



## Lab 2 Homework

# COMPARING MODEL PERFORMANCE: TRADITIONAL MACHINE LEARNING VS. DEEP LEARNING

NAME : 鄭丞恩 / ENZO CHENG @GROUP 83

2024/12/7



Thank you for offering this course with the opportunity provided by Lab 2, which brought a glimmer of hope to my team amidst a crisis. In the last final project progress report, I mentioned that all the members of my group had dropped out of this course, forcing me to seek help from a classmate I previously worked with. He is not taking this course this year but plans to take it next year.

After evaluating our capabilities, we realized that our skills were limited to basic Python programming and some fundamental knowledge of data mining. Through continuous learning this semester, I have made significant progress. Aside from participating in two Google events and completing one Google AI course, I focused intently on the professor's YouTube videos and this lab homework. These efforts allowed me to overcome numerous challenges, including the lack of GPU resources and finding suitable programming environments, which ultimately elevated my skills to a new level.

The entire process was incredibly exciting because I encountered many challenges I had never faced before, but I managed to overcome each one. I am deeply grateful to the professor and teaching assistants for offering this course. I will continue to pursue further learning in this direction in the future.

# OVERCOMING CHALLENGES, EMBRACING OPPORTUNITIES

# STEPS IN THE LEARNING JOURNEY

- 1. Learning and Applying the Procedural Workflow in Lab 2**
- 2. Start writing programs to verify whether some functions are working properly**
  - Dual GPU Programming
  - Noise Handling
  - Over-Fitting and Over-Processing
- 3. Refine the BERT Embedding program to achieve better results in Kaggle competitions.**
  - Progressed from BERT embedding to optimizing the RoBERTa model.
- 4. Reflections After Completing the Assignment**

# 1. LEARNING AND APPLYING THE PROCEDURAL WORKFLOW IN LAB 2

1. Organize / Clean Data



2. Feature Encoding



3. Model Training



4. Model Evaluation



5. Generate Report

These were concepts I was not familiar with before, but through Lab 2, I have gained a basic understanding and practical experience.

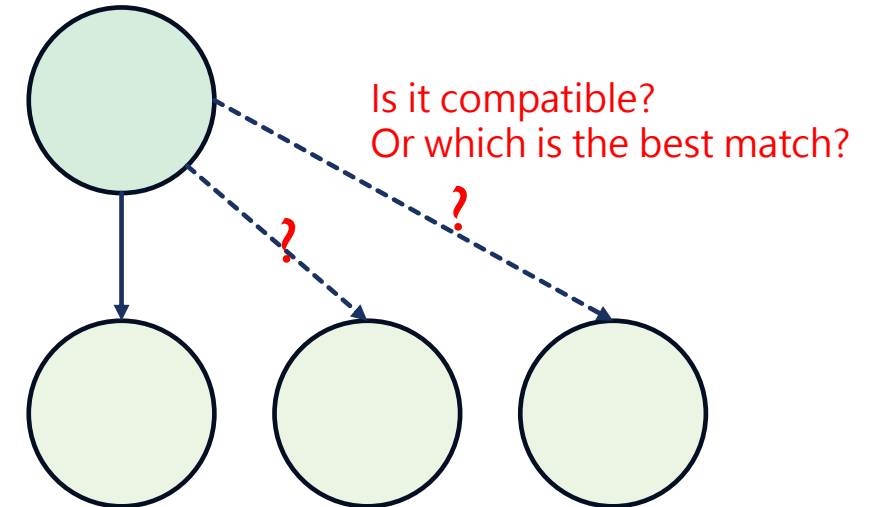


# 1. LEARNING AND APPLYING THE PROCEDURAL WORKFLOW IN LAB 2

## 2. Feature Encoding



There may be  $n$ .



## 3. Model Training



There may be  $k$ .

Is there a compatibility issue  
between feature encoding and  
model training?



Conduct a simple method study before starting to solve the question.

# 1. LEARNING AND APPLYING THE PROCEDURAL WORKFLOW IN LAB 2

Organize correspondences between feature transformations and models before programming.

Method	Feature Engineering	Model	Accuracy Estimation	Advantages	Disadvantages	GPU Support
TF-IDF + Random Forest	Sparse feature representation with term frequency and inverse document frequency weighting	Random Forest	75%-82%	Strong model stability, not sensitive to noise and high-dimensional data	Cannot handle nonlinear patterns, insufficient use of semantic information	Not supported
TF-IDF + Boosting	Sparse feature representation with term frequency and inverse document frequency weighting	XGBoost or LightGBM	78%-85%	Excels at handling sparse features, adapts well to misclassified samples	Slightly high training cost, requires parameter tuning for optimal performance	Supported (significant acceleration, suitable for large datasets)
Word2Vec + Random Forest	Word embeddings, calculating sentence vector averages	Random Forest	72%-80%	Combines semantic features from word embeddings, enhances semantic capture	Requires preprocessing for word embeddings, limited handling of nonlinear semantics in Random Forest	Not supported
Word2Vec + CNN	Word embeddings, preserving word order	Convolutional Neural Network (CNN)	75%-85%	Captures local semantic features, performs well on short texts	Requires significant resources for training, limited effectiveness on long texts	Supported (significant acceleration)
BERT Embedding + Transformer	Contextual semantic embedding, preserving global semantics	Pretrained BERT model	85%-90%	Captures contextual semantics, highest classification accuracy	High training and inference cost, requires large amounts of data and resources	Supported (necessary, otherwise slow)
Tokenizer + LSTM	Digitizes text sequences, retains sequential context	Long Short-Term Memory (LSTM)	80%-88%	Captures text sequence features, suitable for time-series or long texts	Moderate training cost, may encounter gradient vanishing in long texts	Supported (significant acceleration, necessary)
RoBERTa + Transformer	Contextual semantic embedding, optimized for downstream tasks	Pretrained RoBERTa model	88%-92%	Enhanced training optimization, better at handling long sequences	Similar to BERT, high computational cost and resource requirements	Supported (necessary for large-scale tasks)

