2020

NLP Sentiment Analysis

Funniness Estimation for Edited News Headlines

Lab Report

Ziyang Lin

July 25, 2020

Tasks

- **Task one** - Given one edited headline, design a regression model to predict how funny it is
- **Task two** - Given the original headline and two manually edited versions, design a model to predict which edited version is the funnier of the two

Data Preprocessing

Task One

- Convert original headlines into normal sentences (Remove “<” and “/>” by applying RE)
- Get the edited version of headlines by doing word substitution using RE
- Do tokenization and lowercasing for each edited-original headlines pair

Data preprocessing for pre-trained LMs (BERT-liked LMs):

- Version 1 - Concatenate original headlines and new headlines

- Version 2 - Concatenate new headlines and new words
- Version 3 – Contain only new headlines

Task Two

There are 3 versions of data preprocessing:

- The normal version
- The headlines truncated version
- The punctuation removal version

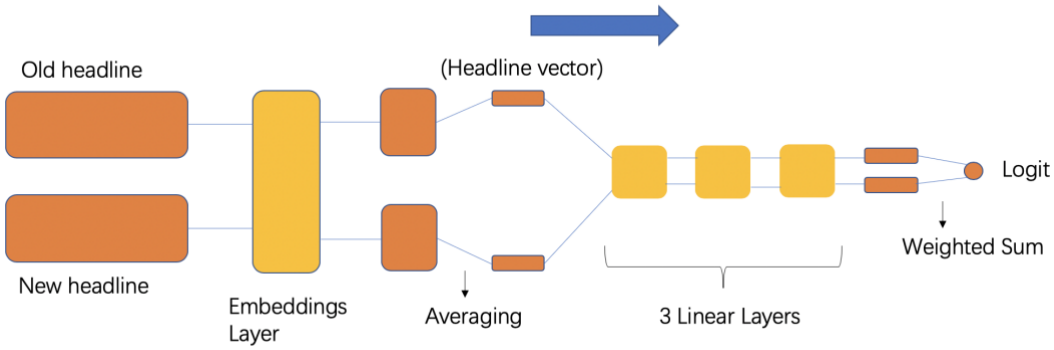
Models Choices & Design

Task One

- Two Inputs FFNN
- Two Inputs CNN
- Two Inputs RNN
- Two Inputs Concatenated RNN
- Pre-trained LM + a regression layer (LMs applied: BERT, ALBERT, XLNet, ELECTRA)

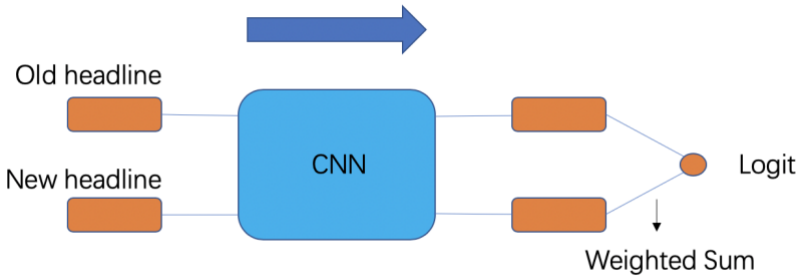
Two Inputs FFNN

This model is a two inputs' feed forward neural network in which two input matrices representing all the original headlines and their corresponding edited headlines respectively are passed simultaneously to the first so called embedding layer of the model to get the word embedding of the fixed dimension for each word in the headline. Following above the model will do averaging for each headline to get the 'document representation (vector)' for each headline. Then these headlines' vector representations are passed to a combination of three concatenated fully connected layers where the information about "how humour they are" are encoded. The Relu activation is applied after output from each of the first two hidden layers to prevent gradient vanishing and gradient exploding. Finally, 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.



