Assessing the humor of edited headlines

Rahul Modak
MT2020014
Rahul.Modak@iiitb.ac.in

Sujit Kumar
MT2020106
Sujit.Kumar@iiitb.ac.in

***Abstract*—We will attempt to solve the SemEval-2020 Task 7: Assessing Humor in Edited News Headlines. Here we will estimate the funniness of news headlines that have been modified by humans using a micro-edit to make them funny. The task include two sub task:**

1) **Regression: Given the original and the edited headline, we are require to predict the mean funniness of the edited headline.**
2) **Predict the funnier of the two edited headlines: Given the original headline and two edited versions, we need to predict which edited version is the funnier of the two.**

***Index Terms*—NLP,BERT,Sequence classification,bert tokenizer**

I. Introduction

Humor detection is difficult task to comprehend in natural language processing. Humor occurs in various intensities, that is, certain jokes are much more funnier than others. Nearly all existing humor datasets are annotated to study whether a chunk of text is funny or not. A system's ability to assess the intensity of humor makes it useful in various applications, for example, humor generation where such a system can be used in a generate-and-test scheme to generate many potentially humorous texts and rank them in terms of funniness.

For semval-2020 task7, We are provided with a dataset that contains news headline with micro-edits to make them funny. Our task comprised of two sub-tasks:

A. Funniness Regression

In this task, given the original and the edited versions of a headline, the we have to estimate the mean funniness of the edited headline on the 0-3 humor scale.

B. Funnier of the two

In this task, given the original headline and two of its edited versions, the system has to predict which edited version is the funnier of the two.

II. Dataset

The data for task1 is Humoroedit[1] dataset. The dataset contains about 5000 original headlines, each having three modified, potentially funny versions for a total of 15,095 edited headline. The headline are collected from Reddit and micro-edits are applied to make them funny. Micro-edits are as follow:

Replaced	Replacement
entity	noun
noun	noun
verb	verb

Further each headline is assigned with grade as one of the following:

Grade	Meaning
0	Not Funny
1	Slightly Funny
2	Moderately Funny
3	Funny

The ground truth funniness of each headline is the mean of its five funniness grades. Sample data points from the training set are shown below:

Id	**Original**	**Edits**	**Grade**	**Mean Grade**
14530	France is ' hunting down its citizens who joined <Isis/>' without trial in Iraq	twins	10000	0.2
13034	Pentagon claims 2,000 % increase in Russian trolls after <Syria/>strikes . What does that mean ?	bowling	33110	1.6
8731	Iceland PM Calls Snap Vote as Pedophile Furor Crashes <Coalition/>	party	22100	1
76	In an apparent first , Iran and Israel <engage/>each other militarily	slap	20000	0.4

For the task1,data set is randomly sampled into train (64%), dev (16%) and test (20%)

For task2 we have training data set which have two edits of same sentence with their mean grades and the label

Labels of task 2
0 Both edits are equally funny
1 1st edit is more funny
2 2nd edit is more funny

Below dig shows the count of labels in the training data set.Which is done using countplot.

