Total Grade: 89.5/100

NLP Coursework Report

Lichen Xue lx320 | **Ziyang Yang** zy820 | **Yiming Zhang** yz12120

Detailed Feedback:
You have done a very good job with this submission! You have conducted a variety of experiments for both approaches and achieved good results. Additionally, you explain all your decisions and build upon your findings. The only weak point of your submission is the quality of your report, which could be improved in a few ways (see detailed feedback). Overall though, this is a great effort..

(1) Clarity and insightfulness of the written report: 15/20
Your report has a mostly clear structure and is easy to follow. However, the language of the text is quite repetetive, the writing is not very academic and at parts gets complicated. It could be improved in a few ways, by making a few passes on the text and polishing, avoiding repetitions and keeping the sentences short. Additionally, you could have included a brief introduction to the data. Finally, you have no citations, but it's good practice to remember adding at least the citations of the algorithms/models you use, even in coursework reports.

(2) Organization and documentation of code: 19/20
You include a readme and some comments, so overall your code is well documented and mostly clear

(3) Creativity of linguistic pipeline (Marks x1.5): 18/20
You have experimented with different types of input as well as different preprocessing pipelines throughout the experiments and mostly explain your choices and build on your findings. There was room for further discussion on the data, especially before starting building your models and some data exploration would have assisted you further on that. Finally, you elaborate on the performance of your various models and analyse your errors, but mostly identified some algorithmic issues. As a complementary analysis, you could look at any false predictions to understand whether there is any pattern on them.

(4) Appropriate use of machine learning methodology (Marks x1.5): 19/20
You experiment with a variety of algorithms and models, inputs, preprocessing and hyperparameters, and you have done so very well. You reason well on your choices and build more sophisticated models based on previous findings. There is plenty of reporting per approach and in comparison, overall a good work.

Best of luck with the module!

different inputs and 3 models (CNN, BiLSTM and GRU). And also there are comparisons about every data preprocessing step on prediction when other hyperparameters stay the same.

Here is the URL of the colab file containing all our training processes and validation results: Colab notebook

2.3 Discussion about data preprocessing and input

In approach 1, with the pretrained tokenizer, all the punctuation will be embedded as well, which means that it is unnecessary to remove punctua-

model	input	RMSE	model	input	RMSE
bert-base-uncased	orgSentence	0.55341	bert-base-uncased	editedSentence	0.52812
bert-base-uncased	orgSen + editWord	0.53879	bert-base-uncased	orgSen + editSen	0.5176
roberta-base	orgSentence	0.57953	roberta-base	editedSentence	0.56991
roberta-base	orgSen + editWord	0.57594	roberta-base	orgSen + editSen	0.56805
xlnet-base-cased	orgSentence	0.62491	xlnet-base-cased	editedSentence	0.60717
xlnet-base-cased	orgSen + editWord	0.58303	xlnet-base-cased	orgSen + editSen	0.57998

Table 1: best performances of each model with their input for Pre-Trained models

tion in approach 1. In the experiment process, the test RMSE with removing is higher than the test RMSE without removing.

There is an interesting phenomenon in the training data set. The grades of some sentences are just zero. There are two possible reasons. The one is that these sentences were actually judged with zero by all the referees, and their mean grades are 0. Another reason is that these sentences were omitted by referees, and there mean grades are not actually 0. During the experiment, we compare the result with and without removing the sentences with just 0 mean grade, and the result without removing the sentences is better. The RMSE gap between the two results is around [0.005, 0.02].

Among these different input methods, the combination of the original sentence and the edited sentence as shown in Table 1 will make the model get the best effect. And also the inputs that include both original information and edit information have better performance than any single input. In details, the two sentences can combine a context relationship, and they are divided by a [SEP] symbol. Such as the two sentences, "I like IC." and "I like UCL.". The combination is "[CLS] I like IC.[SEP]I like UCL.[SEP]".

2.4 Pre-trained model

There are 3 kinds of pre-trained models we have tried, which also are shown in Table 1, BERT model, XLnet model, RoBERTa model. In this coursework, based on bert we got the best performance (RMSE : 0.51760). So we take the bert as the main example to illustrate.