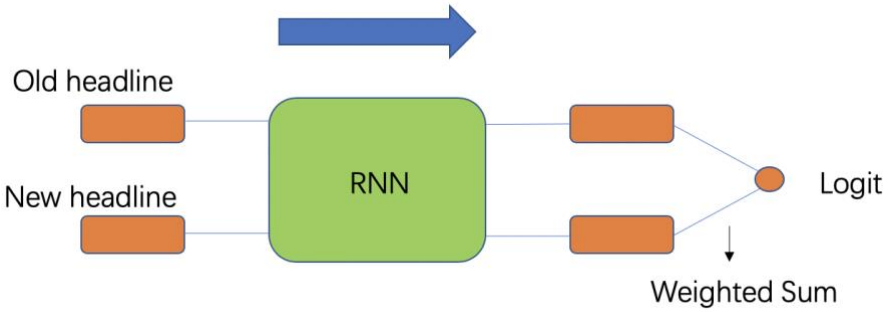
Two Inputs CNN

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.



Two Inputs RNN

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



Two Inputs Concatenated RNN

This model is all the same as Two Inputs RNN 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.

Pre-trained LM + a regression layer

Version 1 - Concatenate original headlines and new headlines

Version 2 - Concatenate new headlines and new words

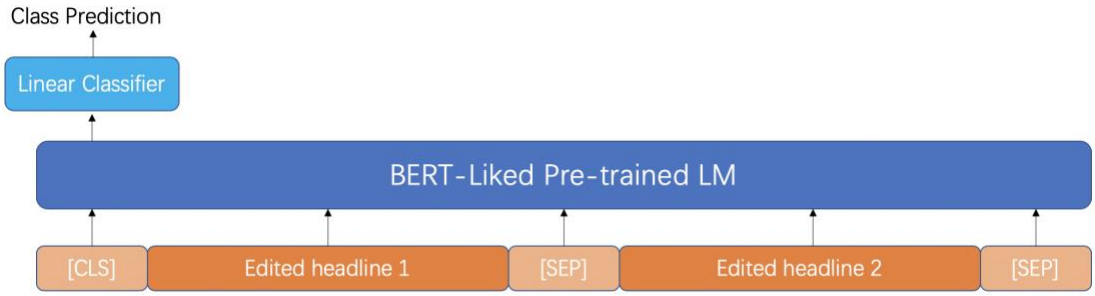
Version 3 – Contain only new headlines

Task Two

- Pre-trained LM + a classification layer (LMs applied: BERT, ALBERT, XLNet, ELECTRA, ROBERTA)

Pre-trained LM + a classification layer

Concatenate edited headline 1 and edited headline 2



Design of Training Processes (for task two only)

Version 1:

- Training the model “Pre-trained LM + a classification layer” straightly for the real classification task

Version 2 (Fake Task + Real Task):

- Firstly, training the model “Pre-trained LM + a regression layer” for a fake regression task on the training dataset
- After training well, get rid of the regression layer and add an initialized classification layer on top of the pre-trained LM
- Finally training the model for the real classification task

Optimizer & Learning Rate Scheduler

For FFNN, CNN, RNN:

- The optimizer ‘AdamW’ and the scheduler ‘CosineAnnealingLR’ provided by pytorch

For pre-trained LMs (BERT-liked LMs):

- The optimizer ‘AdamW’ and the scheduler ‘get_linear_schedule_with_warmup’ from Huggingface transformers

Prime Hyperparameters

- Learning Rate

- Fine-tuning Rate
- Adam Epsilon
- Weight Decay
- Warmup Ratio
- Number of Steps

Loss Function

Task One

Root Mean Square Error

$$\sqrt{\frac{1}{N}\sum_{t=1}^{N}\left(\text{observed}_t - \text{predicted}_t\right)^2}$$

Task Two

Cross-Entropy Loss

$$H(P,Q) = -\sum_i P(y_i)\log Q(y_i)$$

Results

Task One

- Best performance achieved by Two Inputs FFNN

EPOCHS	LRATE	EMBEDDING_DIM	HIDDEN_DIM_1	HIDDEN_DIM_2	HIDDEN_DIM_3
100	0.145	300	100	50	10

```
| Epoch: 100 | Train Loss: 0.575 | Val. Loss: 0.581 |
| Test Loss: 0.576 |
```

- Best performance achieved by Two Inputs CNN

EPOCHS	LRATE	EMBEDDING_DIM	FC_OUT_DIM	N_OUT_CHANNELS	WINDOW_SIZE	DROPOUT
500	5e-3	50	25	100	3	0.7

```
| Epoch: 500 | Train Loss: 0.623988 | Val. Loss: 0.661932 |
```

