Fake News Classification

組長：施宇軒

組員：莊翔麟、楊沛澐

目錄

① 簡介與目標

② WSDM資料集

③ Data preprocessing

④ Keras Version

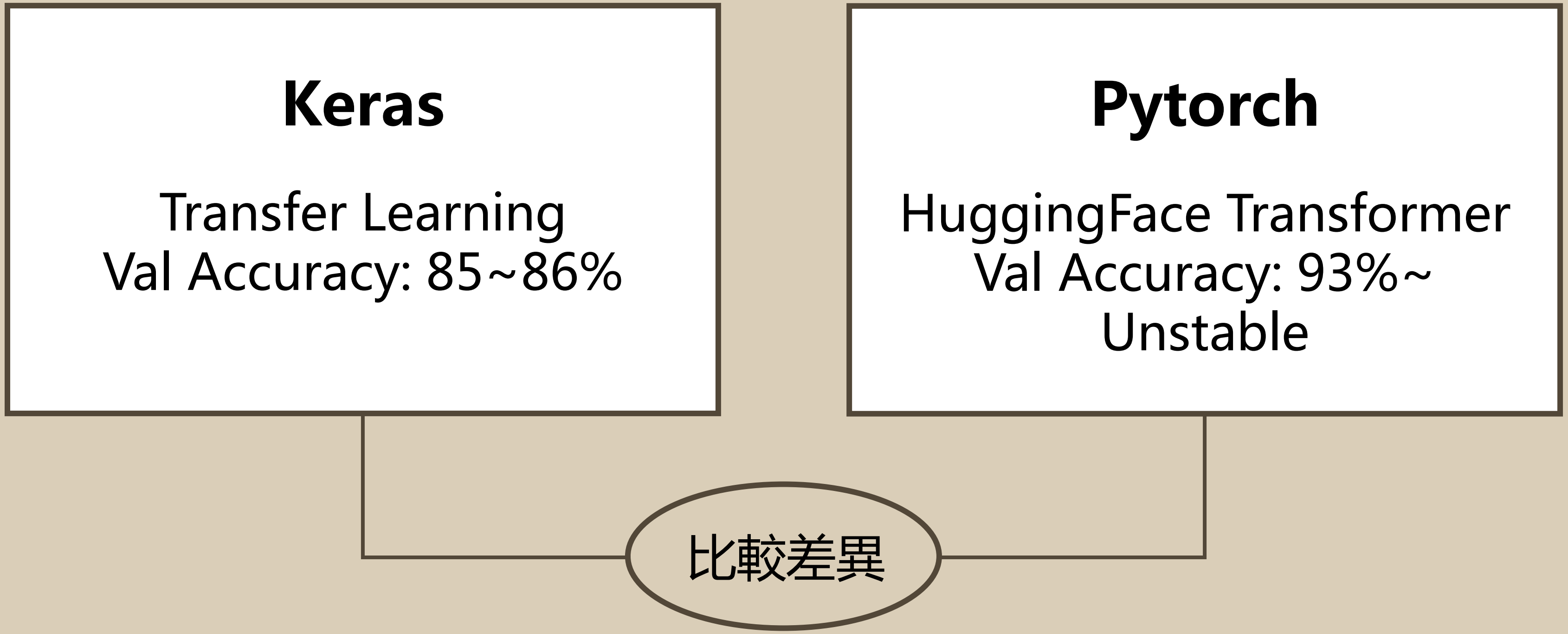
⑤ Pytorch Version

⑥ Keras v.s. Pytorch Version 比較

⑦ 參考資料

03 簡介

藉由Kaggle-WSDM Fake News Detection，深入了解BERT 於Multi-classification應用



Keras

Transfer Learning
Val Accuracy: 85~86%

Pytorch

HuggingFace Transformer
Val Accuracy: 93%~
Unstable

比較差異

04 目標

假新聞定義

狹義定義

中文資料庫

Cofacts謠言查核網站

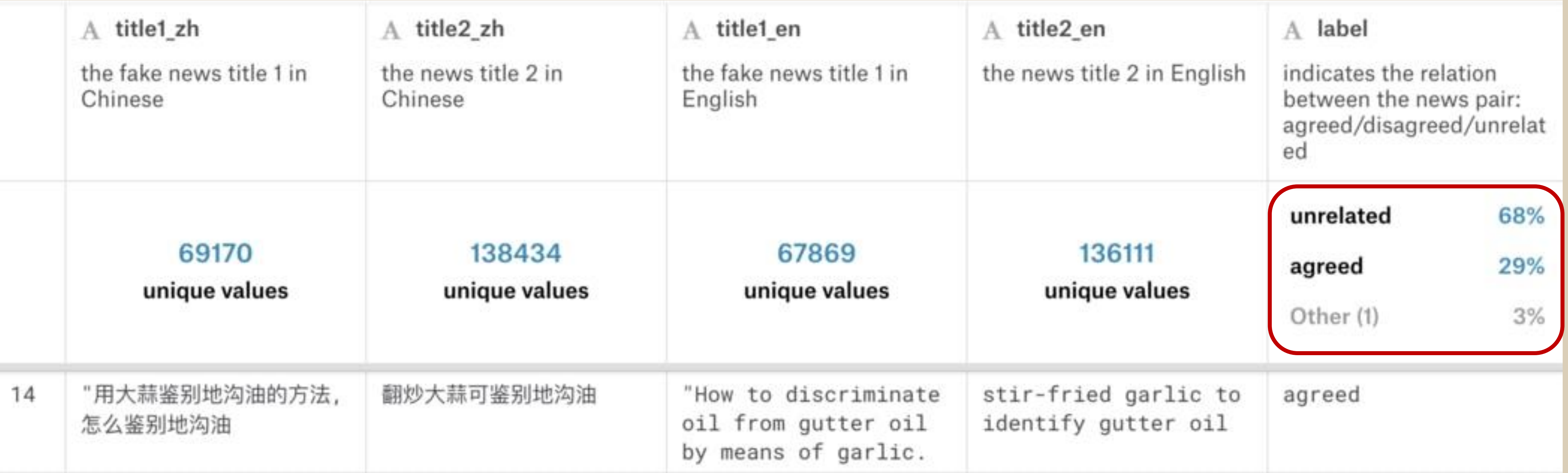
NLI models

BERT與其變形

限縮至NLI Model 進行Multi-class classification的實作
採用 Kaggle-WSDM 2019 Fake News Detection Datasets

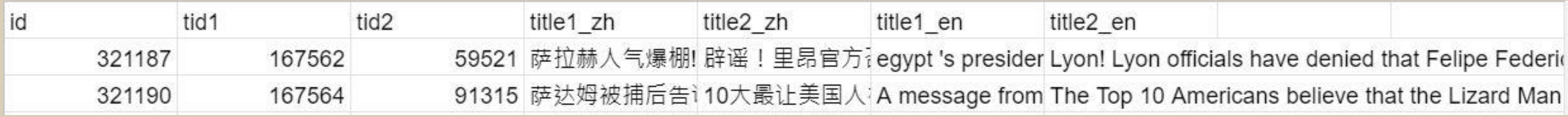
05 WSDM資料集

Train data (約32萬筆)



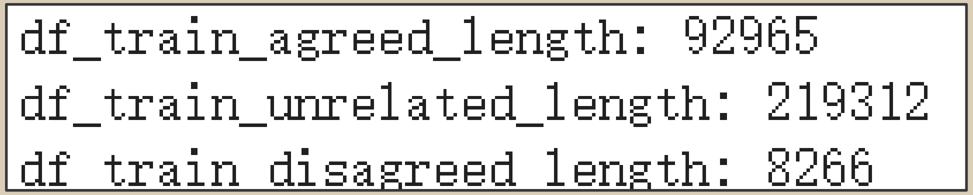
	A title1_zh	A title2_zh	A title1_en	A title2_en	A label
	the fake news title 1 in Chinese	the news title 2 in Chinese	the fake news title 1 in English	the news title 2 in English	indicates the relation between the news pair: agreed/disagreed/unrelated
	69170 unique values	138434 unique values	67869 unique values	136111 unique values	unrelated 68% agreed 29% Other (1) 3%
14	"用大蒜鉴别地沟油的方法，怎么鉴别地沟油	翻炒大蒜可鉴别地沟油	"How to discriminate oil from gutter oil by means of garlic.	stir-fried garlic to identify gutter oil	agreed

Test data (約8萬筆)

