CENTENNIAL COLLEGE

PyTorch

Sentiment Analysis on Movie Reviews : A predictive model with pre-trained Bert by PyTorch

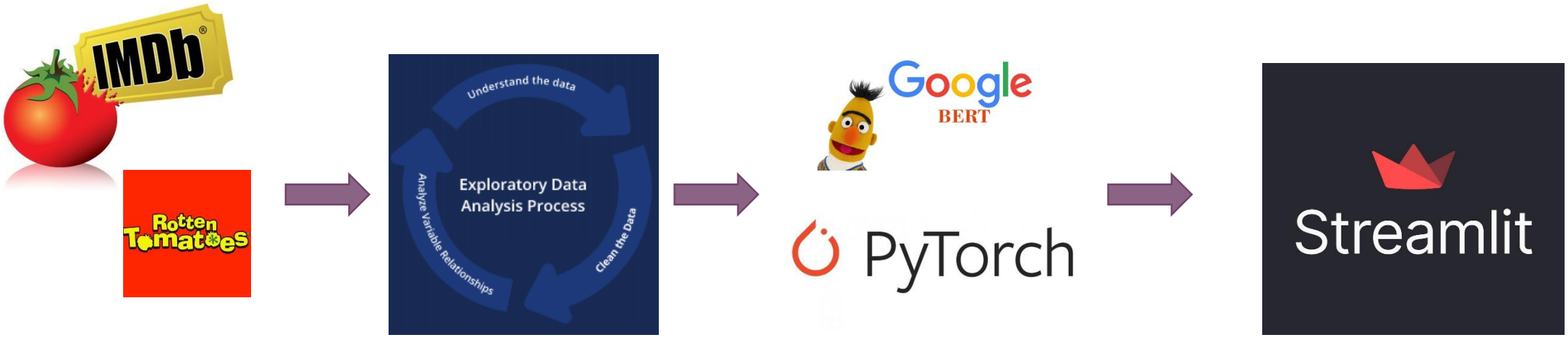
Huaye Zhan[1]

[1] Artificial Intelligence - Software engineer technology, Centennial College

content

1. Introduction
2. Dataset
3. EDA
4. Training
5. Model Deployment
6. Conclusion

Introduction: Workflow



Dataset: Kaggle API

KAGGLE · PLAYGROUND PREDICTION COMPETITION · 10 YEARS AGO

Late Submission

Sentiment Analysis on Movie Reviews

Classify the sentiment of sentences from the Rotten Tomatoes dataset

Overview | Data | Code | Models | Discussion | Leaderboard | Rules | Team | Submissions

Overview

Start
Feb 28, 2014

Close
Feb 28, 2015

Description

"There's a thin line between likably old-fashioned and fuddy-duddy, and The Count of Monte Cristo ... never quite settles on either side."

