# Transformers Tutorial

## Taipei QA Bot with ALBERT-zh

Presenter: UDIC LAB MS1

# 參考資源

- Taipei QA BOT (投影片範例)
  - https://github.com/p208p2002/taipei-QA-BERT
- Transformers
  - https://github.com/huggingface/transformers
- 中文ALBERT
  - https://github.com/p208p2002/albert-zh-for-pytorch-transformers

# 預期進度

1. Tokenizer、Token Embeddings
2. Segment Embeddings、Position Embeddings、Model input
3. Dataset、DataLoader
4. 模型建構與訓練

# Chapter 1

## Tokenizer、Token Embeddings

Presenter: UDIC LAB MS1

# NLP模型建構流程

原始訓練資料 → 轉換成輸入特徵 → 製作Dataset、Dataloader → 模型建構 → 訓練、調整參數

# Bert input

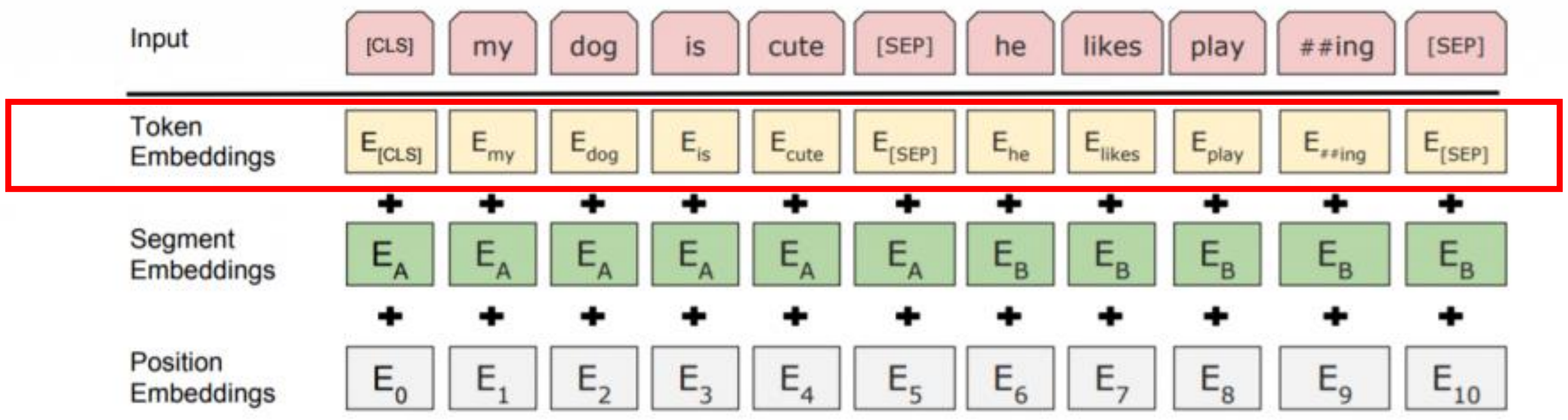


Figure 2: BERT input representation. The input embeddings are the sum of the token embeddings, the segmentation embeddings and the position embeddings.

- BERT的特殊符號
  1. [CLS] 起頭符號
  2. [SEP] 分割符號
  3. [UNK] 未知詞

- 標準的BERT INPUT有三層
  - Token Embeddings
  - Segment Embeddings
  - Position Embeddings

- 本節聚焦在使用Tokenizer建立Token Embeddings
- 標準的Token Embeddings格式
  - [CLS]句子A[SEP]句子C[SEP]
  - [CLS]句子A[SEP]

# Tokenizer

# 標記化

- tokenize.tokenize()
  - 川普於2019年12月18日被眾議院以濫權和藐視國會的名義彈劾

```
>>> tokenizer.tokenize('川普於2019年12月18日被眾議院以濫權和藐視國會的名義彈劾')
```

```
['川', '普', '於', '2019', '年', '12', '月', '18', '日', '被', '眾', '議', '院', '以', '濫', '權', '和', '藐', '視', '國', '會', '
的', '名', '義', '彈', '劾']
```

轉換成 wordpiece 格式

- tokenizer.convert_tokens_to_ids()

```
>>> tokenizer.convert_tokens_to_ids(['川', '普', '於', '2019', '年', '12', '月', '18', '日', '被', '眾', '議', '院', '以', '濫', '
權', '和', '藐', '視', '國', '會', '的', '名', '義', '彈', '劾'])
```

```
[2335, 3249, 3176, 9160, 2399, 8110, 3299, 8123, 3189, 6158, 4707, 6359, 7368, 809, 4093, 3609, 1469, 5967, 6213, 1751, 3298, 4638,
 1399, 5412, 2492, 1231]
```