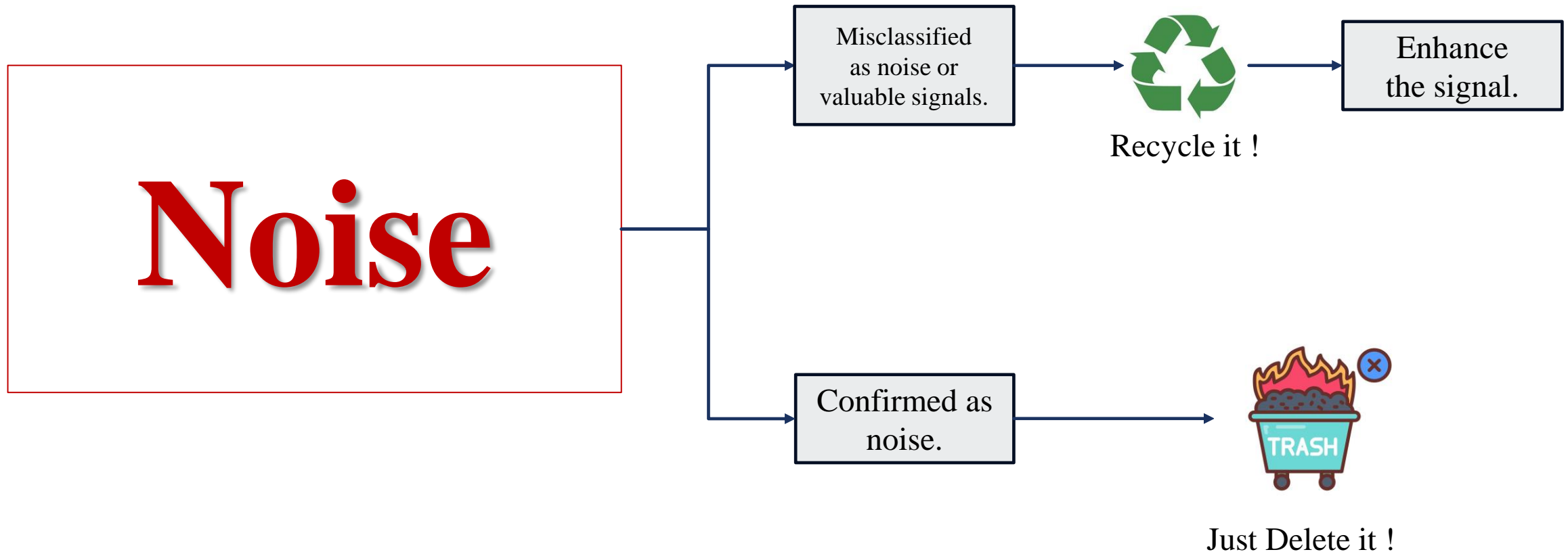
## 2. START WRITING PROGRAMS TO VERIFY WHETHER SOME FUNCTIONS ARE WORKING PROPERLY

Method	Feature Engineering	Model	Accuracy Estimation	Accuracy without data cleaning	GPU support in Kaggle
TF-IDF + Random Forest	Sparse feature representation with term frequency and inverse document frequency weighting	Random Forest	75%-82%	0.4808	No
TF-IDF + Boosting	Sparse feature representation with term frequency and inverse document frequency weighting	XGBoost or LightGBM	78%-85%	0.478	GPU T4 x 2
Word2Vec + Random Forest	Word embeddings, calculating sentence vector averages	Random Forest	72%-80%	0.4586	No
Word2Vec + CNN	Word embeddings, preserving word order	Convolutional Neural Network (CNN)	75%-85%	0.5514	GPU T4 x 2
BERT Embedding + Transformer	Contextual semantic embedding, preserving global semantics	Pretrained BERT model	85%-90%	0.6439	GPU T4 x 2

- We observed that programs capable of GPU acceleration are all neural network-based methods, while those not using GPUs belong to statistical methods.
- Although BERT Embedding achieves higher accuracy, modifying the program to run on dual GPUs and debugging during the actual training phase consumed a significant amount of our time. Therefore, we decided to focus on optimizing BERT Embedding in subsequent work.
- Adopting BERT Embedding indeed yielded better results in the Kaggle competition.

## 2. START WRITING PROGRAMS TO VERIFY WHETHER SOME FUNCTIONS ARE WORKING PROPERLY

For noise handling, we made very simple assumptions.





## 2. START WRITING PROGRAMS TO VERIFY WHETHER SOME FUNCTIONS ARE WORKING PROPERLY

```
import json

data = []
with open('/kaggle/input/dm-2024-isa-5810-lab-2-homework/tweets_DM.json', 'r') as f:
    for line in f:
        data.append(json.loads(line))

# 獲取前 2000 筆資料
first_2000_entries = data[:2000]

# 列印前 2000 筆資料
for entry in first_2000_entries:
    print(entry)
```

```
print(df_tags)
```

