Natural Language Processing and Text Mining:

HW#2

小組成員和負責工作：

109590041 范遠皓 測試和報告

109590043 柯瑞霖 撰寫程式

環境

使用的語言：Python

所需套件：

```
click==8.1.3
colorama==0.4.6
joblib==1.2.0
nltk==3.8.1
numpy==1.24.2
pandas==1.5.3
python-dateutil==2.8.2
pytz==2023.2
regex==2023.3.23
scikit-learn==1.2.2
scipy==1.10.1
six==1.16.0
threadpoolctl==3.1.0
tqdm==4.65.0
```

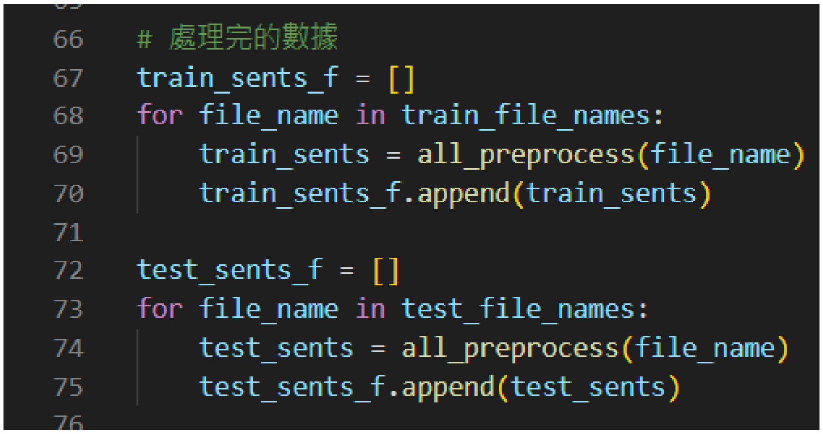
安裝辦法：

Zip 檔內有 requirements.txt

終端機輸入以下指令安裝

```
pip3 install -r requirements.txt
```

資料讀取並進行前置處理

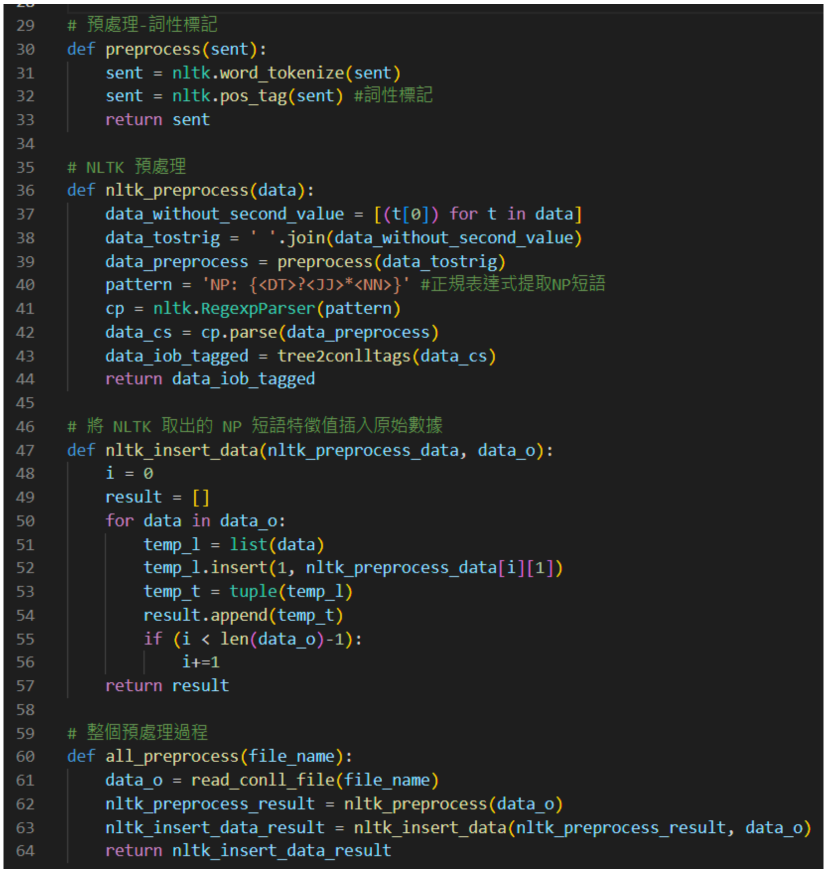


```python
# 處理完的數據
train_sents_f = []
for file_name in train_file_names:
    train_sents = all_preprocess(file_name)
    train_sents_f.append(train_sents)

test_sents_f = []
for file_name in test_file_names:
    test_sents = all_preprocess(file_name)
    test_sents_f.append(test_sents)
```

