Text Classification Report

Ye Yuan # 1700012821

January 9, 2021

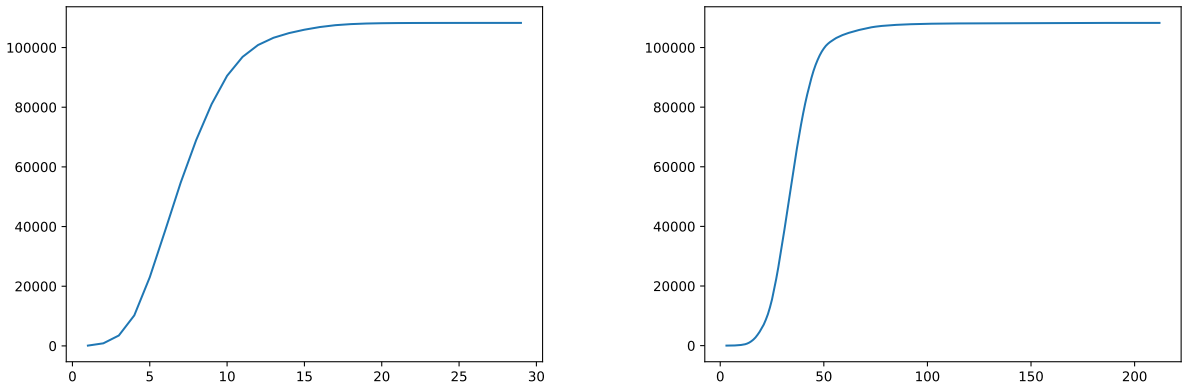
Contents

1 Data Statistics

For this task, I have firstly analyze the statistics of the dataset. Table 1 shows the basic information of the dataset. As we can see, the label distribution is balanced. However the descriptions are a little long. As we need to do truncation or padding to these sentences, so we need to do more investigation of the CDF of lengths. The CDF of title lengths and description length are showed in Figure 1.

Dataset	Max Title Length	Max Description Length	Label Distribution
Train	29	212	(0.2512, 0.2496, 0.2484, 0.2508)
Validation	26	161	(0.2388, 0.2539, 0.2644, 0.2429)
Test	24	153	\

Table 1: Basic Information of Dataset



(a) Title (b) Description

Figure 1: CDF of the lengths of titles and descriptions

2 Preprocessing

I have used *TorchText* and *Spacy* to do preprocessing. Firstly, there are a lot of "\" token in the descriptions, so I replaced them with space. Moreover, I have also replaced the "#39;" with "'" and replaced the "#36;" with "$". For baselines, we just join the titles and descriptions. So I have added a new column merging them together. I use the pre-trained *Spacy* tokenizer "en_core_web_lg" to tokenize the sentences, and use *TorchText* to build vocab, load pre-trained word vectors and do batchify.

3 Experiments

3.1 How To Run

See "README.md" in this folder.

3.2 Baselines

For baselines, I only join the title and description for each data example. There are a lot of choices for running experiments. They are listed in Table 2:

Setting	Description	Choices
Tokenizer	Which tokenizer to use.	nltk.word_tokenize spacy
Vocab	How to build vocab.	Vocab size: 10000, 20000, 30000, 40000, 50000, Full
Max Lengths	Max lengths for truncation or padding.	Max lengths: 20, 50, 100, 200, Full
Optimizer	Which optimizer to use.	Adam AdamW SGD
Other Parameters	Other model parameters.	Model Choice: RNN, LSTM, GRU, CNN

Table 2: Alternative Settings

I have compared these different settings using CNN model. And compare the results and training time of different models. I will introduce CNN experiments firstly. And then compare the results of different models. Notably, this showed results are the results of only one execution. For there is no good solution for PyTorch to avoid randomness and due to time limit, the average score of each setting may be higher or lower than the results listed in the following report. The models are trained on GTX 1080Ti.

3.2.1 CNN experiments

As CNN is not sensitive to "<pad>" token, we just padding all the sentences(the joint text of title and description) to a fixed length. The best CNN model uses full vocab, max length 200 and an exponential decrease learning rate scheduler with 0.1 as beta. The training procedure will train the model for 10 epochs and choose the model with the highest score on validation set. I used the accuracy as my evaluation metrics.The results are showed in Table 3.

As we can see, using pre-trained glove embedding makes a lot of progress. For CNN model has smaller parameters than the embedding layer. So the initialization of embedding layer is very important. Moreover, the optimization method have not much effect on the results except for that SGD need more time to adjust hyper-parameters. Moreover, adding kernels and increasing kernel number does not increase the performance.

