Assignment 3 Report

Ziyang Lin

# **Data Preprocessing**

Besides the preprocessing that had been done in the last version of Funniness Estimation, the dataset now has been splitted into batches using Dataset and DataLoader from torch.utils.data and the batch size is 36.

New class labels are generated by rounding the mean funniness scores of edited headlines for the new pretrain classification task.

# **Models**

TwoInputsNN

TwoInputsCNN

TwoInputsRNN

TwoInputsConcatRNN

ClassifyRNN

ClassifyCNN

**TwoInputsCNN**:

This model uses text CNN architecture with single windows size instead of FFNN for the regression task. The original headlines tensor and the edited headlines tensor are taken as the two inputs. In the output layer, unlike the normal matrix multiplication, all weighted sums or all vector products between the n-th row of the original matrix and the n-th row of the edited matrix are computed such that a vector with the size (origin\_headlines\_num, 1) is returned.

**TwoInputsRNN**:

This model uses single layer bidirectional RNN architecture for the regression task. It is again the same as TwoInputsCNN that takes two tensors as its inputs and does a row-wise weighted summation in the output layer.

**TwoInputsConcatRNN**:

This model is all the same as TwoInputsRNN except it concatenates the two last hidden states for the original headlines and the edited headlines to form a single representation and do a normal matrix multiplication in the output layer.

*The below two models are used for pretrain classification task :*

**ClassifyRNN**:

This model uses single layer bidirectional RNN architecture for the 4-classes classification task. It takes the edited headlines tensor as the input and returns the probability for each of 4 classes.

**ClassifyCNN**:

This model uses text CNN architecture for the 4-classes classification task. The input and the output are the same as the ClassifyRNN.

# **Hyperparameters**

**TwoInputsCNN**:

EPOCHS = 500

LRATE = 5e-3

EMBEDDING\_DIM = 50

FC\_OUT\_DIM = 25

N\_OUT\_CHANNELS = 100

WINDOW\_SIZE = 3

DROPOUT = 0.7

**TwoInputsRNN**:

EMBEDDING\_DIM = 50

HIDDEN\_DIM = 128

FC\_OUTPUT\_DIM = 32

BIDIRECTIONAL = True

DROPOUT = 0.3

LRATE = 1e-4

N\_EPOCHS = 30

**ClassifyRNN**:

EMBEDDING\_DIM = 50

HIDDEN\_DIM = 128

OUTPUT\_DIM = 4

BIDIRECTIONAL = True

DROPOUT = 0.4

LRATE = 1e-4

N\_EPOCHS = 10

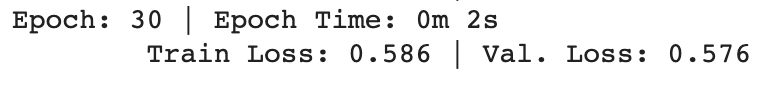
**ClassifyCNN**:

N\_OUT\_CHANNELS = 100

WINDOW\_SIZE = 3

# **Results**

**TwoInputsRNN**:





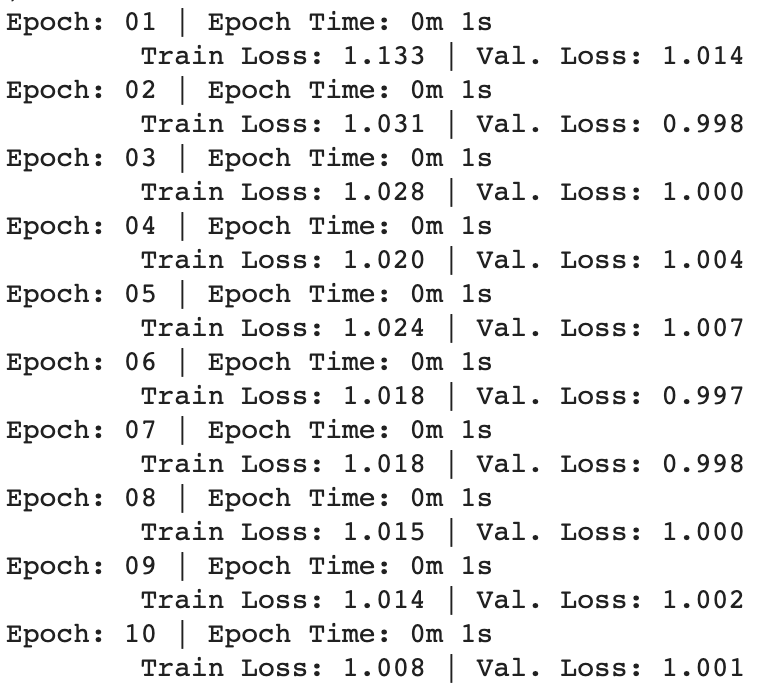
**TwoInputsCNN**:



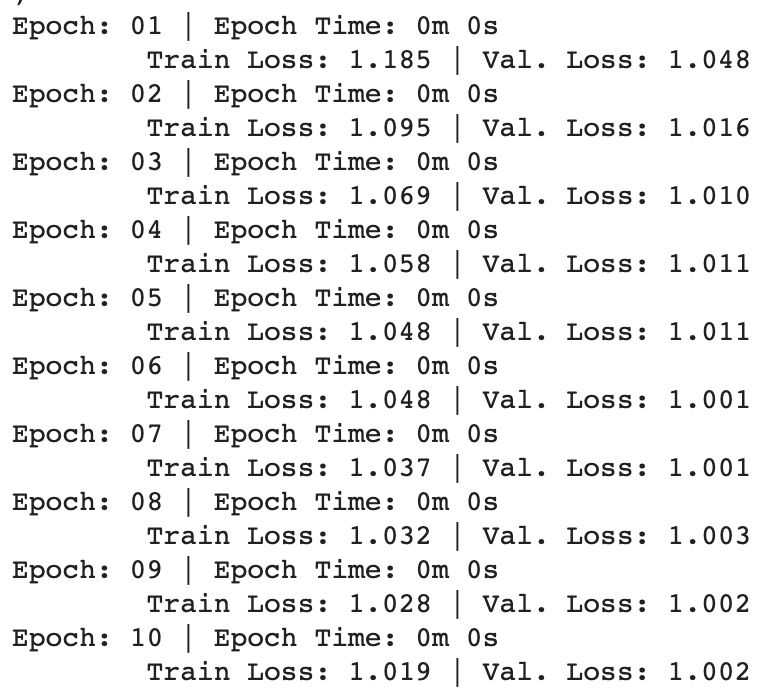
**TwoInputsConcatRNN:**

The overall results are worse than the results of **TwoInputsRNN.**

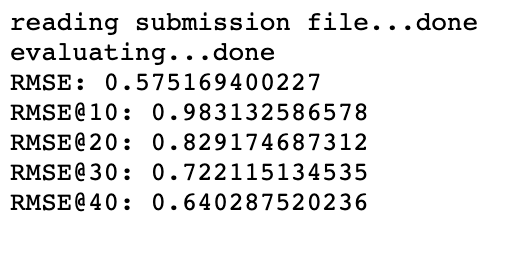
**ClassifyRNN**:



**ClassifyCNN**:



**Scoring Output Log on the Codalab**



# **Discussion**

The best scores or the lowest loss is achieved by the TwoInputsRNN while the performance is just slightly improved compared with that of the previously designed TwoInputsNN (0.5759702196 vs. 0.5751694002 ) and the time complexity is much higher than the TwoInputsNN, at some point the current version of TwoInputsRNN is resources-wasted.

The TwoInputsCNN with a single window size performs worse than the TwoInputsNN and the TwoInputsRNN, and one possible reason is that it only looks at one size of n-gram and hence ignores the knowledge of n-grams with different lengths.

One trial that has been tried to improve the training is firstly make a model to execute a pretrain 4-classes classification task to learn to assign the most likely humor class label to each new edited headline correctly and then take the embeddings layer of the model for later use. Whereas both the models ClassifyRNN and ClassifyCNN failed to converge after tuning parameters and when using their embeddings to initialize the regression model the performance does not change.

# **Prospective**

* Construct a pretrain LM to do a binary classification task in which the model learns to decide whether a word from the edited new headline is original or edited. Take the embeddings out of the pretrain model and use it to initialize the model for the real regression task. By doing so we expect the embeddings can be informed some knowledge about the relationship between original headlines and edited headlines.
* Build up a pretrain LM to do a text translation task on the training dataset and use the embeddings of this model to initialize the model for the real regression task. (Aim to learn the semantics of funniness)
* Intuitively thinking the performance of the TwoInputsCNN might be improved by increasing the number of the window sizes (different n-gram filters).
* Use the SOTA pretrain LM ALBERT / ERNIE
* Will use LaTex and write more formal report next time