- Best performance achieved by Two Inputs RNN

EPOCHS	LRATE	EMBEDDING_DIM	HIDDEN_DIM	FC_OUTPUT_DIM	BIDIRECTIONAL	DROPOUT
30	1e-4	50	128	32	Ture	0.3

```
Epoch: 30 | Epoch Time: 0m 2s
        Train Loss: 0.586 | Val. Loss: 0.576
| Test Loss: 0.570986 |
```

- Best performance achieved by Pre-trained LMs
    a) Without Data Augmentation

        Model: bert_base_uncased

        Inputs structure: new headlines + new words

        Test loss: 0.52937

    b) With Data Augmentation (add “funlines” training dataset)

        Model: bert_base_uncased

        Inputs structure: new headlines + new words

        Test loss: 0.52054 (Best performance achieved. among all trials)

log	Model Name	Batch Size	N_Epochs	lr	fr	eps	wu	wd	Training Loss	Valid. Loss	Test Loss
1	bert-base-uncased	16	1	8e-03	3e-05	1e-08	0.3	0.010	0.59192	0.54155	0.54946
2	bert-base-uncased	16	2	8e-03	3e-05	1e-08	0.3	0.010	0.51444	0.53222	0.54059
3	bert-base-uncased	16	2	8e-03	3e-05	1e-08	0.3	0.010	0.52616	0.53481	0.54498
4	bert-base-uncased	16	2	8e-03	3e-05	1e-08	0.3	0.010	0.52624	0.53648	0.54421
5	bert-base-uncased	16	2	8e-03	3e-05	1e-08	0.3	0.010	0.52289	0.53178	0.54142
6	bert-base-uncased	8	2	8e-03	3e-05	1e-08	0.3	0.010	0.50253	0.52634	0.54085
7	bert-base-uncased	16	2	8e-03	3e-05	1e-08	0.3	0.010	0.51758	0.52993	0.53935
8	bert-base-uncased	32	2	9e-03	3e-05	1e-08	0.3	0.010	0.53778	0.53758	0.54469
9	bert-base-uncased	16	2	9e-03	3e-05	1e-08	0.3	0.010	0.49362	0.52963	0.53184
10	bert-base-uncased	16	2	9e-03	3e-05	1e-08	0.3	0.010	0.49701	0.53673	0.53775
11	bert-base-uncased	16	2	8e-03	3e-05	1e-08	0.3	0.010	0.50129	0.52963	0.52991
12	bert-base-uncased	16	2	8e-03	3e-05	1e-08	0.3	0.010	0.50001	0.53022	0.52937
13	bert-base-uncased	16	2	7e-03	3e-05	1e-08	0.3	0.010	0.50445	0.53777	0.53763
14	bert-base-uncased	16	2	8e-03	3e-05	1e-08	0.3	0.010	0.50526	0.53614	0.53758
15	bert-base-uncased	16	2	9e-03	3e-05	1e-08	0.3	0.010	0.50020	0.52845	0.53042
16	bert-base-uncased	16	2	9e-03	3e-05	1e-08	0.3	0.010	0.50772	0.53581	0.53681
17	bert-base-uncased	8	2	8e-03	3e-05	1e-08	0.3	0.010	0.47793	0.52280	0.53182
18	bert-base-uncased	4	2	8e-03	3e-05	1e-08	0.3	0.010	0.47143	0.50682	0.53211
19	electra	16	2	8e-03	3e-05	1e-08	0.3	0.010	0.58381	0.56950	0.57619
20	electra	16	3	8e-03	1e-04	1e-08	0.3	0.010	0.57928	0.56740	0.57484
21	electra	32	2	1e-02	1e-04	1e-06	0.1	0.000	0.58473	0.57374	0.57525
22	electra	16	7	3e-02	3e-04	1e-06	0.3	0.000	0.57700	0.56668	0.57495
23	electra	16	2	3e-02	8e-04	1e-06	0.3	0.000	0.57743	0.56674	0.57475
24	bert-base-uncased more_data	16	2	8e-03	3e-05	1e-08	0.3	0.010	0.49500	0.54836	0.55294
25	bert-base-uncased more_data	16	1	8e-03	3e-05	1e-08	0.3	0.010	0.57849	0.53592	0.54489
26	bert-base-uncased more_data	16	1	8e-03	3e-05	1e-08	0.5	0.010	0.58669	0.54261	0.55147
27	bert-base-uncased more_data	16	1	8e-03	3e-05	1e-08	0.1	0.010	0.57558	0.54906	0.55774
28	bert-base-uncased more_data	16	2	8e-03	3e-05	1e-08	0.1	0.010	0.47568	0.53434	0.54234
29	BertBaseUncasedMDataStp*2	16	1	8e-03	3e-05	1e-08	0.1	0.010	0.58154	0.51204	0.52356
30	BertBaseUncasedMDataStp*2	16	1	8e-03	3e-05	1e-08	0.2	0.010	0.58520	0.50942	0.52737
31	BertBaseUncasedMDataStp*2	16	1	8e-03	3e-05	1e-08	0.3	0.010	0.58951	0.53278	0.54507
32	BertBaseUncasedMDataStp*2	16	1	8e-03	3e-05	1e-08	0.1	0.005	0.57718	0.50693	0.52054
33	BertBaseUncasedMDataStp*2	16	1	8e-03	3e-05	1e-08	0.1	0.001	0.57701	0.50710	0.52078
34	BertBaseUncasedMDataStp*2	16	1	8e-03	3e-05	1e-08	0.2	0.005	0.58582	0.51137	0.52763