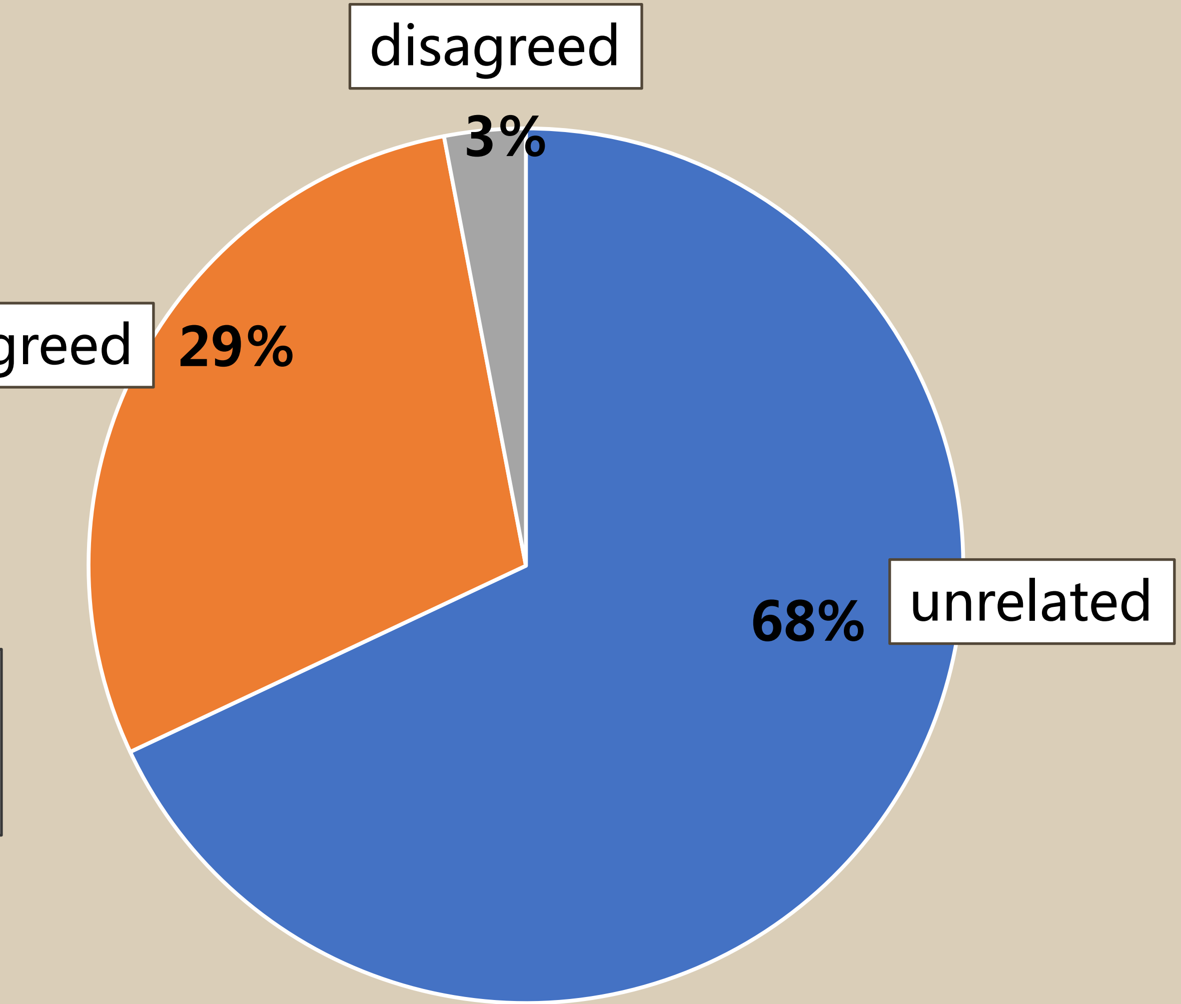


id	tid1	tid2	title1_zh	title2_zh	title1_en	title2_en		
321187	167562	59521	萨拉赫人气爆棚!	辟谣！里昂官方	egypt 's presider	Lyon! Lyon officials have denied that Felipe Federi		
321190	167564	91315	萨达姆被捕后告	10大最让美国人	A message from	The Top 10 Americans believe that the Lizard Man		

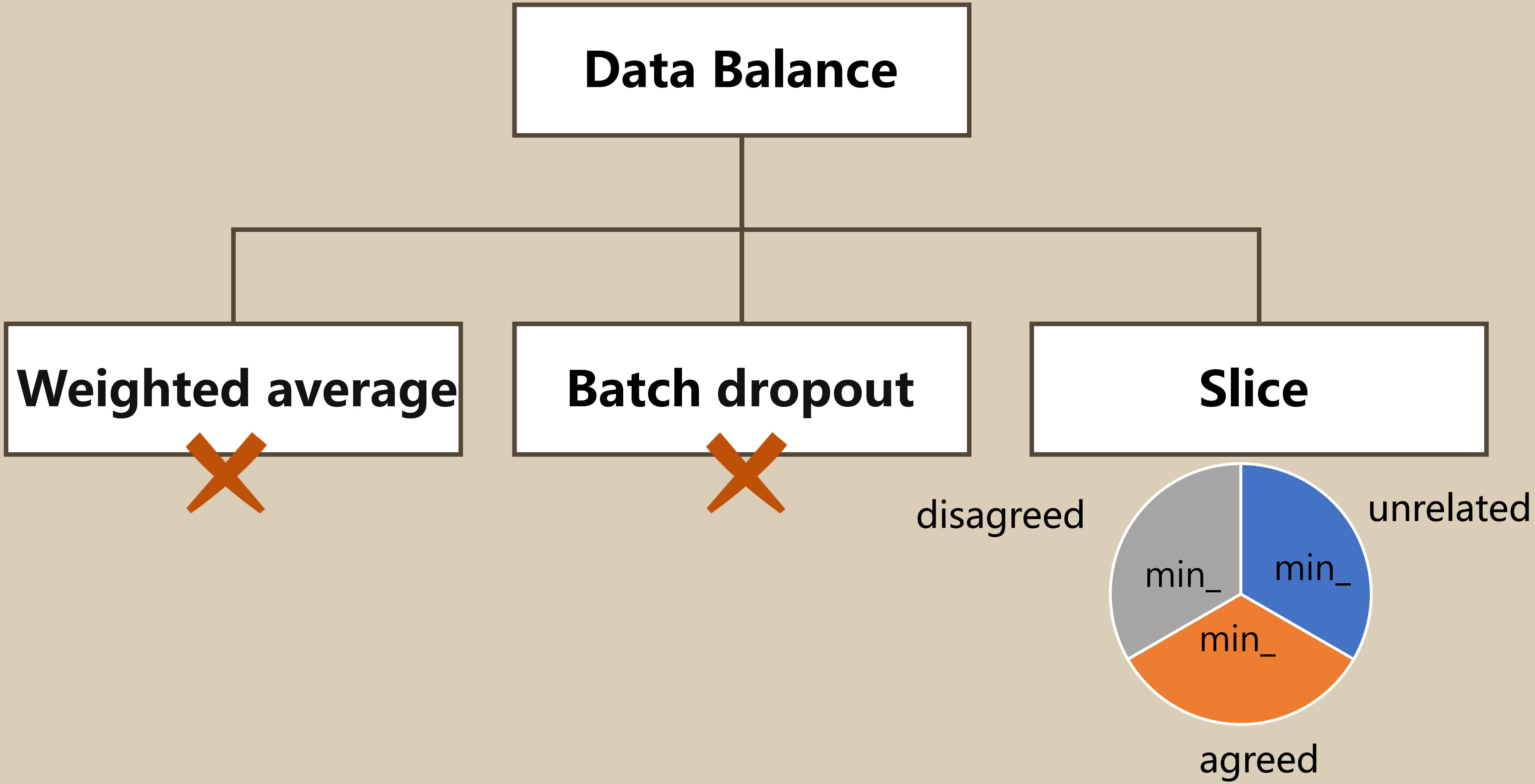
06 Unbalanced Training Data



```
df_train_agreed_length: 92965
df_train_unrelated_length: 219312
df_train_disagreed_length: 8266
```



07 Unbalanced Training Data



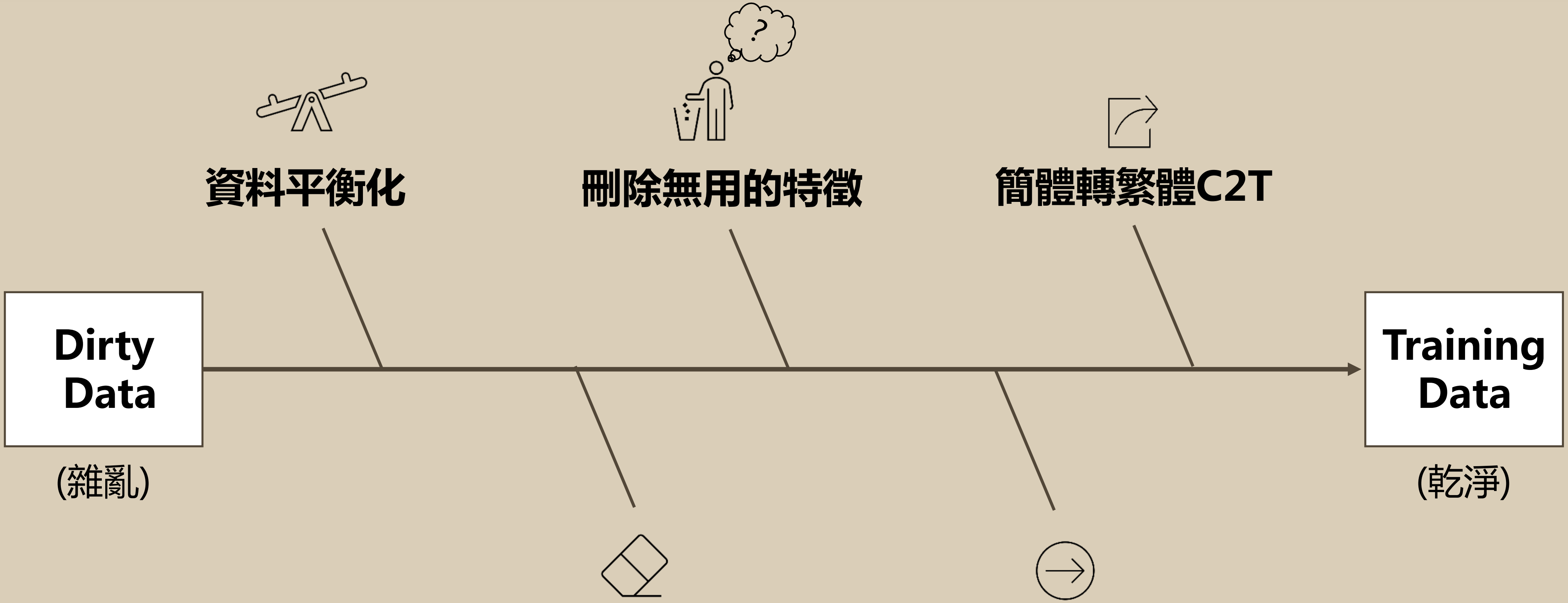
Data Balance

Weighted average ✗

Batch dropout ✗

Slice

08 Data preprocessing



資料平衡化

刪除無用的特徵

簡體轉繁體C2T

Dirty Data

(雜亂)

Training Data

(乾淨)