NLTK 前置處理:

對資料進行詞性標記，然後提取其中的名詞短語 (NP)，並將其特徵值插入原始資料中。



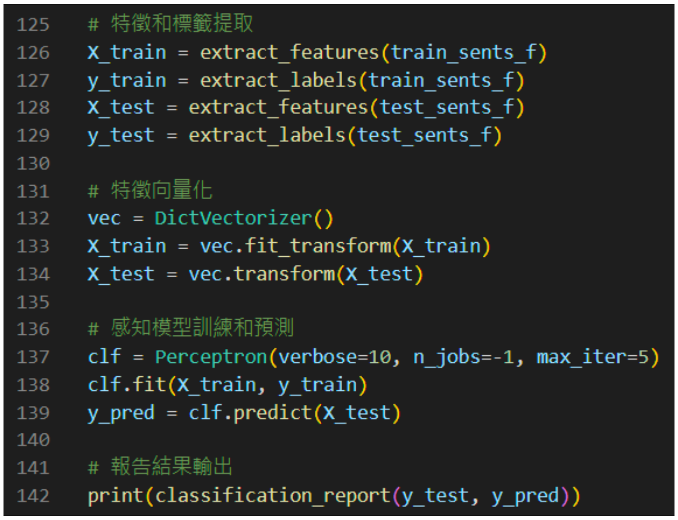
```python
# 預處理-詞性標記
def preprocess(sent):
    sent = nltk.word_tokenize(sent)
    sent = nltk.pos_tag(sent) #詞性標記
    return sent

# NLTK 預處理
def nltk_preprocess(data):
    data_without_second_value = [(t[0]) for t in data]
    data_tostrig = ' '.join(data_without_second_value)
    data_preprocess = preprocess(data_tostrig)
    pattern = 'NP: {<DT>?<JJ>*<NN>}' #正規表達式提取NP短語
    cp = nltk.RegexpParser(pattern)
    data_cs = cp.parse(data_preprocess)
    data_iob_tagged = tree2conlltags(data_cs)
    return data_iob_tagged

# 將 NLTK 取出的 NP 短語特徵值插入原始數據
def nltk_insert_data(nltk_preprocess_data, data_o):
    i = 0
    result = []
    for data in data_o:
        temp_l = list(data)
        temp_l.insert(1, nltk_preprocess_data[i][1])
        temp_t = tuple(temp_l)
        result.append(temp_t)
        if (i < len(data_o)-1):
            i+=1
    return result

# 整個預處理過程
def all_preprocess(file_name):
    data_o = read_conll_file(file_name)
    nltk_preprocess_result = nltk_preprocess(data_o)
    nltk_insert_data_result = nltk_insert_data(nltk_preprocess_result, data_o)
    return nltk_insert_data_result
```

提取特徵後放入 Perceptron 模型進行分類訓練，最後輸出結果。



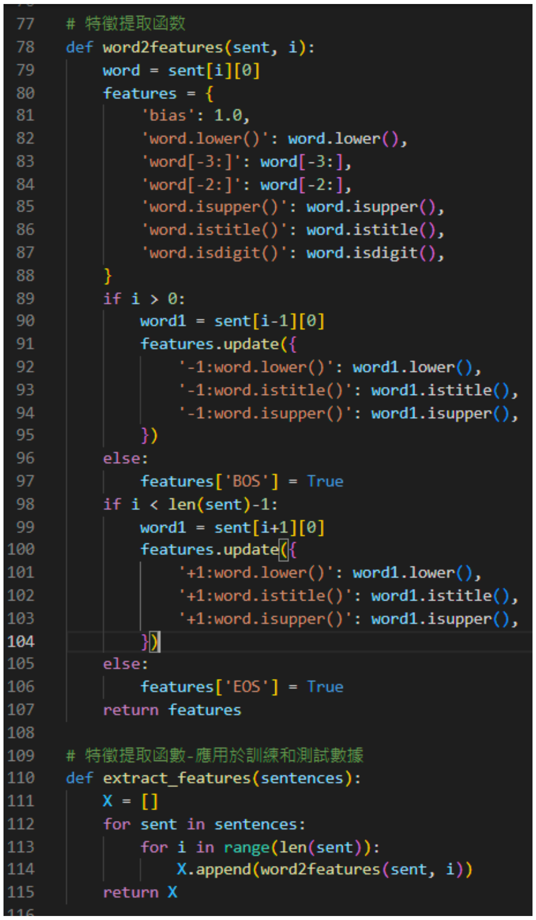
```python
# 特徵和標籤提取
X_train = extract_features(train_sents_f)
y_train = extract_labels(train_sents_f)
X_test = extract_features(test_sents_f)
y_test = extract_labels(test_sents_f)

# 特徵向量化
vec = DictVectorizer()
X_train = vec.fit_transform(X_train)
X_test = vec.transform(X_test)

# 感知模型訓練和預測
clf = Perceptron(verbose=10, n_jobs=-1, max_iter=5)
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)

# 報告結果輸出
print(classification_report(y_test, y_pred))
```