Task Two

- Version 1: Straightly training the model for the real task

Model Name	Batch Size	N_Epochs	lr	eps	wus	wd	Training Loss	Training Accur.	Valid. Loss	Valid. Accur.	Testing Accur.	wu
electra	16	3	5.00E-05	1.00E-08	500	0.01	0.9040663524	56.01575809	1.017055559	45.19988739	50.11415525	
electra	16	3	3.00E-05	1.00E-08	500	0.01	0.7519826203	67.64267462	1.077755947	46.07263514	50.11415525	
bert-base-uncased	16	3	3.00E-05	1.00E-08	500	0.01	0.7519826203	67.64267462	1.077755947	46.07263514	50.11415525	
bert-base-uncased	32	3	3.00E-05	1.00E-08	500	0.01	0.7409755295	68.22916667	1.09057613	46.98168563	50.83713851	
albert-base-v2	32	3	3.00E-05	1.00E-08		0.01	0.9251348149	54.42389457	0.9430917228	51.98035208	51.63622527	0.1
albert-base-v2	32	4	3.00E-05	1.00E-08		0.01	0.7931970478	67.60841837	0.9970703101	54.06294449	50.41856925	0.1
albert-base-v2	16	3	3.00E-05	1.00E-08		0.01	0.9442128167	51.98679728	0.9520759715	51.91441442	50.0761035	0.1
albert-base-v2	16	3	1.00E-05	1.00E-08		0.01	0.8710818268	62.6362862	0.9773926723	50.02815317	49.6194825	0.1
electra	32	3	3.00E-05	1.00E-08		0.01	0.9156131631	55.71853741	0.9853870836	46.29045165	50.64687976	0.1
albert-base-v2	32	4	3.00E-05	1.00E-08		0.01	0.9587668154	46.11607143	0.960945814	43.51662518	51.78843227	0.1