去除空格、空值、0
去除過長樣本

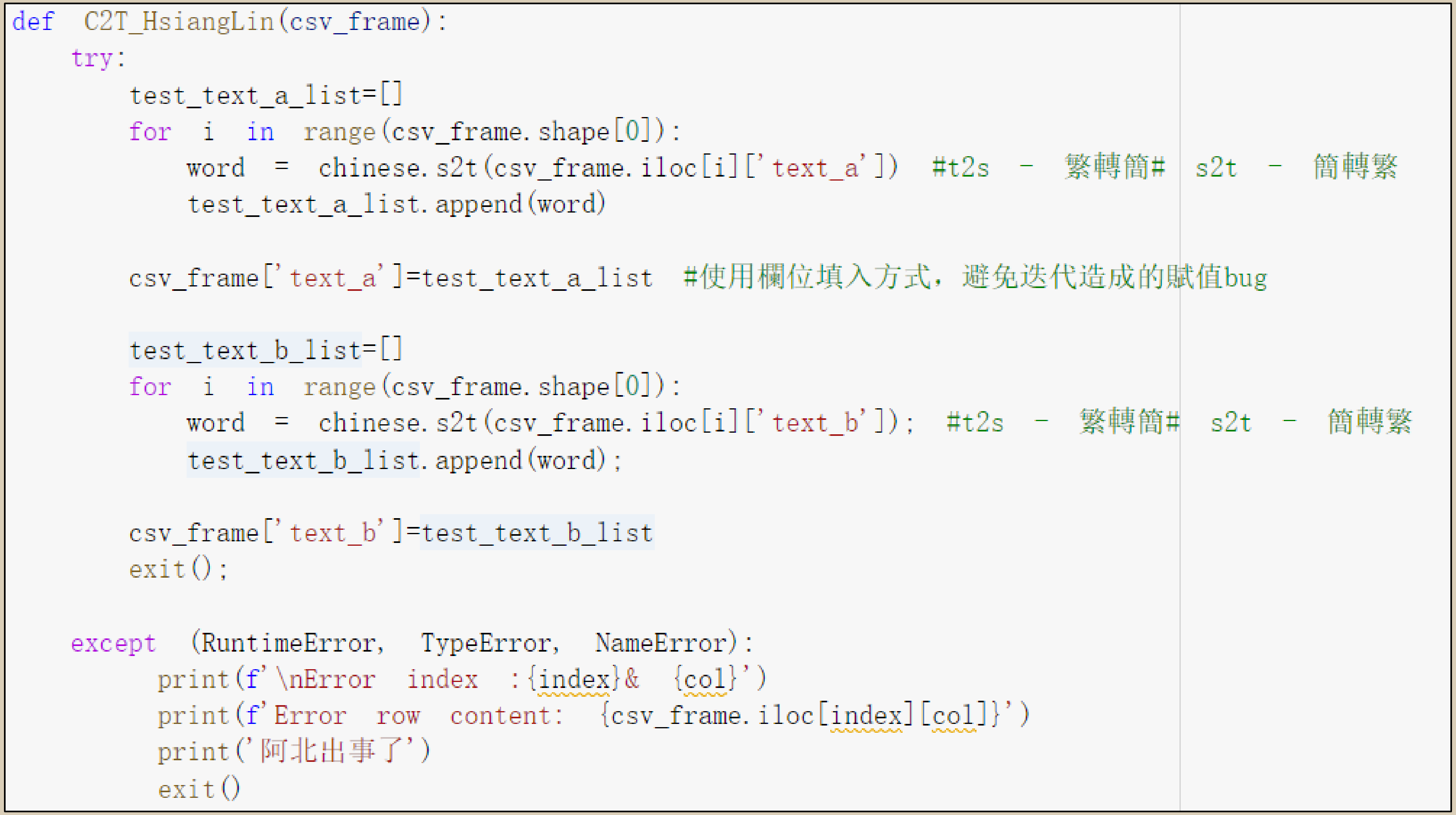
MAX_LENGTH = 250
(limits: 512)
可透過限制單句128來加速

Label向量化

disagreed→0
agreed→1
unrelated→2

09 Data preprocessing - C2T

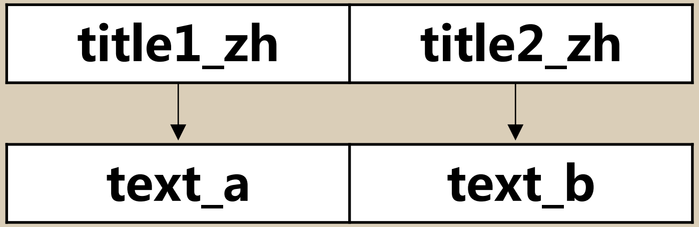
C2T 安裝iNLP的chinese簡轉繁套件



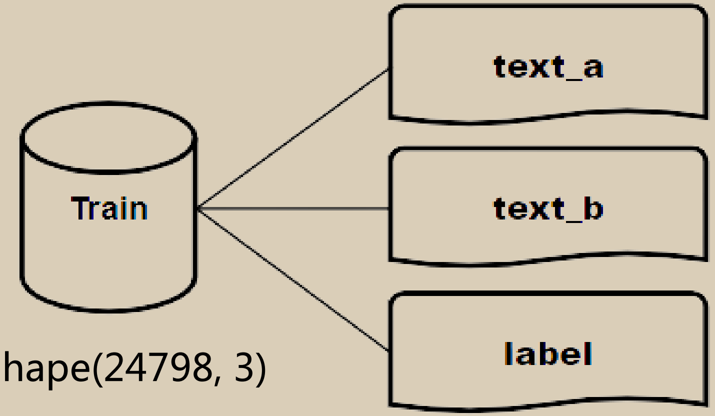
```python
def C2T_HsiangLin(csv_frame):
    try:
        test_text_a_list=[]
        for i in range(csv_frame.shape[0]):
            word = chinese.s2t(csv_frame.iloc[i]['text_a']) #t2s - 繁轉簡# s2t - 簡轉繁
            test_text_a_list.append(word)

        csv_frame['text_a']=test_text_a_list #使用欄位填入方式，避免迭代造成的賦值bug

        test_text_b_list=[]
        for i in range(csv_frame.shape[0]):
            word = chinese.s2t(csv_frame.iloc[i]['text_b']); #t2s - 繁轉簡# s2t - 簡轉繁
            test_text_b_list.append(word);

        csv_frame['text_b']=test_text_b_list
        exit();

    except (RuntimeError, TypeError, NameError):
          print(f'\nError index :{index}& {col}')
          print(f'Error row content: {csv_frame.iloc[index][col]}')
          print('阿北出事了')
          exit()
```

