應用Bidirectional LSTM於NLP之BBC文章分類

108智慧化企業整合

學生：施美全

OUTLINE

PART 01
Background

NLP / 5W1H

Background

NLP(Natural Language Processing)

理解人類語言能力

- ✓ 自然語言理解
- ✓ 自然語言生成
- ✓ 語音辨識
- ✓ 機器翻譯

Background

Problem Definition –5W1H

What

讓電腦學會新聞分類

When

NLP時代來臨

Who

Google、Firefox

Where

服務中心與網路

Why

NLP領域貢獻/過濾大量新聞文章/快速分類

How

資料處理、深度學習

PART 02
Data Preprocessing

Data Preprocessing

Data Resource

BBC新聞網站

- 共有2225個文檔
- 主要為五個主題領域的文章
- 商業/娛樂/政治/體育/科技

Data Preprocessing

將所需工具匯入：CSV / Tensorflow / numpy / Tokenizer / pad_sequences / stopwords

```
import csv
import tensorflow as tf
import numpy as np
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords
STOPWORDS = set(stopwords.words('english'))
```

Data Preprocessing

Data Parameter

將所需共同參數統一置上，以方便統一修改

```
vocab_size = 5000
embedding_dim = 512
max_length = 200
trunc_type = 'post'
padding_type = 'post'
oov_tok = '<OOV>'
training_portion = .6
test_portion= .2
```

Data Preprocessing

Data Preprocessing
Stopwords

Stopwords

- ✓ NLTK 的 stopwords 資料庫支援21種語言
- ✓ Stopwords主要分為兩類:
  - ① 功能詞:如'the'、'is'、'at'、'which'、'on'等
  - ② 詞彙詞:如‘want’，應用較廣泛

```
articles = []
labels = []
with open("bbc-text.csv", 'r') as csvfile:
    reader = csv.reader(csvfile, delimiter=',')
    next(reader)
    for row in reader:
        labels.append(row[0])
        article = row[1]
        for word in STOPWORDS:
            token = ' ' + word + ' '
            article = article.replace(token, ' ')
            article = article.replace('  ', ' ')
        articles.append(article)
print(len(labels))
print(len(articles))
```

```
2225
2225
```

Data Preprocessing

Data Segmentation

訓練集：	測試集：	驗證集
= 6	: 2	: 2
= 1335	: 445	: 445

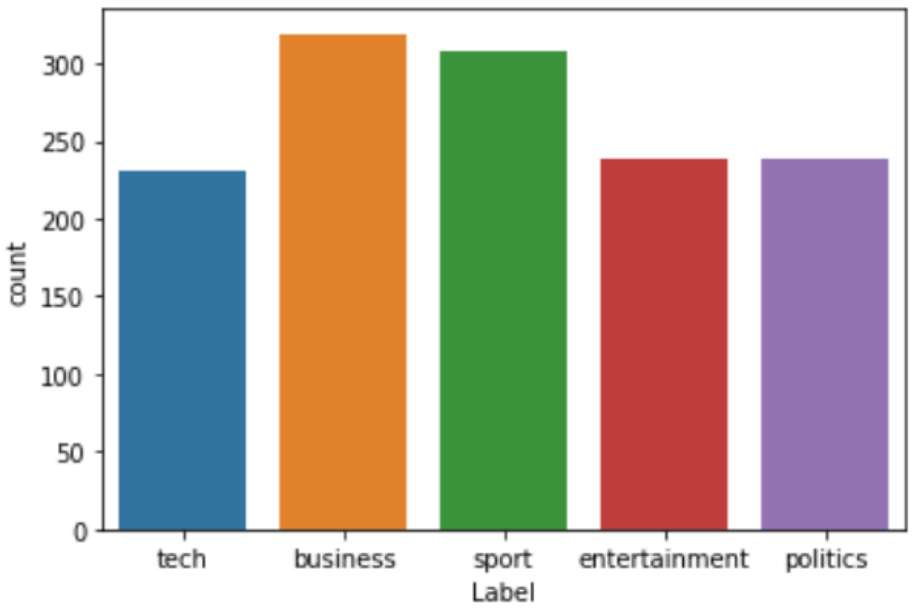
```
1335
445
1335
1335
445
445
445
445
```

```
train_size = int(len(articles) * training_portion)
test_size=int(len(articles) * test_portion)
valid_size=int(len(articles) * test_portion)

train_articles = articles[0: train_size]
train_labels = labels[0: train_size]
test_articles = articles[train_size:(train_size+test_size)]
test_labels = labels[train_size:(train_size+test_size)]
validation_articles = articles[(train_size+test_size):]
validation_labels = labels[(train_size+test_size):]

print(train_size)
print(test_size)
print(len(train_articles))
print(len(train_labels))
print(len(test_articles))
print(len(test_labels))
print(len(validation_articles))
print(len(validation_labels))
```