Log 1

Model Name	Batch Size	N_Epochs	lr	fr	eps	wu	wd	Training Loss	Training Accur.	Valid. Loss	Valid. Accur.	Testing Accur.
xlnet	32	4	3.00E-05	none	1.00E-08	0.1	0.01	0.4545018434	83.26955782	1.459015717	43.38993599	48.89649924
xlnet	32	2	8.00E-03	3.00E-05	1.00E-06	0.3	0.01	1.041007528	43.85841838	0.9648312193	43.24768846	50.95129376
xlnet	32	2	5.00E-03	2.00E-05	1.00E-08	0.1	0.01	1.007680466	44.6237245	0.9634933729	44.73906474	48.89649924
albert-base-v2	32	2	5.00E-03	2.00E-05	1.00E-08	0.1	0.01	1.013667674	44.53869048	0.9644417126	43.24768846	50.95129376
albert-base-v2	32	2	5.00E-03	2.00E-05	1.00E-08	0.1	0.01	0.9626696132	44.72151361	0.9640179244	43.24768846	50.95129376
albert-base-v2	32	3	5.00E-03	2.00E-05	1.00E-08	0.1	0.01	0.9720912097	47.04719389	0.9662280614	43.24768846	50.95129376
albert-base-v2	32	4	3.00E-05	none	1.00E-08	0.1	0.01	0.8882992945	57.72534014	0.9931740898	43.24768846	50.95129376
albert-base-v2	32	2	3.00E-05	none	1.00E-08	0.1	0.01	0.9605611593	48.11437075	0.9521363063	51.54027383	50.41856925
albert-base-v2	32	2	3.00E-05	none	1.00E-08	none	0.01	0.96823891	44.85331633	0.9655782927	43.24768846	50.91324201
albert-base-v2	32	2	3.00E-05	none	1.00E-08	0.3	0.01	0.9561010386	47.67431974	0.9605669492	47.07725817	50.87519026
albert-base-v2	32	3	3.00E-05	none	1.00E-08	0.1	0.01	0.9421355432	50.55484695	0.9653435607	46.58828233	50.87519026
roberta	32	3	3.00E-05	none	1.00E-08	0.1	0.01	0.9555197334	47.1492347	0.9625042894	46.23488623	48.85844749
roberta	32	3	3.00E-05	none	none	0.06	0.01	0.9589914738	45.75467687	0.9630940758	45.79703059	49.04870624
roberta	16	4	1.00E-05	none	none	0.06	0.1	0.8042444122	62.08475298	1.023312278	46.59346848	49.543379
bert-base-uncased	32	4	1.00E-05	none	none	0.06	0.1	0.5708310699	80.08078231	1.118884071	48.38415719	51.56012177
bert-base-uncased	16	4	1.00E-05	none	none	0.06	0.1	0.3808206013	86.85051107	1.378147476	48.60641892	50.2283105
bert-base-uncased	32	4	1.00E-05	none	none	0.01	0.1	0.5582021983	81.05867347	1.135324642	49.24431008	50.68493151
bert-base-uncased	32	4	5.00E-06	none	none	0.01	0.1	0.8313606623	64.90221088	0.996364724	49.23097439	49.88584475
bert-base-uncased	32	4	1.00E-05	none	none	0.06	0.1	0.5205045101	83.46088435	1.132244302	47.69736843	50.68493151

Log 2

- Version 2: Fake Task Training + Real Task Training

Model Name	Batch Size	N_Epochs	lr	fr	eps	wu	wd	Training Loss	Valid. Loss
albert-base-v2	16	3	5.00E-03	2.00E-05	1.00E-08	0.1	0.01	0.5758530823	0.5704045271
albert-base-v2	16	3	5.00E-03	2.00E-05	1.00E-08	0.1	0.01	0.5758985989	0.5704450275
albert-base-v2	16	3	5.00E-03	2.00E-05	1.00E-08	0.1	0.01	0.5173322452	0.5331617039
bert-base-uncased	16	3	5.00E-03	2.00E-05	1.00E-08	0.1	0.01	0.3218292155	0.5424303445
bert-base-uncased	16	1	5.00E-03	2.00E-05	1.00E-08	0.1	0.01	0.5295220188	0.5252422331