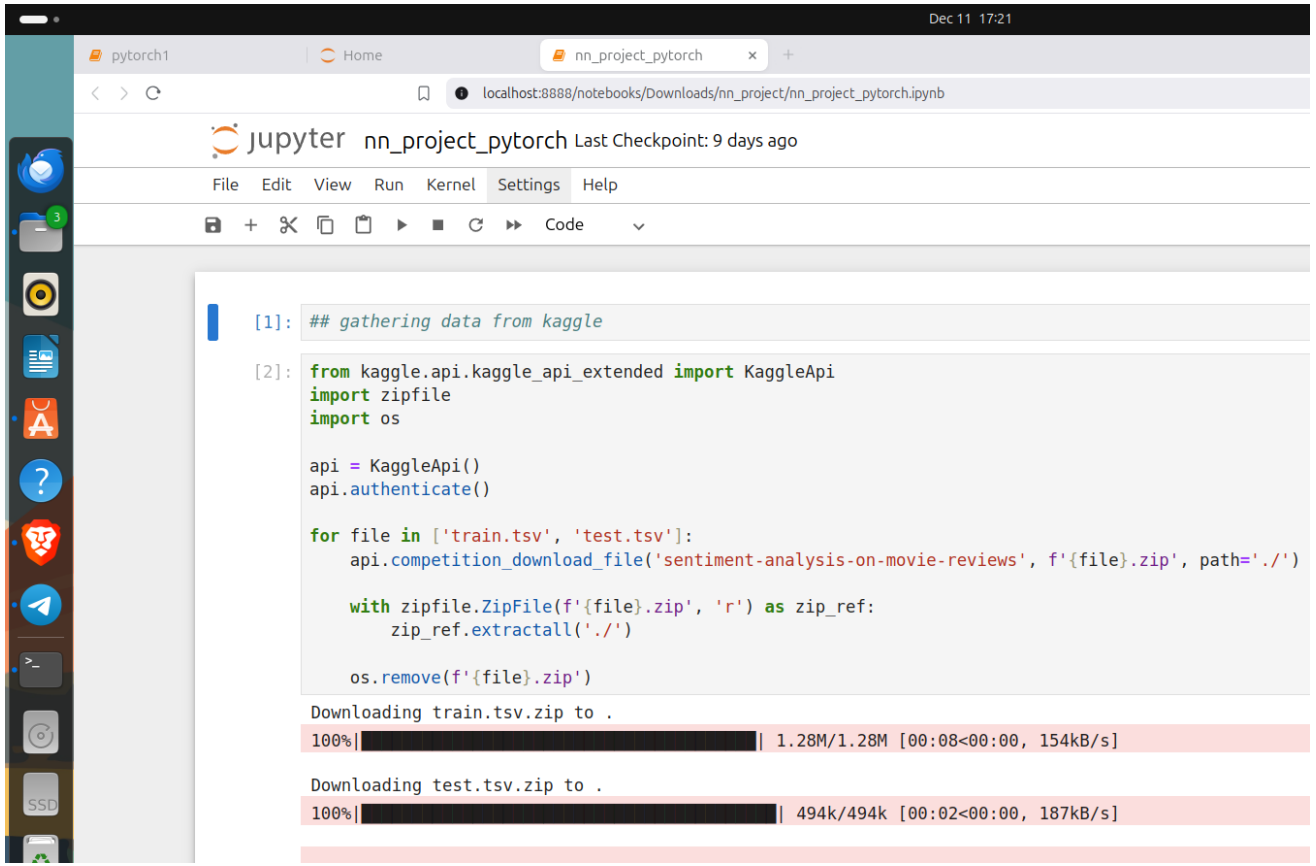
The Rotten Tomatoes movie review dataset is a corpus of movie reviews used for sentiment analysis, originally collected by Pang and Lee [1]. In their work on sentiment treebanks, Socher et al. [2] used Amazon's Mechanical Turk to create fine-grained labels for all parsed phrases in the corpus. This competition presents a chance to benchmark your sentiment-analysis ideas on the Rotten Tomatoes dataset. You are asked to label phrases on a scale of five values: negative, somewhat negative, neutral, somewhat positive, positive. Obstacles like sentence negation, sarcasm,

Competition Host
Kaggle

Prizes & Awards
Knowledge
Does not award Points or Medals

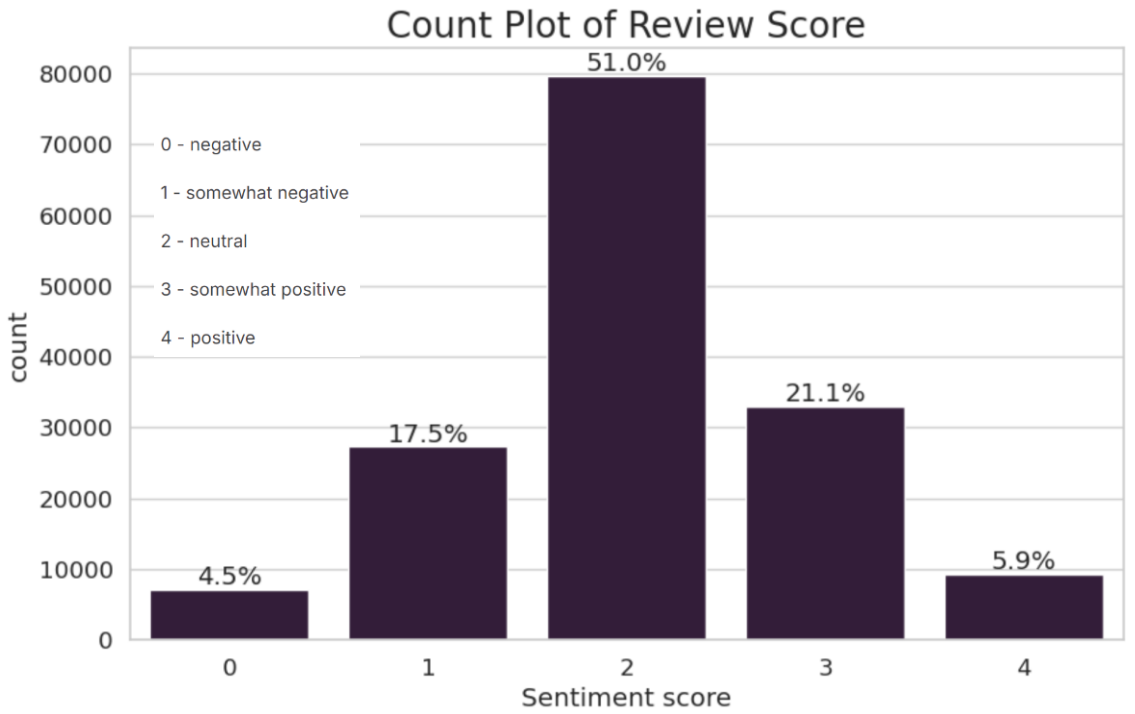
Participation
1,510 Entrants
1,011 Participants
860 Teams
6,813 Submissions