Data Preprocessing

可視畫圖表:查看訓練集各項Label資料

```
import matplotlib.pyplot as plt
import seaborn as sns
plt.figure()
sns.countplot(train_labels)
plt.xlabel('Label')
plt.show()
```



Data Preprocessing
Tokenizer

將文章文字進行標記

- ✓ **Tokenizer**
  將一段文字轉換成一系列的詞彙（Tokens），並將其建立字典 (word_index)
- ✓ **Vocab_size=5000**
  避免字典過大，剩餘的新詞彙視為Unknown
- ✓ **OOV : 有特殊單字及含意，則歸在此類**

```
tokenizer = Tokenizer(num_words = vocab_size, oov_token=oov_tok)
tokenizer.fit_on_texts(train_articles)
word_index = tokenizer.word_index
dict(list(word_index.items())[0:10])
```

```
{'<OOV>': 1,
 'also': 6,
 'mr': 3,
 'new': 8,
 'one': 10,
 'people': 7,
 'said': 2,
 'us': 9,
 'would': 4,
 'year': 5}
```

Data Preprocessing

Tokenizer

將文字轉為序列列表(LSTM所需向量)

```
train_sequences = tokenizer.texts_to_sequences(train_articles)
print(train_sequences[10])
```

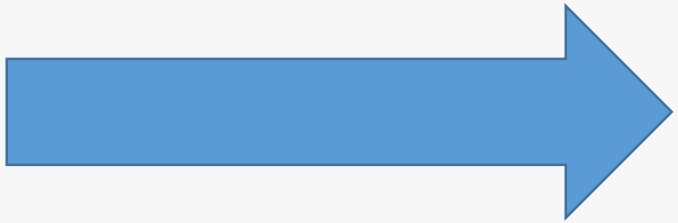
```
[2389, 1, 257, 4144, 19, 633, 524, 257, 4144, 1, 1, 1589, 1, 1, 2389, 19, 496,
```

Data Preprocessing
Zero Padding

由於每個新聞文章的序列長度不相同
採用序列的 Zero Padding進行處理

- Max_length=200
  1. 長度超過200的序列，尾巴會被刪掉
  2. 長度不足的序列，則在詞彙補零

```
train_padded = pad_sequences(train_sequences, maxlen=max_length,
                             padding=padding_type, truncating=trunc_type)
print(len(train_sequences[0]))
print(len(train_padded[0]))
print(len(train_sequences[1]))
print(len(train_padded[1]))
print(len(train_sequences[10]))
print(len(train_padded[10]))
```



```
425
200
192
200
186
200
```

Data Preprocessing

Zero Padding

例如:對於第11條，長度為186，長度控制為200，即添加14個零

```
print(train_padded[10])
```

```
[2389    1  257 4144   19  633  524  257 4144    1    1 1589    1    1
 2389   19  496    1    1  134  265    1  134  265  883  731  662 2632
    1 1019 1732    1 1485    1    1    1    1    1    1 3941    1    1
  104 4402    1    2 4403 1378  320 4145    1   56  375    1  320 2555
 3730   39   19 3553    1    1    1    1  515    1    1    1  794  761
 2154  409 4146    1  325   19    1  746 2390    1    1  142   10    1
 4404  663 4147    1   19    1  477  731  662    1   83   13  593    1
  257 4144    1  576    1 1732  943    1    1  795 2026  115    1    1
    1 3219   19    1  126  265    1 1774    1  515  504    1 1482 4405
  775 1166    1 1873   10   34  633  256    1   70  511  496  285 1557
   19  499    1    1 1909    1  822    1 3220    1 1283    6    1 2389
  496   19 3214    1    1    1    1    1    1  823   43 1969  586  287
   24  893    1  809   19  365   19   13  285 1557 1379  510   20   67
  776 1189 4148  274    0    0    0    0    0    0    0    0    0    0
    0    0    0    0]
```

Data Preprocessing
Zero Padding

對驗證集和測試集作相同資料前處理(長度控制於200)