Tag	Count
0 lh	2296492

There are many mysterious symbols, such as <LH>, but they seem to provide no help in understanding the sentence's semantics.  
Just Delete !

```
{ '_score': 391, '_index': 'hashtag_tweets', '_source': { 'tweet': { 'hashtags': ['Snapchat'], 'tweet_id': '0x376b20', 'text': 'People who post "add me on #Snapchat" must be dehydrated. Cuz man.... that<LH>' }, '_crawldate': '2015-05-23 11:42:47', '_type': 'tweets' } }
{ '_score': 433, '_index': 'hashtag_tweets', '_source': { 'tweet': { 'hashtags': ['freepress', 'TrumpLegacy', 'CNN'], 'tweet_id': '0x2d5350', 'text': '@brianklaas As we see, Trump is dangerous to #freepress around the world. What a <LH> <LH> #TrumpLegacy. #CNN' }, '_crawldate': '2016-01-28 04:52:09', '_type': 'tweets' } }
{ '_score': 232, '_index': 'hashtag_tweets', '_source': { 'tweet': { 'hashtags': ['bibleverse'], 'tweet_id': '0x28b412', 'text': 'Confident of your obedience, I write to you, knowing that you will do even more than I ask. (Philemon 1:21) 3/4 #bibleverse <LH> <LH>' }, '_crawldate': '2017-12-25 04:39:20', '_type': 'tweets' } }
{ '_score': 376, '_index': 'hashtag_tweets', '_source': { 'tweet': { 'hashtags': [], 'tweet_id': '0x1cd5b0', 'text': 'Now ISSA is stalking Tasha 🤔🤔🤔 <LH>' }, '_crawldate': '2016-01-24 23:53:05', '_type': 'tweets' } }
{ '_score': 989, '_index': 'hashtag_tweets', '_source': { 'tweet': { 'hashtags': [], 'tweet_id': '0x2de201', 'text': '"Trust is not the same as faith. A friend is someone you trust. Putting faith in anyone is a mistake." ~ Christopher Hitchens <LH> <LH>' }, '_crawldate': '2016-01-08 17:18:59', '_type': 'tweets' } }
{ '_score': 120, '_index': 'hashtag_tweets', '_source': { 'tweet': { 'hashtags': ['authentic', 'LaughOutLoud'], 'tweet_id': '0x1d755c', 'text': '@RISKshow @TheKevinAllison Thx for the BEST TIME tonight. What stories! Heart breakingly <LH> #authentic #LaughOutLoud good!!' }, '_crawldate': '2015-06-11 04:44:05', '_type': 'tweets' } }
{ '_score': 1021, '_index': 'hashtag_tweets', '_source': { 'tweet': { 'hashtags': [], 'tweet_id': '0x2c91a8', 'text': 'Still waiting on those supplies Liscus. <LH>' }, '_crawldate': '2015-08-18 02:30:07', '_type': 'tweets' } }
{ '_score': 481, '_index': 'hashtag_tweets', '_source': { 'tweet': { 'hashtags': [], 'tweet_id': '0x368e95', 'text': 'Love knows no gender. 🤔🤔 <LH>' }, '_crawldate': '2015-08-20 14:31:27', '_type': 'tweets' } }
{ '_score': 827, '_index': 'hashtag_tweets', '_source': { 'tweet': { 'hashtags': ['LeagueCup'], 'tweet_id': '0x249c0c', 'text': '@DStvNgCare @DStvNg More highlights are being shown than actual sports! Who watches triathlon highlights anyway? <LH> #LeagueCup' }, '_crawldate': '2016-04-18 13:01:02', '_type': 'tweets' } }
{ '_score': 66, '_index': 'hashtag_tweets', '_source': { 'tweet': { 'hashtags': ['materialism', 'money', 'possessions'], 'tweet_id': '0x218443', 'text': 'When do you have enough ? When are you satisfied ? Is you goal really all about money ? #materialism #money #possessions <LH>' }, '_crawldate': '2015-09-09 09:22:55', '_type': 'tweets' } }
{ '_score': 631, '_index': 'hashtag_tweets', '_source': { 'tweet': { 'hashtags': ['SSM', 'gender', 'diversity'], 'tweet_id': '0x359db9', 'text': 'The #SSM debate; <LH> (a manufactured fantasy used to distract the ignorant masses from their mundane lives) V #gender #diversity (a m.....)' }, '_crawldate': '2016-08-05 21:15:36', '_type': 'tweets' } }
{ '_score': 839, '_index': 'hashtag_tweets', '_source': { 'tweet': { 'hashtags': [], 'tweet_id': '0x23b037', 'text': '"I love suffering 🤔🤔 I love when valium does nothing to help 🤔🤔 I love when my doctors say that the y've done all they can 🤔🤔 <LH>' }, '_crawldate': '2016-01-28 08:11:22', '_type': 'tweets' } }
{ '_score': 560, '_index': 'hashtag_tweets', '_source': { 'tweet': { 'hashtags': ['Pissed'], 'tweet_id': '0x1fde89', 'text': 'Can someone tell my why my feeds scroll back to the same 30 tweets that I saw 1 min ago? #Pissed!' }, '_crawldate': '2016-01-26 22:09:45', '_type': 'tweets' } }
{ '_score': 887, '_index': 'hashtag_tweets', '_source': { 'tweet': { 'hashtags': ['justgradstudentthings', 'ecology'], 'tweet_id': '0x37a0a9', 'text': 'You know you research butterflies when predictive text autocorrects "b ut" to "butterfly" #justgradstudentthings #ecology <LH>' }, '_crawldate': '2015-08-11 16:22:15', '_type': 'tweets' } }
```

## 2. START WRITING PROGRAMS TO VERIFY WHETHER SOME FUNCTIONS ARE WORKING PROPERLY

I also wrote a program to check the quantity and distribution of emojis.

```
import json
import regex
import collections
import matplotlib.pyplot as plt

# 定義從文本中提取表情符號的函數
def extract_emojis(text):
    """從給定的文本中提取所有表情符號"""
    emoji_list = []
    data = regex.findall(r'\X', text)
    for word in data:
        if any(char in emoji.EMOJI_DATA for char in word):
            emoji_list.append(word)
    return emoji_list

# 讀取 JSON 檔案並解析為列表
data = []
with open('/kaggle/input/dm-2024-isa-5810-lab-2-homework/tweets_DM.json', 'r') as f:
    for line in f:
        data.append(json.loads(line))

# 提取所有推文中的表情符號
all_emojis = []
for entry in data:
    tweet_text = entry['_source']['tweet']['text']
    emojis_in_tweet = extract_emojis(tweet_text)
    all_emojis.extend(emojis_in_tweet)

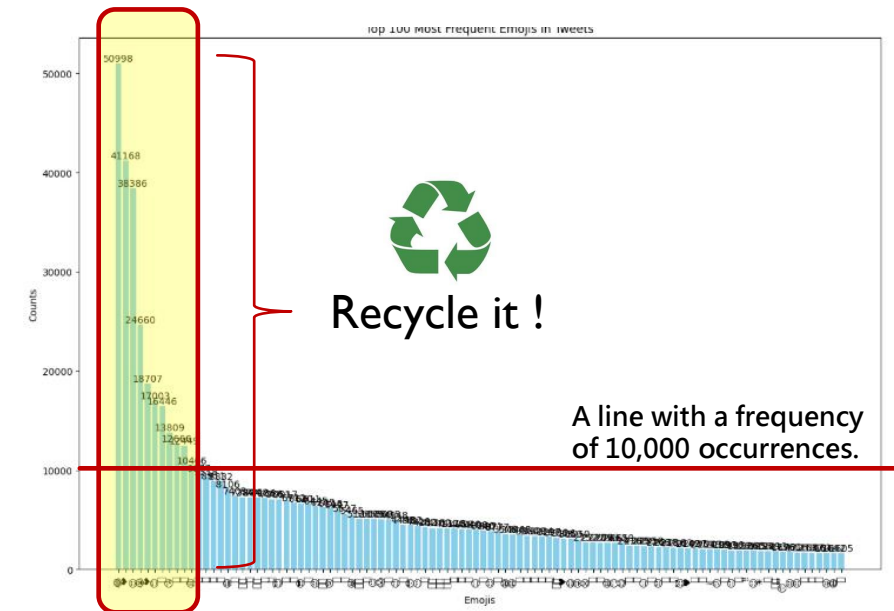
# 計算每個表情符號的出現次數
emoji_counter = collections.Counter(all_emojis)

# 取得出現頻率最高的前 100 個表情符號
top_500_emojis = emoji_counter.most_common(500)

# 分離表情符號和對應的計數
emojis, counts = zip(*top_500_emojis)
```

