4rd Assignment: NLP with RNNs

CONTRIBUTORS: DIMITRA TSAKIRI (f3352123), PANTELEIMON SFAKIANAKIS(f3352121),

DIMITRIOS GKAVERAS(f3352104), ALEXANDROS SKONDRAS (f3352119)

The link for our Google Colab notebook can be found: link

Exercise 1

For the development of the sentiment classifier for this exercise, movie reviews were used from the Cornell Movie Review Data. More specifically, the dataset polarity dataset v1.1 was used, which contains 1386 movie reviews with average review length of 3560 characters, annotated into two mutually exclusive classes as either positive (1) or negative (0).

After downloading the files, the preprocessing of the reviews took place. There are two different types of preprocessing, one for the three baselines and one for the development of our RNN model. For the baselines, single characters and multiple spaces, numbers as well as non-word characters were all removed from the reviews. In addition, lemmatization was used to group the various inflected forms that a word might have as a single item. At this point, some statistics of the documents are shown in the following table.

Characters per Document		Characters per Negative Class			Characters per Positive Class		
Average	Vocabulary	Average	Min	Max	Average	Min	Max
3560	30797	3346	380	10109	3772	476	9919

The documents were randomly split into train, dev and test sets for the training of our classifiers, as shown below along with the class distribution:

Train	Test	Dev
831	278	277

Then, the features from the texts were extracted, using all the single words and bigrams and taking their TF-IDF scores. Stopwords were removed and the 5000 best features were selected according to their TF-IDF scores. Through the stated procedure, the most relevant words and bigrams (with the highest TF-IDF) score will be kept. Consequently, SVD was applied keeping eventually only 100 features, because the train set consists of only 831 reviews and keeping more features would potentially lead to overfitting. The dimensionality reduction that was achieved, is shown in the table of set shapes below:

Train		Test		Dev	
Before SVD	**After SVD**	**Before SVD**	**After SVD**	**Before SVD**	**After SVD**
(831, 5000)	(831, 100)	(278, 5000)	(278, 100)	(277, 5000)	(277, 100)

For the preprocessing we used for our RNN model, firstly, single characters and multiple spaces, numbers as well as non-word characters were all removed from the reviews. Then we split our data into train, test and dev which are the same as we did in the other preprocessing. Then we used the fasttext pre-trained embeddings on our data. Also, we convert our labels to one-hot vectors. At this point, some statistics of the reviews based on this new preprocessing are shown in the following table.

Average Length	Standard Deviation	Vocabulary
322.37	136.93	27316

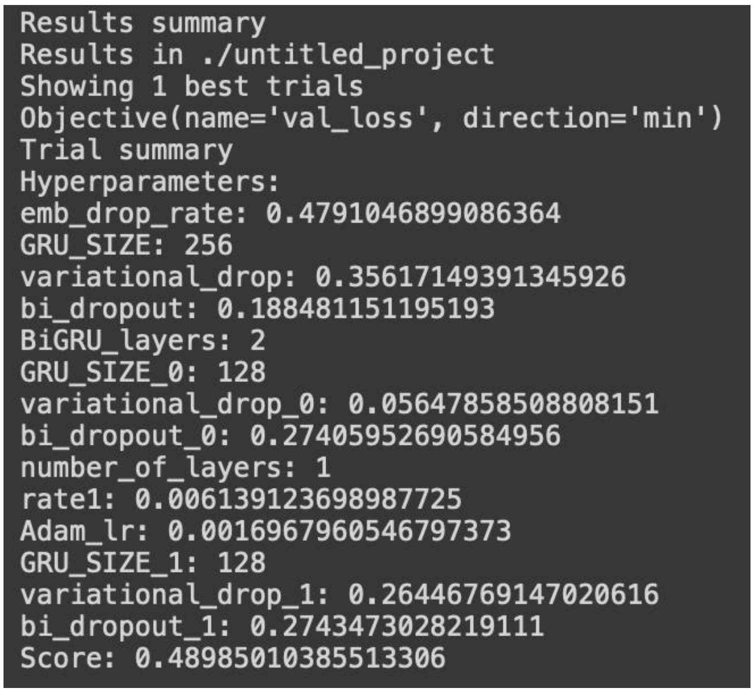
Baselines and RNN model

After the preprocessing phase, a baseline had to be set. For this task, the Dummy Classifier was used with 'most frequent' as strategy, which returns the most frequent class label as prediction for every test set item-review.

