

# Digital Signal Processing Final Project

## FALL SEMESTER 2019

教授：李琳山

組員：B05901011 許秉倫: 爬蟲, 畫embedding散佈圖

B05901082 楊晟甫: Embedding, Cosine Similarity, PRF, 實驗

Github Link: [https://github.com/joeyy5588/DSP2019SPRING/tree/master/dsp\\_final](https://github.com/joeyy5588/DSP2019SPRING/tree/master/dsp_final)

---

## Background

每每到了選課的季節，我們總是會上台大課程網搜尋下學期有興趣修的課，然而，因為課程網的搜尋方式是用逐字比對的，因此，我們往往搜尋不到真正「相關」的課程。

我們登入台大課程網後，搜尋「人工智慧」的結果如下：

人工智慧在臨床資料的分析與應用	人工智慧與心理學書報討論三	人工智慧及深度機器學習之生醫藥產業應用
人工智慧神經網路模型應用	高效能人工智慧系統	人工智慧與法律專題研究二

雖然每堂課真的有包含「人工智慧」這個詞，但與「人工智慧」相關的其他課程諸如「機器學習」、「深度學習」等等，卻一項都沒有出現。

因此，我們的目標就是要用課堂所學的方法做出一個更好的搜尋系統！

## Overview

在開始之前，我們設計了我們的實作流程

1. 登入台大課程網，將所有課程（包含名稱、課號等等）爬取下來，整理為表格。
2. 嘗試使用不同Embedding方法，看何者能對課名有最好的分群（評斷標準為，分群結果是否有和系所符合），最後選用效果最好的分群方法。
3. 使用基本的cosine similarity，取出最大的similarity。

4. 使用PRF(pseudo relevance feedback)做query，取出最符合的項目。
5. 結果分析與討論。

## Step 1. 爬下所有課程資料

```
# 查詢參數
my_params = {
    'current_sem': '107-2',
    'cstype': 1,
    'alltime': 'yes',
    'allproced': 'yes',
    'allsel': 'yes',
    'startrec': 10000,
    'page_cnt': 2500,
    'Submit22': '%Acid%B8%DF',
}

# 將查詢參數加入 GET 請求中
r = requests.post('http://nol2.aca.ntu.edu.tw/nol/coursesearch/search_result.php', params = my_params)
```

