Jiacheng Shen

Professor Mathieu Lauriere

Machine Learning Final Project

23 December 2022

You Only Need One Bert

1. Abstract

One of the most essential features of an online shopping platform is product reviews. They not only help merchants understand the flaws of a product but also help consumers make purchasing decisions. Therefore, in order to determine whether a review is a one or two-star negative review or a four or five-star positive review, in this paper, I trained and selected a relatively good performing, low-cost model and method by comparing different Bert models and training methods. The open-source code for this project is published on my Github:

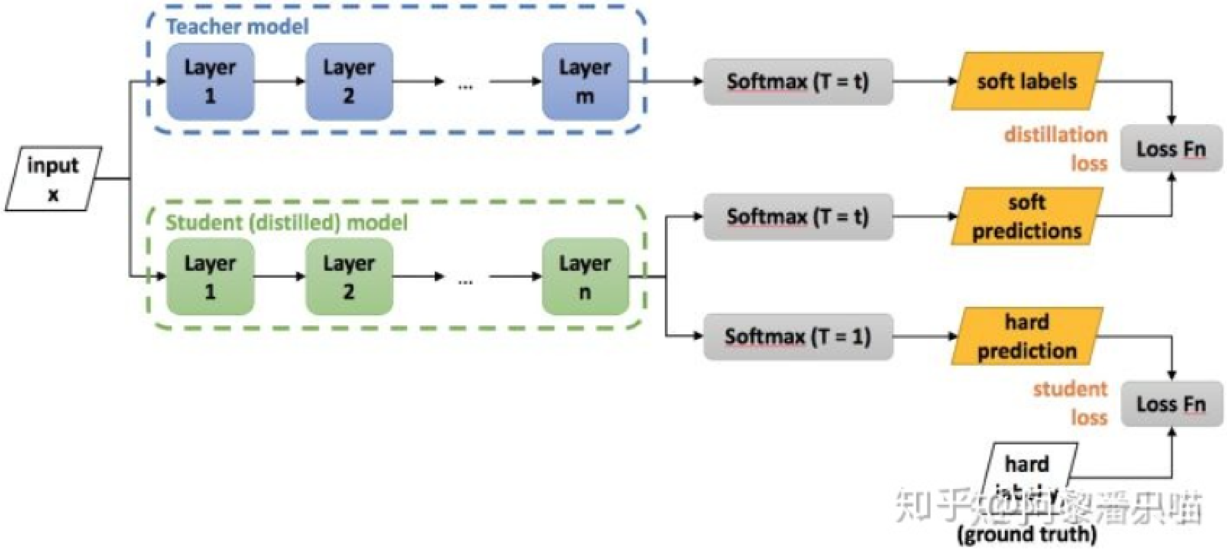
2. Introduction and Background

There are several different models available for natural language processing tasks, and some of the most popular ones are the BERT (Bidirectional Encoder Representations from Transformers), DistilBERT, and TinyBERT models. BERT is a large and complex model that has achieved state-of-the-art performance on a wide range of natural language processing tasks.

2.1 Knowledge Distillation

Knowledge distillation is a machine learning technique that aims to transfer the knowledge of a large and complex model (called the "teacher model") to a smaller and simpler model (called the "student model"). The goal of knowledge distillation is to train the student model to perform as well as the teacher model, but with fewer parameters and a lower memory footprint, making it easier to deploy and run on resource-constrained devices(Hinton et al.).

To perform knowledge distillation, the teacher model is first trained on a large dataset and used to generate predictions for a smaller dataset that is used to train the student model. The student model is then trained based on two loss functions. One is to minimize the difference between its own predictions and the predictions of the teacher model, using a loss function that measures the difference between the two sets of predictions(Hinton et al.). And another loss function performs measuring the distance between the hard label and hard prediction, aiming at improving the performance on the hard label(real label). By minimizing this loss, the student model learns to mimic the behavior of the teacher model and achieves similar performance on the task.



(Zhihu https://zhuanlan.zhihu.com/p/165516661)

2.2 DistilBERT

DistilBERT takes advantage of the knowledge distillation idea above. DistilBERT is a smaller and faster version of BERT that has been trained to retain most of the performance of the original model, but with fewer parameters and a lower memory footprint(Sanh et al.). So the advantage for DistilBERT is simple: It doesn't need tons of corpus to train like Bert. Instead, DistilBERT only needs a small-scale dataset and learns from the strong teacher model to achieve a relatively equal or higher performance compared with base Bert. Additionally, DistilBERT can also save the computation cost of GPU if we don't have adequate computation ability.