Based on the best classifier of our previous analysis, two extra baseline were set, which is a Logistic Regressor Classifier whose parameters were solver = "liblinear", C = 1, max_iter = 100, tol = 0.0001 and and MLP classifier with the use of Keras/TensorFlow. Our MLP model is a Sequential with an input Dense layer of 512 units and "relu" as activation followed by another Dense layer of 512 units and activation "relu". To close out the MLP a Dense layer of 1 unit and "sigmoid" activation was added. The model was compiled using binary cross-entropy loss and accuracy as metric.

Since baselines were set, we implemented an RNN classifier using Keras/TensorFlow. Keras tuner-BayesianOptimization was used in order to hyper-parameter tune the architecture of the model. The BiGRU with MLP model that we formed, was Bidirectional with an embedding dropout in (0, 0.5), an input BiGRU layer of units 256 with recurrent dropout in (0, 0.5) and dropout in (0, 0.5). This was followed by one or two possible BiGRU layers of 128 units and with recurrent dropout in (0, 0.5) and dropout in (0, 0.5) each. Consequently, the option of 1 Deep Attention layer (it can work for the options of 2 or 3 layers as well but in order to save time we tune for only 1 layer) followed by a Dropout with rate between (0, 0.5). To close out the BiGRU with MLP a Dense layer of 1 unit and "sigmoid" activation is added. The model is compiled using binary cross-entropy loss, Adam as optimizer with hypertuned learning rate in (0.001, 0.01) and accuracy as metric. The objective of BayesianOptimization was set to minimize the validation loss and along the tuning, callbacks were applied. More particularly, EarlyStopping with patience=3 (number of max epochs tolerated without improving), min_delta=0.0001, monitoring the validation loss and restoring the best weights from the best performing epoch in each trial was implemented, along with ReduceLROnPlateau with patience=3 and cooldown=0, which reduces the learning rate when the appointed metric has stopped improving. The BiGRU with MLP was hyper-tuned with a maximum of 10 epochs per trial, so that it can train to a satisfactory level before it gets interrupted by EarlyStopping in every trial.

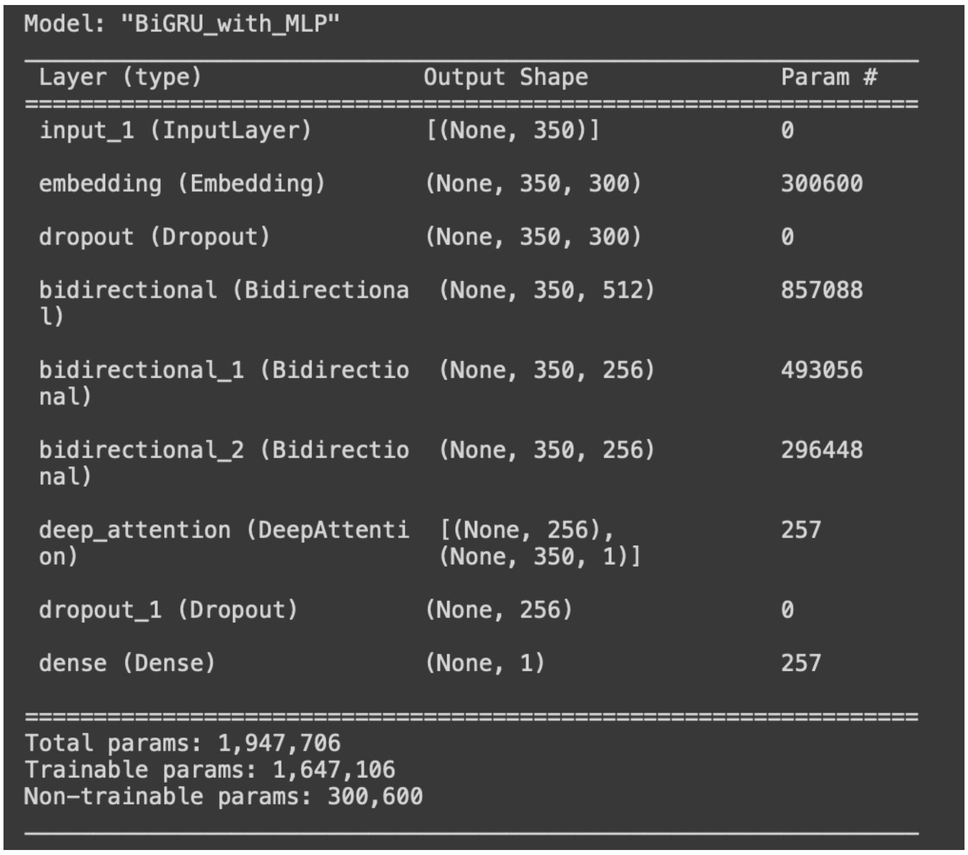
The result summary that defines the final architecture of the best model is showcased hereunder:



```
Results summary
Results in ./untitled_project
Showing 1 best trials
Objective(name='val_loss', direction='min')
Trial summary
Hyperparameters:
emb_drop_rate: 0.4791046899086364
GRU_SIZE: 256
variational_drop: 0.35617149391345926
bi_dropout: 0.188481151195193
BiGRU_layers: 2
GRU_SIZE_0: 128
variational_drop_0: 0.05647858508808151
bi_dropout_0: 0.27405952690584956
number_of_layers: 1
rate1: 0.006139123698987725
Adam_lr: 0.0016967960546797373
GRU_SIZE_1: 128
variational_drop_1: 0.26446769147020616
bi_dropout_1: 0.2743473028219111
Score: 0.48985010385513306
```