驗證集

```
validation_sequences = tokenizer.texts_to_sequences(validation_articles)
validation_padded = pad_sequences(validation_sequences, maxlen=max_length,
                                  padding=padding_type, truncating=trunc_type)
print(len(validation_sequences))
print(validation_padded.shape)
```

測試集

```
test_sequences = tokenizer.texts_to_sequences(test_articles)
test_padded = pad_sequences(test_sequences, maxlen=max_length,
                            padding=padding_type, truncating=trunc_type)
print(len(test_sequences))
print(test_padded.shape)
```

```
445
(445, 200)
445
(445, 200)
```

Data Preprocessing
Label

標籤Label轉為numpy陣列-訓練集/測試集/驗證集

```
label_tokenizer = Tokenizer()
label_tokenizer.fit_on_texts(labels)
training_label_seq = np.array(label_tokenizer.texts_to_sequences(train_labels))
validation_label_seq = np.array(label_tokenizer.texts_to_sequences(validation_labels))
test_label_seq = np.array(label_tokenizer.texts_to_sequences(test_labels))

print(training_label_seq[0])
print(training_label_seq[1])
print(training_label_seq[2])
print(training_label_seq.shape)
print(validation_label_seq[0])
print(validation_label_seq[1])
print(validation_label_seq[2])
print(validation_label_seq.shape)
print(test_label_seq[0])
print(test_label_seq[1])
print(test_label_seq[2])
print(test_label_seq.shape)
```

```
[4]
[2]
[1]
(1335, 1)
[5]
[4]
[3]
(445, 1)
[4]
[5]
[4]
(445, 1)
```