我們使用Http request中的post method向課程網發送請求，課程網就會將一串串課程資料利用html的格式回傳給我們，

```
<tr align="center"><td>96264/</td><td></td><td>TA10320398/</td><td>34/</td><td></td><td><a TARGET="blank" HREF="print_t
13178
13179
13180
13181
13182
13183
13184
13185
13186
13187
13188
13189
13190
13191
13192
13193
13194
13195
13196
13197
13198
13199
13200
13201
13202
13203
13204
13205
13206
13207
13208
13209
13210
13211
</tr align="center"><td>96266/</td><td></td><td>TA10320399/</td><td>35/</td><td></td><td><a TARGET="blank" HREF="print_t
13182
13183
13184
13185
13186
13187
13188
13189
13190
13191
13192
13193
13194
13195
13196
13197
13198
13199
13200
13201
13202
13203
13204
13205
13206
13207
13208
13209
13210
13211
</tr align="center"><td>96462/</td><td></td><td>TA10320399/</td><td>48/</td><td></td><td><a TARGET="blank" HREF="print_t
13182
13183
13184
13185
13186
13187
13188
13189
13190
13191
13192
13193
13194
13195
13196
13197
13198
13199
13200
13201
13202
13203
13204
13205
13206
13207
13208
13209
13210
13211
</tr align="center"><td>96279/</td><td></td><td>TA10320400/</td><td>51/</td><td></td><td><a TARGET="blank" HREF="print_t
13183
13184
13185
13186
13187
13188
13189
13190
13191
13192
13193
13194
13195
13196
13197
13198
13199
13200
13201
13202
13203
13204
13205
13206
13207
13208
13209
13210
13211
</tr align="center"><td>96444/</td><td></td><td>TA10320401/</td><td>56/</td><td></td><td><a TARGET="blank" HREF="print_t
13186
13187
13188
13189
13190
13191
13192
13193
13194
13195
13196
13197
13198
13199
13200
13201
13202
13203
13204
13205
13206
13207
13208
13209
13210
13211
</tr align="center"><td>96465/</td><td></td><td>TA10320401/</td><td>58/</td><td></td><td><a TARGET="blank" HREF="print_t
13187
13188
13189
13190
13191
13192
13193
13194
13195
13196
13197
13198
13199
13200
13201
13202
13203
13204
13205
13206
13207
13208
13209
13210
13211
</tr align="center"><td>96266/</td><td></td><td>TA10320402/</td><td>31/</td><td></td><td><a TARGET="blank" HREF="print_t
13188
13189
13190
13191
13192
13193
13194
13195
13196
13197
13198
13199
13200
13201
13202
13203
13204
13205
13206
13207
13208
13209
13210
13211
</tr align="center"><td>96463/</td><td></td><td>TA10320402/</td><td>41/</td><td></td><td><a TARGET="blank" HREF="print_t
13190
13191
13192
13193
13194
13195
13196
13197
13198
13199
13200
13201
13202
13203
13204
13205
13206
13207
13208
13209
13210
13211
</tr align="center"><td>96428/</td><td></td><td>TA10320403/</td><td>17/</td><td></td><td><a TARGET="blank" HREF="print_t
13191
13192
13193
13194
13195
13196
13197
13198
13199
13200
13201
13202
13203
13204
13205
13206
13207
13208
13209
13210
13211
</tr align="center"><td>96433/</td><td></td><td>TA10320403/</td><td>18/</td><td></td><td><a TARGET="blank" HREF="print_t
13192
13193
13194
13195
13196
13197
13198
13199
13200
13201
13202
13203
13204
13205
13206
13207
13208
13209
13210
13211
</tr align="center"><td>96498/</td><td></td><td>TA10320403/</td><td>19/</td><td></td><td><a TARGET="blank" HREF="print_t
13193
13194
13195
13196
13197
13198
13199
13200
13201
13202
13203
13204
13205
13206
13207
13208
13209
13210
13211
</tr align="center"><td>96720/</td><td></td><td>TA10320403/</td><td>20/</td><td></td><td><a TARGET="blank" HREF="print_t
13194
13195
13196
13197
13198
13199
13200
13201
13202
13203
13204
13205
13206
13207
13208
13209
13210
13211
</tr align="center"><td>96712/</td><td></td><td>TA10320403/</td><td>21/</td><td></td><td><a TARGET="blank" HREF="print_t
13195
13196
13197
13198
13199
13200
13201
13202
13203
13204
13205
13206
13207
13208
13209
13210
13211
</tr align="center"><td>96722/</td><td></td><td>TA10320403/</td><td>22/</td><td></td><td><a TARGET="blank" HREF="print_t
13196
13197
13198
13199
13200
13201
13202
13203
13204
13205
13206
13207
13208
13209
13210
13211
</tr align="center"><td>96723/</td><td></td><td>TA10320403/</td><td>23/</td><td></td><td><a TARGET="blank" HREF="print_t
13197
13198
13199
13200
13201
13202
13203
13204
13205
13206
13207
13208
13209
13210
13211
</tr align="center"><td>96724/</td><td></td><td>TA10320403/</td><td>24/</td><td></td><td><a TARGET="blank" HREF="print_t
13198
13199
13200
13201
13202
13203
13204
13205
13206
13207
13208
13209
13210
13211
</tr align="center"><td>96725/</td><td></td><td>TA10320403/</td><td>25/</td><td></td><td><a TARGET="blank" HREF="print_t
13199
13200
13201
13202
13203
13204
13205
13206
13207
13208
13209
13210
13211
</tr align="center"><td>96726/</td><td></td><td>TA10320403/</td><td>26/</td><td></td><td><a TARGET="blank" HREF="print_t
13200
13201
13202
13203
13204
13205
13206
13207
13208
13209
13210
13211
</tr align="center"><td>96804/</td><td></td><td>TA10620098/</td><td>01/</td><td></td><td><a TARGET="blank" HREF="print_t
13201
13202
13203
13204
13205
13206
13207
13208
13209
13210
13211
</tr align="center"><td>98079/</td><td></td><td>TA10728806/</td><td>16/</td><td></td><td><a TARGET="blank" HREF="print_t
13202
13203
13204
13205
13206
13207
13208
13209
13210
13211
</tr align="center"><td>96016/</td><td></td><td>TA10311697/</td><td>17/</td><td></td><td><a TARGET="blank" HREF="print_t
13203
13204
13205
13206
13207
13208
13209
13210
13211
</tr align="center"><td>96164/</td><td></td><td>TA10320415/</td><td>02/</td><td></td><td><a TARGET="blank" HREF="print_t
13204
13205
13206
13207
13208
13209
13210
13211
</tr align="center"><td>96166/</td><td></td><td>TA10320419/</td><td>05/</td><td></td><td><a TARGET="blank" HREF="print_t
13205
13206
13207
13208
13209
13210
13211
</tr align="center"><td>96368/</td><td></td><td>TA10720181/</td><td>04/</td><td></td><td><a TARGET="blank" HREF="print_t
13207
13208
13209
13210
13211
</tr align="center"><td>96019/</td><td></td><td>TA10720102/</td><td>06/</td><td></td><td><a TARGET="blank" HREF="print_t
13208
13209
13210
13211
</tr align="center"><td>96019/</td><td></td><td>TA10320421/</td><td>06/</td><td></td><td><a TARGET="blank" HREF="print_t
13209
13210
13211
</tr align="center"><td>96334/</td><td></td><td>TA10320422/</td><td>08/</td><td></td><td><a TARGET="blank" HREF="print_t
13210
13211
</tr align="center"><td>98047/</td><td></td><td>TA10320423/</td><td>19/</td><td></td><td><a TARGET="blank" HREF="print_t
13211
```