[7]: top\_500\_emojis

[7]: [('😄', 50998),  
(❤️, 41168),  
(😂, 38386),  
(😭, 24660),  
(💖, 18707),  
(😊, 17003),  
(🙏, 16446),  
(😇, 13809),  
(💕, 12666),  
(🔥, 12449),  
(😓, 10466),  
(😬, 9677),  
(👉, 9318),  
(💙, 8911),  
(😏, 8832),  
(👀, 8106),  
(🙏, 7403),  
(🙏, 7284),  
(🙏, 7284),  
(🙏, 7262),  
(🌟, 7186),  
(💜, 7086),  
(😄, 7051),  
(👉, 7017)]



1. These are groups that appear with high frequency.
2. Frequent use of these emojis can be assumed to convey significant emotions.
3. It can be hypothesized that these are prominent features within the model. **Just Recycle it !**

## 2. START WRITING PROGRAMS TO VERIFY WHETHER SOME FUNCTIONS ARE WORKING PROPERLY

Feature enhancement & retention.

Feature  
enhancement

Feature  
retention

To make the  
features more  
distinct.

Emoji	英文含義	單詞
😂	Face with Tears of Joy	[joy]
❤️	Red Heart	[love]
😍	Smiling Face with Heart-Eyes	[love]
😭	Loudly Crying Face	[cry]
❤️	Red Heart	[love]
😊	Smiling Face with Smiling Eyes	[happy]
🙏	Folded Hands	[pray]
😘	Face Blowing a Kiss	[kiss]
💕	Two Hearts	[love]
🔥	Fire	[fire]
😓	Weary Face	[weary]
😏	Thinking Face	[think]
💯	Hundred Points	[perfect]
💙	Blue Heart	[loyalty]
🙄	Face with Rolling Eyes	[annoyed]
😄	Beaming Face with Smiling Eyes	[happy]
🙌	Raising Hands	[celebrate]
🙏	Folded Hands: Medium-Dark Skin Tone	[pray]
👍	Thumbs Up	[approve]
🙏	Folded Hands: Medium Skin Tone	[pray]

Create a dictionary.

- Incorporate around 10,000 emojis into the dictionary to serve as a filter for transformation purposes.

```
emoji_dict = {  
  '😂': '[joy]',  
  '❤️': '[love]',  
  '😍': '[love]',  
  '😭': '[cry]',  
  '❤️': '[love]',  
  '😊': '[happy]',  
  '🙏': '[pray]',  
  '😘': '[kiss]',  
  '💕': '[love]',  
  '🔥': '[fire]',  
  '😓': '[weary]',  
  '😏': '[think]',  
  '💯': '[perfect]',  
  '💙': '[love]',  
  '🙄': '[annoyed]',  
  '😄': '[happy]',  
  '🙌': '[celebrate]',  
  '🙏': '[pray]',  
  '👍': '[approve]',  
  '🙏': '[pray]'  
}
```

## 2. START WRITING PROGRAMS TO VERIFY WHETHER SOME FUNCTIONS ARE WORKING PROPERLY

Method	Feature Engineering	Model	Accuracy Estimation	Accuracy without data cleaning	Accuracy with data cleaning	GPU support in Kaggle
TF-IDF + Random Forest	Sparse feature representation with term frequency and inverse document frequency weighting	Random Forest	75%-82%	0.4808	0.4708	No
TF-IDF + Boosting	Sparse feature representation with term frequency and inverse document frequency weighting	XGBoost or LightGBM	78%-85%	0.478	0.4675	GPU T4 x 2
Word2Vec + Random Forest	Word embeddings, calculating sentence vector averages	Random Forest	72%-80%	0.4586	0.4566	No
Word2Vec + CNN	Word embeddings, preserving word order	Convolutional Neural Network (CNN)	75%-85%	0.5514	0.5312	GPU T4 x 2
BERT Embedding + Transformer	Contextual semantic embedding, preserving global semantics	Pretrained BERT model	85%-90%	0.6439	0.6294	GPU T4 x 2

- After completing data cleaning (including removing <LH> and filtering out emojis with fewer than 10,000 occurrences) and enhancing signals through the dictionary, we were hopeful of achieving higher accuracy. However, the reality was harsh—the results we obtained were worse than those without cleaning (as shown in the red section).
- At that time, we were very frustrated and had no choice but to abandon data cleaning and continue the competition as is.
- Later, when we came across the concept of over-processing, we felt somewhat relieved as we began to understand where we might have gone wrong. We then regarded **over-fitting** and **over-processing** as key points to watch out for in future data cleaning and programming tasks.