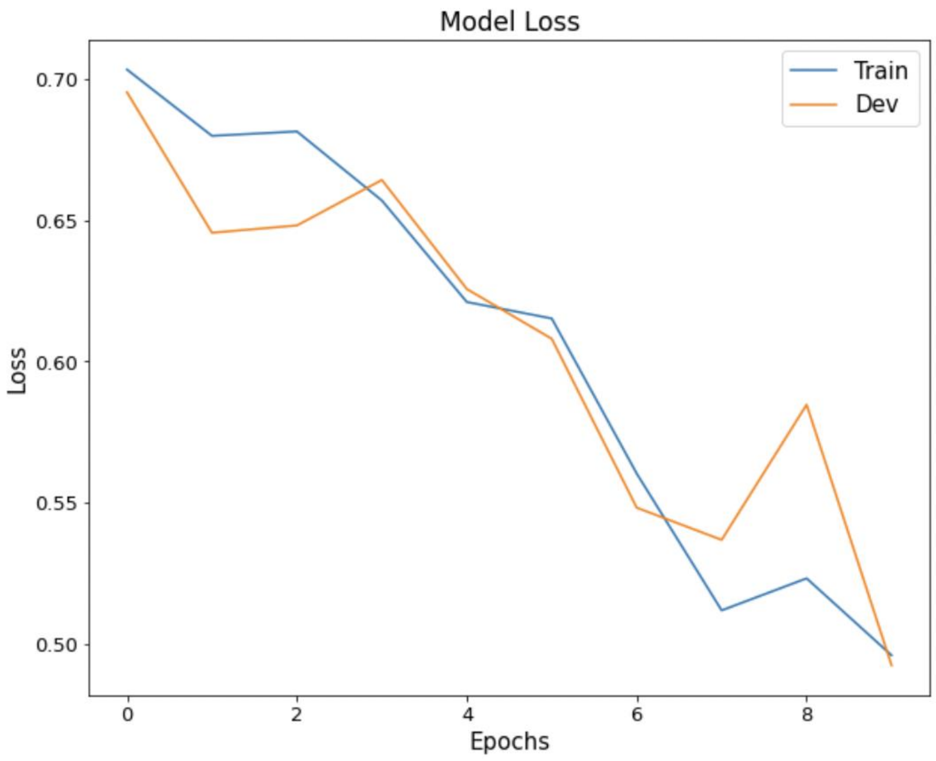
Note: If layers=1, then only parameters of index 0 are considered (matching the 1st layer). If layers=2, then index 1 parameters are also considered in the BiGRU with MLP.

Then, we reach the model with the following architecture, which, of course, agrees with the hyper-parameters summary described above.



```
Model: "BiGRU_with_MLP"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 350)]             0

 embedding (Embedding)       (None, 350, 300)          300600

 dropout (Dropout)           (None, 350, 300)          0

 bidirectional (Bidirectiona  (None, 350, 512)         857088
 l)

 bidirectional_1 (Bidirectio  (None, 350, 256)         493056
 nal)

 bidirectional_2 (Bidirectio  (None, 350, 256)         296448
 nal)

 deep_attention (DeepAttenti  [(None, 256),            257
 on)                          (None, 350, 1)]

 dropout_1 (Dropout)         (None, 256)               0

 dense (Dense)               (None, 1)                 257

=================================================================
Total params: 1,947,706
Trainable params: 1,647,106
Non-trainable params: 300,600
_________________________________________________________________
```

Learning Curves



In the learning curves of the training and development data sets, the loss was depicted as a function of epochs. As expected, the learning curves of both training and dev set tend to descend. Both curves continue to decrease after the 3rd epoch until the 7th epoch when they start to increase until the 8th. Thus EarlyStopping does not interfere and does not interrupt the training of the best BiGRU with MLP model.