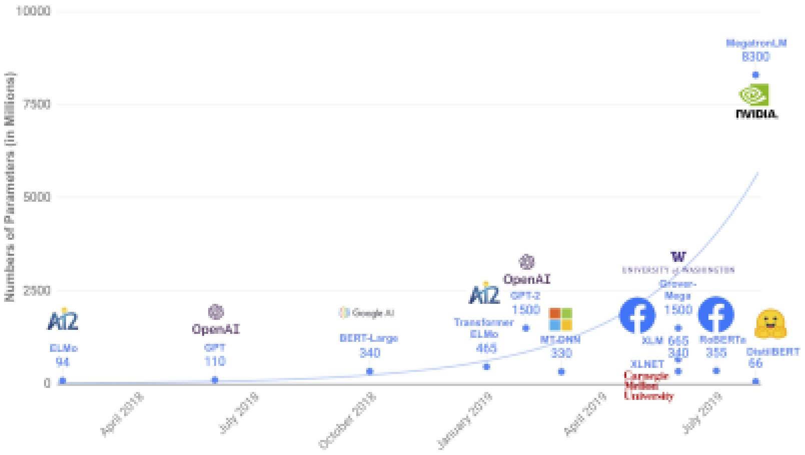


Figure 1: Parameter counts of several recently released pretrained language models.

(Sanh et al. 2019)

2.3 TinyBERT

TinyBERT is an even smaller and faster version of DistilBERT that has been designed for use on resource-constrained devices, such as smartphones or edge devices(Sun et al.). TinyBERT even has a lower training expense than DistilBERT.

2.4 Focal Loss

When training a model for a classification task, such as sentiment analysis, it is common to use a loss function to measure the difference between the model's predictions and the ground truth labels. One popular loss function for classification tasks is the Cross-Entropy loss, which measures the difference between the predicted and true probabilities of the classes. Another variant of the Cross-Entropy loss is the Focal Loss (He et al), which down-weights the loss of well-classified examples and focuses more on hard examples. This can be useful in cases where the classes are imbalanced, as it can help the model to pay more attention to the minority classes.

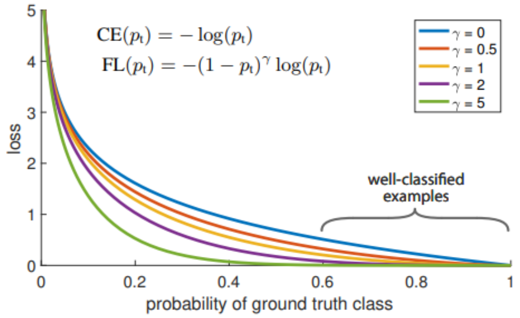


Figure 1. We propose a novel loss we term the *Focal Loss* that adds a factor $(1 - p_t)^{\gamma}$ to the standard cross entropy criterion. Setting $\gamma > 0$ reduces the relative loss for well-classified examples ($p_t > .5$), putting more focus on hard, misclassified examples. As our experiments will demonstrate, the proposed focal loss enables training highly accurate dense object detectors in the presence of vast numbers of easy background examples. (He et. al,2017)

