Deep Learning for Natural Language Processing

Homework 2

Eirini Kolimatsi

Student ID: 7115112200015
Email: eirini[dot]kolimatsi[at]di[dot]uoa[dot]gr

MSc Computer Science
University of Athens
Academic Year: 2022-2023

Contents

1 Data Preparation

Libraries Firstly, all required libraries should be loaded. For the needs of this assignments modules from pytorch (torch, torchtext), scikit-learn (sklearn), nltk, pandas, numpy, re, matplotlib, contractions, google.colab and other utility libraries.

Data Loading In order to access the data provided for the assignment, the csv file was uploaded in Google drive. Then the notebook got "connected" to Google Drive, to allow reading the file from a path. The path that leads to the file is `/content/drive/path/to/file`. When testing the model against other data, the Google Drive file path should be replaced should be replaced in the last section of the notebook.

Pytorch Device If GPUs are available, then the notebook runs on GPU using CUDA to reduce the required time to execute cells through parallelising tasks. In Google Colab, GPU runtime is provided, but in cases it is not available simple CPUs will be used.

1.1 Data Cleaning

The provided data have 3 columns: rating, URL and review. The URL column is not used. Reviews are expressed in a scale from 1 to 10. As a result, all negative reviews (with a score below or equal to 4) are marked with 0 and all positive reviews (with a score above or equal to 7) are marked with 1.
Going through the reviews, it is clear that the data need to be cleaned. Firstly, reviews include HTML tags (eg `<br>`) which cannot help understand the sentiment of the review and need to be removed. Similarly for punctuation points. Also, the characters should all be in lower case to avoid the same word being interpreted as a different one given the case. Lastly, given that reviews are everyday simple text there are contractions present in the text. For example, isn't is used instead of is not. To "expand" these abbreviated words, a dedicated library is used, that is called contractions.
In addition to the above steps that were also performed in the first assignment, the words length is reduced in words with multiple repeated letters. This is applied on the basis that English words usually have at most 2 identical repeated letters, thus letters that occur more

than 2 times in a word are reduced to 2. For example, the word mistaaaaake would be cleaned as mistaake. This step can be useful if spelling checking is applied, to maximize the chances to find the correct spelling. Given that, spelling checks were attempted but given the computational effort that was required and as a result the required time for the data to be prepared, it was decided to omit spelling checks and fixes. The cleaning steps that are applied are:

- HTML tags removal
- lower case adaptation
- punctuation removal
- stopwords removal
- contractions removal
- length reduction of repeated letters
- removal of multiple spaces

Given that this is the same dataset as in the previous assignment, data exploration has already taken place and will not be repeated in this report. However, insights from the previous report/analysis were used throughout in this assignment. For example, when removing stopwords, given the wordcloud that was produced in the previous assignment some additional word were classified as "stopword" and were removed (eg film, movie).

1.2 Data Preparation for Training

Splits The dataset was splitted in 3 sets: train, validation and test sets, using the relevant utility from `sklearn`. First, the set was split in train (90%) and test (10%) sets. Then, the train set was split in train (8/9) and validation (1/9) sets. The split was done with stratification for y, to maintain the class balance. The final sizes of the tests are:

- Train: 80%
- Validation: 10%
- Test: 10%

Embeddings The models that will be presented below use the GloVe (6B tokens, 400k vocab, uncased) embeddings with 50 and 300 dimensions. In order to use the embeddings, they were first downloaded and then for each sentence, the embedding values for each word were found. For words that are not included in GloVe, their vector was filled with zeroes. Then, for each sentence, the final vector was calculated as the average of the embedding vectors of the sentence's words.

Pytorch Tensors After data cleaning and transforming the reviews into vectors using the GloVe embeddings, the reviews and the ratings are transformed into tensors. Then from the tensor, the pytorch datasets are created and given the batch size the data loaders. The batch size used is 32. Batch sizes of 64, 128 and 256 were tested with all of them having similar performance, so 32 was selected.