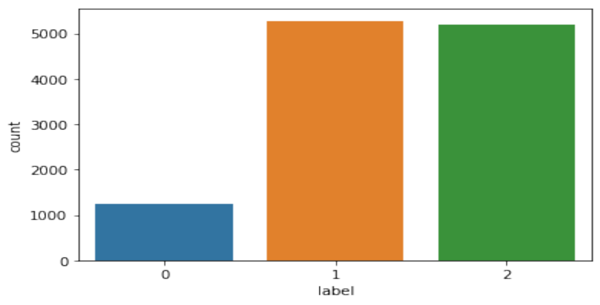


Fig. 1. Count of labels in training data

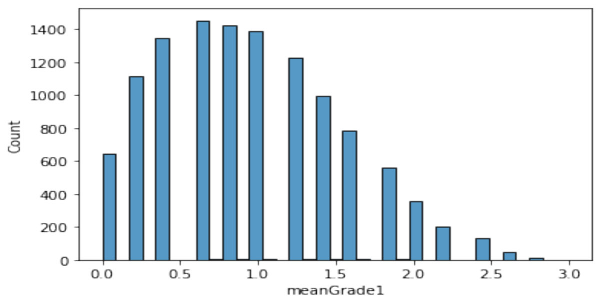


Fig. 2. Histogram plot of meangrade1

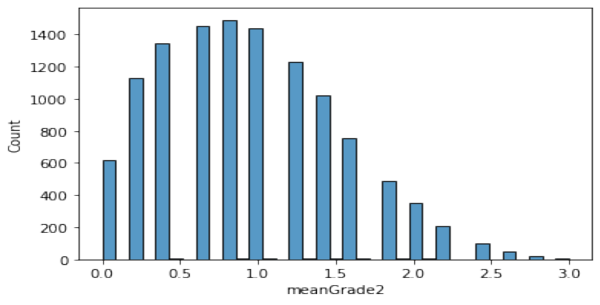


Fig. 3. Histogram plot of meangrade2

Task	Type	Metric	Train	Dev	Test
Subtask1	Regression	RMSE	9653	2420	3025
Subtask2	Classi-cation	Accuracy	9382	2356	2961

III. Data Pre-processing

For both task1 and task2 model training is same i.e fine tuning bert according to our task part is same therefore the code of data processing part will be same.

By assessing the task on our hand we decided that we will be using BERT sequence classification using sentence pair as it perfectly fits our problem.In This sentence pair classification each sample of data set have two sentences and a label/value.Now specific to BERT(as mentioned in the documentation[3]) each sample should look like this.

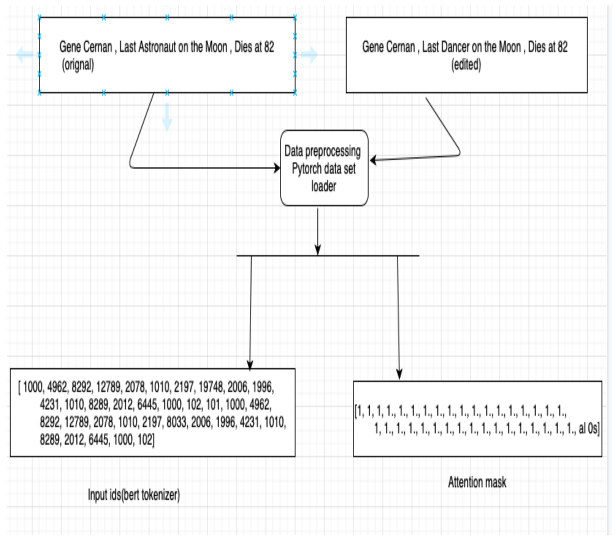
[CLS] sentence 1 [SEP] sentence 2 [SEP]

Now in our case sentence 1 will be the edited sentence and sentence 2 will be the original sentence.

For tokenization i.e converting these texts to numerical vectors we will use BertTokenizer.We will select a max length to bring all the vectors to the same length so for that either we pad shorter length vectors or clip longer length vectors.

Now as in the case where we have to pad shorter length vectors with some pad token which is not necessary in our training so to mask it out we will use attention mask which will be the size of the max length and will have 1's up until original vector was and 0's corresponding to the padded tokens.In the case of vectors bigger than max length attention mask will have all 1's.

Finally we have grade which will act as target variable (eg. 1.2,2.5 etc) for training the model which shows scale of how much funnier the original sentence got after editing.