## 2. START WRITING PROGRAMS TO VERIFY WHETHER SOME FUNCTIONS ARE WORKING PROPERLY

key points to watch out for when programming

Aspect	Over-fitting	Over-processing
Definition	When the model focuses too much on the details and noise of the training data, leading to poor performance on unseen data.	When excessive or unnecessary processing in data preparation leads to loss of crucial information or changes in data distribution.
Scope	Model-related issue.	Data-related issue.
Cause	Overly complex models (e.g., too many parameters or layers).	Overly aggressive data cleaning or transformation.
Symptoms	- High accuracy on training data but poor performance on test or real-world data.	- Loss of key features or patterns due to excessive filtering or cleaning.
Impact	Reduces the model's ability to generalize to new data.	Prevents the model from learning meaningful patterns, leading to underperformance.
Example	A classifier achieving 99% accuracy on training data but only 70% on test data.	Removing all words with low frequency, losing critical <b>long-tail</b> information.
Prevention	- Use regularization techniques.	- Avoid excessive filtering or transformations.
	- Simplify the model architecture.	- Test data processing impacts on model performance before finalizing.
	- Use more training data.	
Focus for Resolution	Optimize the model's complexity and reduce overfitting risks.	Balance data cleaning and transformation to retain meaningful information.

### 1. Avoid Over-fitting (Model-Related)

- **Simplify the Model:**  
Use fewer layers or parameters to prevent over-complexity.
- **Apply Regularization:**  
Techniques like L1/L2 regularization, dropout, or early stopping can reduce over-fitting.
- **Increase Training Data:**  
Incorporate more diverse training samples to improve generalization.
- **Validate Regularly:**  
Continuously monitor validation set performance to avoid over-optimization on training data.

### 2. Avoid Over-processing (Data-Related)

- **Balance Data Cleaning:**  
Avoid excessive filtering that removes rare but important features.
- **Test Data Transformation:**  
Analyze how each processing step affects the model's performance before finalizing it.
- **Preserve Key Features:**  
Ensure critical information isn't lost during cleaning or transformations, especially in long-tail data.
- **Experiment Iteratively:**  
Conduct small experiments to test the impact of data changes before applying them broadly.



### 3. REFINE THE BERT EMBEDDING PROGRAM TO ACHIEVE BETTER RESULTS IN KAGGLE COMPETITIONS.

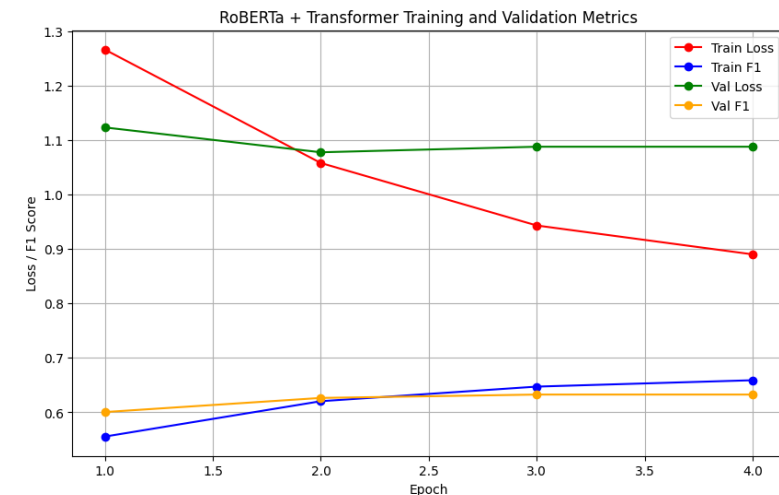
Method	Feature Engineering	Model	Accuracy Estimation	Advantages	Disadvantages	GPU Support
BERT Embedding + Transformer	Contextual semantic embedding, preserving global semantics	Pretrained BERT model	85%-90%	Captures contextual semantics, highest classification accuracy	High training and inference cost, requires large amounts of data and resources	Supported (necessary, otherwise slow)
Tokenizer + LSTM	Digitizes text sequences, retains sequential context	Long Short-Term Memory (LSTM)	80%-88%	Captures text sequence features, suitable for time-series or long texts	Moderate training cost, may encounter gradient vanishing in long texts	Supported (significant acceleration, necessary)
RoBERTa + Transformer	Contextual semantic embedding, optimized for downstream tasks	Pretrained RoBERTa model	88%-92%	Enhanced training optimization, better at handling long sequences	Similar to BERT, high computational cost and resource requirements	Supported (necessary for large-scale tasks)

- Our goal is to achieve higher scores, and based on our previous experiments, it seems that only approaches involving deep learning and GPUs have a chance of scoring high.
- We finally found a Kaggle GPU environment to write programs, but unfortunately, we spent too much time modifying the Tokenizer + LSTM code, which wasted a lot of time.
- Eventually, from some research papers, we discovered that RoBERTa might be more suitable than BERT, so we shifted our approach to adopt RoBERTa.