3. Problem Formulation

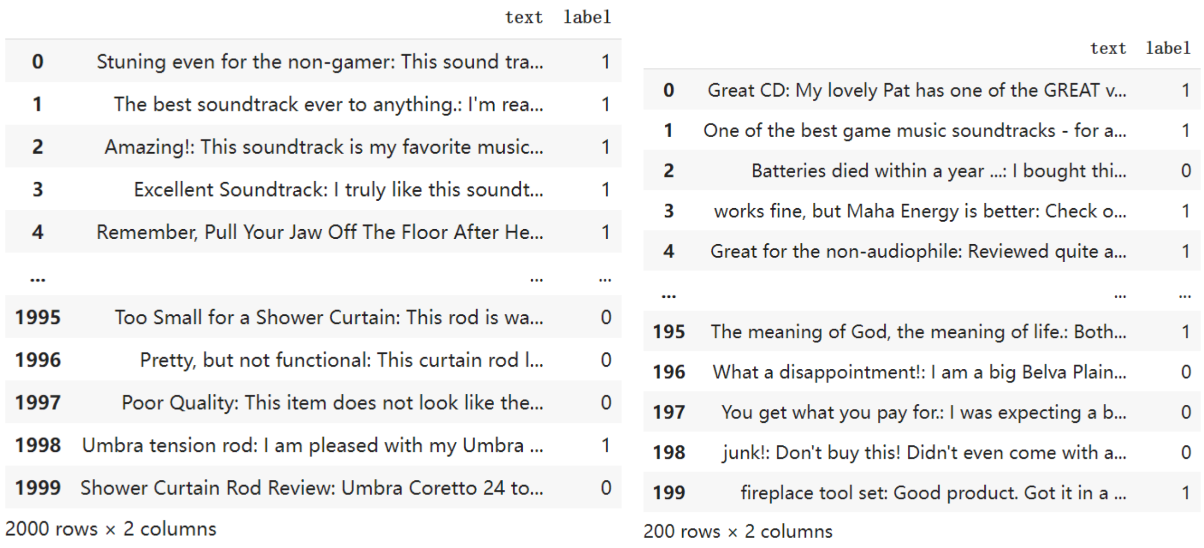
One of the most essential features of an online shopping platform is product reviews. They not only help merchants understand the flaws of a product but also help consumers make purchasing decisions. In my project, I select Amazon review as my dataset, which aims to predict the overall sentiment (positive or negative) of a review based on its text.

The Amazon review dataset consists of a large number of customer reviews and ratings for various products sold on the Amazon website. The reviews are written in natural language and include information about the product, the customer's experience with the product, and their overall rating. It's also available on Kaggle: https://www.kaggle.com/datasets/bittlingmayer/amazonreviews

5. Methodology

5.1 Data Processing

In this project, I decrease both the original label "2" and "1" into "1" and "0", which usually stands for "positive" and "negative". Since the original dataset has several million of data, it's impossible to train the whole dataset in a limited time. I randomly select 1600 data as the training set, 400 data as the validation set, and 200 data as the test set(Figure 1). The test set is selected from a test.txt file, which is a different source from the training set and validation set.



	text	label
0	Stuning even for the non-gamer: This sound tra...	1
1	The best soundtrack ever to anything.: I'm rea...	1
2	Amazing!: This soundtrack is my favorite music...	1
3	Excellent Soundtrack: I truly like this soundt...	1
4	Remember, Pull Your Jaw Off The Floor After He...	1
...	...	...
1995	Too Small for a Shower Curtain: This rod is wa...	0
1996	Pretty, but not functional: This curtain rod l...	0
1997	Poor Quality: This item does not look like the...	0
1998	Umbra tension rod: I am pleased with my Umbra ...	1
1999	Shower Curtain Rod Review: Umbra Coretto 24 to...	0