歸屬在哪些主題

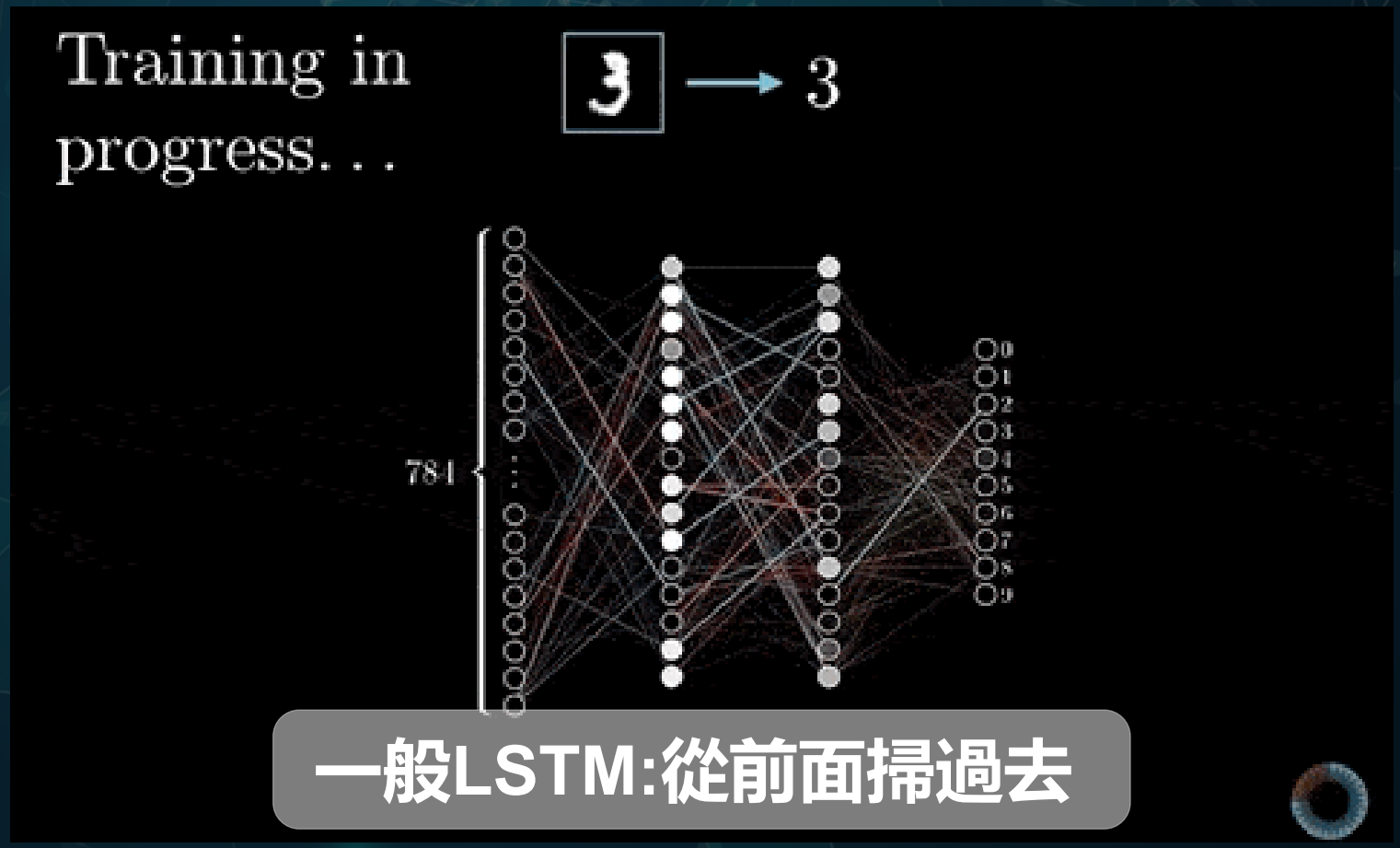
PART 03
Model Structure

Bidirectional LSTM

Model Structure

Bidirectional LSTM(反向傳播算法)

Bidirectional LSTM:從前面掃過去，亦從後面掃回來



一般LSTM:從前面掃過去

Model Structure

Bidirectional LSTM(反向傳播算法)

使用Sequential()建置的Bidirectional LSTM

```
model = tf.keras.Sequential([
    tf.keras.layers.Embedding(vocab_size, embedding_dim),
    tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(embedding_dim)),
    tf.keras.layers.Dense(embedding_dim, activation='relu'),
    tf.keras.layers.Dropout(0.5),
    tf.keras.layers.Dense(10, activation='softmax'),])
model.summary()
```

5000

512

Model Structure

Bidirectional LSTM(反向傳播算法)

Model Summary

```
Model: "sequential_8"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_9 (Embedding)      (None, None, 512)         2560000
_________________________________________________________________
bidirectional_7 (Bidirection (None, 1024)              4198400
_________________________________________________________________
dense_17 (Dense)             (None, 512)               524800
_________________________________________________________________
dropout_9 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_18 (Dense)             (None, 10)                5130
=================================================================
Total params: 7,288,330
Trainable params: 7,288,330
Non-trainable params: 0
```

Model Structure

Kearas—Deep Learning

Keras 建立深度學習模型感覺就像是在玩疊疊樂

PART 04
Analysis
Brainstorming

Analysis
Training & Validation Result

```
model.compile(loss='sparse_categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
num_epochs = 10
history = model.fit(train_padded, training_label_seq, epochs=num_epochs, validation_data=(
    validation_padded, validation_label_seq), verbose=2)
```