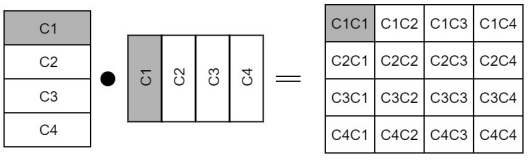

Figure 1: Dot multiply to get self attention matrix

There is a significant component in the transformer, the multi-head self-attention. The input will do the dot multiply to the word vector matrix and its transposed matrix in one sentence (let batch size = 1), which could yield a square matrix N*N (N = the sentence length). As shown in Figure 1, N=4. In this way, we could get the bidirectional relationship between different words in a sentence. By increasing the batch size, this process can be executed more efficiently.

In fact, each word is embedded in 768 dimensions. And the smallest unit is one word. BERT has 12 transformer layers, 12 heads and 30,000 tokens. Each head generates embedding of length 64(768/12), and with the three embeddings generated "Q, K, V", and additional weight, the encoder has 2362368 parameters. More details about BERT can be found in GitHub.

We also used RoBERTa, and expect it to give us a better performance. Because it improves the mask to dynamic, and its batch size could be larger. And we also hope XLNet, which uses two-stream Self-Attention instead of the Mask, could give us a better result. But the result was not well as BERT with the same hyperparameters. One reason might be both two models need more training data and more epochs to train.

2.5 Training process and Training results

In approach 1, for all the pretrained model (XLnet, RoBERTa, Bert) we used, we set the label number to 1 and add a regression layer to calculate the output. During the training, we define the RMSE as the loss function and AdamW as the optimizer. Additionally, the scheduler is used to update the learning rate during the training process.

And we also noticed an interesting phenomenon, the max length of input also influences the result. For example, we notice the actual max length of all single sentences is 27, so the actual max length of input (two sentences) is 54. But instead of the actual max length, we set the max input length to 60. After converting the length of all sentences into the longest sentence length, we use [pad] to fill the vacancy of input. Accord-

ing to the result, taking 60 but not 54 as the max input length has a better performance. We think the phenomenon might relate to the label smoothing. Proper zero padding can reduce the influence of extreme training samples on the model. But when the number of extra pads grows too large, the empty pads also take part in the precision. Table 2 shows some example during the hyperparameters tuning for BERT, and the RMSE of the test data.

lr	input	WDeacy	maxlen	RMSE
0.007	editW+newS	0.005	35	0.53879
0.005	oriS+newS	0.005	52	0.53551
0.008	oriS+newS	0.0001	60	0.53209
0.01	oriS+newS	0	60	0.51951
0.01	oriS+newS	0	60	0.5176

Table 2: bert-base-uncased hyperparameters tuning

**WDeacy = weight decay, oriS = original sentence, newS = edited sentence, editW = edited word.
After struggling with the hyperparameter tuning (random grid searching manually), we get the best performance with RMSE = 0.5176, ranked 5th on codalab competition website.

3 Approach 2: non pre-trained model

Without the help of pre-trained and corpus, we tried different machine learning models to see if different models have a decisive impact on the prediction results. At the same time, we use the method of controlling variables to observe the results of different inputs to the same model. Through this method, we want to find out whether there is a preprocessing step that has a great impact on the results.

3.1 Data Preprocessing

After composing the original sentence with edit word to form a new sentence, There will be four preprocessing steps to conduct comparative experiments on the impact of the prediction results, as shown below:
(1) : Remove rows that value in "grades" equals '0' in the training data frame.
(2) : Lower case all words in both original and edited series.
(3) : Remove punctuation in original series.
(4) : Remove punctuation in edited series.

The reason we try to remove the sentences which score equal to exactly 0 is it might help the model predicting extreme values.

Table 4 shows the combination of several data preprocessing steps we tried and the prediction results of the GRU models using three kinds of inputs as the test object. The first column in the table is the combination of preprocessing steps, and the digital coding remains the same as step 1 to 4 above.

Overall, the four steps have no negative impression on the predicted results. The removal of all punctuation (step 3 and 4) does not significantly improve the prediction effect, while the removal of zero score sentences (step 1) has a significant improvement.

Even though step 1 seems to have a significant effect, it deleted the data for 500 rows. This operation reduces the data of the training set. For the final model training, whether the smaller training set version is more suitable for the current hyperparameters is also a point to consider. Therefore, it does not seem to be able to directly conclude that deleting rows with a score of 0 can help prediction. In order to verify this step, we added it to the data preprocessing of approach 1 and observed the results. What we got here is that deleting so many rows will directly reduce the predictive ability of the model, and the loss value will increase significantly.