2 Models

Three model architecture were created:

- Two Layers Feed Forward NN
- Five Layers Feed Forward NN
- Two Layers Feed Forward NN + One Dropout Layer

All models take as a parameter an activation function to be applied between the linear layers. Regarding the hidden layers size, it is passed as a parameter and in the second model (5 layers) it is getting halved at each layer. For example, a model with input size 300 and hidden size 512 would have the following layers: $L_1 = (300, 512)$, $L_2 = (512, 256)$, $L_3 = (256, 128)$, $L_4 = (128, 64)$, $L_5 = (64, 1)$.

3 Training

The training included evaluation in the validation set for each epoch. At the end of the training, each model was evaluated on the test set for the metrics (F1-score, Precision, Recall, Accuracy). The confusion matrix and ROC curve was plotted as well.
At first, training was done using the datasets with the 50-dimension GloVe embeddings. However, the results were substantially lower compared to the results from the first assignment, so it was decided to continue training with the embeddings with dimension 300. In order to select the models that perform well to further explore, a for loop was built to check the performance using different combinations of the training elements. The combinations that were checked used parameters that are described below.

Loss Function The loss function used in training is BCEWithLogitsLoss, which combines a Sigmoid layer and BCELoss. It is chosen over BCELoss because of the increased numerical stability that provides compared to applying a sigmoid layer in the model and then selecting the BCELoss criterion.

Optimizer Adam, SGD

Learning Rate $1e-2$, $1e-3$, $1e-4$

Epochs 10 for model searching. When fine tuning models larger numbers were used according to each model's results

Activation Functions ReLU, Leaky ReLU, Tanh

Hidden Layers + Neurons Given that models described above have either 2 or 5 linear layers, for the models with 2 layers (1 hidden) the sizes tested are 25, 200, 500. For the model with 5 layers (4 hidden) the initial hidden layer size was tested for 512 and 1024 neurons. These sizes were used mostly for the training in the dataset with the 50 dimensions GloVe embeddings.

4 Results

4.1 50d GloVe Embeddings

Grid Search To get an idea about which direction to follow in training and model settings, a search is done against a set of parameters and values. More specifically, the model is trained on all the possible combinations of the following settings:

- Architecture: 2 linear layers (hidden size = 500) + 1 dropout layer (p=0.25)
  - 2 Linear Layers (1 hidden)
    Hidden Layer Sizes: 25, 200, 500
  - 2 Linear Layers (1 hidden) + 1 Dropout Layer
    Hidden Layer Sizes: 25, 200, 500
  - 5 Linear Layers (4 hidden)
    Hidden Layer Sizes: 512, 1024
- Activation Function: ReLU, Leaky ReLU, Tanh
- Optimizer: Adam, SGD
- Learning Rate: 0.01, 0.001, 0.0001
- Epochs: 10

From these 144 models, there are multiple that have similar scores and quality of performance. The detailed results and learning curves for each model can be found on the notebook submitted along with this report. After experimenting further, some of them are presented below as the final models that perform the best.

Batch Size Before proceeding to training more models, it is important to checkout if the selected batch size works properly or a change is needed. To do so, a candidate model is selected and it will be trained with different batch sizes.

- Architecture: 2 linear layers (hidden size = 250)
- Activation Function: ReLU
- Optimizer: Adam
- Learning Rate: 0.0001
- Epochs: 30

The results from training the 4 models show that there are not huge differences in performance when batch size is changing. However, the batch size 32 has the best f1-score and recall, so it is selected as the batch size for the rest of the training. Even though a batch size is selected, at later points in the training of models, other batch sizes were checked again, without causing any performance improvement.

	Batch size			
Metric (%)	**32**	**64**	**128**	**256**
F1-score	77.144	76.079	77.113	76.643
Recall	78.721	73.612	77.832	75.655
Precision	75.630	78.717	76.407	77.656
Accuracy	76.672	76.850	76.894	76.939

Model A1 From the manual grid search results and further experimentation with learning rates and epochs, the model following model was trained:

- Architecture: 2 linear layers (hidden size = 500) + 1 dropout layer (p=0.25)
- Activation Function: ReLU
- Optimizer: Adam
- Learning Rate: 0.00005
- Epochs: 100

The training seems successful given the results and the Loss vs Epochs and Accuracy vs Epochs plots. There are no signs of overfitting or underfitting.

