse the influence of the number of hidden layers on the model's performance and time efficiency.

From these different structures, we selected the best one and trained it three times and will be used to predict unseen data.

Data Processing

A data module was used to analyse and extract features from the documents. First, the dataset *The Signal Media One-Million News Articles* which contains articles from september 2015 was used as training data. Then, the raw data was stripped (tokenized) of part of speech tags, and the most common words were saved under a vocabulary vectorizer (`vectorizer.pickle`) of size 3000, which can be used also on test and unknown data. The document's source (gold labels) are used as target labels to evaluate the predictions.

The functionality for processing the data can be found in the Python package called 'data.py'. This code reads the datafile, stripping the POS tags from the documents, builds the vocabulary and set of labels, and then maps the vocabulary to the documents and encodes the sources to their numeric labels.

Models

Feature Tuning

Before changing the structure of the model, we explored differents Bag of Words features implementation by varying the vocabulary size, the preprocessing and building the vocabulary before and after the split. For example, with a vocabulary size of 4000 we did not see improvement in the performance. In addition, using the Part of Speech (POS) tags did not help to optimize the results. Because of difficulties and time restriction we did not create the vocabulary after splitting into training and development parts.

Hyperparameters

A brief training session to evaluate the performance with different hyper parameters was firstly performed. The hyper parameters used are described on the table below.

Table 1: Hyperparameters

Parameter	Value
Split	Train, Dev = .9, .1
Vocabulary	3000
Batch size	32
Learning Rate	.09
Epochs	250

Architectures

Then, five different models were trained on Saga's server with different layout. These models differ in their number of hidden layers, and their architectures are presented in the tables below.

Table 2: Model 1 Architecture

Layers	Neurons	Activation
Input	3000	ReLU
Hidden Layer 1	150	Linear
Output	20	Softmax

Table 3: Model 2 Architecture

Layers	Neurons	Activation
Input	3000	ReLU
Hidden Layer 1	150	ReLU
Hidden Layer 2	150	Linear
Output	20	Softmax

Table 4: Model 3 Architecture

Layers	Neurons	Activation
Input	3000	ReLU
Hidden Layer 1	150	ReLU
Hidden Layer 2	150	ReLU
Hidden Layer 3	150	Linear
Output	20	Softmax

Table 5: Model 4 Architecture

Layers	Neurons	Activation
Input	3000	ReLU
Hidden Layer 1	150	ReLU
Hidden Layer 2	150	ReLU
Hidden Layer 3	150	ReLU
Hidden Layer 4	150	Linear
Output	20	Softmax

Table 6: Model 5 Architecture

Layers	Neurons	Activation
Input	3000	ReLU
Hidden Layer 1	150	ReLU
Hidden Layer 2	150	ReLU
Hidden Layer 3	150	ReLU
Hidden Layer 4	150	ReLU
Hidden Layer 5	150	Linear
Output	20	Softmax

Evaluation

After we trained all models, there was no specific model that stood out. However, we tried to differenciate them regarding four indicators : the accuracy, macro-F1, precision and recall.