Evaluation of the RNN model

The following tables show the precision, recall, F1 and precision-recall AUC scores, per class and classifier, separately for the training, development, and test sets along with the macro-averaged precision, recall, F1 and precision-recall AUC scores. Precision can be interpreted as the proportion of the data points our model says were relevant and actually were relevant. Recall can define the ability to find all relevant instances in a dataset. F1 score is the harmonic mean of precision and recall. Lastly, precision recall AUC scores are the area below the precision-recall curves, using various thresholds. For this exercise all these metrics were calculated for both the positive and the negative classes. It is worth noting that the higher these values are simultaneously, the better a model is.

Train data set

Est.	Precision		Recall		F1		PR-AUC		Macro Average			
	N	P	N	P	N	P	N	P	Prec	Recall	F1	PR-AUC
Dum	0.51	0.00	1.00	0.00	0.68	0.00	0.75	0.74	0.26	0.50	0.34	0.75
LR	0.85	0.86	0.86	0.84	0.86	0.85	0.93	0.93	0.85	0.85	0.85	0.93
MLP	0.90	0.95	0.96	0.89	0.93	0.92	0.98	0.98	0.98	0.93	0.93	0.93
RNN	0.87	0.82	0.82	0.87	0.84	0.85	0.90	0.91	0.85	0.85	0.84	0.91

Dev set

Est.	Precision		Recall		F1		PR-AUC		Macro Average			
	N	P	N	P	N	P	N	P	Prec	Recall	F1	PR-AUC
Dum	0.49	0.00	1.00	0.00	0.66	0.00	0.74	0.75	0.25	0.50	0.33	0.75
LR	0.76	0.80	0.81	0.75	0.78	0.77	0.86	0.84	0.78	0.78	0.78	0.85
MLP	0.78	0.80	0.80	0.78	0.79	0.79	0.88	0.86	0.87	0.79	0.79	0.79
RNN	0.77	0.73	0.70	0.79	0.73	0.76	0.84	0.82	0.75	0.75	0.75	0.83

Test data set

Est.	Precision		Recall		F1		PR-AUC		Macro Average			
	N	P	N	P	N	P	N	P	Prec	Recall	F1	PR-AUC
Dum	0.47	0.00	1.00	0.00	0.64	0.00	0.73	0.76	0.23	0.50	0.32	0.75
LR	0.74	0.84	0.84	0.74	0.78	0.78	0.83	0.89	0.79	0.79	0.78	0.86
MLP	0.77	0.85	0.85	0.78	0.81	0.82	0.85	0.90	0.81	0.81	0.81	0.87
RNN	0.75	0.78	0.75	0.78	0.75	0.78	0.82	0.86	0.76	0.76	0.76	0.84

Considering the previous tables , it can be concluded that, the MLP classifier accomplishes slightly better results compared to our RNN and to the other baselines. But, this is somthing expected because we could not hypertuned our model efficient due to lack of resources and time.

Thus, the final results are affected, of course, to a great degree in the hyper-parameters. It is worth mentioning that the results of the RNN are significantly better in the training set (e.g. Macro F1->0.84) and not as high as in the development and test sets (Macro F1-> 0.75 and 0.76, respectively). This notable difference in the results between sets though, could be attributed to the fact that our dataset consists only of very few examples (1386). As a consequence, small variations in the partitioning into train-dev-test of the initial dataset could easily affect the final results of the RNN metrics.

Manner of working on the assignment:

The presented assignment was processed with the equal participation of all team members/contributors. The team worked together in group Discord calls, during which the understanding and structure of the assignment were discussed and cleared upon. Subsequently, more Discord calls took place with one member each time sharing the screen, during which the code was created with the simultaneous attention/participation of all members. The code was each day at night worked upon or modified by each member individually, to eventually reach its final form.

The report was written in a similar manner.