Gene Cernan , Last Astronaut on the Moon , Dies at 82 (orignal)
Gene Cernan , Last Dancer on the Moon , Dies at 82 (edited)
Data preprocessing Pytorch data set loader
[1000, 4962, 8292, 12789, 2078, 1010, 2197, 19748, 2006, 1996, 4231, 1010, 8289, 2012, 6445, 1000, 102, 101, 1000, 4962, 8292, 12789, 2078, 1010, 2197, 8033, 2006, 1996, 4231, 1010, 8289, 2012, 6445, 1000, 102]
[1, 1, 1, 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1, 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., al 0s]
Input ids(bert tokenizer)
Attention mask

Fig. 4. Data processing pipeline

Above figure shows one pair of original and edited sentence processed using data processing pipeline and in the last we got two things input ids and attention mask.

IV. Model Used

For dealing with this problem we are using BERT[2] Bidirectional Encoder Representations from Transformers which are designed to pre-train deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers. As a result, the pre-trained BERT model can be fine-tuned with just one additional output layer to create state-of-the-art models for a wide range of tasks, such as question answering and language inference, without substantial task-specific architecture modifications.Specifically we used ”BERTForSequenceClassification”[3] which is a Bert Model transformer with a sequence classification/regression head on top of a linear layer on top of the pooled output.For the tokenization purpose i.e converting text to vectors ”Bert-Tokenizer” is used.

V. Training

In this we are using Bert for sequence classifier which takes sentence pair as input and give a output by extracting semantic linking between these two sentences.Now this output can be a classification or a regression output,in our case we want a regression output i.e predicting the grade(funniness of the edited sentence).For this in model parameters we will set num labels=1 which will make the model work as a regressor and we will use MSE loss for training.Now we can use this for training our model for task1 using train.csv as our training data and predict grades for task1 test data.Now for training we used Pytorch autograd[5] module which makes back propagating the loss really easy.Now from BERT for Sequence classifier we get many outputs out of which we will use two loss and logits.This loss will be MSE loss and we will back propagate this loss and train BERT for our specific task.Now in this not the whole BERT model gets trained but only the a nn(pytorch) layer gets trained which is specified inside “BertForSeqClassification” class(mentioned in documentation).See the image below we have three layers first one BERT model which is freezed(doesn’t get trained), second is drop out layer and third one is nn.Linear layer which is a pytorch layer[5] which gets trained.We used GPU for training.

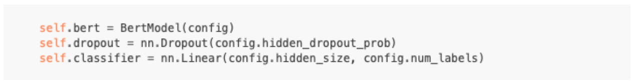

```
self.bert = BertModel(config)
self.dropout = nn.Dropout(config.hidden_dropout_prob)
self.classifier = nn.Linear(config.hidden_size, config.num_labels)
```

Fig. 5. Inisde ”BERTForSequenceClassification” class from doc

For task2 we will use the same approach as in task1 and train.csv from task2 data set for training our model.In task2 finally we want to predict 0,1,2 which we will do on the basis of this predicted grade for each edit.Task2 pipeline dig shows the steps for task2.

Now, we will run this trained model on our test data set for predicting the grade of each edited sentence in each test sample.

After that we will label each test sample using labeling logic which is described as follows

We have a sentence S1 that has two edits E1 and E2 now our model will predict the grade for this E1 and E2.

We will decide the label following way

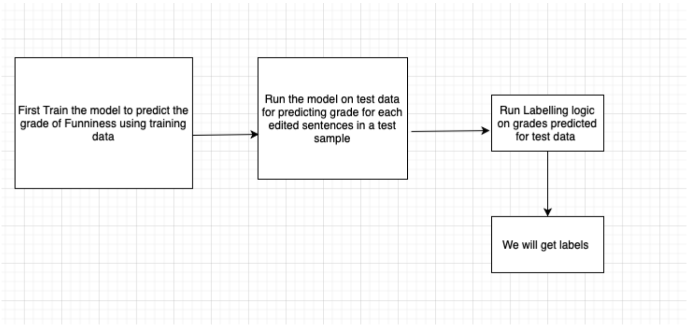

First Train the model to predict the grade of Funniness using training data
Run the model on test data for predicting grade for each edited sentences in a test sample
Run Labelling logic on grades predicted for test data
We will get labels