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

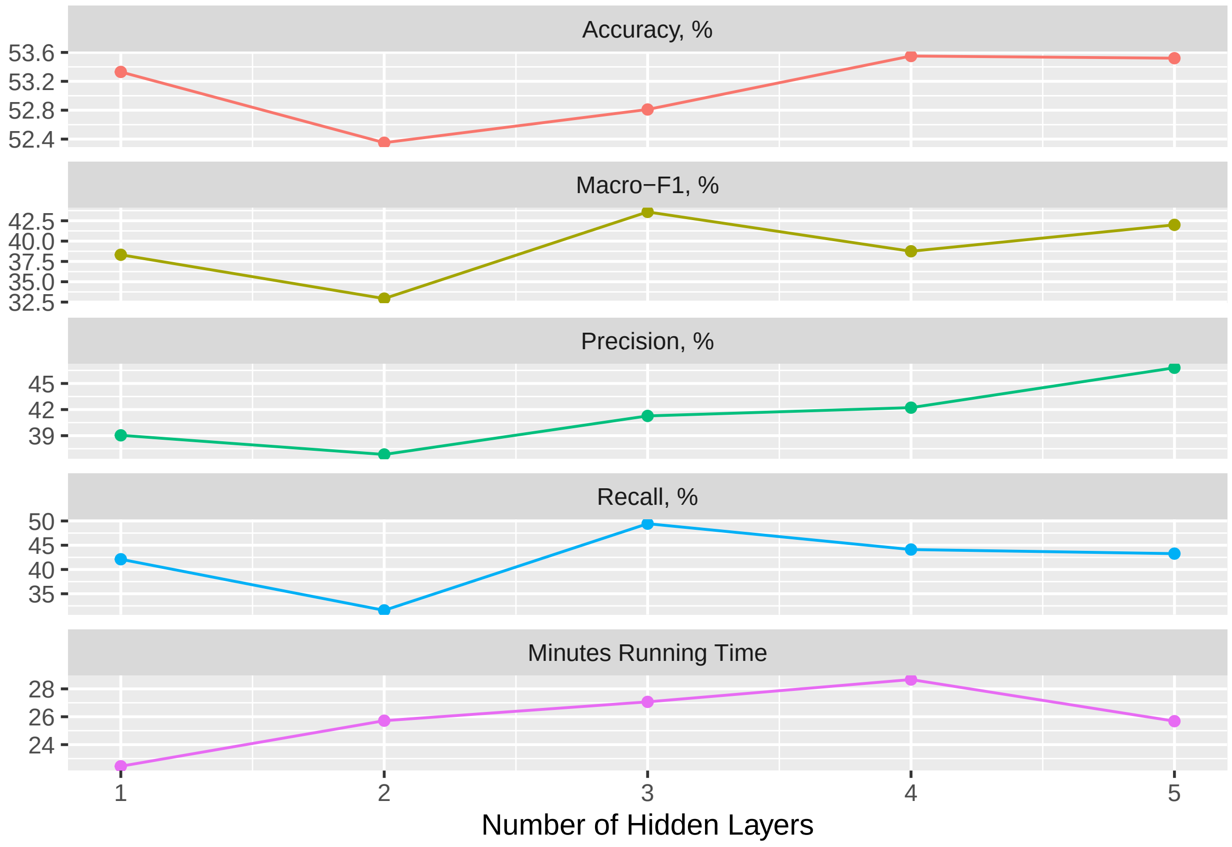
$$Accuracy = \frac{\#Correct}{N}$$

And Macro-F1 is the weighted average between precision and recall.

Results

As shown in the table below, the model 4 presents the best accuracy. Nevertheless, this might not be the only criterion to consider, especially because of the different sources frequency. The Macro-F1 score might be less sensitive to imbalanced class frequencies. In that case, the model 3 performs better even if its precision is not optimal. Due to these performance, we choose the model 3 to push further the training.

Model	Accuracy	Macro-F1	Precision	Recall	Run Time
Model 1	53.33	38.32	39.04	42.11	00:22:27
Model 2	52.35	32.93	36.84	31.58	00:25:43
Model 3	52.81	43.59	41.27	49.44	00:27:04
Model 4	53.55	38.75	42.22	44.11	00:28:40
Model 5	53.52	42.00	46.80	43.27	00:25:41



Model Training

The mean and standard deviations of the metrics when running the chosen model three times is displayed in the table below.

Table 8: Averages and Standard Deviations of Final Runs

Metric	Average	Std.dev
Accuracy	53.31333	0.2871121
Macro-F1	34.07333	4.0730865
Precision	33.85333	4.1304883
Recall	37.27000	5.4732897
Run Time	42.73333	10.2779294

Conclusion

As shown in the results, the performance are really different from one training to the other. Most likely this is due to us no setting a random seed, leading to different weights, bias and also affecting the optimizer.