資料非常之雜亂，因此我們使用“beautiful soup”這個套件，對html做parsing，最終取出課名和課號。

整理過後的資料會是這種形式，課名對應到課號。

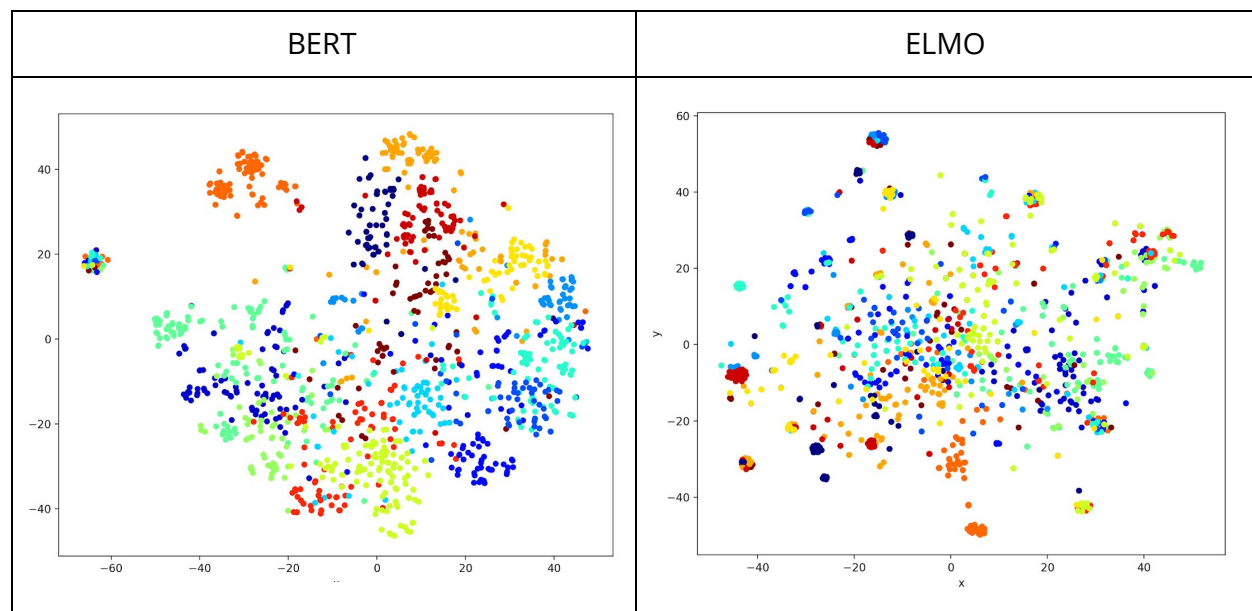
```
{  
  '國文領域': 'Common1012',  
  '外文領域下': 'Common1010',  
  '英文領域下': 'Common1004',  
  '適應體育(生理班)': 'PE1007',  
  '適應體育(肢體班)': 'PE1005'  
}
```

