全部展開
回到頂部
移至底部

HW5: Fake News Detection 2

309554001 劉雨恩

1. Data Preprocessing

1. 將 `train.csv` 、 `test.csv` 和 `sample_submission.csv` 的資料 (`id` , `text` , `label`) 提取出來，並消除 HTML tag 及 stop words (利用 `spacy` 提供的停頓詞列表)。

```
import pandas as pd
import spacy

spacy.load('en_core_web_sm')
spacy_stopwords = spacy.lang.en.stop_words.STOP_WORDS

def data_preprocess(data_path, mode, stop_words):

    def rm_tags_stops(text, stop_words):
        # remove tags
        re_tag = re.compile(r'<[^>]+>')
        out_text = re_tag.sub(' ', text.lower())

        # remove stop words
        out_text = (" ").join([word for word in out_text.split(' ') if not wor

        return out_text

    # read data
    fp = open(f'{data_path}{mode}.csv')
    data_lines = fp.readlines()
    fp.close()

    # data preprocess
    data_df = pd.DataFrame([], columns=['id', 'text', 'label'])
    avg_text_len = 0
    if mode == 'train':
        for i, l_i in enumerate(data_lines):
            if i == 0:
                continue
            try:
                dict_i = {}
                text_i, label = l_i.split('\t')
                text_i = rm_tags_stops(text_i.strip(), stop_words)
                avg_text_len += len(text_i.split(' '))

                dict_i['id'] = [str(i)]
                dict_i['text'] = [text_i]
                dict_i['label'] = [int(label.strip())]
                data_df = pd.concat([data_df, pd.DataFrame.from_dict(dict_i, o
            except:
                pass
    else:
        label_data = pd.read_csv(f'{data_path}sample_submission.csv')

        for i, (test_li, label_i) in enumerate(zip(data_lines, label_data['lab
            if i == 0:
                continue
            dict_i = {}
            id, text_i = test_li.split('\t')
            text_i = rm_tags_stops(text_i.strip(), stop_words)
            avg_text_len += len(text_i.split(' '))

            dict_i['id'] = [id.strip()]
            dict_i['text'] = [text_i]
            dict_i['label'] = [int(label_i)]
            data_df = pd.concat([data_df, pd.DataFrame.from_dict(dict_i, orien
    print(f'{mode} avg text len: {avg_text_len / len(data_df)}')

    return data_df['text'], np.array(data_df['label'].values)
```

2. 使用 `Tokenizer` 模組建立 token，建立一個 3800 (`hyper_params['dict_len']`) 字的字典：讀取所有訓練文檔資料之後，會依照每個英文字在資料出現的次數進行排序，並將前 3800 名的英文單字加進字典中。
3. 透過 `texts_to_sequences` 可以將訓練和測試集資料中的文檔轉換為數字列表。
4. 每一篇影評文字字數不固定，但後續進行深度學習模型訓練時長度必須固定，因此需要截長補短 `sequence.pad_sequences` ：長度小於 380 (`hyper_params['content_len']`) 的，前面的數字補 0；長度大於 380 的，截去前面的數字

```
from keras.preprocessing import sequence
from keras.preprocessing.text import Tokenizer

# create token
token = Tokenizer(num_words=hyper_params['dict_len'])
token.fit_on_texts(text_train.values)
x_train = token.texts_to_sequences(text_train.values)
x_train = sequence.pad_sequences(x_train, maxlen=hyper_params['content_len'])
x_test = token.texts_to_sequences(text_test.values)
x_test = sequence.pad_sequences(x_test, maxlen=hyper_params['content_len'])
```

2. Model Training

- 訓練兩種模型 `SimpleRNN` 和 `LSTM`，並將每個 Epoch 的 `training loss` 、 `training accuracy` 和 `testing accuracy` 記錄下來。
- Hyperparameters
  - epoch: 100
  - batch size: 4987
  - drop out: 0.7
  - optimizer: Adam (learning rate: 1e-3)