轉換成對應的wordpiece ids

# 快速建立BERT INPUT

- tokenizer.build_inputs_with_special_tokens()

**AB句用法**

tokenizer.build_inputs_with_special_tokens(SENTENCE_A, SENTENCE_ B)

=>[CLS] SENTENCE_A[SEP] SENTENCE_B[SEP]

單句用法

tokenizer.build_inputs_with_special_tokens(A)

=>[CLS] SENTENCE_A[SEP]

```
>>> tokenizer.build_inputs_with_special_tokens([2335, 3249, 3176, 9160, 2399, 8110, 3299, 8123, 3189, 6158, 4707, 6359, 7368, 809,
4093, 3609, 1469, 5967, 6213, 1751, 3298, 4638, 1399, 5412, 2492, 1231])
[101, 2335, 3249, 3176, 9160, 2399, 8110, 3299, 8123, 3189, 6158, 4707, 6359, 7368, 809, 4093, 3609, 1469, 5967, 6213, 1751, 3298,
4638, 1399, 5412, 2492, 1231, 102]
```

# 反向查詢

- tokenizer.convert_ids_to_tokens()

```
>>> tokenizer.convert_ids_to_tokens([101, 2335, 3249, 3176, 9160, 2399, 8110, 3299, 8123, 3189, 6158, 4707, 6359, 7368, 809, 4093,
3609, 1469, 5967, 6213, 1751, 3298,4638, 1399, 5412, 2492, 1231, 102])
```

```
['[CLS]', '川', '普', '於', '2019', '年', '12', '月', '18', '日', '被', '眾', '議', '院', '以', '濫', '權', '和', '藐', '視', '國',
 '會', '的', '名', '義', '彈', '劾', '[SEP]']
```

- 更多的Tokenizer用法
  - https://huggingface.co/transformers/main_classes/tokenizer.html

# 小結

- 使用Tokenizer將輸入轉換成BERT Token Embeddings
- 標準的NLP模型架構中都會包含Tokenizer
- Tokenizer負責了Token與id之間的互轉
- 好的Tokenizer會削減不常用的詞彙，使用“[UNK]”表示以節省記憶體

# Chapter 2

## Segment Embeddings、Position Embeddings、Model input

Presenter: UDIC LAB MS1

# Bert input

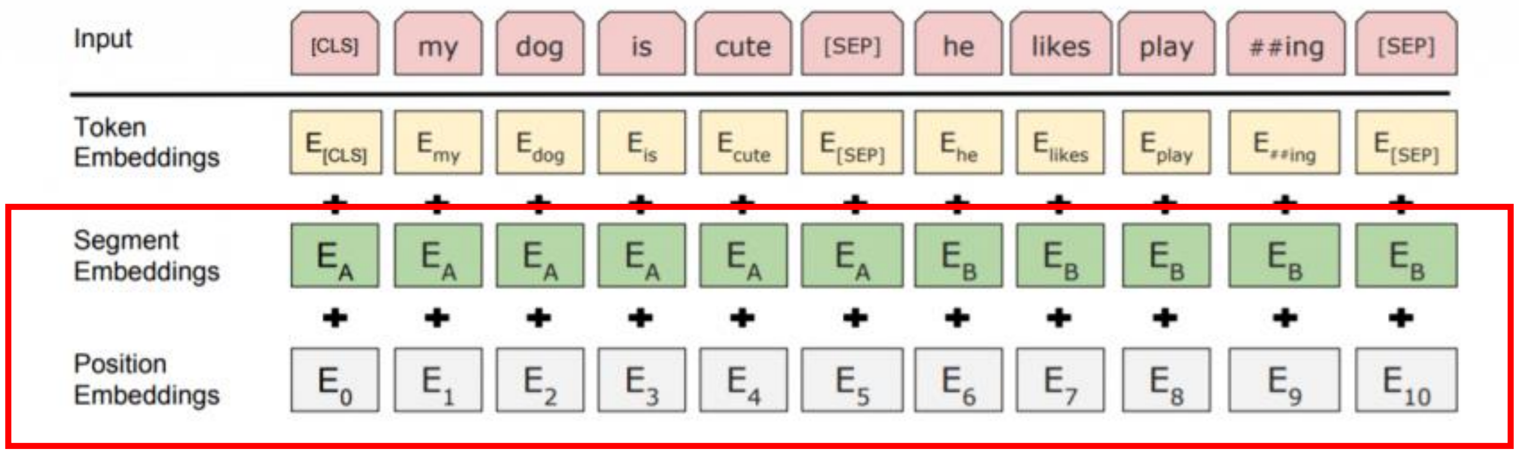


Figure 2: BERT input representation. The input embeddings are the sum of the token embeddings, the segmentation embeddings and the position embeddings.