Tags
Text | Multiclass Classification | Categorization Accuracy

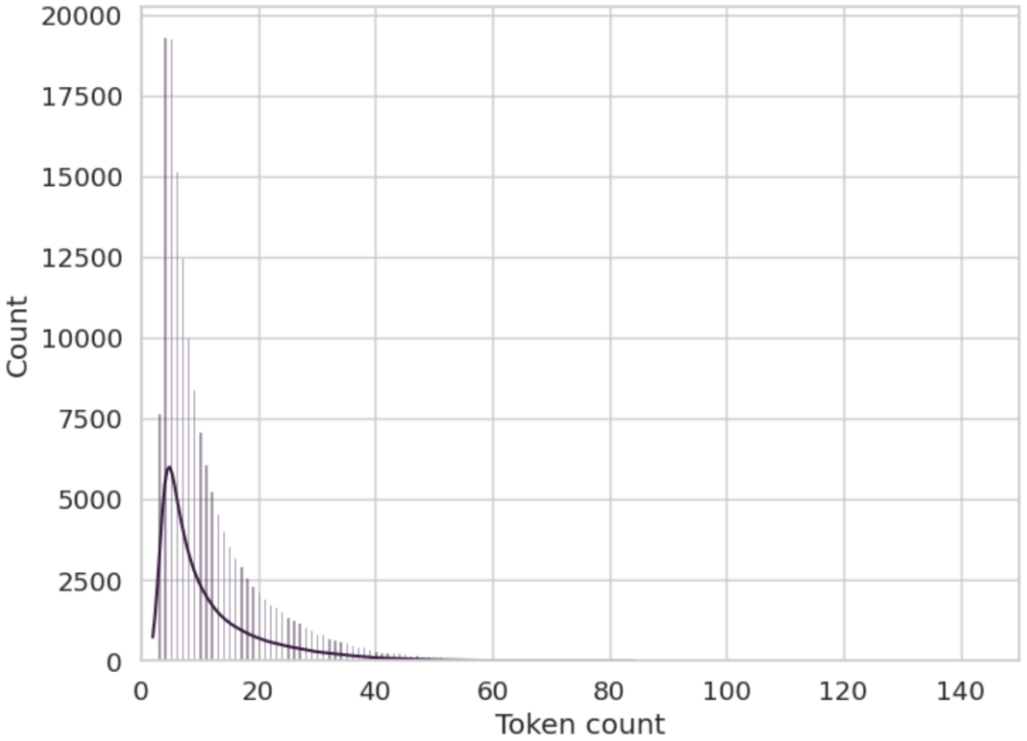


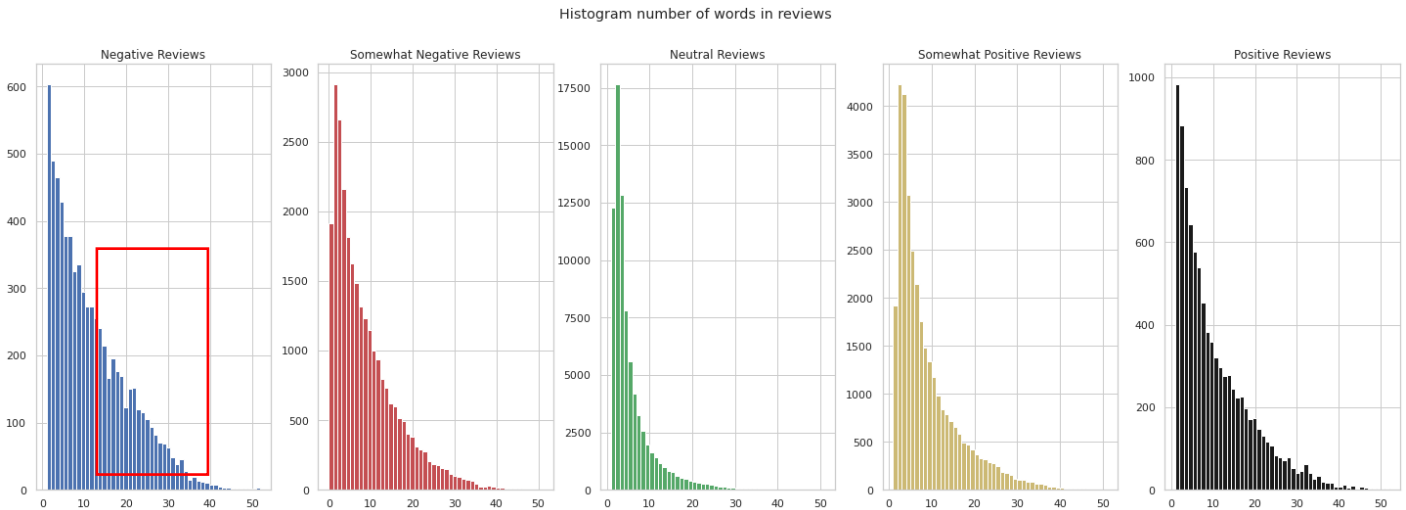
EDA

	PhraseId	SentenceId	Phrase	Sentiment
0	1	1	A series of escapades demonstrating the adage ...	1