Metric	Score (%)
F1-score	77.841
Recall	78.499
Precision	77.195
Accuracy	77.649

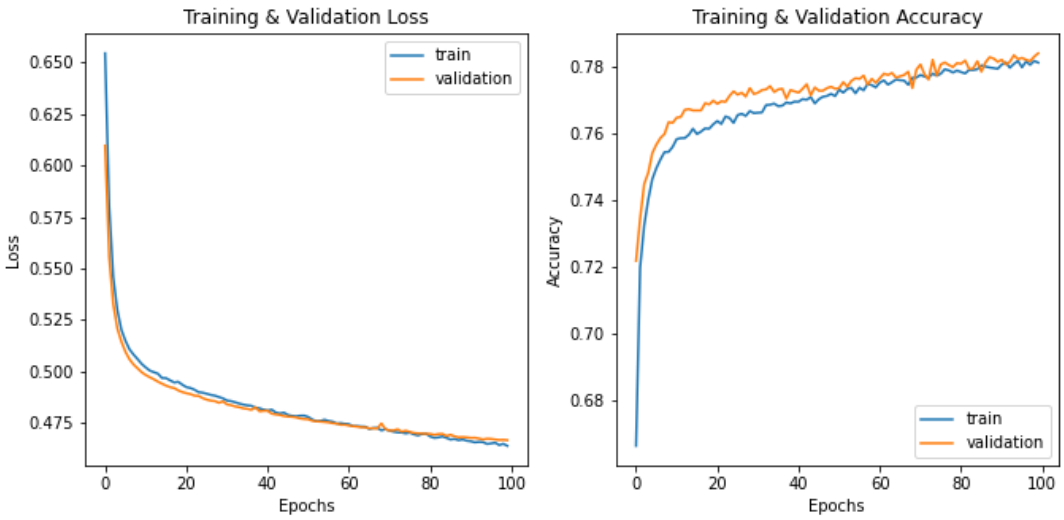


Figure 1: Model A1 - Loss and Accuracy in Training & Validation sets

Model A2 Through the plots that were produced from the manual grid search results, it can be noticed that many of the models using the SGD optimizer seem to have chances to be good for the task, but the 10 epochs used in the grid search seem insufficient. This is due to the fact that usually SGD needs more time to converge compared to Adam optimizer.

- Architecture: 2 linear layers (hidden size = 750)
- Activation Function: ReLU
- Optimizer: SGD
- Learning Rate: 0.004
- Epochs: 160

Metric	Score (%)
F1-score	77.089
Recall	77.654
Precision	76.532
Accuracy	76.916

The result from this model are slightly worse compared to model A1. This model is underfitting a bit, but not significantly and it's underfitting shouldn't be considered as concerning about the model's performance. We can also notice the difference in the number of epochs compared to the 1st model, given the slower convergence of the SGD.