模型會自動學習positon embeddings，所以不需提供
而為了正確的告知模型輸入的長度，我們在實作中另外提供attention mask

- BERT的特殊符號
  1. [CLS] 起頭符號
  2. [SEP] 分割符號
  3. [UNK] 未知詞

- 標準的BERT INPUT有三層
  - Token Embeddings
  - Segment Embeddings
  - Position Embeddings

- 標準的Token Embeddings格式
  - [CLS]句子A[SEP]句子C[SEP]
  - [CLS]句子A[SEP]

Segment Embeddings、Position Embeddings

# [CLS]你今天好嗎[SEP]、[CLS]還不錯[SEP]

| | | | | |
|---|---|---|---|---|
| Input | [CLS]你今天好嗎[SEP] | Input | [CLS]還不錯[SEP] | |
| Token Embeddings | [101,1,2,3,4,5,102] | Token Embeddings | [101,6,7,8,102, | 0,0] |
| Segment Embeddings | [ 0 , 0 , 0 , 0 , 0 , 0 , 0 ] | Segment Embeddings | [ 0 , 0 , 0 , 0 , 0 , | 0,0] |
| Attention Mask | [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ] | Attention Mask | [ 1 , 1 , 1 , 1 , 1 , | 0,0] |
| | 長度7 | | 長度5 | 補齊 |

Batch

# [CLS]你今天好嗎[SEP]還不錯[SEP]

| | |
|---|---|
| Input | [CLS]你今天好嗎[SEP]還不錯[SEP] |
| Token Embeddings | [101,1,2,3,4,5,102,6,7,8,102] |
| Segment Embeddings | [0,0,0,0,0,0,0,1,1,1,1] |
| Attention Mask | [1,1,1,1,1,1,1,1,1,1,1] |

# 模型加載、輸入

# 加載、輸入模型

由於albert-zh沒有被整合在transformers
因此使用修改過的module
但是用法、接口與文件與transformers通用

https://github.com/p208p2002/albert-zh-for-pytorch-transformers

```
from albert_zh import AlbertConfig,AlbertForSequenceClassification,AlbertTokenizer
import torch
if __name__ == "__main__":
    tokenizer = AlbertTokenizer.from_pretrained('albert_tiny/vocab.txt')
    model_config = AlbertConfig.from_json_file('./albert_tiny/config.json')
    model = AlbertForSequenceClassification.from_pretrained('./albert_tiny',config = model_config)

    intput_str = '周杰倫，臺灣著名華語流行歌曲男歌手、音樂家、唱片製片人。同時是演員、導演，也是電競團隊隊長兼老闆
    input_ids = torch.tensor(tokenizer.encode(intput_str)).unsqueeze(0)  # Batch size 1
    labels = torch.tensor([1]).unsqueeze(0)  # Batch size 1
    outputs = model(input_ids, labels=labels)
    loss, logits = outputs[:2]

    print(loss,logits)
```

```
root@5bd13fd0a321:/tf/albert-zh-for-pytorch-transformers# python usage_example.py
tensor(0.3286, grad_fn=<NllLossBackward>) tensor([[-0.2912,  0.6528]], grad_fn=<AddmmBackward>)
```

https://github.com/p208p2002/albert-zh-for-pytorch-transformers/blob/master/usage_example.py

# 完整的輸入接口

| Transformers Model Input | 對應的 Bert Embeddings | 說明/輸入建議 |
| --- | --- | --- |
| input_ids | Token_embedings | 必要 |
| token_type_ids | Segment_embeddings | 建議提供 |
| position_ids (optional) | Postion_embeddings | 根據訓練過程學習，不需提供 |
| attention_mask | - | 聚焦非padding(補零)的輸入，強烈建議提供 |

官方文件
https://huggingface.co/transformers/v2.3.0/model_doc/albert.html#albertforsequenceclassification

# 補充-修正jupyter import

在jupyter、colab等環境可能會遇到import問題

增加系統路徑避免找不到自訂package

```
In [2]: !git clone https://github.com/p208p2002/albert-zh-for-pytorch-transformers.git albert
```

```
In [19]: import sys
         sys.path.append('.')
         from albert.albert_zh import AlbertConfig, AlbertTokenizer, AlbertForSequenceClassification
```

https://github.com/p208p2002/albert-zh-for-pytorch-transformers#%E5%B8%B8%E8%A6%8B%E5%95%8F%E9%A1%8C