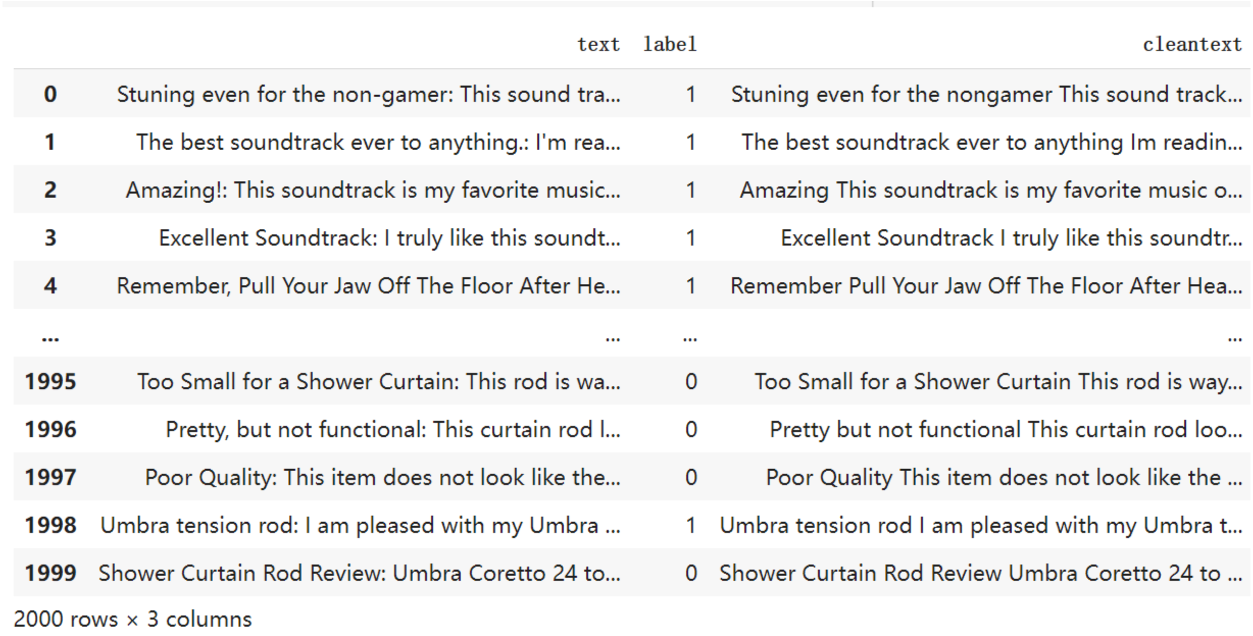
2000 rows × 2 columns

	text	label
0	Great CD: My lovely Pat has one of the GREAT v...	1
1	One of the best game music soundtracks - for a...	1
2	Batteries died within a year ...: I bought thi...	0
3	works fine, but Maha Energy is better: Check o...	1
4	Great for the non-audiophile: Reviewed quite a...	1
...	...	...
195	The meaning of God, the meaning of life.: Both...	1
196	What a disappointment!: I am a big Belva Plain...	0
197	You get what you pay for.: I was expecting a b...	0
198	junk!: Don't buy this! Didn't even come with a...	0
199	fireplace tool set: Good product. Got it in a ...	1

200 rows × 2 columns

(Figure 1)

Then after selecting the text, I import the "re" package to remove all numbers and punctuations in the dataset(Figure 2). And it's ready for the training step.



	text	label	cleantext
0	Stuning even for the non-gamer: This sound tra...	1	Stuning even for the nongamer This sound track...
1	The best soundtrack ever to anything.: I'm rea...	1	The best soundtrack ever to anything Im readin...
2	Amazing!: This soundtrack is my favorite music...	1	Amazing This soundtrack is my favorite music o...
3	Excellent Soundtrack: I truly like this soundt...	1	Excellent Soundtrack I truly like this soundtr...
4	Remember, Pull Your Jaw Off The Floor After He...	1	Remember Pull Your Jaw Off The Floor After Hea...
...	...	...	...
1995	Too Small for a Shower Curtain: This rod is wa...	0	Too Small for a Shower Curtain This rod is way...
1996	Pretty, but not functional: This curtain rod l...	0	Pretty but not functional This curtain rod loo...
1997	Poor Quality: This item does not look like the...	0	Poor Quality This item does not look like the ...
1998	Umbra tension rod: I am pleased with my Umbra ...	1	Umbra tension rod I am pleased with my Umbra t...
1999	Shower Curtain Rod Review: Umbra Coretto 24 to...	0	Shower Curtain Rod Review Umbra Coretto 24 to ...

2000 rows × 3 columns

(Figure 2)

5.2 Pre-training

To save time, I directly use pre-trained models of Bert, DistilBERT, and TinyBERT from huggingface. https://huggingface.co/docs/transformers/v4.25.1/en/index

5.3 Training Algorithm

I. Firstly, using a tokenizer to convert the input text data into a numerical representation that can be processed by the model. The tokenizer here depends on the corresponding model, for example, the base Bert model uses the Bert Tokenizer.

II. Convert training and test datasets into tensors using the tokenizer and storing in TensorDatasets.

III. Split the training dataset into the training set and validation set by keeping 80% of the original training dataset and taking 20% as the validation set. This is where 1600 and 400 come from in the previous section. And use dataloaders to load training, validation, and test set.

IV. In the training step, I use “Adam” as the optimizer and cross-entropy or focal loss as the loss function. As we discussed during Recitation 12, Adam has a generally better performance compared with SGD. Then in every epoch, the model will use corresponding input_ids, attention_mask, and labels in the dataloader to perform forward propagation and backward propagation. Before performing the optimizer step(parameter updating), I add a step to do gradient clipping to help prevent gradient vanishing, as we discussed in Recitation 11.