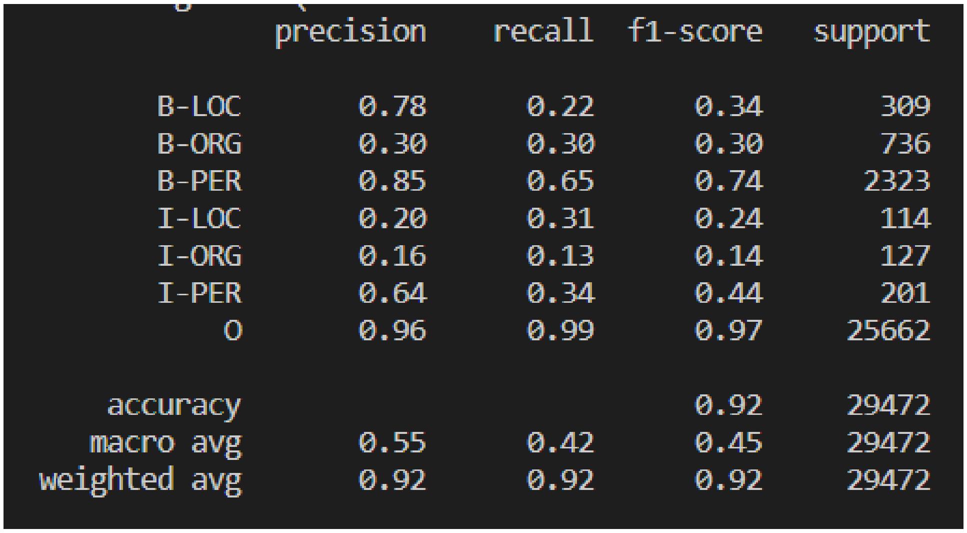
特徵提取



```python
# 特徵提取函数
def word2features(sent, i):
    word = sent[i][0]
    features = {
        'bias': 1.0,
        'word.lower()': word.lower(),
        'word[-3:]': word[-3:],
        'word[-2:]': word[-2:],
        'word.isupper()': word.isupper(),
        'word.istitle()': word.istitle(),
        'word.isdigit()': word.isdigit(),
    }
    if i > 0:
        word1 = sent[i-1][0]
        features.update({
            '-1:word.lower()': word1.lower(),
            '-1:word.istitle()': word1.istitle(),
            '-1:word.isupper()': word1.isupper(),
        })
    else:
        features['BOS'] = True
    if i < len(sent)-1:
        word1 = sent[i+1][0]
        features.update({
            '+1:word.lower()': word1.lower(),
            '+1:word.istitle()': word1.istitle(),
            '+1:word.isupper()': word1.isupper(),
        })
    else:
        features['EOS'] = True
    return features

# 特徵提取函數-應用於訓練和測試數據
def extract_features(sentences):
    X = []
    for sent in sentences:
        for i in range(len(sent)):
            X.append(word2features(sent, i))
    return X
```

執行結果：



```
              precision    recall  f1-score   support

       B-LOC       0.78      0.22      0.34       309
       B-ORG       0.30      0.30      0.30       736
       B-PER       0.85      0.65      0.74      2323
       I-LOC       0.20      0.31      0.24       114
       I-ORG       0.16      0.13      0.14       127
       I-PER       0.64      0.34      0.44       201
           O       0.96      0.99      0.97     25662

    accuracy                           0.92     29472
   macro avg       0.55      0.42      0.45     29472
weighted avg       0.92      0.92      0.92     29472
```