WSDM資料集 - 處理後

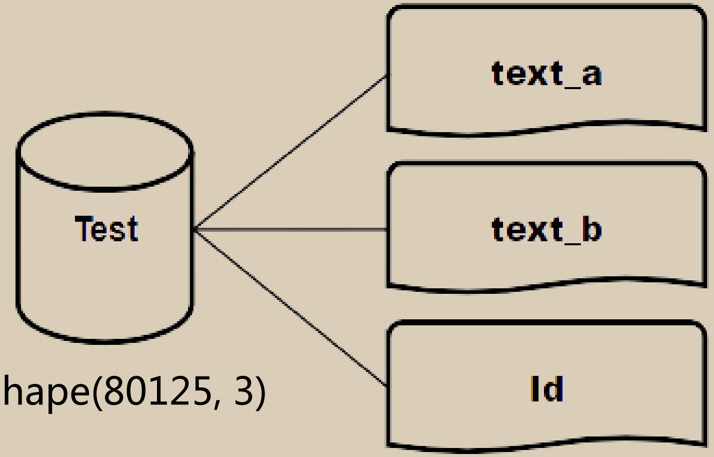
重新命名欄位名稱



title1_zh	title2_zh
↓	↓
text_a	text_b

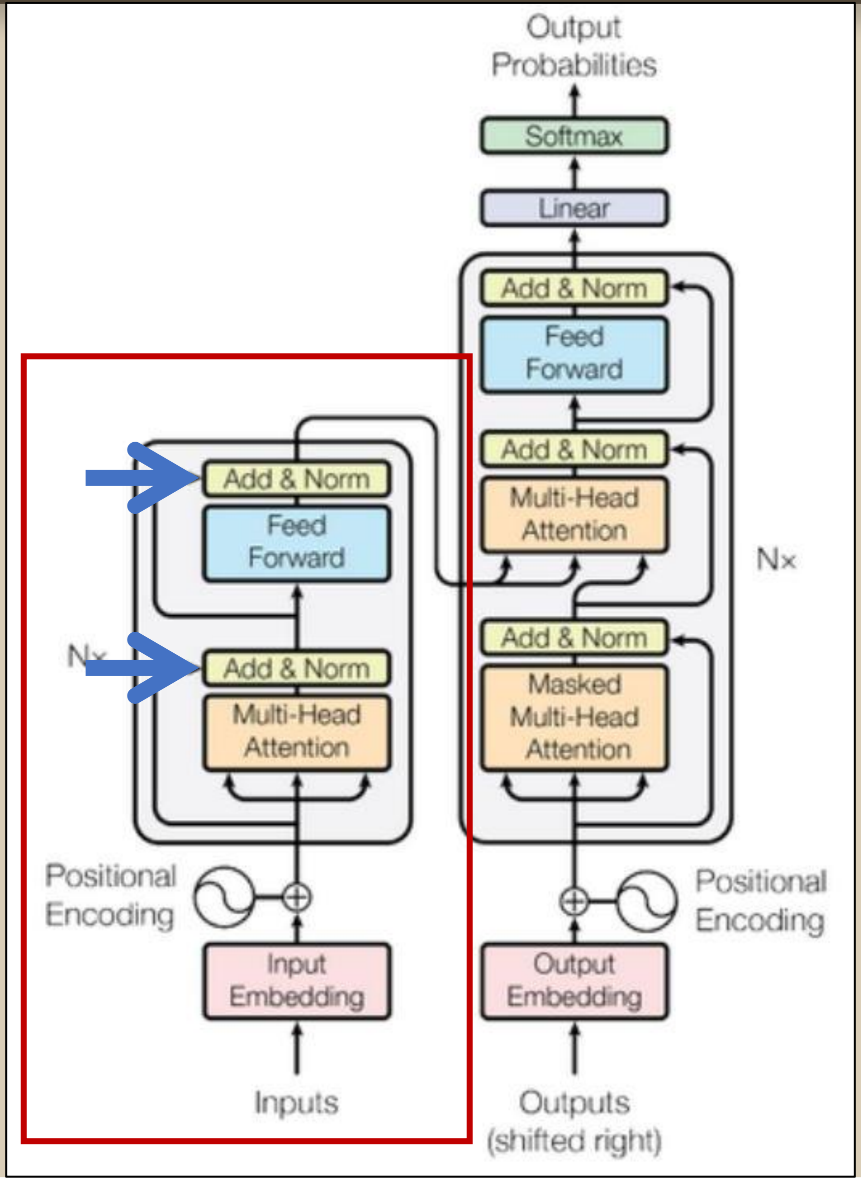


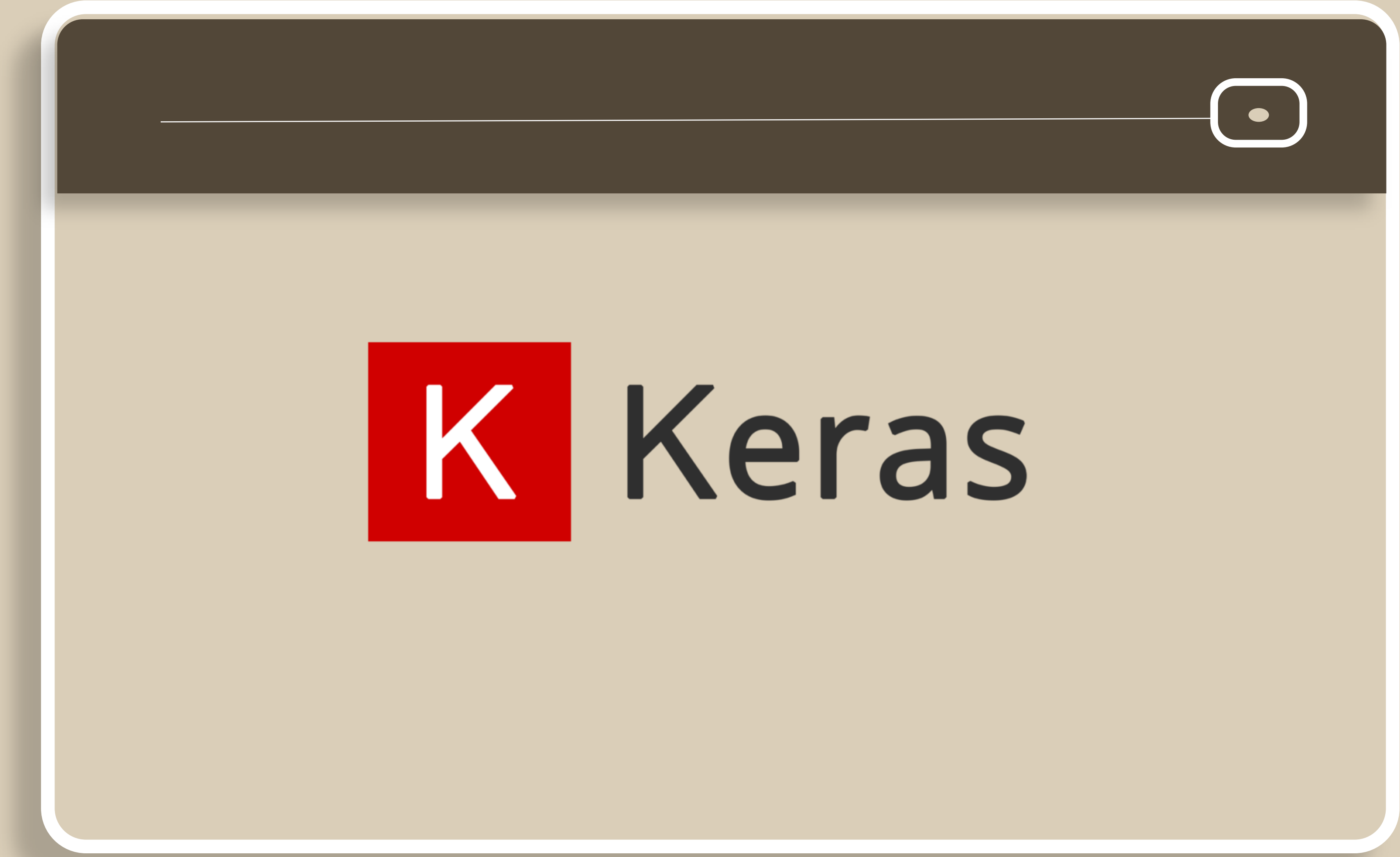
Shape(24798, 3)



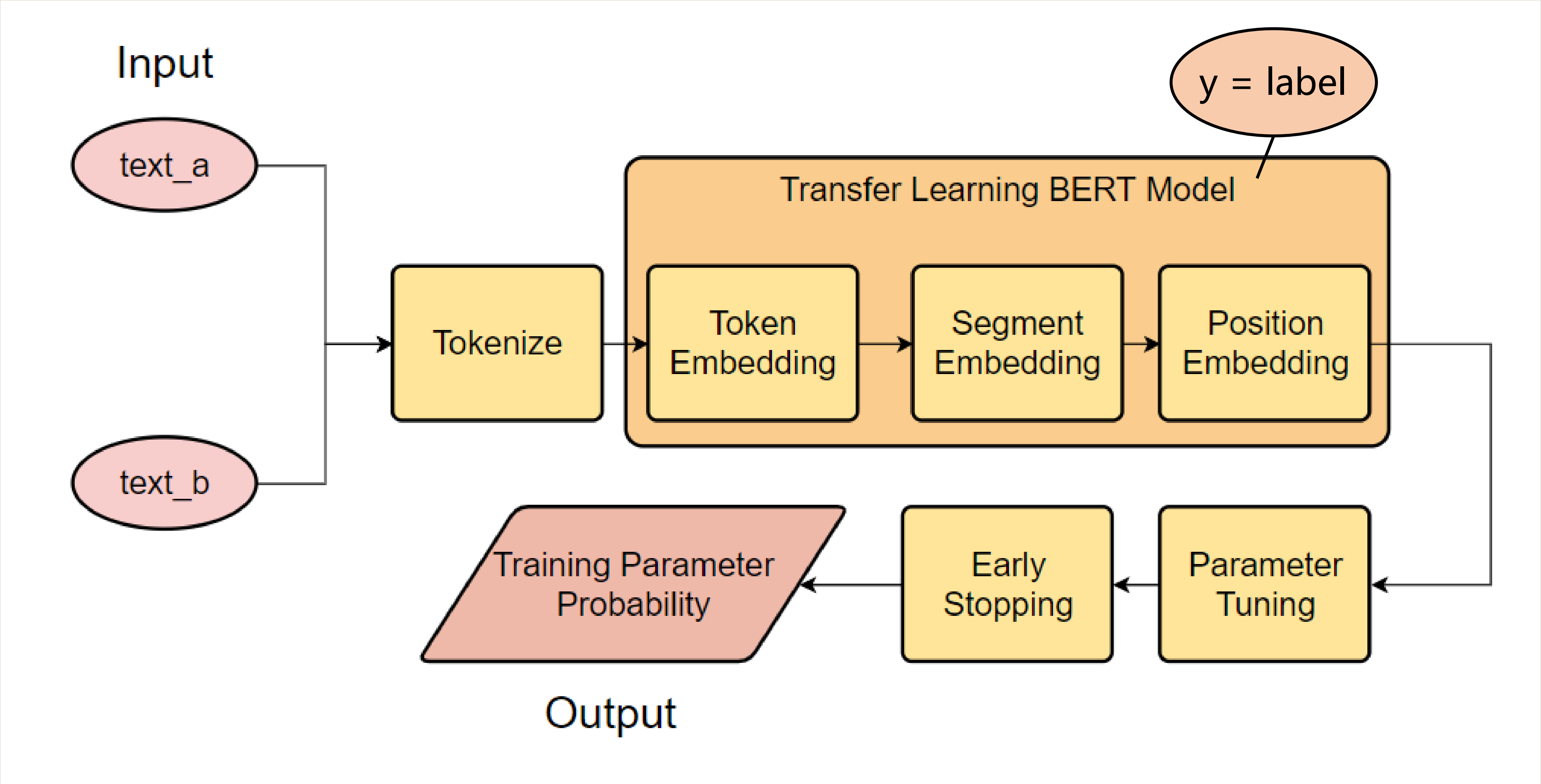
Shape(80125, 3)