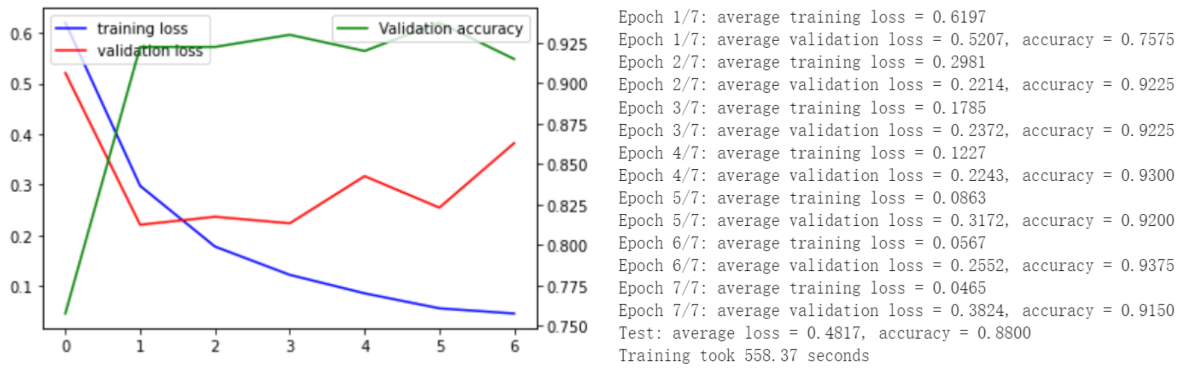
V. Finally, I calculated the average loss and accuracy on the validation set and test set.

VI. Repeat step 1 to 5 on Bert with focal loss, DistilBERT with focal loss, and TinyBERT with focal loss.

6. Results

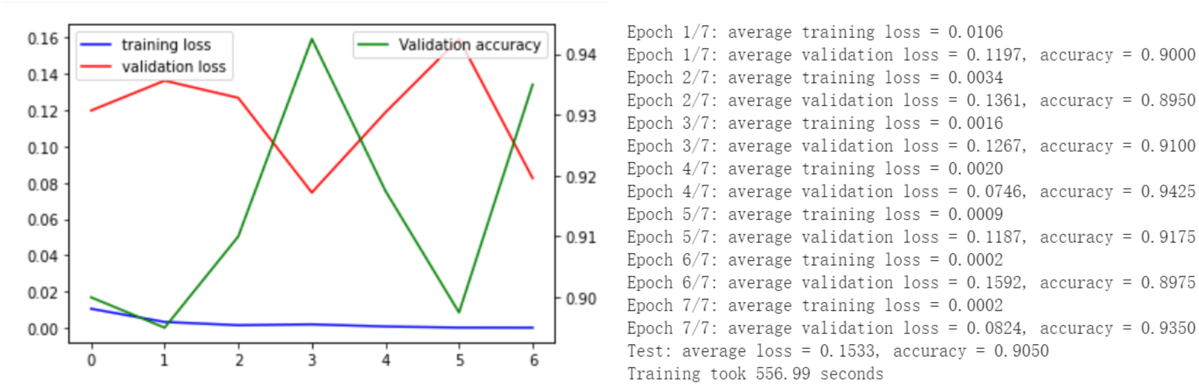
6.1 Cross-entropy and Focal Loss

To compare whether Focal loss or Cross-entropy performs better, I combine them with base Bert and train separately, by taking the same parameters(num_epoches=7,batch_size=16,alpha=0.2, gamma=3,lr=0.5e-5):



```
Epoch 1/7: average training loss = 0.6197
Epoch 1/7: average validation loss = 0.5207, accuracy = 0.7575
Epoch 2/7: average training loss = 0.2981
Epoch 2/7: average validation loss = 0.2214, accuracy = 0.9225
Epoch 3/7: average training loss = 0.1785
Epoch 3/7: average validation loss = 0.2372, accuracy = 0.9225
Epoch 4/7: average training loss = 0.1227
Epoch 4/7: average validation loss = 0.2243, accuracy = 0.9300
Epoch 5/7: average training loss = 0.0863
Epoch 5/7: average validation loss = 0.3172, accuracy = 0.9200
Epoch 6/7: average training loss = 0.0567
Epoch 6/7: average validation loss = 0.2552, accuracy = 0.9375
Epoch 7/7: average training loss = 0.0465
Epoch 7/7: average validation loss = 0.3824, accuracy = 0.9150
Test: average loss = 0.4817, accuracy = 0.8800
Training took 558.37 seconds
```

