Deep Learning for Natural Language Processing

Homework 3

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.
The data cleaning steps remain the same as in the 2nd assignment:

- 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 two previous assignment, data exploration has already taken place and will not be repeated in this report. However, insights from the previous reports/analysis were used throughout this assignment.

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 300 dimensions. In order to use the embeddings, they were first downloaded through Pytorch and then for each sentence, words were matched with the same id the word is represented under in the GloVe embeddings. For words that are not included in GloVe, an unknown token was introduced, that was mapped with 0 and is the default token for the vocabulary. Also, a padding token was added (1). Then, with an Embedding layer in the model the sentences are translated to GloVe vectors.

Pytorch Tensors After data cleaning and tokenization, 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. Experiments with different batch sizes were also done, but some larger batch sizes led to out of memory errors, so it was not possible to be considered for training.

Maximum sequence length Given the very diverse lengths of the reviews feeding the models with the full reviews and by doing the necessary padding led to low scores (lower than the previous two assignments). This is an expected behavior and to resolve it a maximum sequence length is selected. The length picked is 115 tokens per review, given that after data cleaning the average review size is around that number. So, reviews with length above 115 are limited to the first 115 tokens, while smaller -in length- reviews are padded to reach 115 tokens.

2 Models

A main model has been created, with the ability to produce different architecture through parameters:

- Embedding Layer (GloVe)
- Bidirectional Stacked RNNs
- Linear Layer

The models examined included experimenting with LSTM and GRU cells, number of hidden layers, number of stacked RNNs, gradient clipping and dropout probability.

3 Training

In the previous assignment, the 300-dimension GloVe embeddings led a significantly better model, so in this assignment we're going to train model only using the 300-dimension embeddings. In order to select the models that perform well to further explore, for loops were built to check the performance in 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. Finally, it is chosen over the simple CrossEntropyLoss as it directly take cares of the needs of binary classification that we aim to achieve.

Optimizer Adam

Learning Rate Different learning rated were used, according to the training need of each model (eg GRU models were performing more consistently with the learning rate at $1e-4$, while LSTMs at $1e-3$).

Epochs 10 for model searching. When fine tuning and if required the epochs were adjusted.

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 required metrics (F1-score, Precision, Recall, Accuracy).

Type of cells LSTM and GRU cells

Number of Hidden Layers 60, 80, 100, 120, 140 and 160. Larger numbers were also tested, but due to low scores are not presented.

Number of Stacked RNNs 2, 3, 4 & 6

Gradient Clipping -1, 1, 2, 5 & None

Dropout Probability 0.25, 0.3, 0.4, 0.5 & 0.75

4 Results

Hidden Size To evaluate the hidden size, training was done for an LSTM and a GRU based model with 2 layers. The LSTM model's training was becoming less smooth as the hidden size was increasing. Regarding the GRU, larger hidden sizes results to better performance, however after 120 the models started to overfit.

Number of stacked Layers To evaluate the number of layers, training was done for an LSTM and a GRU based model given the most prominent results from experimenting with hidden size. The LSTM model's training showed that only the 2 and 3 layers models have quality results. For GRU, it was the best model was with 3 layers. This is expected due to the size of the dataset. To have more layer in the models, larger datasets are usually trained.

Gradient Clipping Gradient clipping seems an important factor that helps the model perform better. Out of all values, also taking into consideration the final fine tuning, are 1 and 5.

Dropout Probability Different probabilities were tried, however large probabilities do not produce as good results as smaller ones, as it was expected. So, probabilities up to 0.5 produce better results. Especially, for the GRU help the model have smoother learning plots.

5 Model Selection

The best model that is selected as the most appropriate for the task is a bidirectional stacked RNN, with GRU cells, hidden size 120, 3 layers, dropout probability 0.45 and gradient clipping size 5.
After training the selected model again and fine tuning, the final result is the following.

Metric	Score (%)
F1-score	84.941
Recall	85.829
Precision	84.073
Accuracy	84.781

The learning curves, as well as the confusion matrix and the ROC curve can be found below.