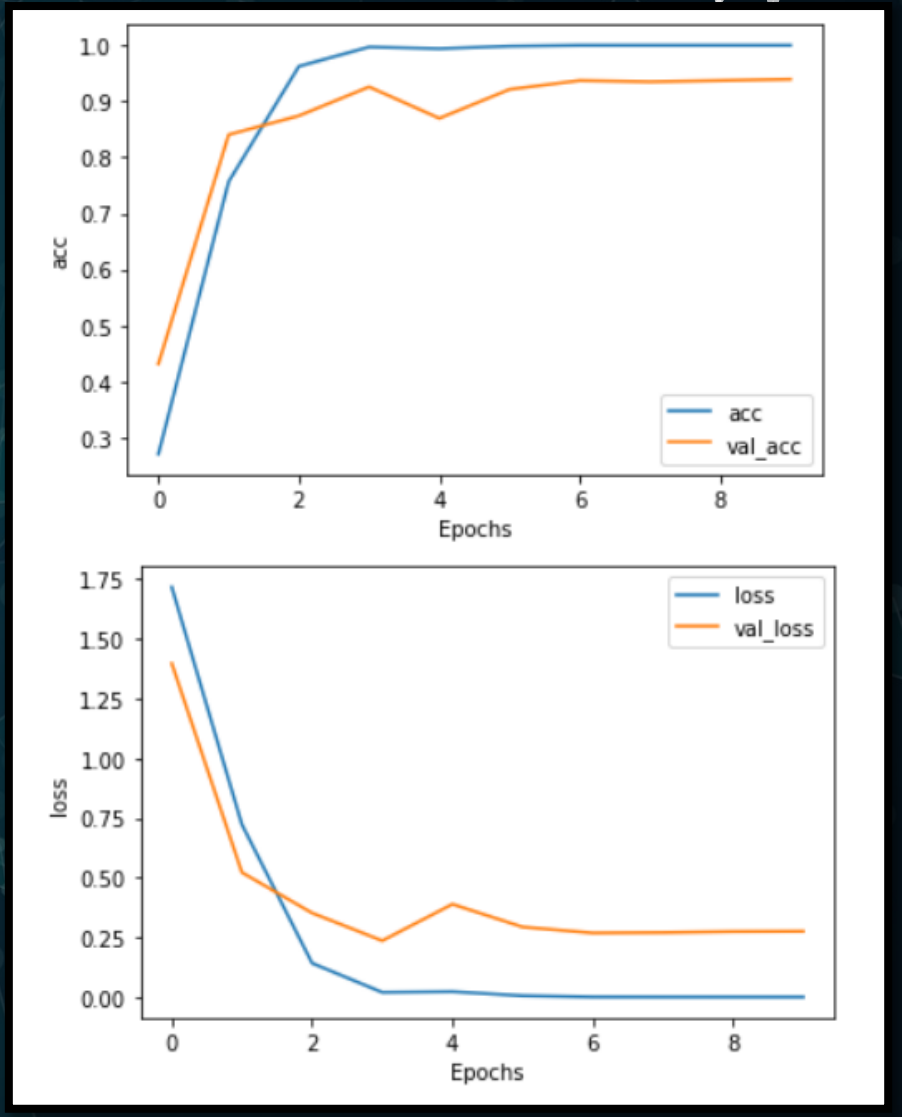
```
Train on 1335 samples, validate on 445 samples
Epoch 1/10
1335/1335 - 502s - loss: 1.7157 - acc: 0.2704 - val_loss: 1.3963 - val_acc: 0.4315
Epoch 2/10
1335/1335 - 493s - loss: 0.7221 - acc: 0.7566 - val_loss: 0.5215 - val_acc: 0.8404
Epoch 3/10
1335/1335 - 494s - loss: 0.1418 - acc: 0.9625 - val_loss: 0.3520 - val_acc: 0.8742
Epoch 4/10
1335/1335 - 496s - loss: 0.0198 - acc: 0.9970 - val_loss: 0.2358 - val_acc: 0.9258
Epoch 5/10
1335/1335 - 493s - loss: 0.0225 - acc: 0.9940 - val_loss: 0.3888 - val_acc: 0.8697
Epoch 6/10
1335/1335 - 493s - l                                                    acc: 0.9213
Epoch 7/10
1335/1335 - 489s - l                                                    val_acc: 0.9371
Epoch 8/10
1335/1335 - 489s - l                                                    val_acc: 0.9348
Epoch 9/10
1335/1335 - 485s - loss: 1.4053e-04 - acc: 1.0000 - val_loss: 0.2741 - val_acc: 0.9371
Epoch 10/10
1335/1335 - 483s - loss: 1.0150e-04 - acc: 1.0000 - val_loss: 0.2753 - val_acc: 0.9393
```

Train accuracy : 100.00%
Validation accuracy : 93.93%

Analysis

可視化圖示:
訓練集與驗證集accuracy及loss

```
import matplotlib.pyplot as plt
def plot_graphs(history, string):
  plt.plot(history.history[string])
  plt.plot(history.history['val_'+string])
  plt.xlabel("Epochs")
  plt.ylabel(string)
  plt.legend([string, 'val_'+string])
  plt.show()
plot_graphs(history, "acc")
plot_graphs(history, "loss")
```



Analysis

Testing Result

Test accuracy : 94.61%

```
scores = model.evaluate(test_padded, test_label_seq, verbose=2)
print('Test accuracy:', scores[1])
```

```
445/445 - 29s - loss: 0.2362 - acc: 0.9461
Test accuracy: 0.9460674
```

PART 05
Conclusion

Conclusion

Background

01

NLP之趨勢

5W1H

Data Preprocessing

02

Stopwords/Tokenizer /Zero Padding

Data Segmentation/Label轉為numpy陣列

Model Structure

03

Bidirectional LSTM

Analysis

04

- Training accuracy : 100.00%
- Testing accuracy : 94.61%
- Validation accuracy : 93.93%

Conclusion
未來發展方向

持續優化及訓練模型，提升其準確度

分類顧客之回饋與客訴留言

- ✓ 工業4.0—以需而至
- ✓ 快速反應顧客需求

NLP領域

應用層面:文章過濾器

網路文章多樣化，保障未成年心靈

Conclusion

未來發展方向——NLP應用

Thanks