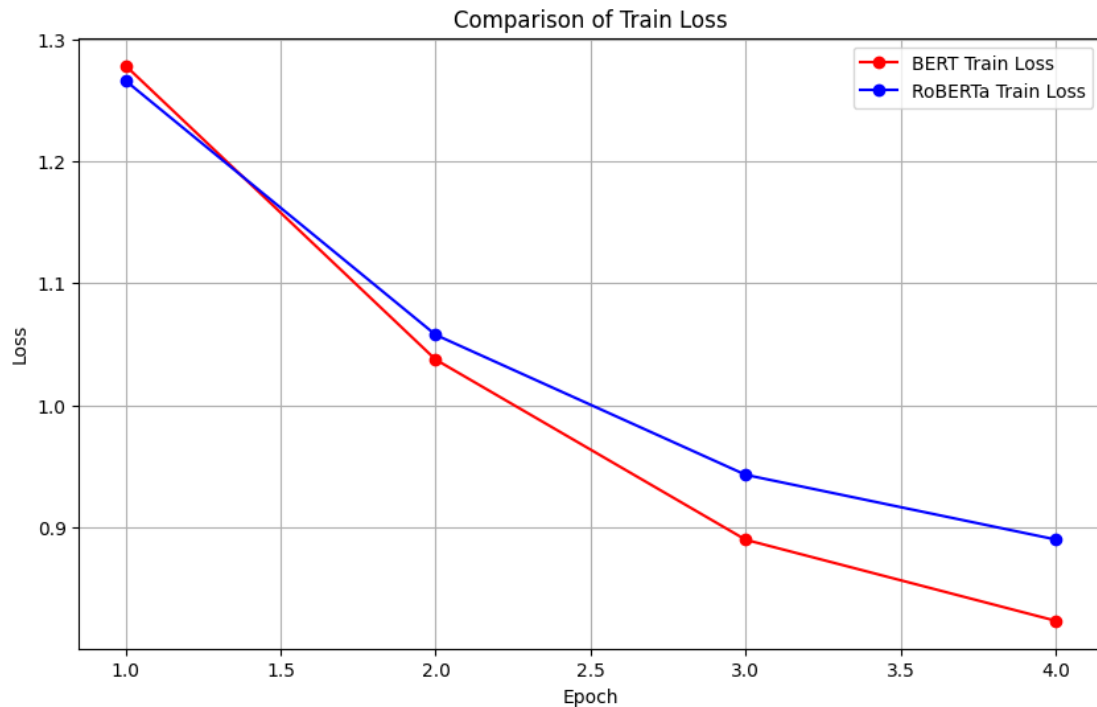
### 3. REFINE THE BERT EMBEDDING PROGRAM TO ACHIEVE BETTER RESULTS IN KAGGLE COMPETITIONS.

Aspect	BERT-based Encoding with Transformer	RoBERTa-based Encoding with Transformer
Model Architecture	BERT uses bidirectional encoding to understand context in both directions.	RoBERTa is an improved version of BERT with dynamic masking and larger datasets.
Pretraining Objective	Masked Language Model (MLM) and Next Sentence Prediction (NSP).	Only Masked Language Model (MLM); removes NSP to improve training.
Training Data	Trained on 16GB of data (BooksCorpus and English Wikipedia).	Trained on 160GB of data, including additional datasets like OpenWebText and CC-News.
Tokenizer	WordPiece tokenizer.	Byte-Pair Encoding (BPE), enabling better handling of rare words.
Performance	Strong contextual embeddings but slightly weaker in generalization compared to RoBERTa.	Improved contextual understanding and generalization across various tasks.
Speed	Faster during inference due to fewer pretraining optimizations.	Slower during inference but more accurate due to enhanced pretraining.
Suitability	Better suited for tasks with a smaller dataset or where NSP is relevant.	Ideal for tasks requiring more robust contextual understanding and generalization.
Example Use Case	Sentence-pair classification or smaller-scale NLP tasks.	Large-scale NLP tasks, sentiment analysis, or question answering.
Pretrained Model Size	Original BERT has smaller model variants like BERT-base (110M parameters).	RoBERTa often uses larger models (e.g., RoBERTa-large, 355M parameters).

- While working on RoBERTa-based Encoding with Transformer, we encountered some issues with JAX and excessively long processing times. We faced several instances of deadlocks and spent considerable time debugging them.
- Fortunately, we took the programming key points for over-fitting into account, allowing the model to early stop at **epoch = 4**. However, each run took more than 10 hours, and eventually, the process was successfully completed in around 8 hours.
- An epoch count of 4 might seem low, but it was a necessary choice since **the GPU time on Kaggle was running out!**