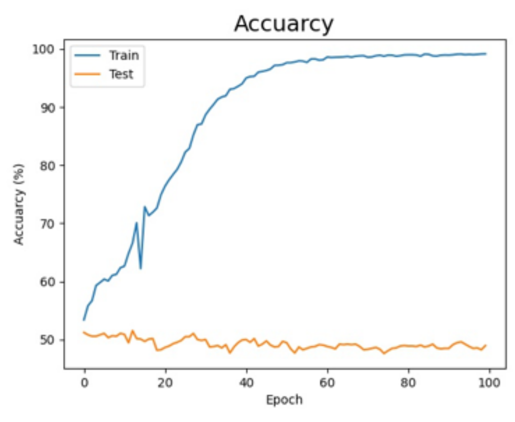
```
import keras
from keras.models import Sequential
from keras.layers.core import Dense, Dropout
from keras.layers.embeddings import Embedding
from keras.layers.recurrent import SimpleRNN, LSTM

hyper_params = {'epoch': 100, 'batch_size': 4987, 'drop_out': 0.7,
                'dict_len': 3800, 'content_len': 380}

def train(target_path, model_type, params, x_train, y_train, x_test, y_test):
    # set up model
    model = Sequential()
    model.add(Embedding(output_dim=128,
                        input_dim=params['dict_len'],
                        input_length=params['content_len']))
    model.add(Dropout(params['drop_out']))
    if model_type == 'RNN':
        model.add(SimpleRNN(units=128))
    else:
        model.add(LSTM(units=128))
    model.add(Dense(units=128, activation='relu'))
    model.add(Dropout(params['drop_out']))
    model.add(Dense(units=1, activation='sigmoid'))
    model.summary()

    # define model
    model.compile(loss='binary_crossentropy',
                  optimizer=keras.optimizers.Adam(learning_rate=1e-3),
                  metrics=['accuracy'])

    # training
    fp_train_loss = open(f'{target_path}{model_type}_train_loss.txt', 'w')
    fp_train_acc = open(f'{target_path}{model_type}_train_acc.txt', 'w')
    fp_test_acc = open(f'{target_path}{model_type}_test_acc.txt', 'w')

    for epoch_i in range(1, params['epoch'] + 1):
        train_history = model.fit(x_train, y_train,
                                  epochs=1, batch_size=params['batch_size'],
                                  shuffle=True, workers=12,
                                  verbose=0)
        train_loss = train_history.history["loss"][0]
        train_acc = train_history.history["accuracy"][0]
        print(f'\nEpoch {epoch_i}/{params["epoch"]}')
        print(f'Loss: {train_loss}, Accuracy: {train_acc}')
        fp_train_loss.write(f'{train_loss}\n')
        fp_train_acc.write(f'{train_acc}\n')

        score = model.evaluate(x_test, y_test, verbose=0, workers=12)
        print(f'Testing accuracy: {score[1]}')
        fp_test_acc.write(f'{score[1]}\n')

    fp_train_loss.close()
    fp_train_acc.close()
    fp_test_acc.close()
```

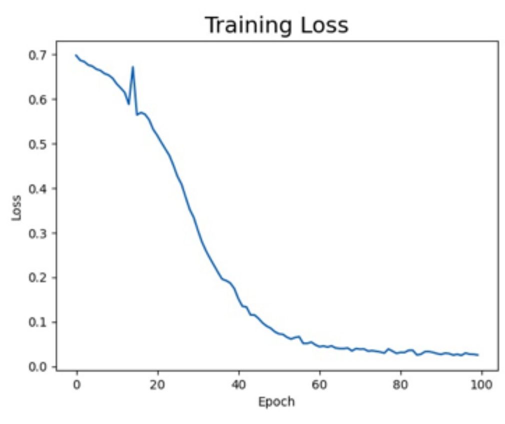
3. Results

最後雖然兩種模型 `training accuracy` 都有提升至將近百分百、 `loss` 也都趨近 0，但 `testing accuracy` 都維持在 50% 左右，有 overfitting 的現象。

- RNN

	Train	Test
Accuracy	99.16%	51.52%





- LSTM

	Train	Test
Accuracy	97.03%	50.80%