1	2	1	A series of escapades demonstrating the adage ...	2
2	3	1	A series	2
3	4	1	A	2
4	5	1	series	2

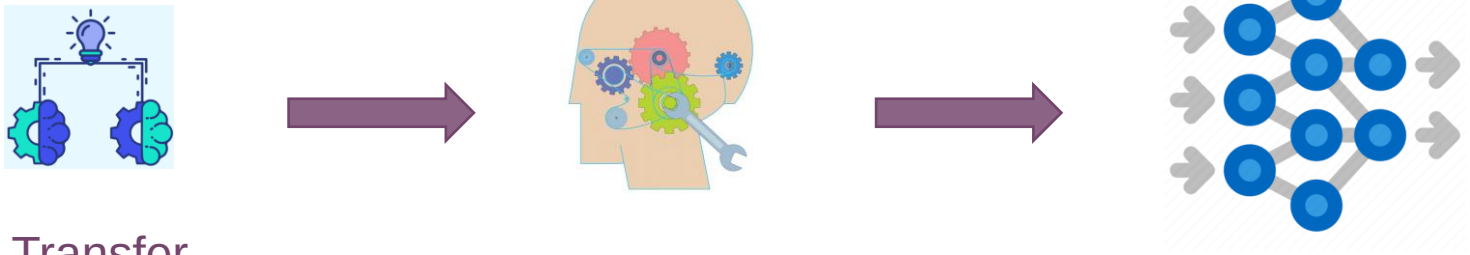


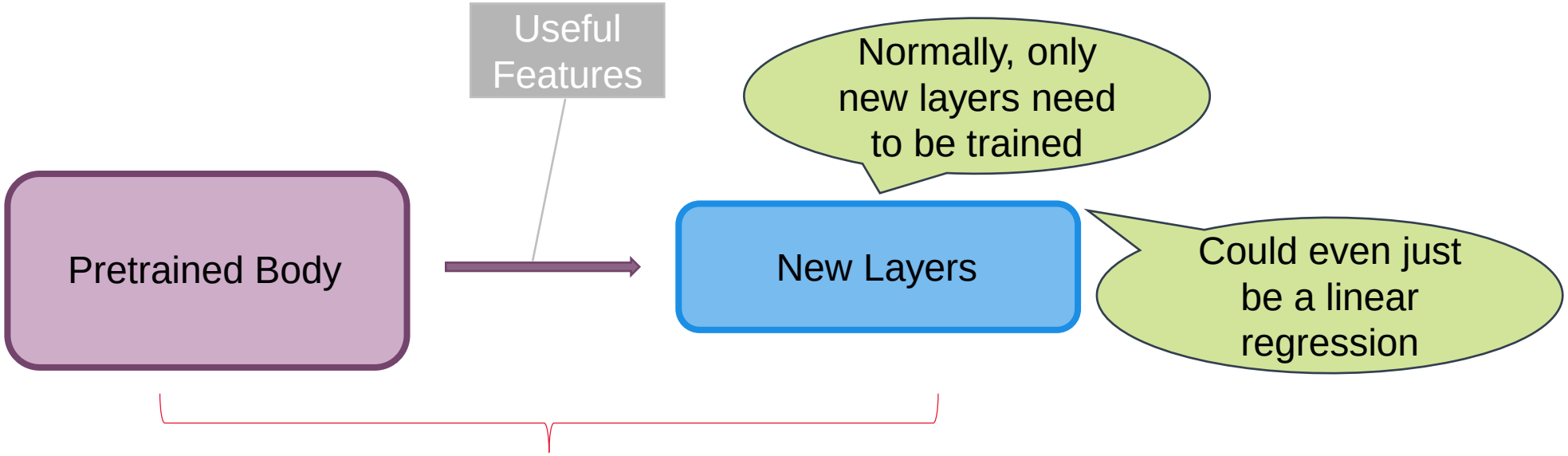
EDA


Count
Token count
0
2500
5000
7500
10000
12500
15000
17500
20000
0
20
40
60
80
100
120
140