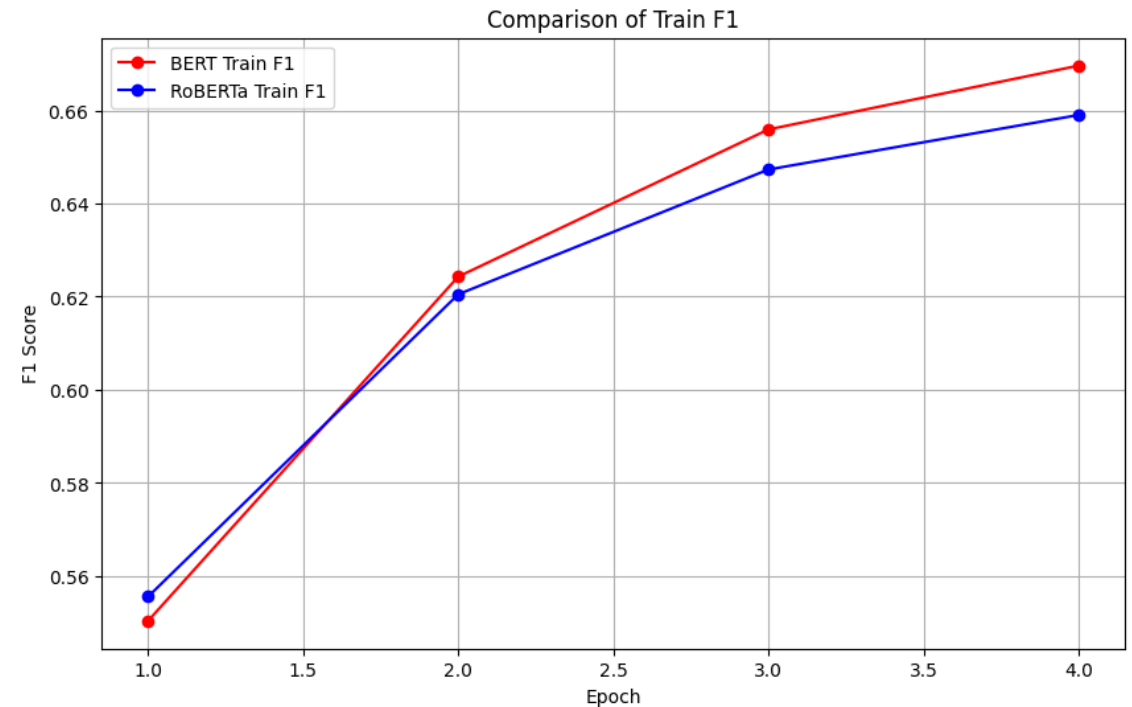


### 3. REFINE THE BERT EMBEDDING PROGRAM TO ACHIEVE BETTER RESULTS IN KAGGLE COMPETITIONS.



#### Training Loss Comparison:

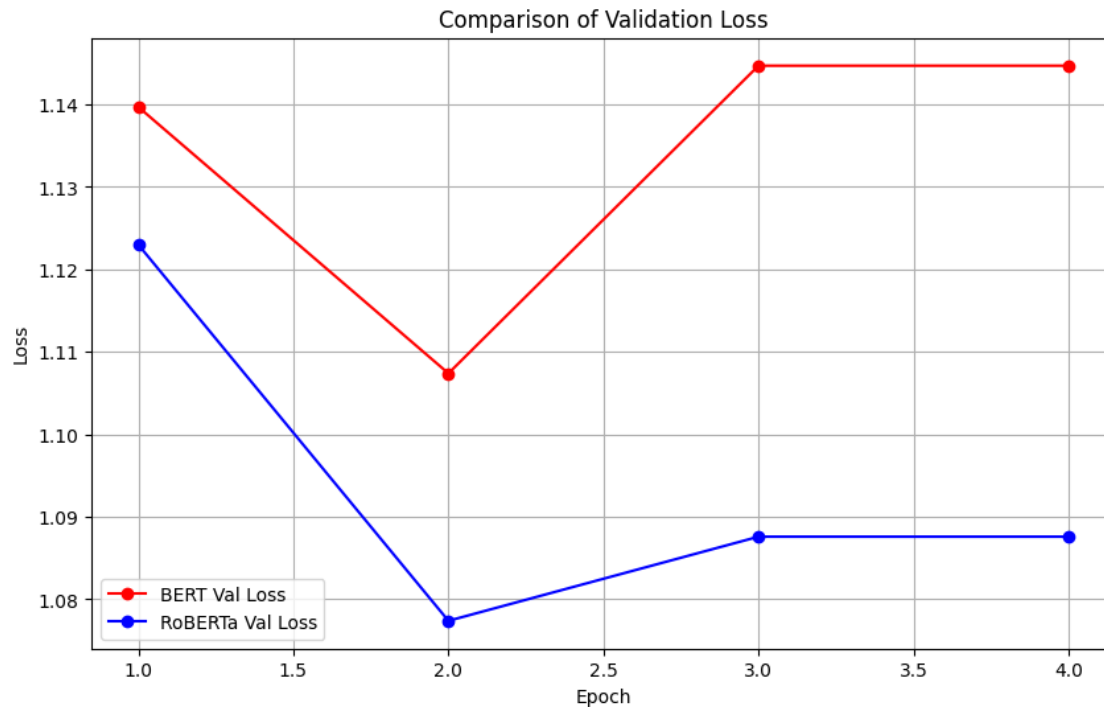
- BERT's training loss decreases slightly faster initially but eventually performs similarly to RoBERTa.
- RoBERTa starts with a slightly lower loss but decreases more gradually.



#### Training F1 Comparison:

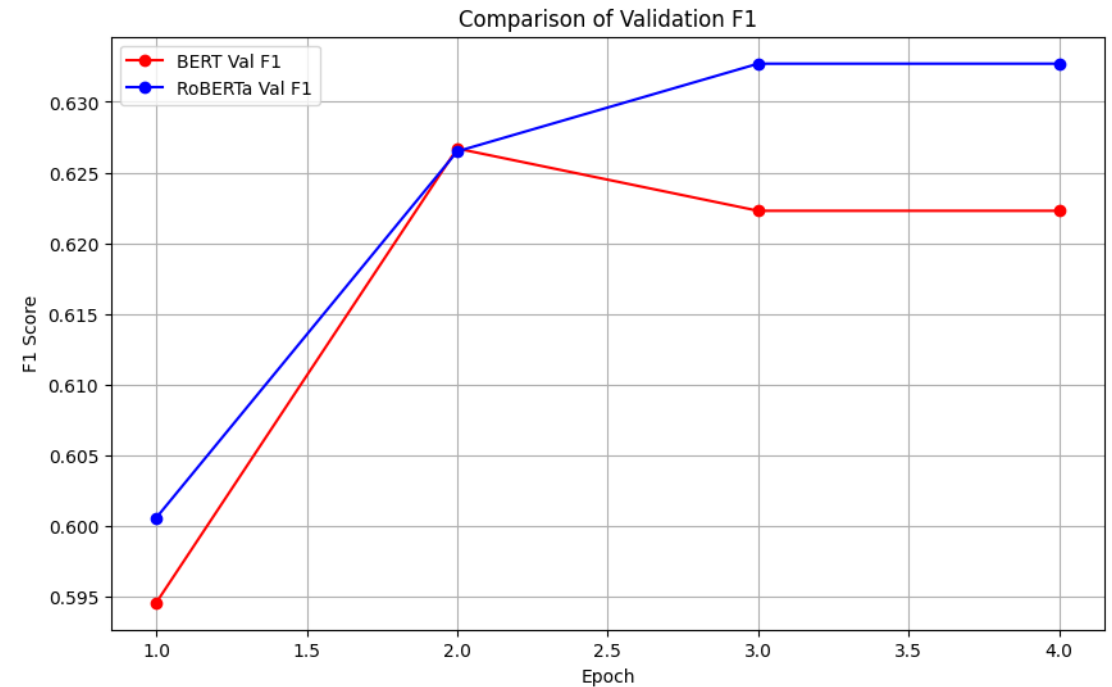
- Both BERT and RoBERTa show gradual improvements in training F1 as the number of epochs increases, with BERT slightly outperforming RoBERTa in the end.