Fig. 6. Project pipeline

1) If grade(E1) ¿ grade(E2) our label for this sample will be 1
2) If grade(E1) ¡ grade(E2) our label for this sample will be 2
3) If grade(E1) = grade(E2) our label for this sample will be 0

VI. Evaluation

For Subtask 1, systems are ranked using the root mean squared error (RMSE) between the mean grade and the rating estimated by the system for the headlines. Given N test samples, and given the ground truth funniness y_i and the predicted funniness $\hat{y}_i$ for the i-th sample:

$$RMSE = \sqrt{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2/N}$$

For Subtask 2, which attempts to find the funnier of the two modified versions of a headline, the evaluation metric is classification accuracy. We also report another auxiliary metric called the reward. Given N test samples with correct predictions, and given the i-th sample, the funniness ratings of its two edited headlines $f^{(1)}{}_i$ and $f^{(2)}{}_i$, its ground truth label y_i and its predicted label $\hat{y}_i$:

$$Accuracy = C/N$$

VII. Result

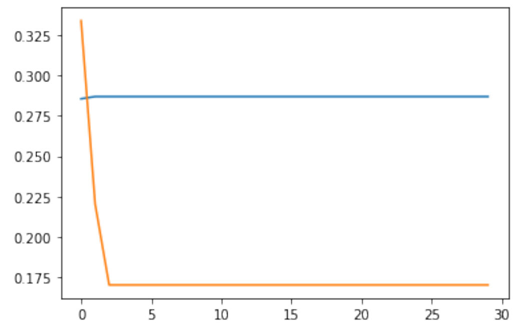


Fig. 7. Training and validation loss graph for task1(blue: validation loss, orange: training loss)

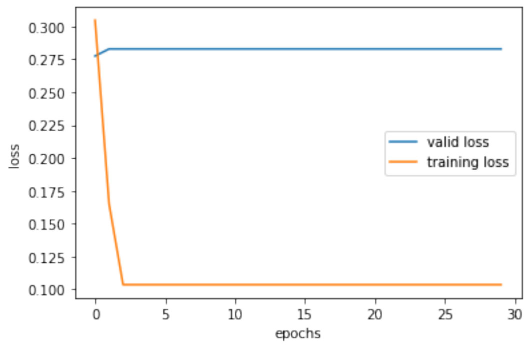


Fig. 8. Training and validation loss graph for task2(blue: validation loss, orange: training loss)

BASELINE: assigns the mean rating (Sub-task 1) or the majority label (Subtask 2) fromthe training set.

Model	Task1: RMSE	Task2: Acc
Baseline	0.575	0.490
Team at 1st	0.49725	0.67428
Our model	0.5416	0.57

VIII. Conclusion

Through BertForSequenceClassification, we have successfully completed task1 with RMSE of 0.5416(top scoring team in evaluation-task1: 0.49725) and task2 with accuracy of (top scoring team in evaluation-task2: 0.67428).

IX. Acknowledgement

We would like to thank Professor G Srinivasaraghavan for giving us the opportunity to work on this project and help us whenever we were struck by giving ideas and pointing out to specific resources that we could learn from. Also, we would like to thank the all teams for sharing their ideas and having open ended discussions. We really had a great learning experience while working on this project.

References

[1] ”President Vows to Cut Taxes Hair”: Dataset and Analysis of Creative Text Editing for Humorous Headlines $https://www.aclweb.org/anthology/N19-1012.pdf$

[2] BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding $https://arxiv.org/abs/1810.04805$

[3] Hugging face Documentation $https://huggingface.co/bert-base-uncased$

[4] The Annotated Transformer $https://nlp.seas.harvard.edu/2018/04/03/attention.html$

[5] $https://pytorch.org/tutorials/beginner/deep_learning_nlp_tutorial.html$