Histogram number of words in reviews
Negative Reviews
Somewhat Negative Reviews
Neutral Reviews
Somewhat Positive Reviews
Positive Reviews

Training: Transfer learning and Fine tuning

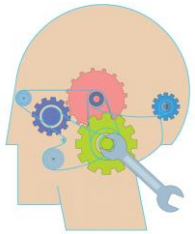




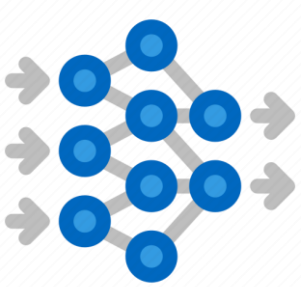
In this project, I fine-tune the entire model, including the BERT layers

Training: pytorchinfo

Transfer Learning

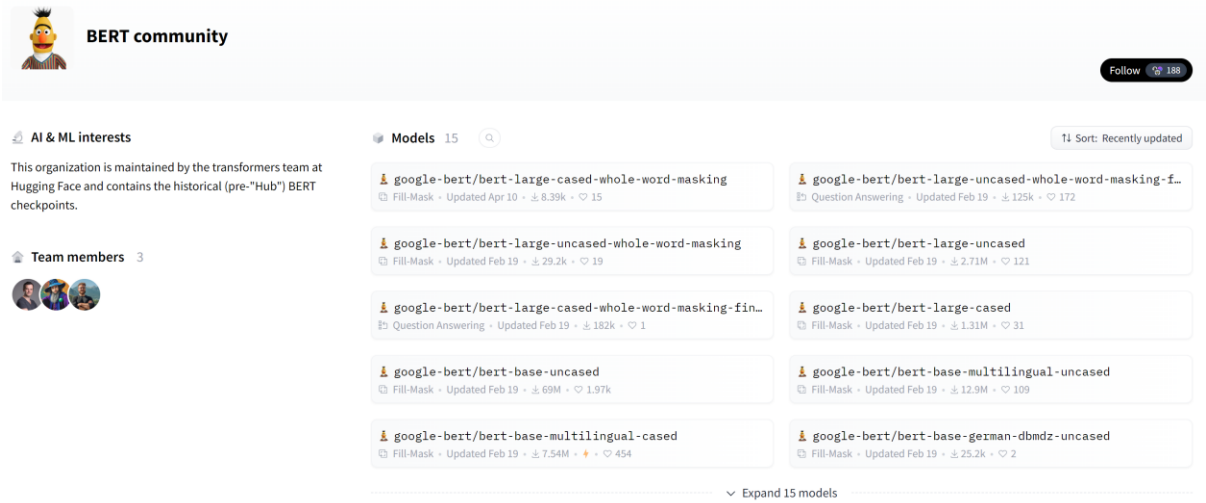


Fine-Tuning



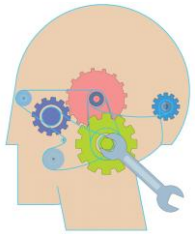
Output

```
==========================================================================================
Layer (type:depth-idx)                                   Param #
==========================================================================================
SentimentClassifier                                      --
├─BertModel: 1-1                                         --
│    └─BertEmbeddings: 2-1                               --
│    │    └─Embedding: 3-1                               22,268,928
│    │    └─Embedding: 3-2                               393,216
│    │    └─Embedding: 3-3                               1,536
│    │    └─LayerNorm: 3-4                               1,536
│    │    └─Dropout: 3-5                                 --
│    └─BertEncoder: 2-2                                  --
│    │    └─ModuleList: 3-6                              85,054,464
│    └─BertPooler: 2-3                                   --
│    │    └─Linear: 3-7                                  590,592
│    │    └─Tanh: 3-8                                    --
├─Dropout: 1-2                                           --
├─Linear: 1-3                                            3,845
==========================================================================================
Total params: 108,314,117
Trainable params: 108,314,117
Non-trainable params: 0
==========================================================================================
```