Fake Task Log

Model Name	Batch Size	N_Epochs	lr	fr	eps	wu	wd	Training Loss	Training Accur.	Valid. Loss	Valid. Accur.	Testing Accur.
bert-base-uncased	16	3	3.00E-05	5.00E-05	1.00E-08	0.1	0.01	0.2301243856	92.14224872	1.686486853	51.29504506	46.34703196
bert-base-uncased	16	1	3.00E-05	5.00E-05	1.00E-08	0.1	0.01	0.3065944731	88.45400341	1.793892831	49.2257883	44.7108067
bert-base-uncased	16	1	3.00E-05	5.00E-05	1.00E-08	0.1	0.01	0.1073978884	96.72913118	2.533883247	46.77646398	41.55251142
bert-base-uncased	16	1	3.00E-05	5.00E-05	1.00E-08	0.1	0.1	0.04975398582	98.69889268	2.844575617	47.62105857	41.93302892
bert-base-uncased	16	1	3.00E-05	5.00E-05	1.00E-08	0.1	0.1	0.9632197996	47.40630324	0.9466330961	52.36486486	49.42922374
bert-base-uncased	16	3	3.00E-05	5.00E-05	1.00E-08	0.1	0.1	0.293213008	89.28875639	1.634537949	53.40653154	46.27092846
bert-base-uncased	16	2	3.00E-05	5.00E-05	1.00E-08	0.1	0.5	0.8229573291	63.49446337	0.9505428632	56.58783784	50
bert-base-uncased	16	2	5.00E-03	2.00E-05	1.00E-08	0.1	0.01	0.8640907449	60.43654174	0.9335295485	56.84121622	50.53272451
bert-base-uncased	16	2	5.00E-03	2.00E-05	1.00E-08	0.1	0.7	0.8653311929	60.06388416	0.9337685056	57.01013514	50.34246575
bert-base-uncased	16	2	5.00E-03	2.00E-05	1.00E-08	0.3	0.01	0.8961150421	57.24446337	0.9304311582	56.02477479	49.6194825
bert-base-uncased	16	2	5.00E-03	2.00E-05	1.00E-08	0.1	0.01	0.9535369074	49.12265758	0.9624514358	46.73423425	49.9238965
bert-base-uncased	16	3	5.00E-03	2.00E-05	1.00E-08	0.1	0.9	0.7048996182	70.56005111	1.053826613	52.02702703	50.03805175
bert-base-uncased more_data	16	3	5.00E-03	2.00E-05	1.00E-08	0.1	0.9	0.2767301854	90.17295597	1.453697178	50.7882883	49.80974125
bert-base-uncased more_data	16	1	5.00E-03	2.00E-05	1.00E-08	0.1	0.1	0.9473693124	49.64061096	0.9366792163	53.23761263	50.19025875
bert-base-uncased more_data	16	2	5.00E-03	2.00E-05	1.00E-08	0.1	0.1	0.6993590668	71.25355646	1.040675076	55.22240992	50.41856925
bert-base-uncased more_data	16	1	8.00E-03	3.00E-05	1.00E-08	0.3	0.05	0.9500814976	49.99812819	0.9438883147	54.56081081	50.304414

Real Task Log

Discussion

Task One

- The performance of Two Inputs RNN is just slightly better compared with that of the Two Inputs FFNN (0.5759702196 vs. 0.5751694002) while the time complexity of the Two Inputs RNN is much higher than the Two Inputs FFNN, at some point the current version of Two Inputs RNN is resources-wasted.
- The Two Inputs CNN with a single window size performs worse than the Two Inputs FFNN and the Two Inputs RNN, and one of possible reasons is that it only looks at one size of n-gram and hence ignores the knowledge of n-grams with different lengths.

Task Two

- For different preprocessing methods, the headlines truncated version and the punctuation removal version have the same performance as the normal one except that truncating headlines will reduce the training time for a single epoch.

- The issue of overfitting on the training dataset is hard to overcome when applying BERT-liked pre-trained LMs (Although several methods, such as data augmentation, weight decay and dropout increase have been tried to mitigate this problem.
- Surprisingly, the fake task training for pre-trained LMs does not help to improve the performance of the model in real task even a little bit.
- With the same hyperparameters setting for the certain task, the performance of the newly proposed pre-trained LM is not necessarily the best.

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 Two Inputs CNN might be improved by increasing the number of the window sizes (different n-gram filters).
- Applying the pre-trained LM Longformer rather than other BERT-liked models for the task two, in which the Longformer has the 'global attention mask' and it can probably better model the relationship between the edited word and the other words in a headline (e.g. **How important is the edited word for the whole headline in order to make it funnier?** / **How does the edited word contribute to the meaning of the whole sentence**).