11 Transformer Encoder 系統架構

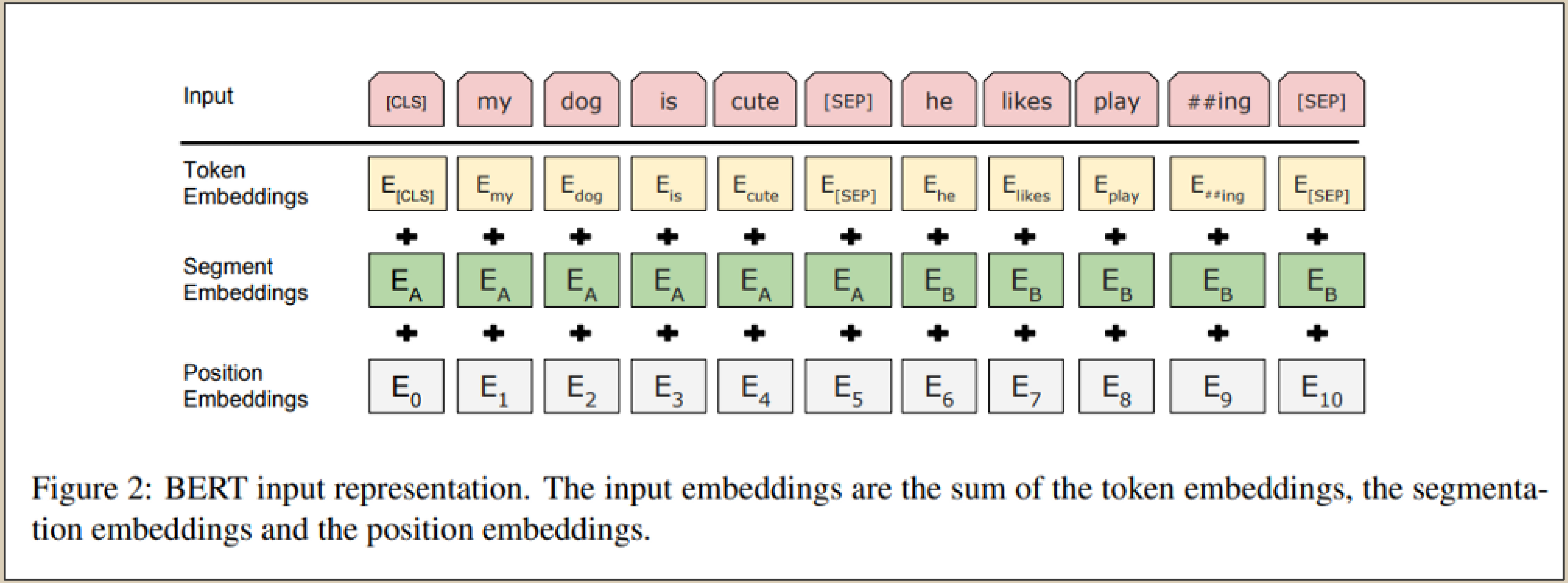



K
Keras

Keras 版本 + 遷移學習



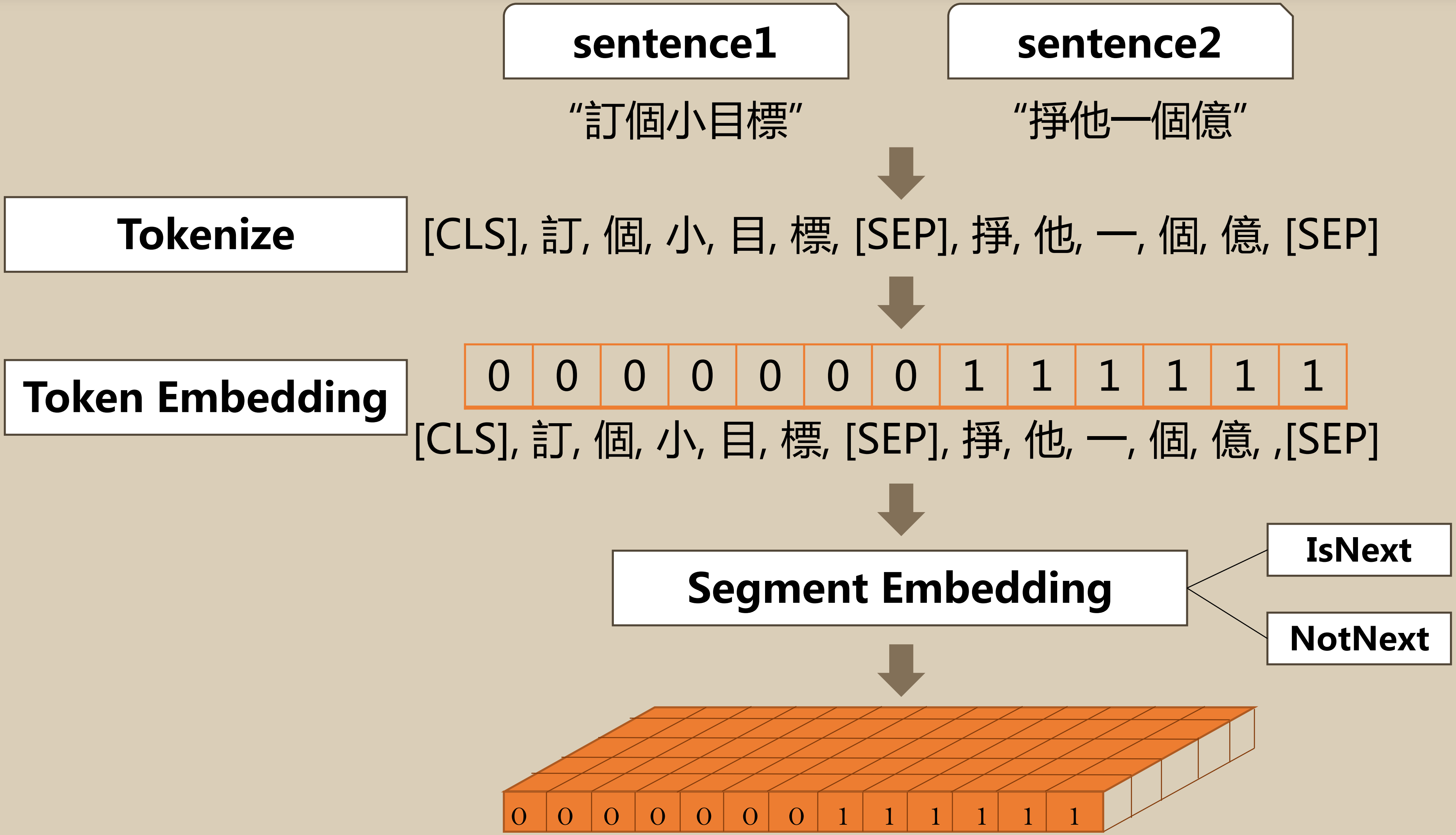
Tokenizer



Input	[CLS]	my	dog	is	cute	[SEP]	he	likes	play	##ing	[SEP]
Token Embeddings	$E_{[CLS]}$	E_{my}	E_{dog}	E_{is}	E_{cute}	$E_{[SEP]}$	E_{he}	E_{likes}	E_{play}	$E_{\#\#ing}$	$E_{[SEP]}$
	+	+	+	+	+	+	+	+	+	+	+
Segment Embeddings	E_A	E_A	E_A	E_A	E_A	E_A	E_B	E_B	E_B	E_B	E_B
	+	+	+	+	+	+	+	+	+	+	+
Position Embeddings	E_0	E_1	E_2	E_3	E_4	E_5	E_6	E_7	E_8	E_9	E_{10}

Figure 2: BERT input representation. The input embeddings are the sum of the token embeddings, the segmentation embeddings and the position embeddings.

Tokenizer



sentence1

“訂個小目標”

sentence2

“掙他一個億”

Tokenize

[CLS], 訂, 個, 小, 目, 標, [SEP], 掙, 他, 一, 個, 億, [SEP]

Token Embedding

0	0	0	0	0	0	0	1	1	1	1	1	1

[CLS], 訂, 個, 小, 目, 標, [SEP], 掙, 他, 一, 個, 億, ,[SEP]

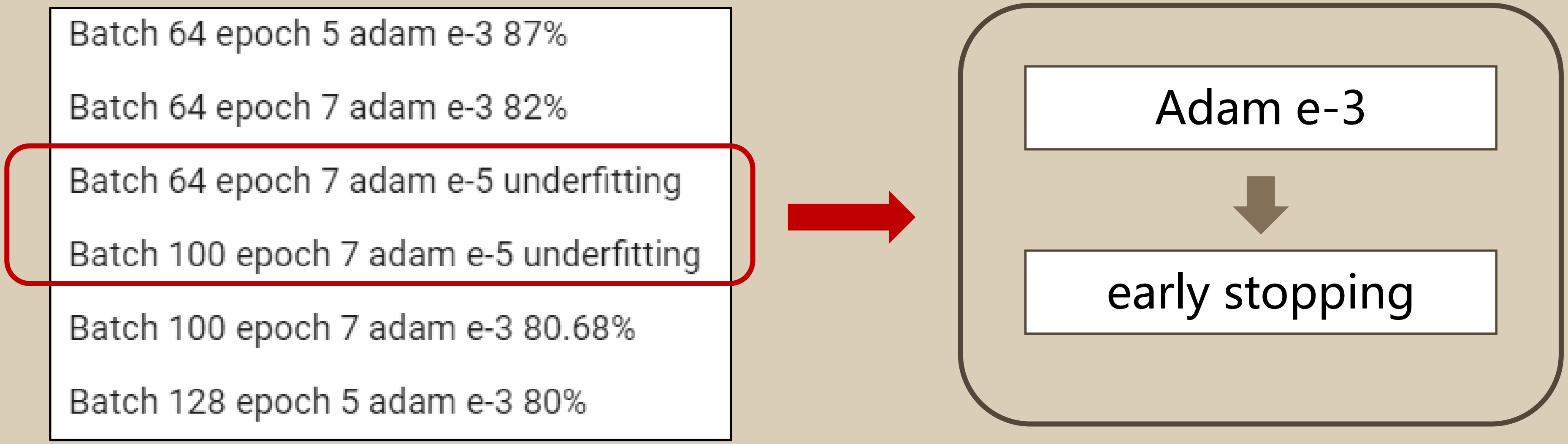
Segment Embedding

IsNext

NotNext

0	0	0	0	0	0	0	1	1	1	1	1	1

16 Parameter tuning



Batch 64 epoch 5 adam e-3 87%

Batch 64 epoch 7 adam e-3 82%

Batch 64 epoch 7 adam e-5 underfitting

Batch 100 epoch 7 adam e-5 underfitting

Batch 100 epoch 7 adam e-3 80.68%

Batch 128 epoch 5 adam e-3 80%

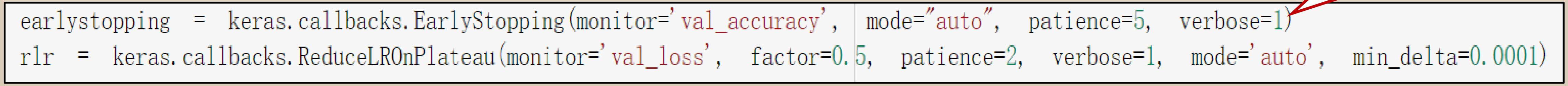
Adam e-3

early stopping

17 Early stopping

因為在少了Batch Normalization (以minibatch 64)的狀況下，若預先設置較小學習率（1e-5）會有underfitting問題

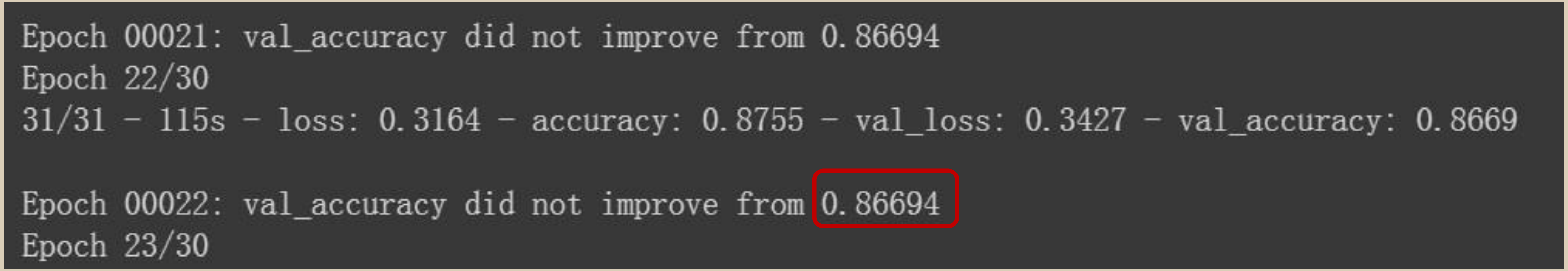
warmup



```
earlystopping = keras.callbacks.EarlyStopping(monitor='val_accuracy', mode="auto", patience=5, verbose=1)
rlr = keras.callbacks.ReduceLROnPlateau(monitor='val_loss', factor=0.5, patience=2, verbose=1, mode='auto', min_delta=0.0001)
```

學習率：每兩個epochs若val_loss沒有顯著的進步，則調降一半

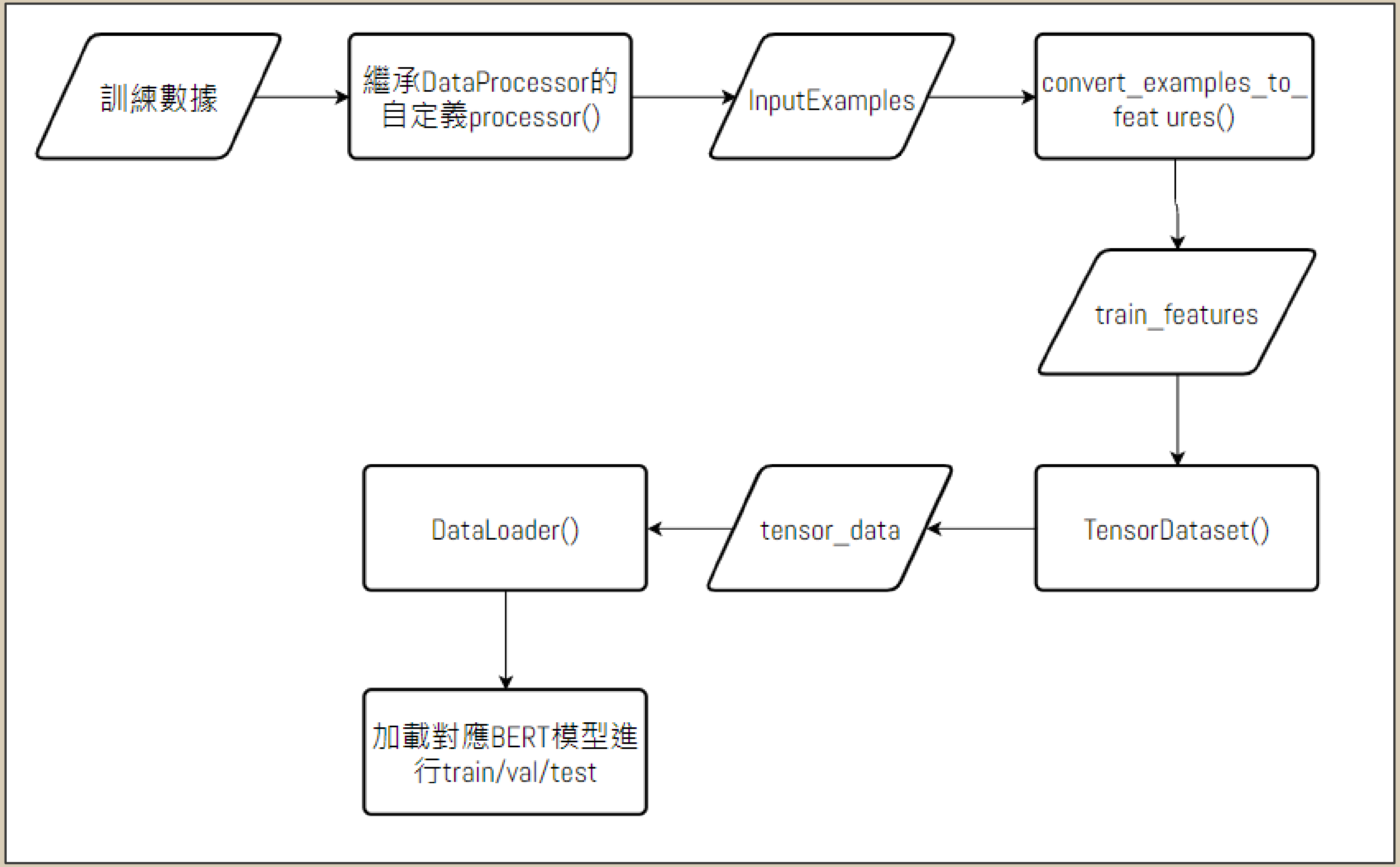
Keras 驗證集準確度



```
Epoch 00021: val_accuracy did not improve from 0.86694
Epoch 22/30
31/31 - 115s - loss: 0.3164 - accuracy: 0.8755 - val_loss: 0.3427 - val_accuracy: 0.8669

Epoch 00022: val_accuracy did not improve from 0.86694
Epoch 23/30
```