Next, let's talk about the effect of vocab size and max padding length. As the maximum length of the text is not longer than 250, so I have tried 50,100,150,200,250 for the max padding length. For the vocab size, I have tried 10000-80000 every 10000 and use full vocab size, which is 84849. The results are showed in Table 4 and Table 5. As we can see, increase vocab size have little influence on the time, and can improve the performance slightly. Increasing the max padding length can also improve the performance slightly, but need a lot more time.

3.2.2 Different Models Comparison

In this section, I will introduce the validation results of different models and the time efficiency. Each type is trained to the best as I can. The results are showed in Table 6.

Model	Optimizer	Use Glove	Kernels	Accuracy
CNN	AdamW, lr=0.001 0.1 exponential decrease	True	(2,3,4) (256,256,256)	93.039%
CNN	AdamW, lr=0.001 0.1 exponential decrease	True	(2,3,4,5) (1024,1024,1024,1024)	92.783%
CNN	AdamW, lr=0.001 0.1 exponential decrease	False	(2,3,4) (256,256,256)	90.082%
CNN	AdamW, lr=0.001 no decrease	False	(2,3,4) (256,256,256)	89.707%
CNN	AdamW, lr=0.001 0.5 exponential decrease	False	(2,3,4) (256,256,256)	90.653%
CNN	Adam, lr=0.001 0.1 exponential decrease	True	(2,3,4) (256,256,256)	93.004 %
CNN	SGD, lr=0.001, momentum=0.9 0.1 exponential decrease	True	(2,3,4) (256,256,256)	87.858%
CNN	SGD, lr=0.001, momentum=0.9 0.5 exponential decrease	True	(2,3,4) (256,256,256)	89.153%
CNN	SGD, lr=0.001, momentum=0.9 0.9 exponential decrease	True	(2,3,4) (256,256,256)	90.874%
CNN	SGD, lr=0.0025, momentum=0.9 0.9 exponential decrease	True	(2,3,4) (256,256,256)	91.837%

Table 3: The results of CNN experiments

Vocab Size	10000	20000	30000	40000	50000	60000	70000	80000	84849 (full)
Time (s/epoch)	37	37	38	38	39	39	40	40	41
Performance (accuracy %)	92.26	92.61	92.61	93.02	92.82	92.83	92.92	93.02	92.95

Table 4: The effects of vocab size.

Max Length	50	100	150	200	250
Time (s/epoch)	22	25	33	41	47
Performance (accuracy %)	92.64	92.74	92.80	93.09	92.92

Table 5: The effects of max length.

Model	CNN	RNN	LSTM	GRU	BiRNN	BiLSTM	BiGRU
Time (s/epoch)	41	35	48	44	61	97	94
Performance (accuracy %)	93.039	89.340	91.914	92.212	82.831	92.152	92.791
Parameters (M)	26.1	26.7	29.5	28.5	28.4	35.6	33.2

Table 6: The comparison of different models.

As we can see, CNN has the least parameters but has the best performance. I think this may be because of the glove initialization. For there are about 27M parameters are from the embedding layer. RNN has a bad performance for the simple structure. Moreover, GRU is faster and better than LSTM, and has less parameters.

3.3 More Ways to Use Titles and Descriptions

I have explored more ways to use titles and descriptions. Firstly, I will compare the results of using only titles or descriptions. Moreover, I will try some attention-based models based on GRU and Transformer.

3.3.1 Use Titles or Descriptions only

In this section, I will show the results of title and description only using CNN model. The results is showed in Table 7. The results shows that descriptions contribute to the prediction more, and join them together can improve the accuracy.

Settings	CNN Full Text Max Length: 200	CNN Title Only Max Length: 50	CNN Description Only Max Length: 150
Performance (accuracy %)	93.039	87.764	92.076

Table 7: The comparison of using titles and descriptions.

3.3.2 Attention-Based Models

In the attention-based models, I used BiGRU as my encoder. The structure of this model is from [2] but replace the BiLSTM with BiGRU. Moreover, I run the experiments using another attention-based models which is Transformer [1]. And I compared the results of these models showed in Table 8. For transformer model, I only used the encoder with self-attention layers, and concatenate all the output embeddings as sentence embedding, and then feed it into a FC layer. 6 layers model is too large for this task and hard to adjust hyperparameters, so I only used 2 self attention layers.

The results shows that maybe this dataset is too easy for models, and adding attention mechanism does not help to performance.

Finally, I used CNN-based model to generate my prediction file.

Settings	BiGRU	BiGRU+Attn	Transformer
Performance (accuracy %)	92.791	92.757	91.590

Table 8: The comparison of using titles and descriptions.

References

[1] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[2] Peng Zhou, Wei Shi, Jun Tian, Zhenyu Qi, Bingchen Li, Hongwei Hao, and Bo Xu. Attention-based bidirectional long short-term memory networks for relation classification. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 207–212, 2016.