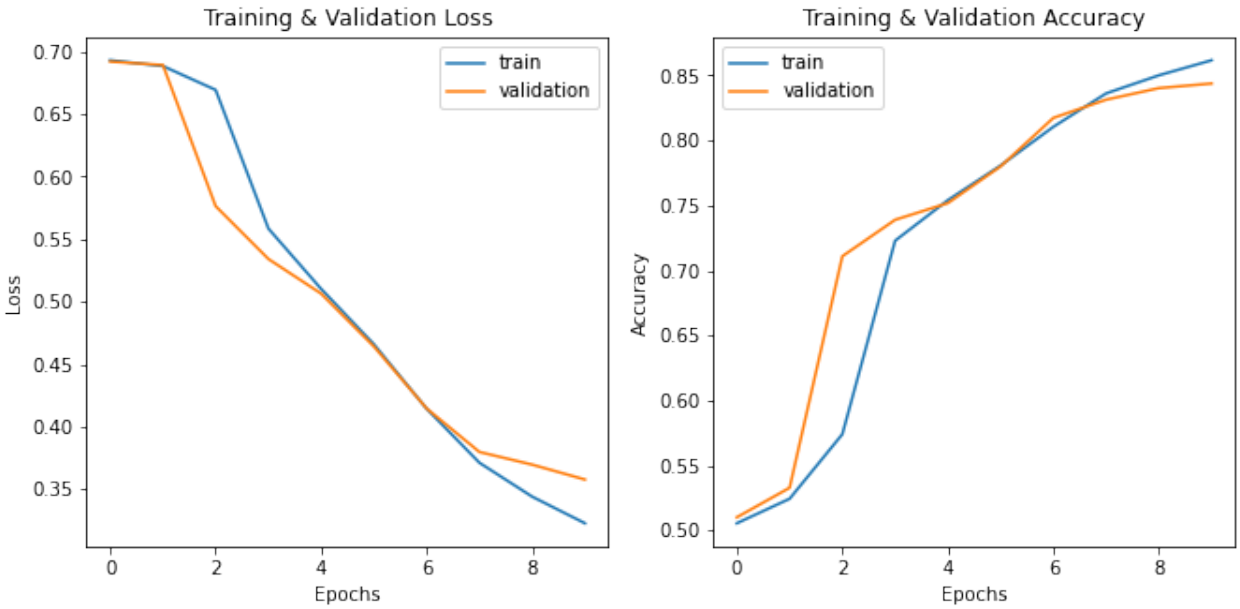
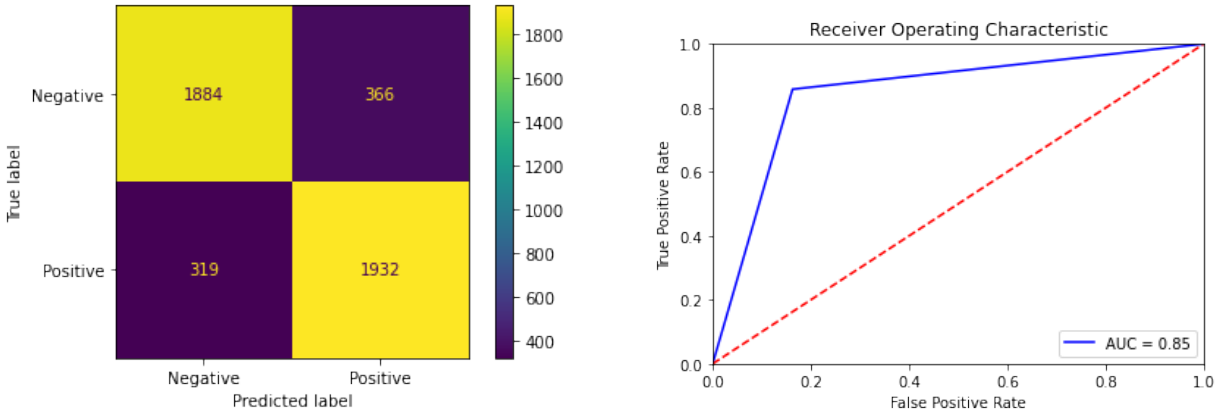


Figure 1: Final Model - Loss and Accuracy in Training & Validation sets



(a) Confusion Matrix

(b) ROC Curve

Figure 2: Final Model - Results in Test set

The model produced in the second assignment was slightly worse than the one produced now. I would expect the LSTM/GRU model have a bigger gap in performance compared to the simple Feed Forward, due to the increased abilities of RNNs. Even though I tried many things to get a better score, I couldn't get above 86% for F1 without overfitting, so I expect that maybe a more elaborate data cleaning and preparation would help (eg spellchecking).

Compared to the first assignment, the score is lower (F1: 88.33). Again I would expect an RNN to perform better than a simple logistic regression model, but it seems like it adds complexity that is not required.

Metric	Assignment 2	Assignment 1
F1-score	83.333	88.332
Recall	82.852	88.892
Precision	83.820	87.531
Accuracy	83.426	88.203

In conclusion, there might be a better data preparation or combination of hyperparameters to achieve better results as it would be expected from an LSTM, 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, but above the second.
To run the model on hidden test data, relevant instructions are given at the end of the notebook.

6 References

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