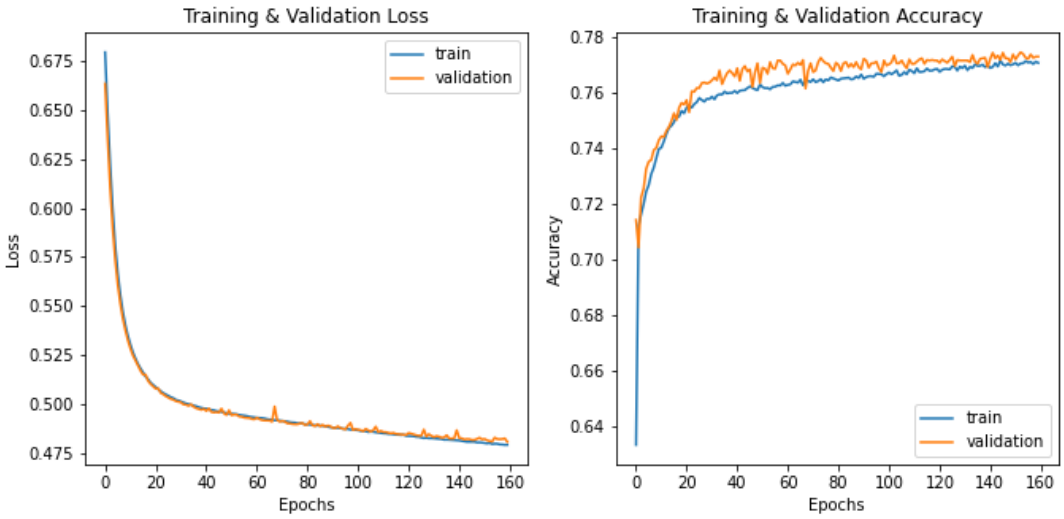


Figure 2: Model A2 - Loss and Accuracy in Training & Validation sets

Many more training attempts were done, with different parameters and architectures, but these were the best out of all, thus are selected to be included in the report. However, when comparing the results with those from the first assignment (eg F1-score@88.33%) these results are significantly worse. This seems quite odd given that Neural Networks are used in this assignment, as well as GloVe embeddings which are considered much more effective for NLP tasks compared to simple Logistic Regression. In an attempt to explore ways to improve these scores, instead of using the GloVe embeddings with 50 dimensions, below we will embeddings with 300 dimensions to check if it improves the results.

4.2 300d GloVe Embeddings

Firstly, we're going to check how our best models up to this point perform with the 300d embeddings.

	Score (%)			
	50d GloVe		300d GloVe	
Metric	**Model A1 2L+D+Adam**	**Model A2 2L + SGD**	**Model A1 2L+D+Adam**	**Model A2 2L + SGD**
F1-score	77.841	77.089	85.502	83.956
Recall	78.499	77.654	86.850	83.918
Precision	77.195	76.532	84.195	83.993
Accuracy	77.649	76.916	85.270	83.959

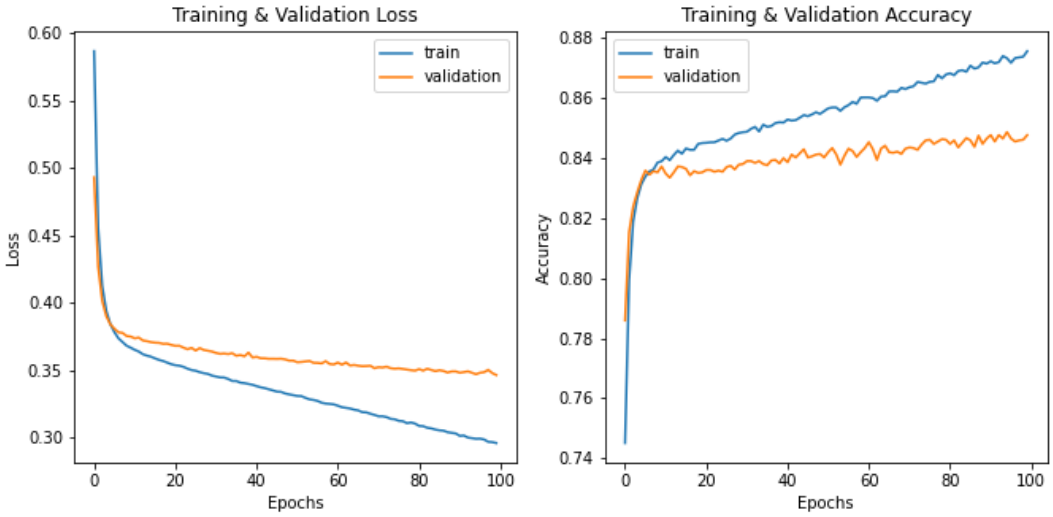


Figure 3: Model A1 & 300d GloVe - Loss and Accuracy in Training & Validation sets