## Step 2. 選取最佳Embedding

我們將課名經過“BERT”和“ELMO”兩種方式Embedding，使用「系所」當做label。

使用TSNE將結果畫出，每種顏色對應的是一個系所，經觀察比較過後我們認為BERT在我們的task中，較能將資料區分開，具有較強的分群能力（課程名稱的embedding有較高的semantic information）。

因此，接下來我們會使用BERT繼續實作！



---

## Step 3. 使用cosine similarity 做query

```
query = "人工智慧"
bc = BertClient()
vector = bc.encode([query])

rank_list = []
for key in embedding_dict:
    rank_list.append([embedding_dict[key]])
rank_list = np.array(rank_list).squeeze()

top_k_index, bottom_k_index = topK(vector, rank_list, K)

top_result, bottom_result = topK_result(top_k_index, bottom_k_index, K)
```

我們將所有課程名稱以及query text轉成word embedding，將query embedding與course name embedding做cosine similarity 的計算，並由大到小排序，並顯示top-K個以及bottom-K的搜尋結果

```
def topK(vector, rank_list, K):
    dot = np.dot(rank_list, vector.transpose())
    norma = np.linalg.norm(vector)
    normb = np.linalg.norm(rank_list, axis = 1)
    cos_sim = dot.flatten() / (norma * normb)
    cos_sim_rank = np.flip(np.argsort(cos_sim))

    top_k_index = cos_sim_rank[:K]
    bottom_k_index = cos_sim_rank[-K:]
    return top_k_index, bottom_k_index

def topK_result(top_k_index, bottom_k_index, K):
    original_dict = load_obj('course')
    top_result = [0] * K
    bottom_result = [0] * K
    for i, (k, v) in enumerate(original_dict.items()):
        if i in top_k_index:
            top_result[np.where(top_k_index == i)[0][0]] = v
        elif i in bottom_k_index:
            bottom_result[np.where(bottom_k_index == i)[0][0]] = v
    return top_result, bottom_result
```

---

在這個階段中，我們使用"高雄發大財"作為query，搜尋到的結果依序為：公司理財一、現代農業體驗一、臺灣農業、資管專題一、社會痛苦、國際農業體驗二、現代農業體驗二、燈光技術一、集水區經營、阿美語一下

## Step 4. 使用PRF 優化

PRF的流程大致可以分為以下四步：

1. 計算cosine similarity並對document(in our case, course name embedding) 做排序
2. 取出top K 個並假定其是跟我們的query相關的，而bottom K個則是無關的
3. 使用relevance feedback，我們使用的是Rocchio Algorithm，其公式如圖：

Note:  $q_0$ 是原query， $D_r$ 為relevant set， $D_{nr}$ 為irrelevant set，alpha, beta, gamma為參數，我們分別將其設定為1, 0.75, 0.15

$$\vec{q}_m = \alpha \vec{q}_0 + \beta \frac{1}{|D_r|} \sum_{\vec{d}_j \in D_r} \vec{d}_j - \gamma \frac{1}{|D_{nr}|} \sum_{\vec{d}_j \in D_{nr}} \vec{d}_j$$

4. 使用feedback 修正我們的query，並重複以上1~4步驟

```
def PRF(query, rank_list, top_k_index, bottom_k_index, K):
    alpha = 1
    beta = 0.75
    gamma = 0.15
    rel = np.sum(rank_list[top_k_index], axis = 0) / K
    irrel = np.sum(rank_list[bottom_k_index], axis = 0) / K
    query = (alpha * query + beta * rel - gamma * irrel) / (alpha + beta - gamma)
    return query
```

