國立臺東專科學校
National Taitung Junior College

自動化自然語言國際船運港口辨識

指導教授:廖強棋
學生:梁子奕、何政岳、江文正、姚宇龍

摘要

我們的研究動機是為了解決國際船運港口名稱，因*自然語言*口語化稱呼間的不同，導致*國際船運港口名稱配對*無法相同的問題，於是我們先在Spyder內建立一個panda環境再將所需的套件設定安裝在讀取csv檔，接著將資料中的錯誤筆數刪除不列入收集，然後設定需收集的資料數後利用Rapidfuzz演算法進行模糊匹配並轉換結果，最後設定閾值就可以進行成功率計算，最終將結果儲存於csv檔，*來找出正確的標準國際船運港口名稱*。

壹、研究動機

自古以來水運就是最大貿易的通路之一，隨著現代科技越來越發達，讓水運的貿易量愈發頻繁，語言溝通就成為了其中一個麻煩。世界上有197個國家，雖然英文是國際通用語言，但本地的工作人員還是會使用較為口語化的說法去記錄港口，導致同一港口卻有不同名稱的窘態。所以我們收集了各地港口的*標準*英文名稱和*很多不同人使用的船運貨單港口名稱*來做辨識配對，希望能解決不同語言習慣在溝通上的困擾。藉由得到的這些港口資料，先是透過SPYDER來編輯所需程式後，分別使用模糊匹配來連結那些港口名稱，幫助船運公司理解各式港口名稱應對不同*自然語言*口語化稱謂，能夠省下更多時間。

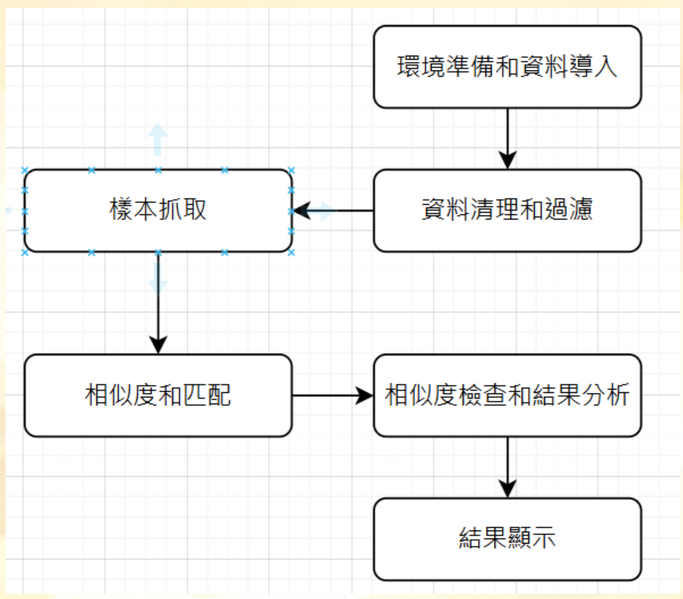
貳、匹配運用

將環境建立後，利用rapidfuzz演算法可以將獲得的樣本進行比對，接著設置隨機種子來保證結果能夠穩定，再將檔案中的空值和亂碼移除不列入比較當中，然後再控制每筆資料的大小和格式並確保每一筆資料都是唯一的，接著設定所需的樣本數去進行配對，最後等待資料累積完成，就可以將配對出來的匹配結果和正確率寫入csv檔中。

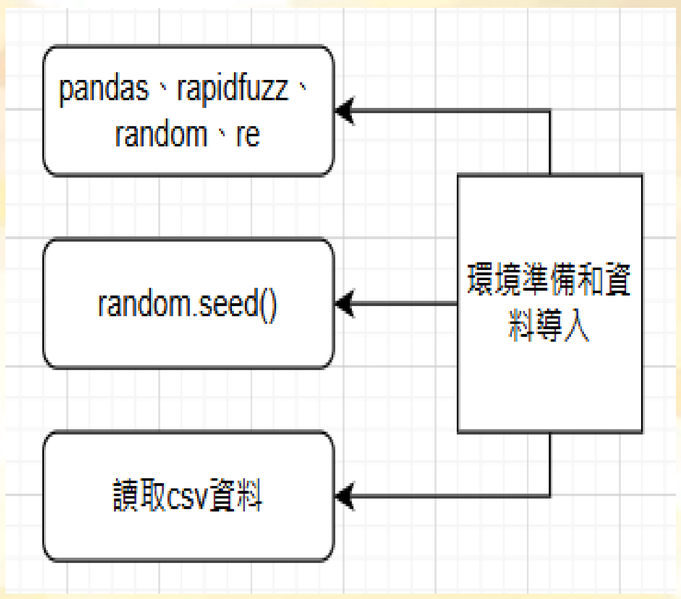
參、結果分析

一、環境準備和資料導入

1. 載入相關的套件和模組
   導入pandas(處理資料框)、rapidfuzz(模糊匹配)、random（隨機數生成）、re（正則表達式）等模組。
2. 設定隨機種子
   設置隨機種子數random.seed(50)的用意是為了確保每次運行的結果能一致，尤其是在抓取隨機選擇樣本。
3. 讀取csv文件
   分別從我們的unlocode_list_df和unlocode_mapping_df，抓取這兩筆資料的code和unlocode欄位資料。