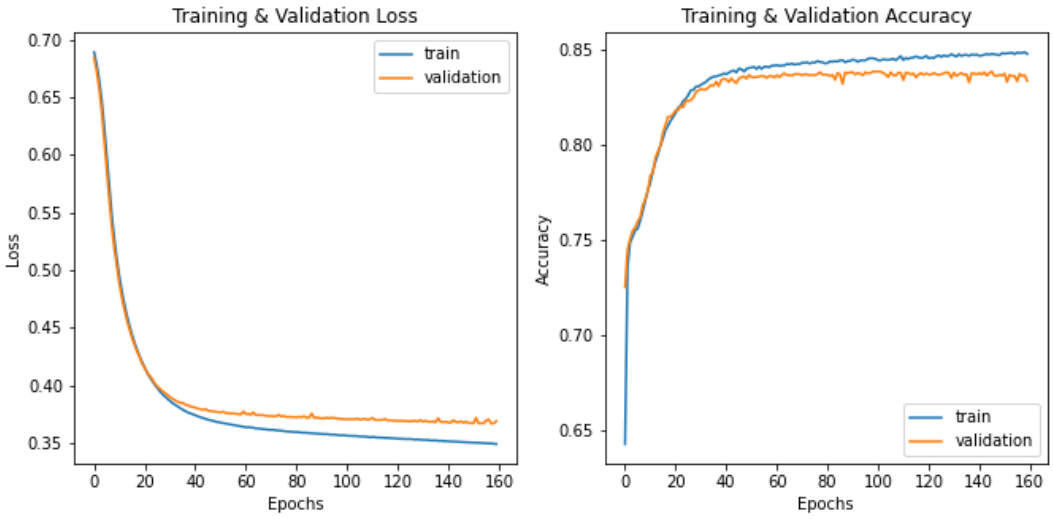


Figure 4: Model A2 & 300d GloVe - Loss and Accuracy in Training & Validation sets

Based on the plots and table above, there is substantial improvement in performance, but both models overfit. Especially, the 1st model overfits significantly. The reason why this

happens most likely is the fast convergence of Adam optimizer combined with the increased number of features (from 50 to 300). It is clear that using the 300d embeddings we will get a better model compared t 50d embeddings, but we should fine tune these models or experiment with different ones, in order to select the best for the task.

Grid Search A search will be done again to understand whether there are any models similar to the ones used to train for the 50d dataset that are appropriate for the 300d. Due to memory constraints less combinations will be tried. The parameters are selected considering the results from the previous grid, for the 50d GloVe embeddings.

- Architecture: 2 linear layers (hidden size = 500) + 1 dropout layer (p=0.25)
  - 2 Linear Layers (1 hidden)
    Hidden Layer Sizes: 200, 500
  - 2 Linear Layers (1 hidden) + 1 Dropout Layer
    Hidden Layer Sizes: 200, 500
  - 5 Linear Layers (4 hidden)
    Hidden Layer Sizes: 512, 1024
- Activation Function: ReLU, Leaky ReLU
- Optimizer: Adam
- Learning Rate: 0.01, 0.0001
- Epochs: 10

Most of the models trained seem to overfit, with the ones showing that they're likely to easily get adapted being the ones using the 2 Linear Layers + 1 Dropout Layer architecture. Thus, we're going to experiment on this direction, as well as changing the parameters a lot (eg increasing a lot the size of the hidden layers).

Model B1.1

- Architecture: 2 linear layers (hidden size = 1200) + 1 Dropout Layer (p=0.6)
- Activation Function: ReLU
- Optimizer: Adam
- Learning Rate: 0.00001
- Epochs: 40

This model provides improved results compared to the 50d best models, while maintaining the training quality.