（Bert with cross-entropy）



```
Epoch 1/7: average training loss = 0.0106
Epoch 1/7: average validation loss = 0.1197, accuracy = 0.9000
Epoch 2/7: average training loss = 0.0034
Epoch 2/7: average validation loss = 0.1361, accuracy = 0.8950
Epoch 3/7: average training loss = 0.0016
Epoch 3/7: average validation loss = 0.1267, accuracy = 0.9100
Epoch 4/7: average training loss = 0.0020
Epoch 4/7: average validation loss = 0.0746, accuracy = 0.9425
Epoch 5/7: average training loss = 0.0009
Epoch 5/7: average validation loss = 0.1187, accuracy = 0.9175
Epoch 6/7: average training loss = 0.0002
Epoch 6/7: average validation loss = 0.1592, accuracy = 0.8975
Epoch 7/7: average training loss = 0.0002
Epoch 7/7: average validation loss = 0.0824, accuracy = 0.9350
Test: average loss = 0.1533, accuracy = 0.9050
Training took 556.99 seconds
```

(Bert with focal loss)

Based on the figures above, I can conclude the results:

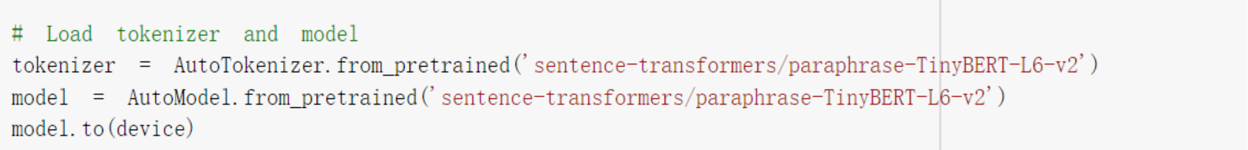
a. The validation loss and accuracy of Bert with focal loss are not as stable as Bert with cross-entropy. That might be because Bert with focal loss adjusts the weights of uncertain categories dynamically during training.

b. Comparing their test accuracy, Bert with focal loss increases by 2.5% accuracy and has less test loss than Bert with cross-entropy.

Therefore, focal loss generally will perform better than cross-entropy on this limited-size dataset.

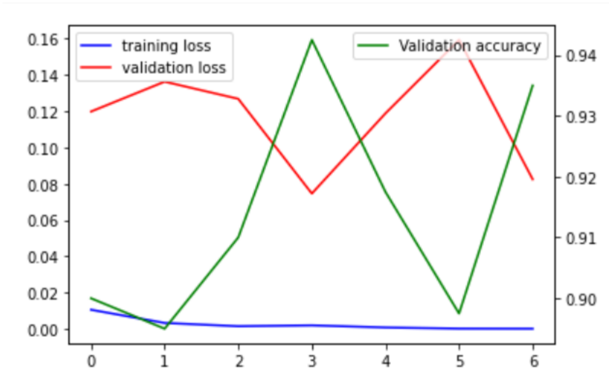
6.2 base Bert, DistilBERT, and TinyBERT, trained with focal loss

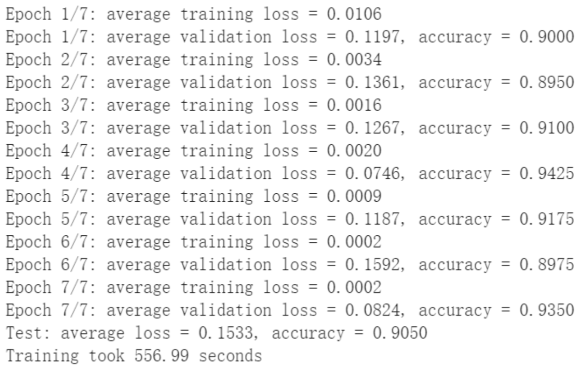
For Bert and DistilBERT, both of them are using parameters: num_epoches=7,batch_size=16,alpha=0.2, gamma=3,lr=0.5e-5. But for TinyBERT, I use num_epoches=7,batch_size=16,alpha=0.2, gamma=3,lr=2e-5. Because TinyBERT is so new that huggingface doesn’t have its corresponding well-pre-trained tokenizer and model. The performance is very bad if I use the same parameter as DistilBERT and BERT. So maybe it’s also because the unofficial pre-trained model doesn’t suit my dataset and task well.



```
# Load tokenizer and model
tokenizer = AutoTokenizer.from_pretrained('sentence-transformers/paraphrase-TinyBERT-L6-v2')
model = AutoModel.from_pretrained('sentence-transformers/paraphrase-TinyBERT-L6-v2')
model.to(device)
```