# Chapter 3

## Dataset、DataLoader

Presenter: UDIC LAB MS1

# 特徵轉換、類別反查

- 將資料由text經由tokenizer轉換成對應的ids
- 並且根據資料格式組建 segment_ids、attention_masks(input_masks)
- 幫資料編號，還有對應的反查字典

```python
def convert_data_to_feature(tokenizer, train_data_path):
```

```python
    data_features = {'input_ids':input_ids,
                     'input_masks':input_masks,
                     'input_segment_ids':input_segment_ids,
                     'answer_lables':answer_lables,
                     'question_dic':question_dic,
                     'answer_dic':ans_dic}

    output = open('trained_model/data_features.pkl', 'wb')
    pickle.dump(data_features,output)
    return data_features
```

由answer_dic而來

主要用途為保存answer_dic實例
用於將預測的id轉換為對應的label

https://github.com/p208p2002/taipei-QA-BERT/blob/master/core.py#L80

```python
#
data_feature = convert_data_to_feature(tokenizer,'Taipei_QA_new.txt')
input_ids = data_feature['input_ids']
input_masks = data_feature['input_masks']
input_segment_ids = data_feature['input_segment_ids']
answer_lables = data_feature['answer_lables']
```

https://github.com/p208p2002/taipei-QA-BERT/blob/master/train.py#L34

DataDic功能:

- 將list內重複元素刪除
- 為每一個元素編號
- 提供 to_id、to_text查詢方法

輸入為text list
例如
['第1類', '第2類', '第2類', '第1類']

```python
class DataDic(object):
    def __init__(self, answers):
        self.answers = answers #全部答案(含重複)
        self.answers_norepeat = sorted(list(set(answers))) # 不重複
        self.answers_types = len(self.answers_norepeat) # 總共多少類
        self.ans_list = [] # 用於查找id或是text的list
        self._make_dic() # 製作字典

    def _make_dic(self):
        for index_a,a in enumerate(self.answers_norepeat):
            if a != None:
                self.ans_list.append((index_a,a))

    def to_id(self,text):
        for ans_id,ans_text in self.ans_list:
            if text == ans_text:
                return ans_id

    def to_text(self,id):
        for ans_id,ans_text in self.ans_list:
            if id == ans_id:
                return ans_text
```

# PyTorch-Dataset & DataLoader

- Dataset
  - PyTorch封裝資料集的格式
  - 輸入的格式必須先轉換為tensor

- DataLoader
  - 定義如何從Dataset取出資料
  - 將資料從Dataset中以迭代方式取出
  - 每次迭代的資料得到的資料會是一個batch

```python
def make_dataset(input_ids, input_masks, input_segment_ids, answer_lables):
    all_input_ids = torch.tensor([input_id for input_id in input_ids], dtype=torch.long)
    all_input_masks = torch.tensor([input_mask for input_mask in input_masks], dtype=torch.long)
    all_input_segment_ids = torch.tensor([input_segment_id for input_segment_id in input_segment_ids], dtype=torch.long)
    all_answer_lables = torch.tensor([answer_lable for answer_lable in answer_lables], dtype=torch.long)
    return TensorDataset(all_input_ids, all_input_masks, all_input_segment_ids, all_answer_lables)
```

https://github.com/p208p2002/taipei-QA-BERT/blob/master/core.py#L32

```python
full_dataset = make_dataset(input_ids = input_ids, input_masks = input_masks, \
                            input_segment_ids = input_segment_ids, answer_lables = answer_lables)
train_dataset, test_dataset = split_dataset(full_dataset, 0.9)
train_dataloader = DataLoader(train_dataset,batch_size=16,shuffle=True)
test_dataloader = DataLoader(test_dataset,batch_size=16,shuffle=True)
```

https://github.com/p208p2002/taipei-QA-BERT/blob/master/train.py#L41

補充:更精簡的make_dataset寫法
https://gist.github.com/p208p2002/d588d6eec034f0f89ad41e6fab3cb948