Metric	**Score (%)**
F1-score	83.155
Recall	82.674
Precision	83.645
Accuracy	83.248

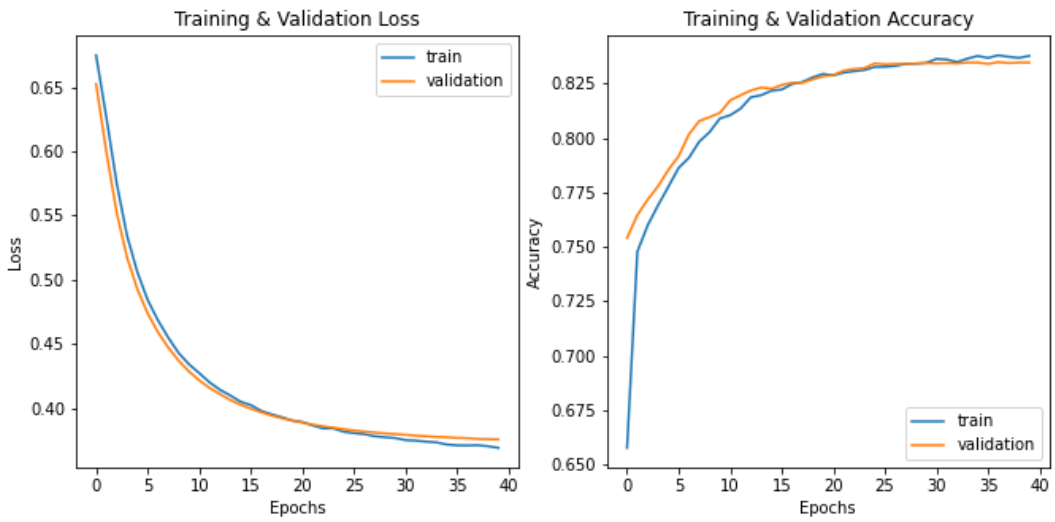


Figure 5: Model B1.1 - Loss and Accuracy in Training & Validation sets

Model B1.2

- Architecture: 2 linear layers (hidden size = 1000)
- Activation Function: ReLU
- Optimizer: SGD
- Learning Rate: 0.001
- Epochs: 170

This model provides slighlty better results compared to model B1.1. There is no underfitting or overfitting, but we can see that there was need for significantly more epochs in training to get to these results.

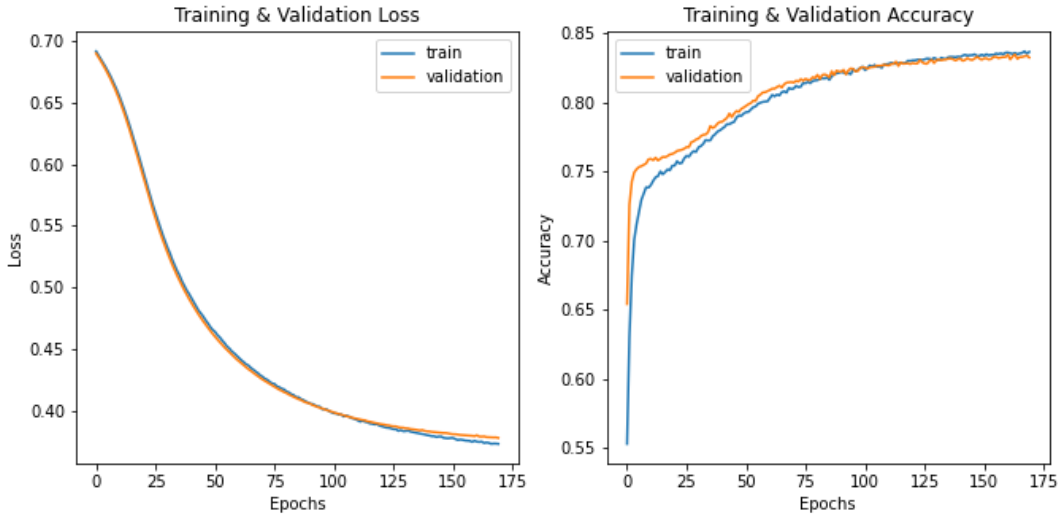


Figure 6: Model B1.2 - Loss and Accuracy in Training & Validation sets

Metric	Score (%)
F1-score	83.396
Recall	83.341
Precision	83.452
Accuracy	83.404

Model B2

- Architecture: 2 linear layers (hidden size = 850)
- Activation Function: ReLU
- Optimizer: Adam
- Learning Rate: 0.00001
- Epochs: 40

This model's results are comparable to B1.1 and B1.2 results. However, overfitting occurs, so models B1.1 and B1.2 are preferable.