PYTORCH

Pytorch 版本



訓練數據

繼承DataProcessor的自定義processor()

InputExamples

convert_examples_to_feat ures()

train_features

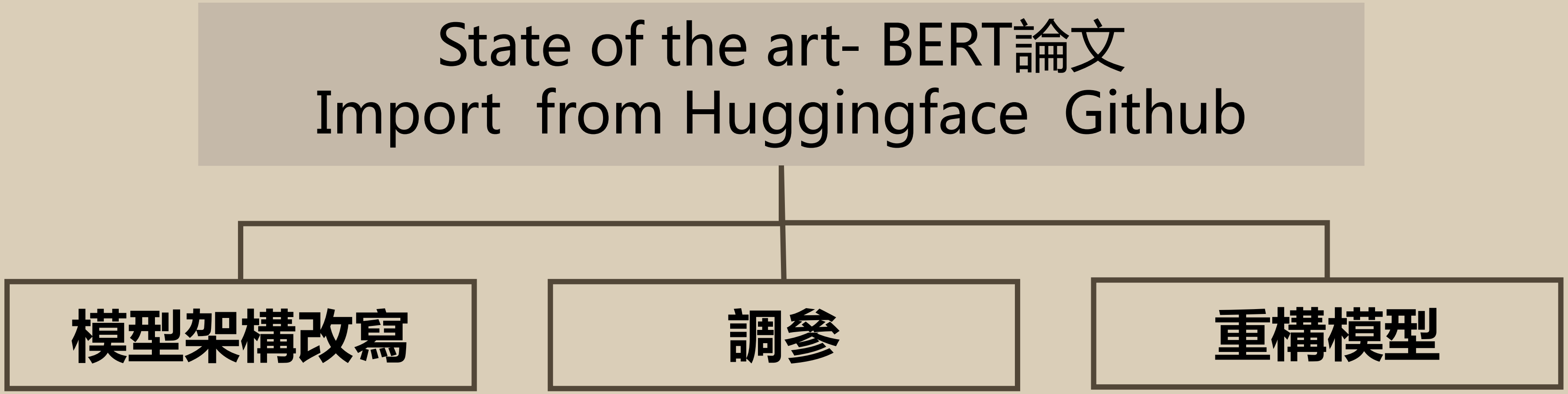
TensorDataset()

tensor_data

DataLoader()

加載對應BERT模型進行train/val/test

Pytorch 版本

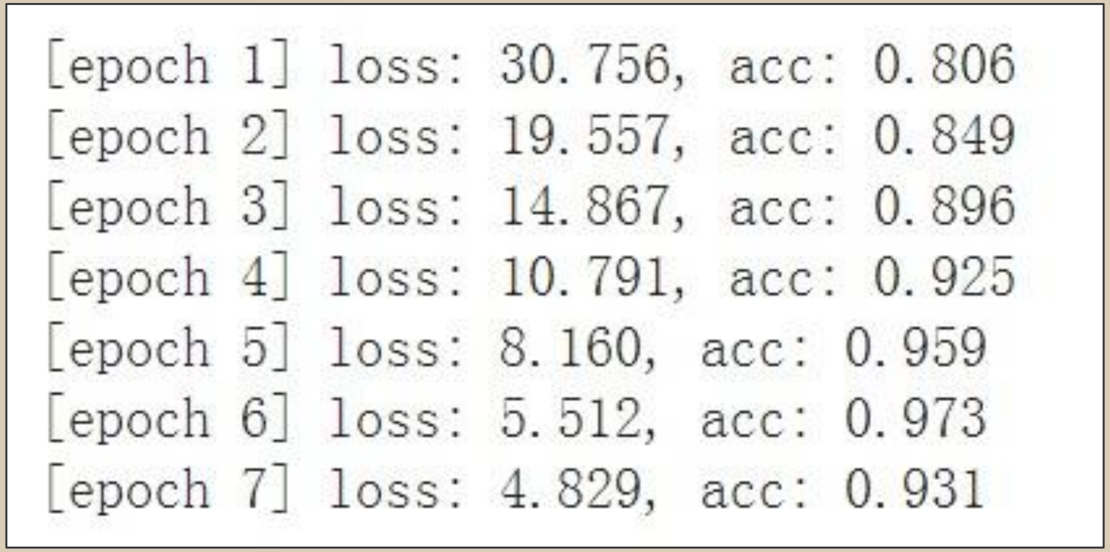


State of the art- BERT論文
Import from Huggingface Github

模型架構改寫

調參

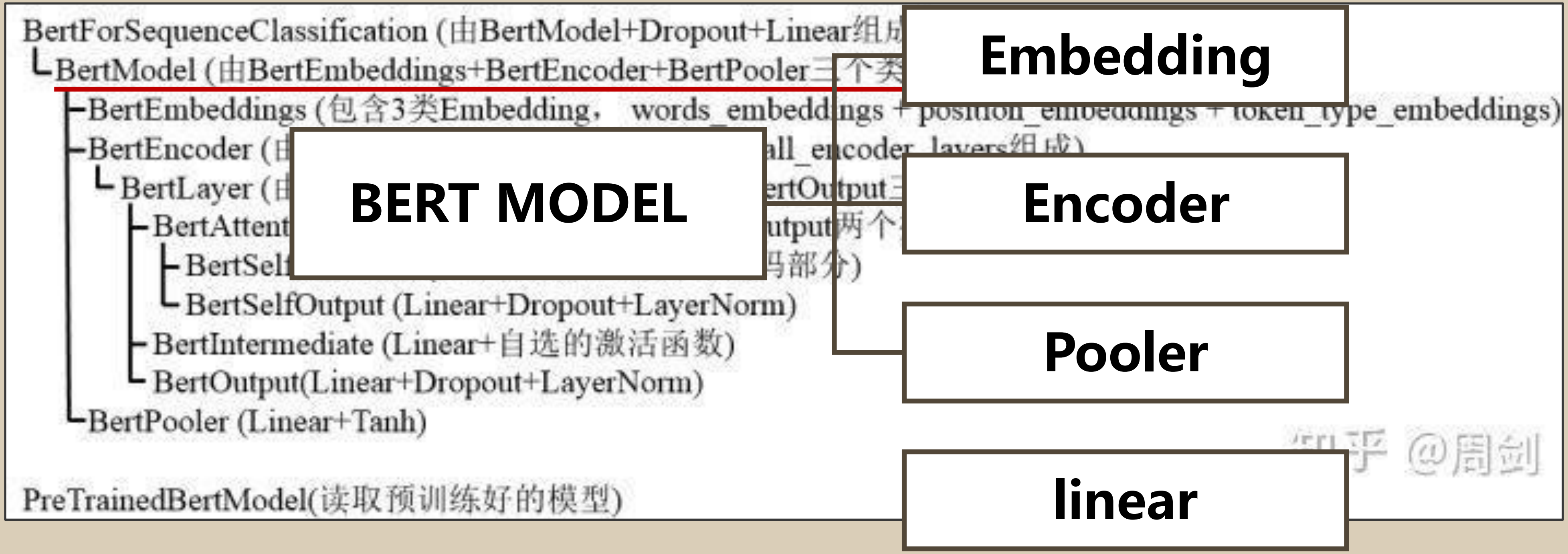
重構模型



```
[epoch 1] loss: 30.756, acc: 0.806
[epoch 2] loss: 19.557, acc: 0.849
[epoch 3] loss: 14.867, acc: 0.896
[epoch 4] loss: 10.791, acc: 0.925
[epoch 5] loss: 8.160, acc: 0.959
[epoch 6] loss: 5.512, acc: 0.973
[epoch 7] loss: 4.829, acc: 0.931
```