# 迭代Dataloader

https://github.com/p208p2002/taipei-QA-BERT/blob/master/train.py#L59

https://github.com/p208p2002/taipei-QA-BERT/blob/master/core.py#L37

```
# TensorDataset(all_input_ids, all_input_masks, all_input_segment_ids, all_answer_lables)
# 注意丟入Dataset的順序
for batch_index, batch_dict in enumerate(train_dataloader):
  # 從DataLoader取出的時候也是同樣的順序
  batch_input_ids = batch[0]
  batch_input_masks = batch[1]
  batch_input_segment_ids = batch[2]
  batch_answer_lables = batch[3]
  outputs = model(batch_input_ids ,labels = batch_answer_lables)
```

https://gist.github.com/p208p2002/1ee58d90cb93c0219749de94d0aa4897

# Chapter 4

## 模型建構與訓練

Presenter: UDIC LAB MS1

# 訓練流程建構

```
model.zero_grad() # 執行梯度歸零確保模型沒有記錄梯度資訊
for epoch in range(30):
  # 訓練環節
  for batch_index, batch_dict in enumerate(train_dataloader): # 從Dataloader取出batch
    model.train() # 允許模型更新權重
    outputs = model(batch_dict['bert_input_ids'], labels = batch_dict['answer_ids']) # 將輸入餵給模型
    loss,logits = outputs[:2] # 得到loss與logits(輸出)
    loss.sum().backward() # 反向傳播
    optimizer.step() # 使用優化器進行模型權重更新
    model.zero_grad() # 梯度歸零

  # 測試環節
  for batch_index, batch_dict in enumerate(test_dataloader):
    model.eval() # 鎖定權重
    outputs = model(batch_dict['bert_input_ids'], labels = batch_dict['answer_ids']) # 將輸入餵給模型
    loss,logits = outputs[:2] # 得到loss與logits(輸出)
    # 測試是為了觀察數值變化，無須更新或優化
```

程式碼僅為示範、無法執行，完整程式請參考連結

https://github.com/p208p2002/taipei-QA-BERT/blob/master/train.py https://github.com/p208p2002/taipei-QA-BERT/blob/master/train.py

# 訓練架構流水線總覽

```python
model_setting = {
    "model_name":"bert",
    "config_file_path":"bert-base-chinese",
    "model_file_path":"bert-base-chinese",
    "vocab_file_path":"bert-base-chinese-vocab.txt",
    "num_labels":149  # 分幾類
}
```

```python
model, tokenizer = use_model(**model_setting)
```

1. 初始化、加載模型

```python
# setting device
device = torch.device('cuda') if torch.cuda.is_available() else torch.device('cpu')
print("using device",device)
model.to(device)
```

2. 指定硬體裝置

```python
#
data_feature = convert_data_to_feature(tokenizer,'Taipei_QA_new.txt')
input_ids = data_feature['input_ids']
input_masks = data_feature['input_masks']
input_segment_ids = data_feature['input_segment_ids']
answer_lables = data_feature['answer_lables']
```

3. 將訓練資料讀入
並且組建BERT輸入格式

```python
#
full_dataset = make_dataset(input_ids = input_ids, input_masks = input_masks, input_segment_ids = input_segment_ids, answer_lables = an
train_dataset, test_dataset = split_dataset(full_dataset, 0.9)
train_dataloader = DataLoader(train_dataset,batch_size=16,shuffle=True)
test_dataloader = DataLoader(test_dataset,batch_size=16,shuffle=True)
```

4. 將組建好的輸入格式轉換成tensor格式，
並且建立dataset與dataloader

```python
model.zero_grad()
for epoch in range(30):
    running_loss_val = 0.0
    running_acc = 0.0
    for batch_index, batch_dict in enumerate(train_dataloader):
        model.train()
        batch_dict = tuple(t.to(device) for t in batch_dict)
        outputs = model(
            batch_dict[0],
            # attention_mask=batch_dict[1],
            labels = batch_dict[3]
            )
        loss,logits = outputs[:2]
        loss.sum().backward()
        optimizer.step()
        # scheduler.step()  # Update learning rate schedule
        model.zero_grad()

        # compute the loss
        loss_t = loss.item()
        running_loss_val += (loss_t - running_loss_val) / (batch_index + 1)

        # compute the accuracy
        acc_t = compute_accuracy(logits, batch_dict[3])
        running_acc += (acc_t - running_acc) / (batch_index + 1)

        # log
        print("epoch:%2d batch:%4d train_loss:%2.4f train_acc:%3.4f"%(epoch+1, batch_index+1, running_loss_val, running_acc))

    running_loss_val = 0.0
    running_acc = 0.0
```

5. 建構training loop、開始訓練

# 作業

電影評論情緒分析

Presenter: UDIC LAB MS1

# 參與在kaggle上的競賽項目

- https://www.kaggle.com/c/sentiment-analysis-on-movie-reviews/overview

- 要求:
  - 使用英文ALBERT或是BERT進行
  - 完成後提交"test.tsv"中題目的預測結果予kaggle並獲得分數
  - 網路有很多針對此題目的討論、範例請試著不要參考或使用
  - 對於程式中運用到的func至少理解其作用

- 可用運算資源:
  - Google Colab
  - Kaggle Notebook