Metric	Score (%)
F1-score	83.430
Recall	82.985
Precision	83.880
Accuracy	83.515

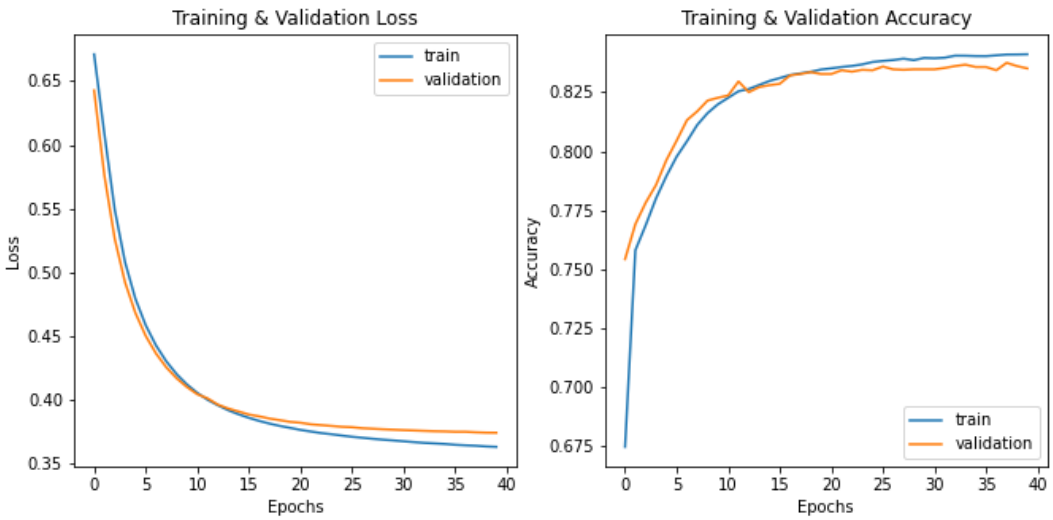


Figure 7: Model B2 - Loss and Accuracy in Training & Validation sets

Model B3

- Architecture: 5 linear layers (initial hidden size = 880)
- Activation Function: ReLU
- Optimizer: AdamW
- Learning Rate: 0.0000008

- Epochs: 50

Finally, this model utilizes the 3rd architecture that is suggested on the models section, which didn't produce many models with acceptable results (many were over/underfitting a lot). For this model, instead of Adam, the optimizer used is AdamW which improves Adam's generalization performance according to Loshchilov and Hutter [2019].