3.2 Input and how it influence the prediction

In this approach, we have only compared three types of inputs, original sentence, edited sentence and both of them.

Different from the situation where "original sentence + edited sentence" is the best input in approach 1, in the internal comparison of the three neural network models in this part, it is better to only input the edited sentence. In this part, all three kinds of neural networks are not performing well enough in two inputs. By observing the loss value of the training set, we found all models have been underfitting during the training. Insufficient network dimensions may be the direct reason they cannot achieve learning effects.

3.3 Training models, hyperparameters tuning and results

Table 3 records the best scores of each model in approach 2 and their important hyperparameters.

In approach2, we used the RMSE as the loss function, adam as optimizer, and there is a linear layer in the output layer to achieve regression. From a model type point of view, BiLSTM and

model	inputData	lr	embedDim	outChan	scheduleStep	epoch	validLoss	testLoss
TF-IDF	original	/	/	/	/	/	0.604	0.607
CNN	both	0.0008	100	100	20	1500	0.597	0.615
CNN	original	0.00085	90	100	30	220	0.595	0.614
CNN	edited	0.009	700	100	25	300	0.598	0.617
BiLSTM	original	0.0001	30	10	12	15	0.549	0.587
BiLSTM	edited	0.0001	30	10	12	6	0.554	0.579
GRU	both	0.0001	50	10	30	10	0.591	0.613
GRU	original	0.0001	25	10	18	18	0.554	0.578
GRU	edited	0.0001	25	10	18	18	0.552	0.577

Table 3: best performances of each model and their hyperparameters

preprocessing	2 inputs	1 in(edit)	1 in Org
none	0.612	0.620	0.628
1	0.601	0.587	0.588
2	0.624	0.612	0.612
1,2	0.584	0.581	0.583
2,3	0.631	/	0.613
2,4	0.627	0.811	/
1,3,4	0.597	0.574	0.576
1,2,3,4	0.586	0.577	0.578

Table 4: uses GRU as the test object, and all other hyperparameters are kept consistent with those recorded in Table 3

GRU models had a big problem at the beginning. They were never converging at the beginning of our training. The minimum loss values appear in the first epoch, thus we think there was an exploding gradient. After smaller embedding dimensions from 300 to 30 and a smaller learning rate to 0.0001, we solved that problem. Judging from the test results conducted on CodaLab, GRU performs slightly better than BiLSTM. There is not much difference between the two.

To deal with the exploding gradient problem, when we design the CNN models, the learning rate schedule has been added during training to help optimize the model. The results show this strategy works. There is a clear downtrend of validation loss during training. And reducing the schedule step to a certain extent will also help CNN get a lower loss value.

However, CNN does not perform well in the prediction results, even if the learning rate is dynamically adjusted through the scheduler method and we also tuned other hyperparameters. This is clear from the results in Table 3. On the other hand, Table 3 records the best performance results of each model, and from our many attempts, the prediction results of the three CNN models are often lower than the baseline. This may be due to the fact that CNN cannot process word information over long distances, which results in the lack of information of position, resulting in greater deviation. Therefore, we think CNN is not suitable for this problem of predicting the level of humour.

4 Two approaches compare

The reason that pre-trained models are better than non pre-trained models we boil down to two points:
(1): In BERT, the self-attention mechanism can obtain the relationship between a word in a sentence with another word in the same sentence. However, CNN cannot identify the relationship of the context, and this input sequence affects the location information originally contained in the training set.
(2): Using the Bert-base-uncased word vector provided by Transformers, the word dimension is 768. So many dimensions enable the data to be better learned to obtain high-precision prediction results. However, the obvious point in approach 2 is that when we use the original sentence and the edited sentence, all models cannot learn well. It directly reflects in the loss of the training set, which means the loss does not decrease during the whole training. Therefore, the final training results were not satisfactory. Two other models under the same network with only one input at a time can achieve better training effects through fine-tuning of parameters.

5 Conclusion

After trying the two approaches, we found that the overall performance of the pre-trained model is better, and the uncased BERT base is the most suitable for this problem. In approach 1, without removing the punctuation and sentences with 0 points, the result is better. But as shown in the comparison of approach2, the biggest influence on the result is to delete the items that are having 0 in the average score.