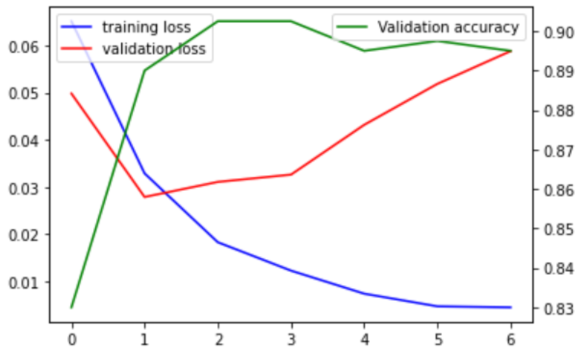
Therefore, if we observe the result of TinyBERT isolatedly(Figure TinyBERT trained by focal loss), we may find it probably has a larger opportunity to achieve higher accuracy, if we increase num_epoches or pre-training it by ourselves. Because its validation accuracy increases along the time of training.

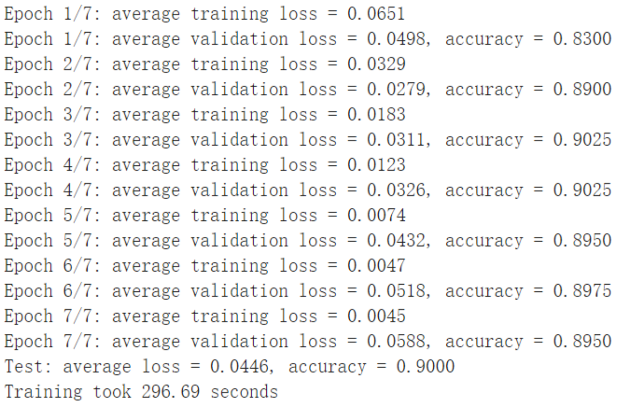




```
Epoch 1/7: average training loss = 0. 0106
Epoch 1/7: average validation loss = 0. 1197, accuracy = 0. 9000
Epoch 2/7: average training loss = 0. 0034
Epoch 2/7: average validation loss = 0. 1361, accuracy = 0. 8950
Epoch 3/7: average training loss = 0. 0016
Epoch 3/7: average validation loss = 0. 1267, accuracy = 0. 9100
Epoch 4/7: average training loss = 0. 0020
Epoch 4/7: average validation loss = 0. 0746, accuracy = 0. 9425
Epoch 5/7: average training loss = 0. 0009
Epoch 5/7: average validation loss = 0. 1187, accuracy = 0. 9175
Epoch 6/7: average training loss = 0. 0002
Epoch 6/7: average validation loss = 0. 1592, accuracy = 0. 8975
Epoch 7/7: average training loss = 0. 0002
Epoch 7/7: average validation loss = 0. 0824, accuracy = 0. 9350
Test: average loss = 0. 1533, accuracy = 0. 9050
Training took 556. 99 seconds
```

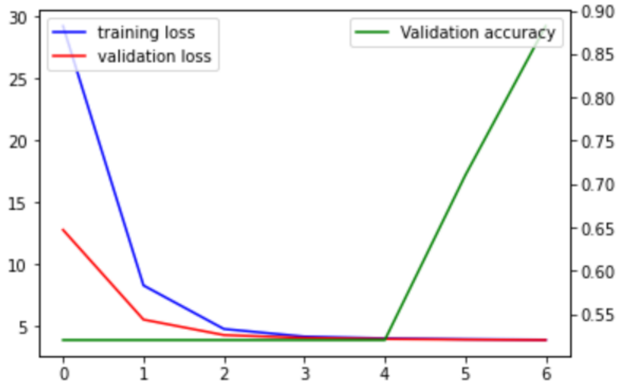
(Figure base Bert trained by focal loss)