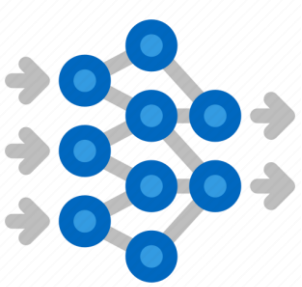


Training: Result

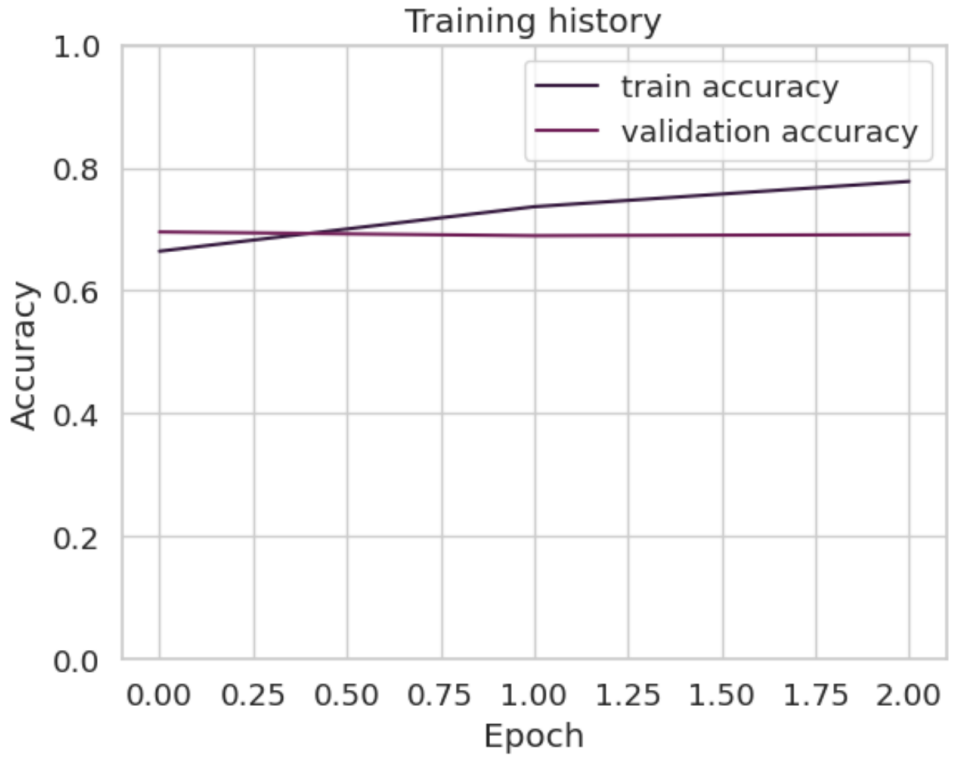
Transfer Learning



Fine-Tuning



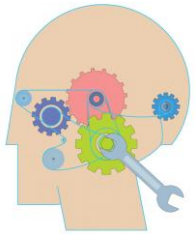
Output



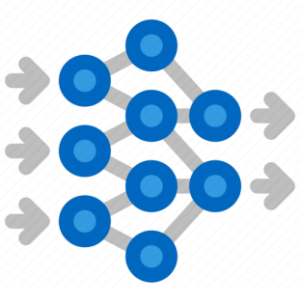

```
Epoch 1/3
----------------------------------

Train loss 0.8075 accuracy 0.6642

Val loss   0.7417 accuracy 0.6956

Epoch 2/3
----------------------------------

Train loss 0.6434 accuracy 0.7367

Val loss   0.7612 accuracy 0.6895

Epoch 3/3
----------------------------------

Train loss 0.5542 accuracy 0.7779

Val loss   0.8015 accuracy 0.6913
```