---

## Step 5. 結果與討論

在使用word embedding 前，學校的系統是無法查到字沒有完全一樣的結果的，透過word embedding，我們可以搜尋出相關的課程，再透過prf做進一步的優化，可以發現在embedding的搜尋排序中，有完全相同的字的，分數也會較高，但是經過prf後，此一現象便消失。

	Original	Only cosine similarity	With PRF
Top 1	人工智慧在臨床資料的分析與應用	人工智慧	人工智慧
Top 2	人工智慧及深度機器學習之生醫藥產業應用	機器學習	深度學習於電腦視覺
Top 3	人工智慧神經網路模型應用	高效能人工智慧系統	機器學習
Top 4	人工智慧	機器人視覺	先進機器人感測與控制
Top 5	高效能人工智慧系統	社會機器人	機器人視覺
Top 6	人工智慧與法律專題研究	深度學習於電腦視覺	高效能人工智慧系統
Top 7	人工智慧與管理創新	行動學習	機器人控制
Top 8	None	機器人控制	行動學習
Top 9	None	人工智慧神經網路模型應用	人工智慧神經網路模型應用
Top 10	None	先進機器人感測與控制	社會機器人

在PRF algorithm 中，Top-K的K值與iteration的次數是task-dependent的參數，為此，我們做了一些實驗來決定我們的模型最終要用什麼K值以及iteration次數。

### Varying K value, fix iteration number(i = 10)

(see next page)

	K=10	K=30	K=50
Top 1	人工智慧	數位學習概論	資訊理論與編碼技巧
Top 2	深度學習於電腦視覺	資訊理論與編碼技巧	數位學習概論
Top 3	機器學習	數位系統與實驗	資訊計量學導論
Top 4	先進機器人感測與控制	資料科學程式設計	資料科學程式設計
Top 5	機器人視覺	多媒體設計與評估	量子計算與資訊導論
Top 6	高效能人工智慧系統	數位人文概論	數位系統與實驗
Top 7	機器人控制	數位系統設計	電子計算機概論
Top 8	行動學習	數位語音處理概論	多媒體設計與評估
Top 9	人工智慧神經網路模型應用	進階生物晶片操作與資料分析	資料結構與演算法
Top 10	社會機器人	數位邏輯實驗	數位人文概論

我們可以發現，隨著K值越來越大，與我們的query“人工智慧”完全相符的結果已經越來越少，這跟我們在網路上查到的結果相符，prf容易造成query drift的現象(查詢目標偏移)，所以我們選用k值=10

### Varying iteration number, fix K value (K = 10)

	K=10	K=30	K=50
Top 1	人工智慧	人工智慧	人工智慧
Top 2	深度學習於電腦視覺	深度學習於電腦視覺	深度學習於電腦視覺
Top 3	機器學習	先進機器人感測與控制	先進機器人感測與控制
Top 4	先進機器人感測與控制	機器學習	機器學習

---

Top 5	機器人視覺	機器人視覺	機器人視覺
Top 6	高效能人工智慧系統	機器人控制	機器人控制
Top 7	機器人控制	高效能人工智慧系統	高效能人工智慧系統
Top 8	行動學習	行動學習	行動學習
Top 9	人工智慧神經網路模型應用	人工智慧神經網路模型應用	人工智慧神經網路模型應用
Top 10	社會機器人	社會機器人	社會機器人

我們可以發現iteration number對於結果的影響較小，這是可以預期的，因為top-K的選取會影響最後query vector收斂的位置，iteration number並不會造成整個搜尋結果的偏移。

## Vision

這個final project 其實還蠻好玩的，我們後來有一個發想是利用課程概述，先將其作text summarization，再把它當成我們的query feature，或著是以同學的feedback來做搜尋，但礙於並不是每堂課都有課程概述或是充足的feedback，我們最後選用了課名來當作搜尋的基準，也許日後可以再對這個PRF模型進行改進。

## How to execute our code

1. 爬蟲部分

```
$ cd scraper/  
$ python3 scrape.py
```

2. PRF部分：先開啟一個cmd視窗呼叫bert server，再開啟另一個視窗執行search.py

```
$ bert-serving-start -model_dir chinese_L-12_H-768_A-12/  
$ python3 search.py
```