### 3. REFINE THE BERT EMBEDDING PROGRAM TO ACHIEVE BETTER RESULTS IN KAGGLE COMPETITIONS.



#### Validation Loss Comparison:

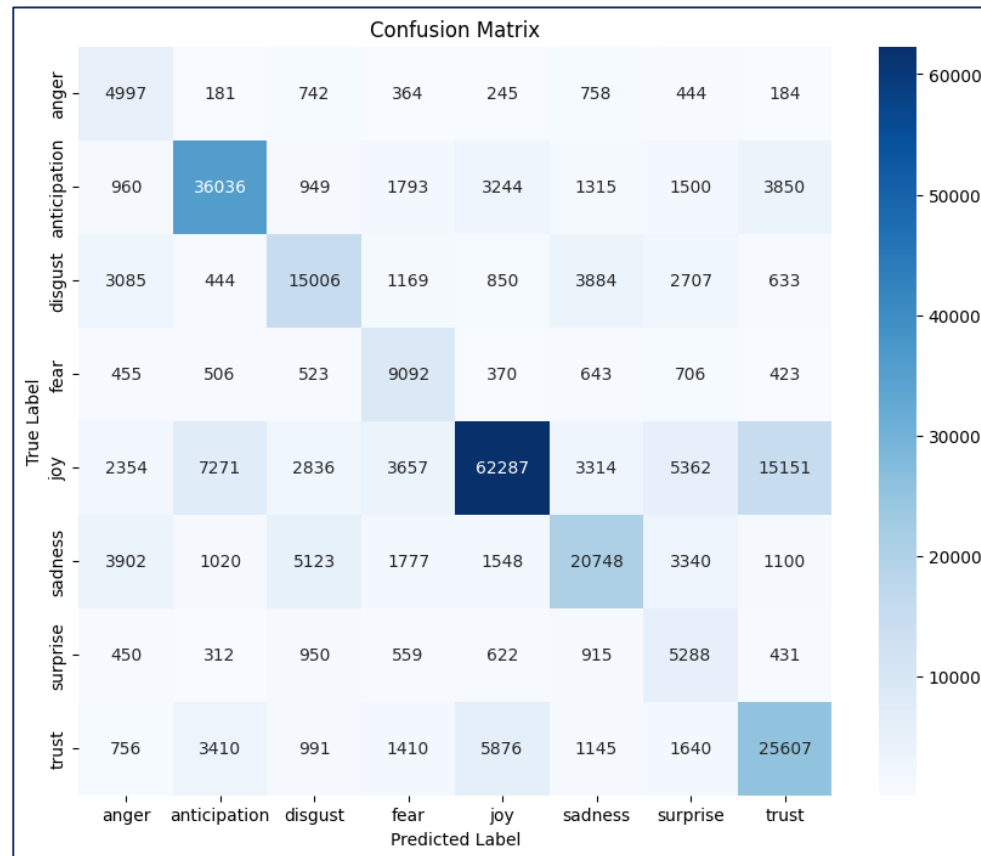
- RoBERTa's validation loss is slightly lower and more stable compared to BERT.
- BERT's validation loss shows some fluctuations in the later stages.



#### Validation F1 Comparison:

- RoBERTa's validation F1 performs slightly better than BERT, especially in the early epochs.
- RoBERTa's performance eventually stabilizes, while BERT shows a slight decline.

### 3. REFINE THE BERT EMBEDDING PROGRAM TO ACHIEVE BETTER RESULTS IN KAGGLE COMPETITIONS.



Confusion Matrix:

```
[[ 4997  181  742  364  245  758  444  184]
 [  960 36036  949 1793 3244 1315 1500 3850]
 [ 3085  444 15006 1169  850 3884 2707  633]
 [  455  506  523 9092  370  643  706  423]
 [ 2354 7271 2836 3657 62287 3314 5362 15151]
 [ 3902 1020 5123 1777 1548 20748 3340 1100]
 [  450  312  950  559  622  915 5288  431]
 [  756 3410  991 1410 5876 1145 1640 25607]]
```

Classification Report:

	precision	recall	f1-score	support
anger	0.29	0.63	0.40	7915
anticipation	0.73	0.73	0.73	49647
disgust	0.55	0.54	0.55	27778
fear	0.46	0.71	0.56	12718
joy	0.83	0.61	0.70	102232
sadness	0.63	0.54	0.58	38558
surprise	0.25	0.56	0.35	9527
trust	0.54	0.63	0.58	40835
accuracy			0.62	289210
macro avg	0.54	0.62	0.56	289210
weighted avg	0.67	0.62	0.63	289210


Validation Accuracy: 0.6191



### 3. REFINE THE BERT EMBEDDING PROGRAM TO ACHIEVE BETTER RESULTS IN KAGGLE COMPETITIONS.


30

EnzoCheng222



0.51428

6



Your Best Entry!

Your most recent submission scored 0.51428, which is an improvement of your previous score of 0.50585. Great job!

Tweet this


RoBERTa-based Encoding with Transformer

- **Data cleaning** : only remove <LH>
- **Epoch** = 4
- **F1 average** : **0.51428**

Submissions

Select up to 2 submissions that will count towards your final leaderboard score. If less than 2 are selected, Kaggle will automatically select from your best scoring submissions. [Learn More](#)

0/2

☒ Auto-selection candidates 



All

Successful

Selected

Errors

Recent

Submission and Description	Public Score 	Select
<div> <b>BERT Transformer epoch 4.csv</b> Complete · 1m ago · BERT + Transformer + epoch 4</div>	0.50585	<input type="checkbox"/>

BERT-based Encoding with Transformer

- **Data cleaning** : only remove <LH>
- **Epoch** = 4
- **F1 average** : **0.50585**



### **Challenges and Overcoming Obstacles**

- Faced team changes as all original members dropped the course, requiring support from a past classmate.
- Continuously encountered technical challenges and overcame them step by step..

### **Growth in Skills and Knowledge**

- Mastered the complete procedural workflow: data cleaning, feature encoding, model training, and evaluation.
- Progressed from BERT embedding to optimizing the RoBERTa model.

### **Lessons from Mistakes**

- Recognized the negative impact of over-processing and overfitting through experimental failures.
- Improved stability and performance by adopting iterative experimentation strategies.

### **Future Directions and Gratitude**

- Plans to delve deeper into deep learning techniques, especially embedding and GPU training.
- Expressed heartfelt gratitude to the professor and teaching assistants for the learning opportunities provided by the course.

## **REFLECTIONS AFTER COMPLETING THE ASSIGNMENT**



THANK YOU.

ENZOC.MG11@NYCU.EDU.TW