圖一、專題流程圖



圖二、環境準備和資料導入示意圖

```
# 第一階段:環境準備和資料導入
# 設置隨機種子,以保證結果的穩定性
random.seed(50)

# 讀取CSV文件,並只保留unlocode和code欄位
unlocode_list_df = pd.read_csv(file2, encoding='utf-8', usecols=['code'])
unlocode_mapping_df = pd.read_csv(file1, encoding='latin1', usecols=['unlocode'])
```

圖三、導入資料程式碼

自動化自然語言國際船運港口辦識

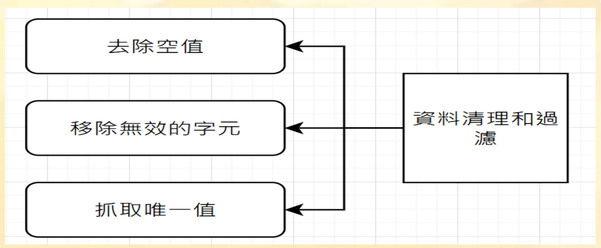
國立臺東專科學校
National Taitung Junior College

指導教授:廖強棋
學生:梁子奕、何政岳、江文正、姚宇龍

參、結果分析(續)

二、資料清理和過濾

1. 去除空值
   由於這兩份檔案中的一些資料比較殘缺需要進行欄位清理，我們對code和unlocode欄位進行清理，去除掉包含空值的行。
2. 移除無效的字符
   使用正則表達式^[A-Za-z0-9]+$，過濾掉非字母或數字的值，保留英文字母或數字的字串，若有空格或標點符號則不匹配，利用此方法可確保code和unlocode都是有效的字母和數字排列組合。
3. 抓取唯一值
   使用.unique()函數，抓取code和unlocode欄位的唯一值，使我們的資料抓取更直接，不會有重複性的問題。



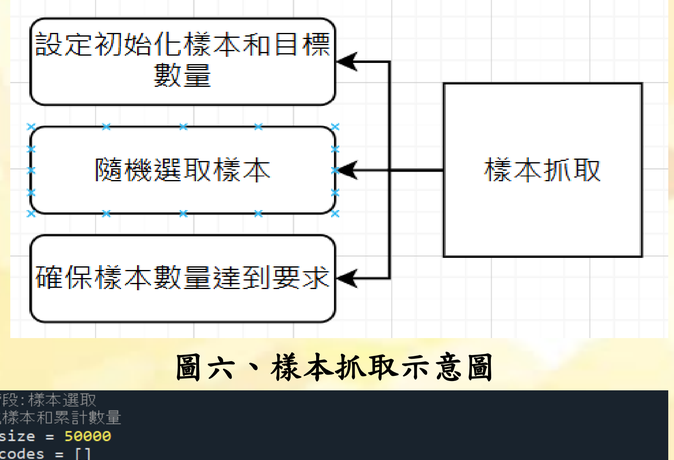
圖四、資料清理和過濾示意圖

```
# 第二階段:資料清理和過濾
# 去除空值
unlocode_list_df.dropna(subset=['code'], inplace=True)
unlocode_mapping_df.dropna(subset=['unlocode'], inplace=True)

# 移除含有非字母或數字字符的值
unlocode_list_df = unlocode_list_df[unlocode_list_df['code'].apply(lambda x: bool(re.match('^[A-Za-z0-9]+$', str(x))))]
unlocode_mapping_df = unlocode_mapping_df[unlocode_mapping_df['unlocode'].apply(lambda x: bool(re.match('^[A-Za-z0-9]+$', str(x))))]

# 提取唯一值
codes = unlocode_list_df['code'].unique()
unlocodes = unlocode_mapping_df['unlocode'].unique()
```

圖五、資料清理和過濾程式碼

三、樣本抓取

1. 設定初始化樣本和目標數量
   設定sample_size為100000，並初始化sample_codes和sample_unlocodes存儲當作符合條件的樣本元素。
2. 隨機選取樣本
   使用random.choice()從codes和unlocodes中隨機選取code_sample和unlocode_sample。篩選分析並檢查是否有無重複。
3. 確保樣本數量達到要求
   將樣本列表sample_code和sample_unlocodes丟到sample_size，來保證樣本的數量。