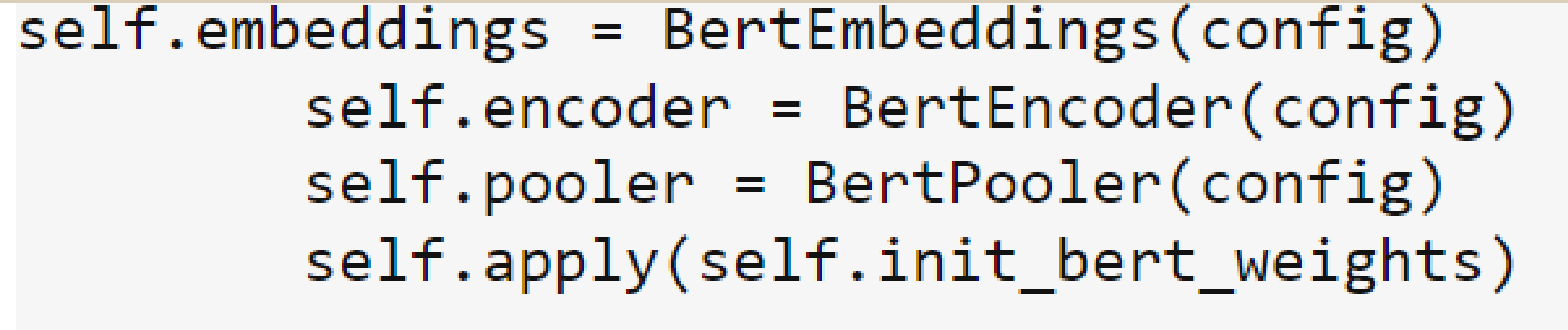
Huggingface BERT 模型架構

環境：Python 3.5+, Pytorch 0.4.1/1.0.0



```
BertForSequenceClassification (由BertModel+Dropout+Linear组成
 └BertModel (由BertEmbeddings+BertEncoder+BertPooler三个类
   ├BertEmbeddings (包含3类Embedding， words_embeddings + position_embeddings + token_type_embeddings)
   ├BertEncoder (由                all_encoder_layers组成)
   │ └BertLayer (由                ertOutput三
   │   ├BertAttent                 utput两个
   │   │ ├BertSelf                 码部分)
   │   │ └BertSelfOutput (Linear+Dropout+LayerNorm)
   │   ├BertIntermediate (Linear+自选的激活函数)
   │   └BertOutput(Linear+Dropout+LayerNorm)
   └BertPooler (Linear+Tanh)

PreTrainedBertModel(读取预训练好的模型)
```

Pytorch – BertModel



```
self.embeddings = BertEmbeddings(config)
        self.encoder = BertEncoder(config)
        self.pooler = BertPooler(config)
        self.apply(self.init_bert_weights)
```

Attention_mask	Embedding	Encoder	Pooler
將訓練集mask處理	詞向量層加載 預訓練好的詞向量		用來訓練兩句話的 BERT模型

Pytorch Preprocessing

保留 mask

＋

加入 attention dropout

加速收斂
提高準確度

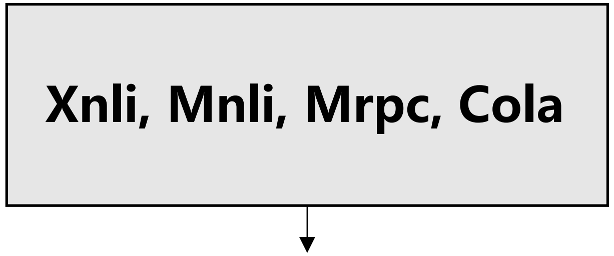
Masked LM – Position Embedding

BERT論文	實務上
1. 隨機Mask掉15%的WordPiece Token	確定要Mask掉的單詞後 ⬇ 80%會直接替換為[Mask] 10%將其替換為其他任意單詞 10%保留原始Token

Keras v.s. Pytorch
版本比較

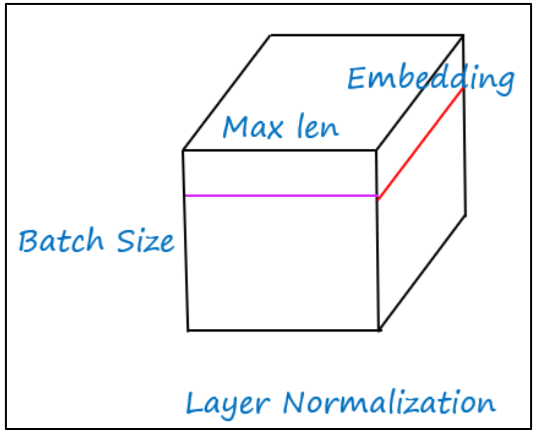
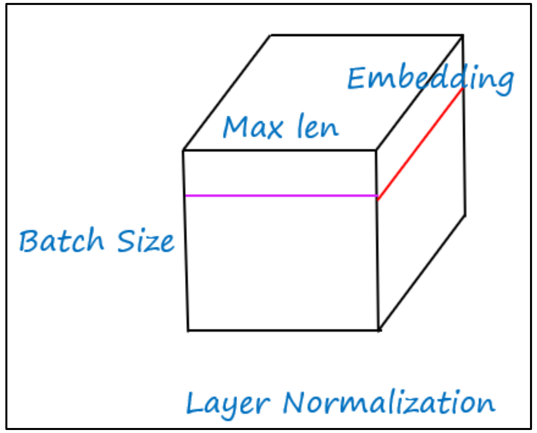
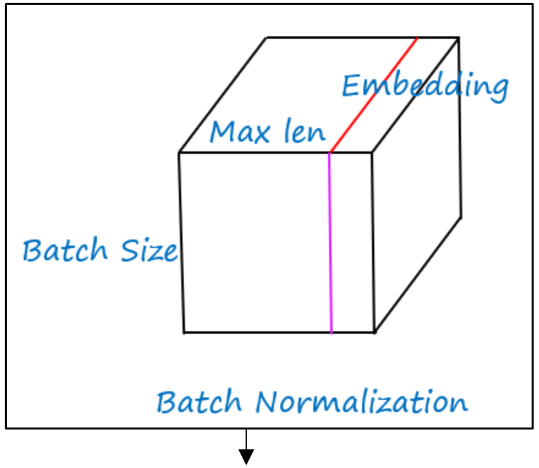
Keras v.s. Pytorch Version 比較

① Input

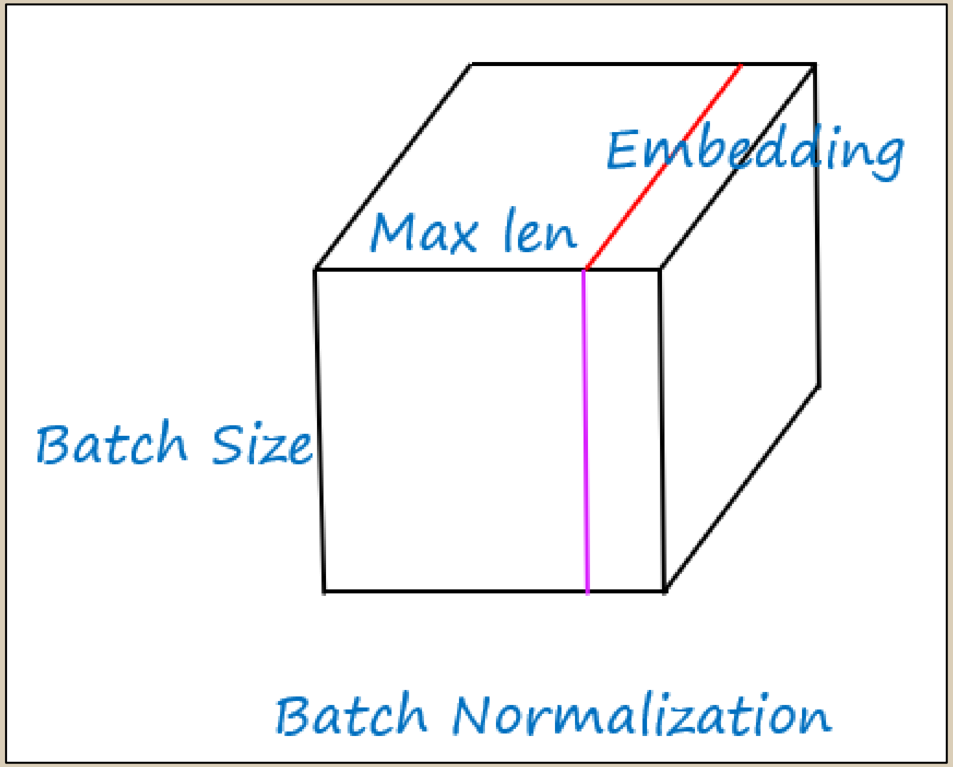
Keras	Pytorch
WSDM	 Xnli, Mnli, Mrpc, Cola WSDM
Preprocessing	

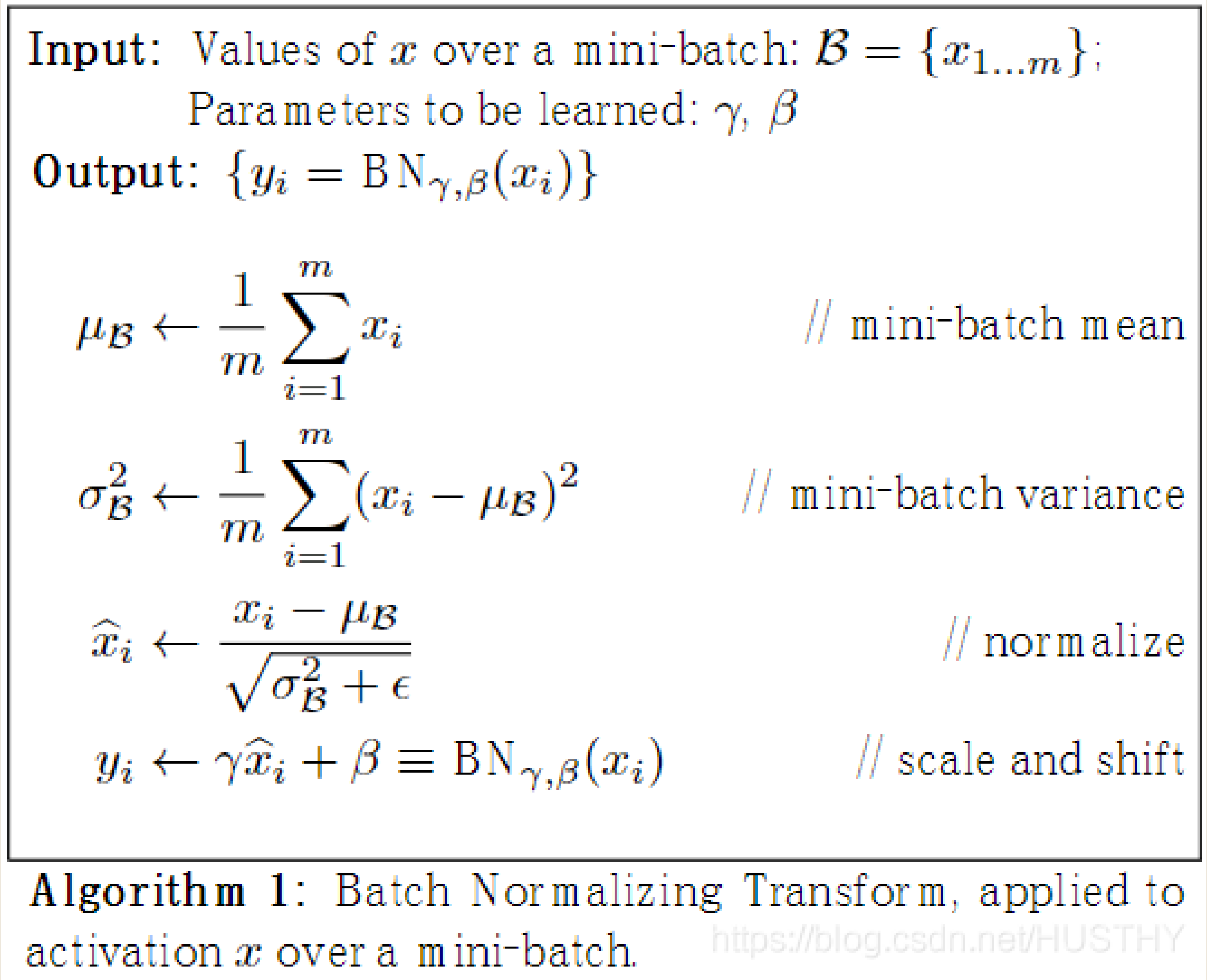
Keras v.s. Pytorch Version 比較

② LN & BN

Keras	Pytorch
 	  增加mini-batch進行正規化 1. 增加神經網路穩定性 2. 拉平資料分佈、加速收斂

Keras v.s. Pytorch Version 比較





Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1\ldots m}\}$;
Parameters to be learned: γ, β

Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

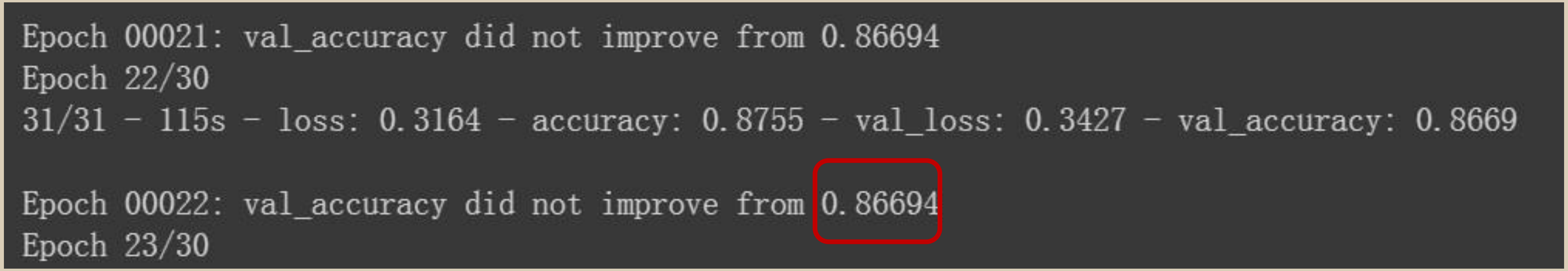
$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma\widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

Algorithm 1: Batch Normalizing Transform, applied to activation x over a mini-batch.

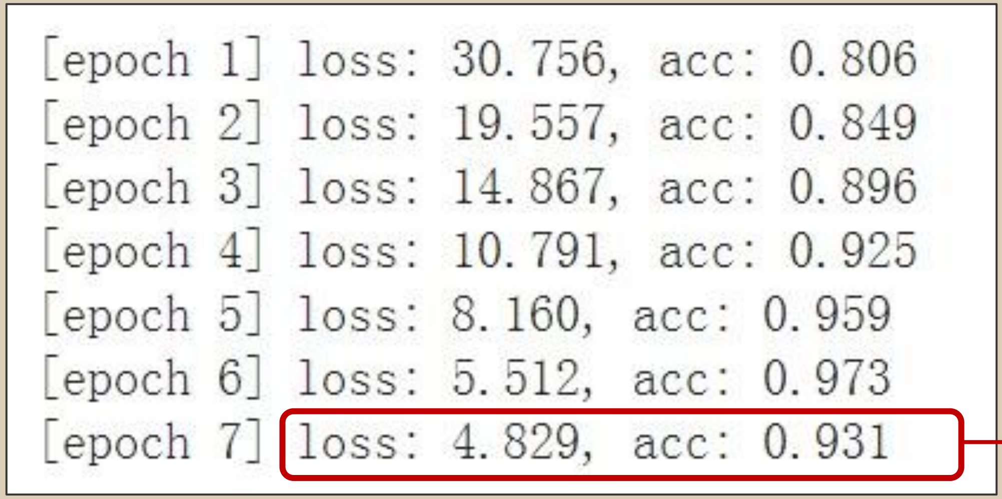
Keras v.s. Pytorch Version 比較

Keras



```
Epoch 00021: val_accuracy did not improve from 0.86694
Epoch 22/30
31/31 - 115s - loss: 0.3164 - accuracy: 0.8755 - val_loss: 0.3427 - val_accuracy: 0.8669

Epoch 00022: val_accuracy did not improve from 0.86694
Epoch 23/30
```

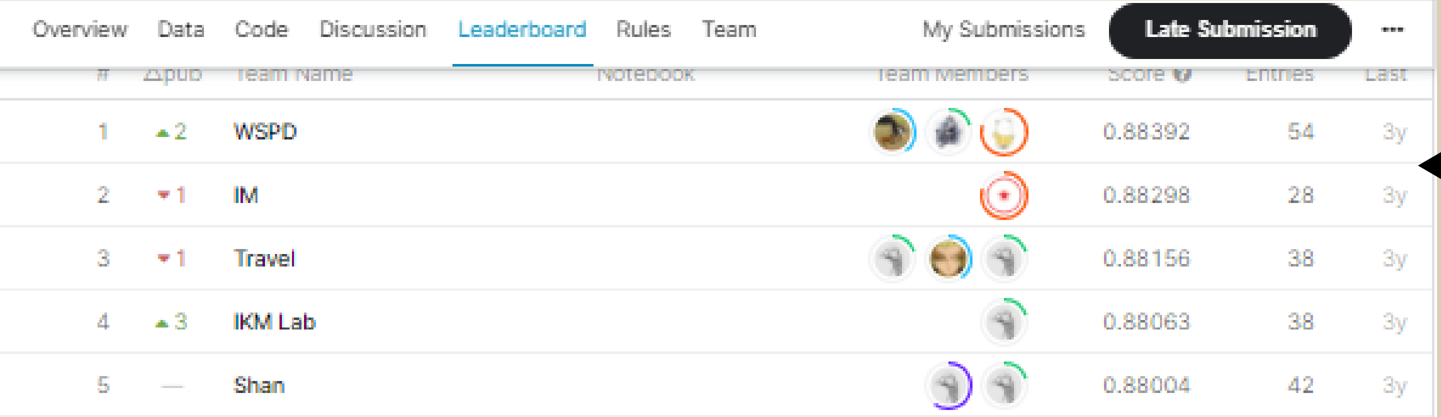
Pytorch



```
[epoch 1] loss: 30.756,  acc: 0.806
[epoch 2] loss: 19.557,  acc: 0.849
[epoch 3] loss: 14.867,  acc: 0.896
[epoch 4] loss: 10.791,  acc: 0.925
[epoch 5] loss: 8.160,  acc: 0.959
[epoch 6] loss: 5.512,  acc: 0.973
[epoch 7] loss: 4.829,  acc: 0.931
```

手刻程式因素：
Epoch 7 準確度是自己承接
val_loss,val_accuracy印出來的

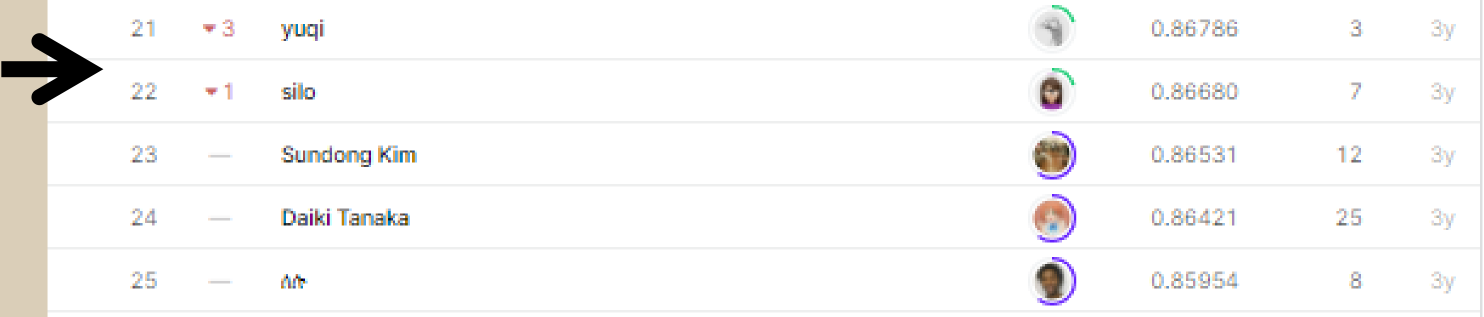
31 Keras v.s. Pytorch Version 比較



#	△pub	Team Name	Notebook	Team Members	Score	Entries	Last
1	▲2	WSPD			0.88392	54	3y
2	▼1	IM			0.88298	28	3y
3	▼1	Travel			0.88156	38	3y
4	▲3	IKM Lab			0.88063	38	3y
5	—	Shan			0.88004	42	3y

Pytorch version

·
·
·
·



#	△pub	Team Name	Notebook	Team Members	Score	Entries	Last
21	▼3	yuqi			0.86786	3	3y
22	▼1	silo			0.86680	7	3y
23	—	Sundong Kim			0.86531	12	3y
24	—	Daiki Tanaka			0.86421	25	3y
25	—	ሰሱ			0.85954	8	3y

Keras version

參考資料

- A Study of the Effect of Dropout on Imbalanced Data Classification using Deep Neural Networks
  http://www.jmest.org/wp-content/uploads/JMESTN42352707.pdf

- Batch Normalization和Layer Normalization归一化原理
  https://zhuanlan.zhihu.com/p/101570806

- BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding
  https://arxiv.org/abs/1810.04805

- 多分类模型Accuracy, Precision, Recall和F1-score的超级无敌深入探讨
  https://zhuanlan.zhihu.com/p/147663370

- 模型压缩实践系列之——layer dropout
  https://zhuanlan.zhihu.com/p/106198038

參考資料

- Transfer Learning for NLP: Fine-Tuning BERT for Text Classification
  https://www.analyticsvidhya.com/blog/2020/07/transfer-learning-for-nlp-fine-tuning-bert-for-text-classification/

- Text classification with transformers in Tensorflow2：
  https://laptrinhx.com/text-classification-with-transformers-in-tensorflow-2-bert-2931716339/

- 進入NLP世界的最佳橋樑：寫給所有人的自然語言處理與深度學習入門指南
  https://leemeng.tw/shortest-path-to-the-nlp-world-a-gentle-guide-of-natural-language-processing-and-deep-learning-for-everyone.html#%E6%84%8F%E6%96%99%E4%B9%8B%E5%A4%96%E7%9A%84-Kaggle-%E7%AB%B6%E8%B3%BD

- 【深度学习】BERT详解
  https://zhuanlan.zhihu.com/p/130913995

- BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding
  https://arxiv.org/pdf/1810.04805.pdf

THANK YOU