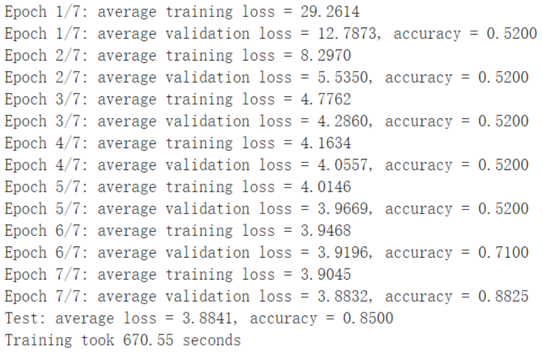




```
Epoch 1/7: average training loss = 0. 0651
Epoch 1/7: average validation loss = 0. 0498, accuracy = 0. 8300
Epoch 2/7: average training loss = 0. 0329
Epoch 2/7: average validation loss = 0. 0279, accuracy = 0. 8900
Epoch 3/7: average training loss = 0. 0183
Epoch 3/7: average validation loss = 0. 0311, accuracy = 0. 9025
Epoch 4/7: average training loss = 0. 0123
Epoch 4/7: average validation loss = 0. 0326, accuracy = 0. 9025
Epoch 5/7: average training loss = 0. 0074
Epoch 5/7: average validation loss = 0. 0432, accuracy = 0. 8950
Epoch 6/7: average training loss = 0. 0047
Epoch 6/7: average validation loss = 0. 0518, accuracy = 0. 8975
Epoch 7/7: average training loss = 0. 0045
Epoch 7/7: average validation loss = 0. 0588, accuracy = 0. 8950
Test: average loss = 0. 0446, accuracy = 0. 9000
Training took 296. 69 seconds
```

(Figure DistilBERT trained by focal loss)





```
Epoch 1/7: average training loss = 29. 2614
Epoch 1/7: average validation loss = 12. 7873, accuracy = 0. 5200
Epoch 2/7: average training loss = 8. 2970
Epoch 2/7: average validation loss = 5. 5350, accuracy = 0. 5200
Epoch 3/7: average training loss = 4. 7762
Epoch 3/7: average validation loss = 4. 2860, accuracy = 0. 5200
Epoch 4/7: average training loss = 4. 1634
Epoch 4/7: average validation loss = 4. 0557, accuracy = 0. 5200
Epoch 5/7: average training loss = 4. 0146
Epoch 5/7: average validation loss = 3. 9669, accuracy = 0. 5200
Epoch 6/7: average training loss = 3. 9468
Epoch 6/7: average validation loss = 3. 9196, accuracy = 0. 7100
Epoch 7/7: average training loss = 3. 9045
Epoch 7/7: average validation loss = 3. 8832, accuracy = 0. 8825
Test: average loss = 3. 8841, accuracy = 0. 8500
Training took 670. 55 seconds
```

(Figure TinyBERT trained by focal loss)

Based on the result of base Bert and DistilBERT trained by the limited data, it's obvious that:

a. DistilBERT can achieve almost the same accuracy on the test set as base BERT but only needs half-time compared with base BERT. It also means we will have more time and computation space to train more data if we use DistilBERT.
b. DistilBERT has a less average loss on the test set, which means the true label and the prediction label are very close. The prediction is very stable. Because I only include 200 data in the test set. So the distance of their accuracy is very tiny and can be ignored. If I enlarge the test set, their performances may have a larger and more obvious distance.

7. Conclusion

In conclusion, DistilBERT trained by focal loss can provide a faster and low-cost approach to text classification, compared with Bert trained by cross-entropy or focal loss. TinyBert also has the potential ability to perform better than base Bert.

In general, it is important to consider the specific characteristics of the dataset and task when choosing a model and loss function, as different models and loss functions may be better suited for different types of data and tasks. It is also important to consider the computational resources available and the desired level of performance when deciding which model to use.

Furthermore, if I have more strong GPU or a long time, it's possible to verify these Bert Family models on multi-classification and discover more about TinyBert if I pretrain the TinyBert on my own.

Reference

Liu, Weijie, et al. "Fastbert: a self-distilling bert with adaptive inference time." *arXiv preprint arXiv:2004.02178* (2020).

Devlin, Jacob, et al. "Bert: Pre-training of deep bidirectional transformers for language understanding."*arXiv preprint arXiv:1810.04805* (2018).

Hinton, Geoffrey, Oriol Vinyals, and Jeff Dean. "Distilling the knowledge in a neural network." *arXiv preprint arXiv:1503.02531* 2.7 (2015).

Sanh et al. "DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter" *arXiv preprint arXiv: 1910.01108* .

Sun et al. “TinyBERT: Distilling BERT for Natural Language Understanding”. *arXiv preprint arXiv: 2010.11934*

Pretraining model reference:

https://huggingface.co/docs/transformers/v4.25.1/en/model_doc/distilbert#overview