圖六、樣本抓取示意圖

```
# 第三階段:樣本選取
# 初始化樣本和累計數量
sample_size = 50000
sample_codes = []
sample_unlocodes = []

# 累積符合條件的樣本
while len(sample_codes) < sample_size and len(sample_unlocodes) < sample_size:
    # 隨機選取一個符合條件的code和unlocode
    code_sample = random.choice(codes)
    unlocode_sample = random.choice(unlocodes)

    # 檢查是否已經包含該樣本,以避免重複
    if code_sample not in sample_codes:
        sample_codes.append(code_sample)
    if unlocode_sample not in sample_unlocodes:
        sample_unlocodes.append(unlocode_sample)

# 確保樣本數量達到要求
sample_codes = sample_codes[:sample_size]
sample_unlocodes = sample_unlocodes[:sample_size]
```

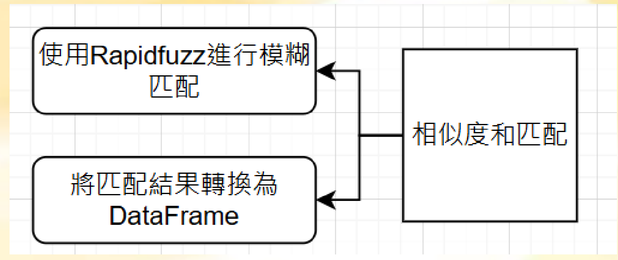
圖七、樣本抓取程式碼

致謝

感謝聖學科技股份有限公司（GoFreight）提供實際資料供我們研究分析，
聖學科技股份有限公司（GoFreight）介紹網址：
https://www.104.com.tw/company/1a2x6bj764

四、相似度和匹配

1. 使用 Rapidfuzz 進行模糊匹配
   一開始我們原本是使用fuzzywuzzy但由於抓去資料的效率和在面對大量字串匹配的效益太低，最後我們使用的是rapidfuzz這套演算法，可用來處理大量的資料和快速的在兩字串間進行快速配對，使用process.extractOne()，計算所有資料上的code和sample_unlocodes表格上最相似的unlocode，儲存相似度分數(similarity)和配對結果進行篩選和分析。
2. 將匹配結果轉換為 DataFrame
   將匹配結果轉成表格形式sample_matches_df，並命名欄位名稱（code, matched_unlocode, similarity, match_score）。



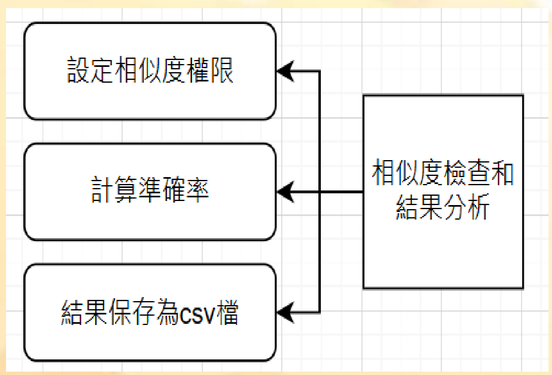
圖八、相似度和匹配示意圖

```
# 第四階段:相似度計算和匹配
# 使用 rapidfuzz 計算相似度,取最接近的匹配
sample_matches = [(code, *process.extractOne(code, sample_unlocodes, scorer=fuzz.ratio)) for code in sample_codes]

# 將結果轉換為DataFrame,並設置四個欄位名稱
sample_matches_df = pd.DataFrame(sample_matches, columns=['code', 'matched_unlocode', 'similarity', 'match_score'])
```

圖九、相似度和匹配程式碼

五、相似度檢查和結果分析

1. 設定相似度閾值
   設定相似度閾值threshold為80當作資料的標界，需要達到此資料的這個數值才會被認定是最具符合的資料。
2. 計算準確率
   這邊使用mean()函數來計算is_correct_match指定欄位和所有資料的平均值，當做準確率的轉換，其中包括布林值(true或false)，來表達每種匹配是否有達到預想的指定相似度閾值。
3. 結果保存為csv檔
   把sample_matches_df資料框演算完的所有資料轉換成csv文件，以當作結果參考。



圖十、相似度檢查和結果保存示意圖

```
# 第五階段:相似度檢查和結果分析
# 設定相似度閾值並計算正確匹配的比例
threshold = 80  # 使用更高的相似度閾值
sample_matches_df['is_correct_match'] = sample_matches_df['similarity'] >= threshold
sample_accuracy = sample_matches_df['is_correct_match'].mean()

# 將結果儲存為CSV
output_file = 'D:\\專題\\sample_matching_results.csv'
sample_matches_df.to_csv(output_file, index=False, encoding='utf-8')
```

圖十一、相似度檢查程式碼