Metric	Score (%)
F1-score	82.793
Recall	82.186
Precision	83.409
Accuracy	82.915

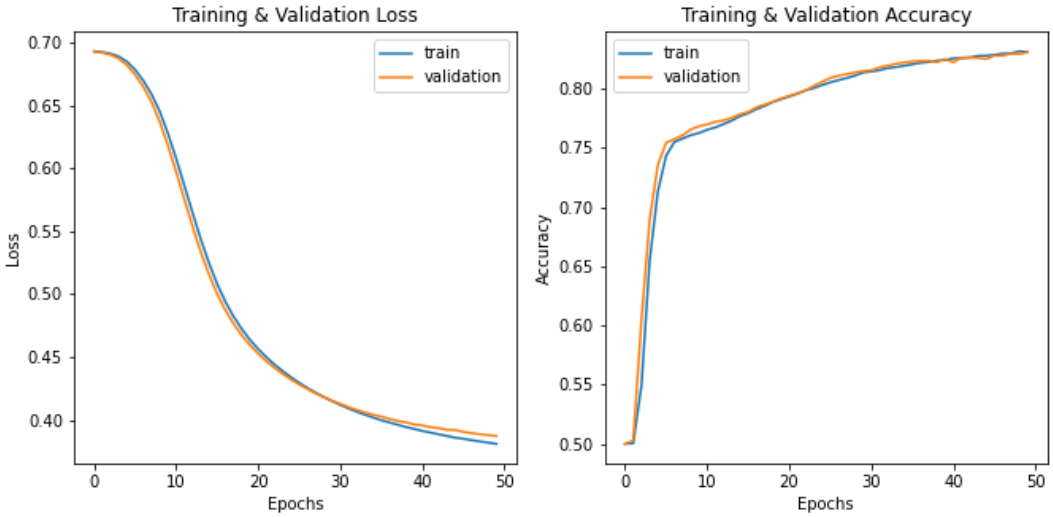


Figure 8: Model B3 - Loss and Accuracy in Training & Validation sets

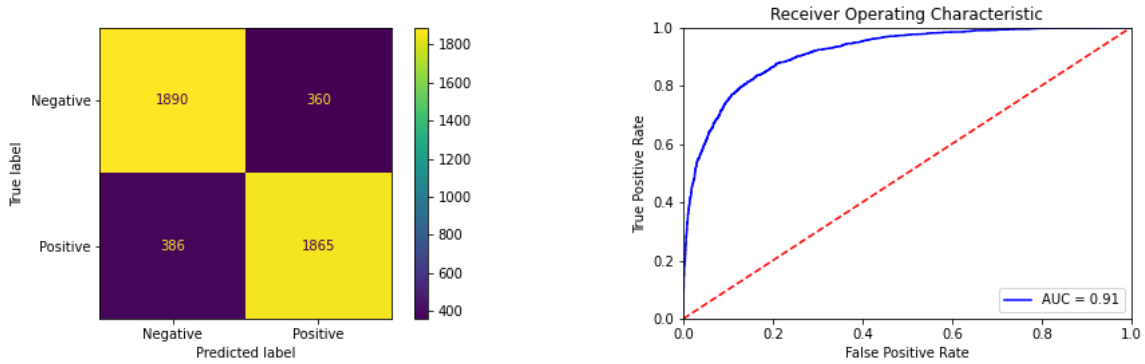
5 Model Selection

After training these models, to compare the results we can see the table below:

		Score (%)			
		F1-score	**Recall**	**Precision**	**Accuracy**
50d GloVe	**A1**	77.841	78.499	77.195	77.649
	A2	77.089	77.654	76.532	76.916
300d GloVe	**B1.1**	83.155	82.674	83.645	83.248
	B1.2	83.396	83.341	83.452	83.404
	B2	83.430	82.985	83.880	83.515
	B3	82.793	82.186	83.409	82.915

The best model that is selected as the most appropriate for the task is B1.2. Although B2 might have slightly better F1, Precision and Accuracy, it overfitts so will stick to B1.2. When

evaluating the ROC curve, we notice that the model has quite good ability of understanding the differences between positive and negative reviews.



(a) Confusion Matrix

(b) ROC Curve

Figure 9: Final Model - Results in Test set

After training the selected model again, there are slight changes in the results, so our final result is the following.

Metric	**Score (%)**
F1-score	83.333
Recall	82.852
Precision	83.820
Accuracy	83.426

In conclusion, there might be alternative architectures and hyperparameters to might achieve better results, but the suggested model has a robust ability to understand the sentiment of the reviews and achieve scores that are considered fine, even though they're below the scores from the first assignment.

6 References

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

Websites

- Pytorch documentation
- https://learn.microsoft.com/en-us/windows/ai/windows-ml/tutorials/pytorch-analysis-train-model
- https://www.kaggle.com/getting-started/251213
- https://medium.com/@martinpella/how-to-use-pre-trained-word-embeddings-in-pytorch-71ca59249f76