Training: Result

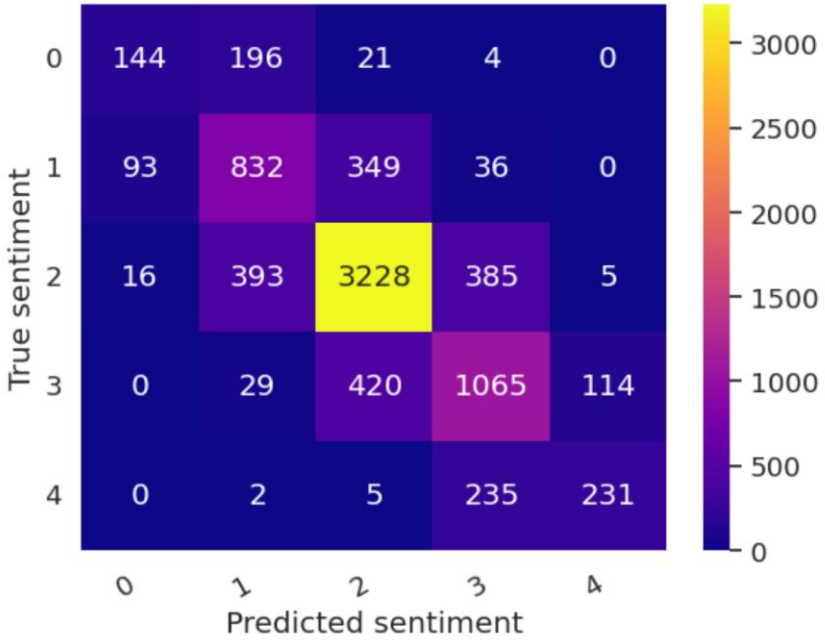
Transfer Learning



Fine-Tuning

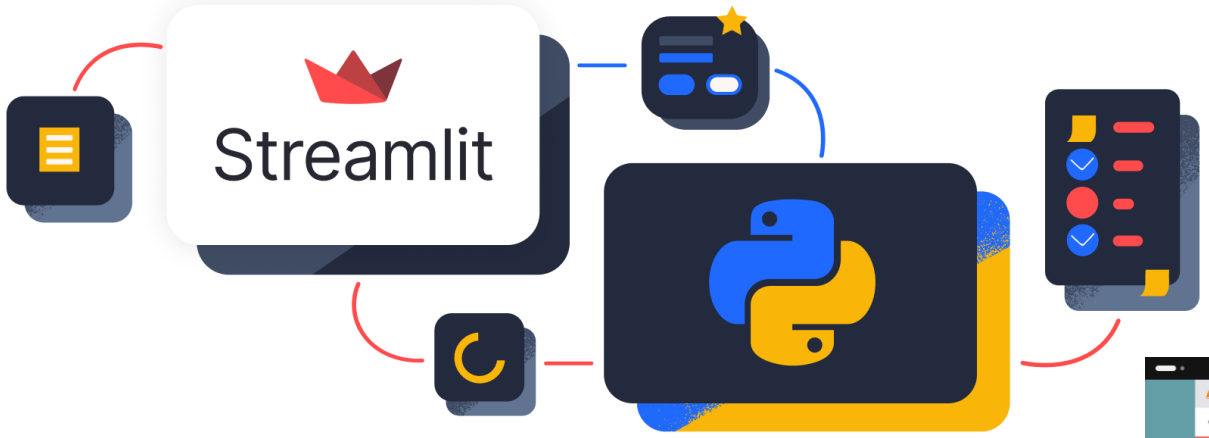


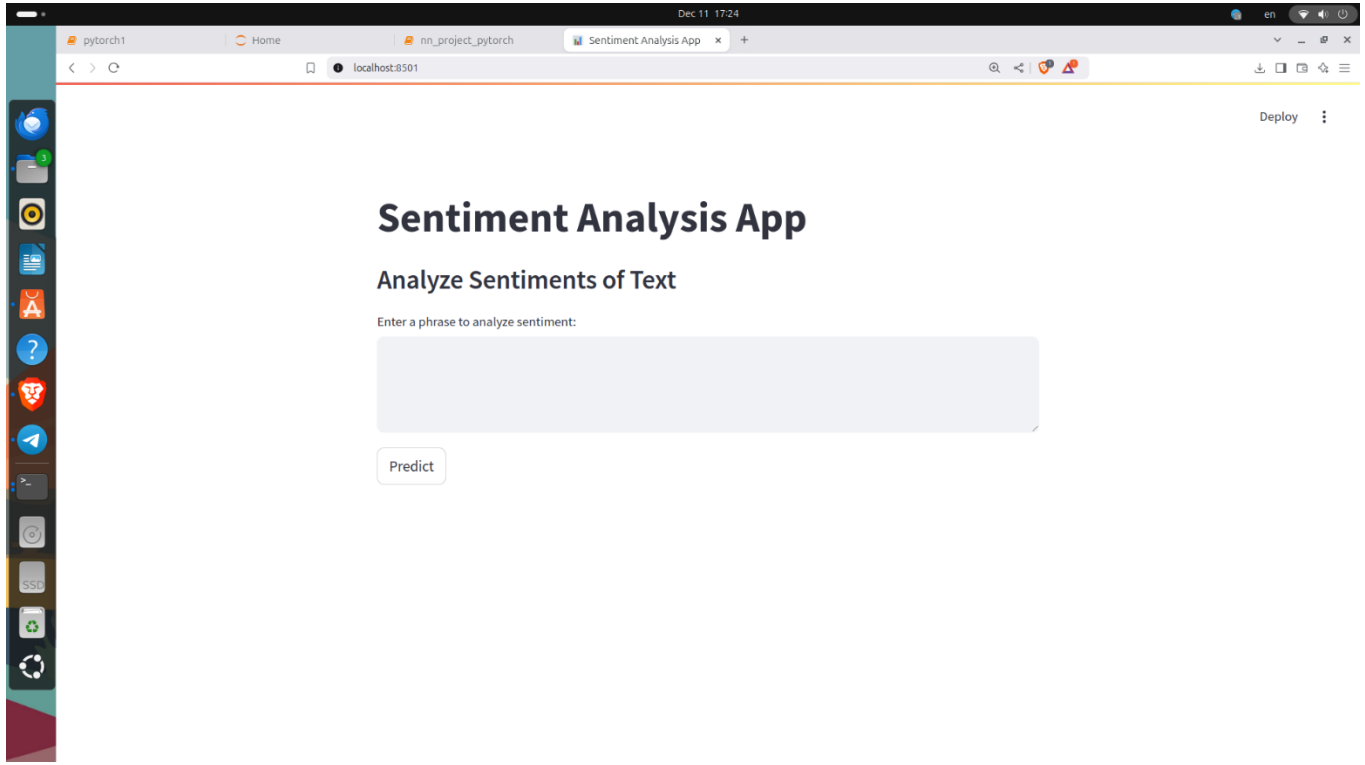
Output



Class	Precision	Recall	F1-Score	Support	Accuracy
0	0.57	0.39	0.47	365	
1	0.57	0.64	0.60	1310	
2	0.80	0.80	0.80	4027	
3	0.62	0.65	0.64	1628	
4	0.66	0.49	0.56	473	
					0.70

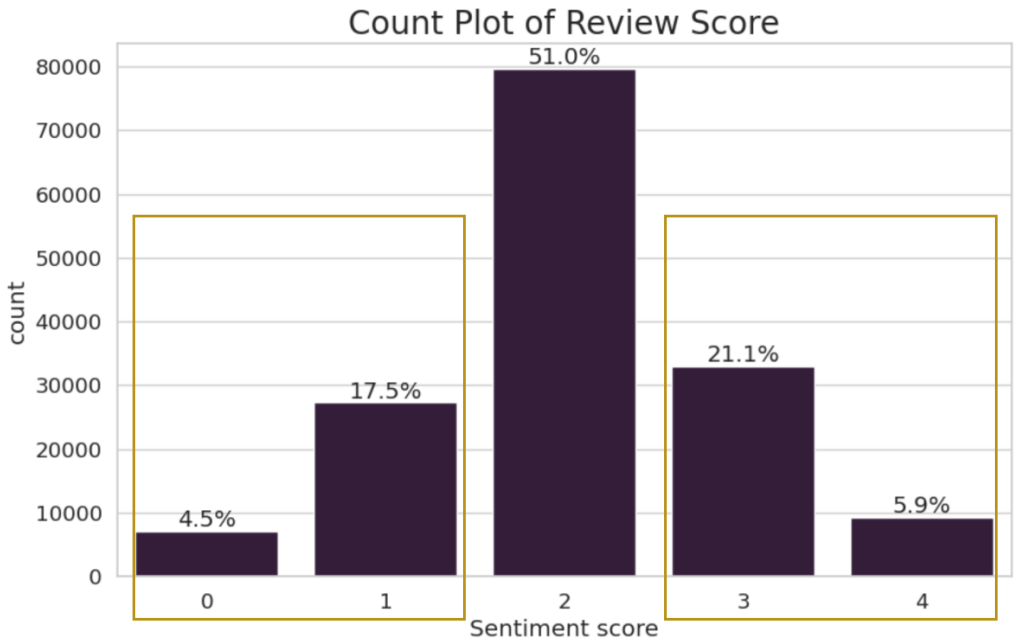
Model Deployment: Live Demo





Conclusion: limitation and Improvement

- **Data binning and Data processing**: bin 0 into 1, bin 3 into 4, to make the data less imbalanced in preprocessing data.
- **Model comparison**: compared with other pre-training models
- Train more epochs to observe the result




```
Epoch 1/10
----------

Train loss 0.8136 accuracy 0.6617

Val loss   0.7496 accuracy 0.6935

Epoch 2/10
----------

Train loss 0.6601 accuracy 0.7296

Val loss   0.7827 accuracy 0.6900

Epoch 3/10
----------

Train loss 0.5709 accuracy 0.7709

Val loss   0.8427 accuracy 0.6822

Epoch 4/10
----------

Train loss 0.4885 accuracy 0.8084

Val loss   0.9527 accuracy 0.6742

Epoch 5/10
----------

Train loss 0.4144 accuracy 0.8420

Val loss   1.0937 accuracy 0.6606

Epoch 6/10
----------
```

Thank you!