

通識課程問卷調查

TKUGERS

摘要

現有問卷製作平台僅能增加選項或做簡單的顯示不同選項，但未有串接 API 的功能讓我們可以彈性的去增加問卷的功能性。我們因此製作了一個自己的「通識課程問卷」平台，目的在於收集學生選修過的通識課並請他們評分，再將其資料透過先前訓練好的深度學習推薦系統模型推薦出新的課程，再請學生們從五門課程中選他們有興趣或是有可能會選修的。藉著此方法，若學生選的課程是我們推薦的表示我們此課程推薦系統是有一定效益的。我們將其平台命名為 **TKUGERS (TKU General Education Recommendation System**，淡江大學通識課程推薦系統)。

目錄

摘要	1
目錄	2
動機	3
背景知識	4
介面流程	6
問題排除	6
小巧思	16
心得與結論	17
未來展望	19
參考資料	20

動機

在大學的生涯中一定會有修通識課的經驗，如何選到一門適合自己或是感興趣的通識課是學生關注的議題。身為學生的我們，在面對五花八門的通識課時，多半都是先看自己有興趣的課程名稱或是先上社群網站去尋找推薦。若有一推薦功能像現今電商平台，可以根據過去購物過的經驗及評分推薦新商品那樣，能夠推薦課程提供選擇多樣性及便利性，肯定會大受學生喜愛。此外，若能夠讓學生選擇到自己較喜歡的課程也能讓授課老師上起課來更有收穫。

這次的期末作品是延續閔立中同學所負責的高教深耕計畫，該計畫全名為「淡江大學 109 年度『教師跨領域研究社群』」。此研究計畫旨在透過人工智慧分析與深度學習的優點，進行教師教學與學生學習資料的分析，提供教師教學策略的推薦，讓教師在實際教學能掌握學生學習狀況，運用適切的教學策略與方法，增進教學效果。就本研究計畫的參與老師們來說，許多老師常會從自身的教學經驗來決定合適的教學策略，然而，當遇到新開設課程或是不同背景的學生時，就需要再經歷嘗試錯誤、累積教學經驗的階段。因此，若能夠過 AI 推薦系統的研發，提供教師教學策略與方法建議，不僅能提升學生學習成果，也能增強教師教學效能。

經過學期初的研究與分析，掌握了初步的教師教學策略與學生

學習資料，也經由多次的修正問卷，再蒐集資料，讓問卷更加精緻。但仍缺乏足夠的大樣本資料，以及學生部分的中介變項資料。前次的探究多針對教師的教學部分，此次研究希望聚焦在學生學習導向，分析學生學習型態與策略，結合教師與學生資料，希望能夠獲得更精準的預測效果。

因既有的問卷製作平台及網站無法根據使用者的回應更改後續的問題或者是能夠串接自己的 API 等功能，故我們開發了自己的問卷網站來滿足我們的需求。除了網站外，也為了此課程開發出了一個 Android 應用程式來增加使用者的便利性。

背景知識

推薦系統是網際網路時代的一種資訊檢索工具，自上世紀 90 年代起，人們便認識到了推薦系統的價值，經過了二十多年的積累和沉澱，推薦系統逐漸成為一門獨立的學科在學術研究和業界應用中都取得了很多成果。1994 年明尼蘇達大學 GroupLens 研究組推出第一個自動化推薦系統 GroupLens^[1]。提出了將協同過濾作為推薦系統的重要技術，這也是最早的自動化協同過濾推薦系統之一。1997 年 Resnick 等人^[2]首次提出推薦系統（recommender system，RS）一詞，自此，推薦系統一詞被廣泛引用，並且推薦系統開始成為一

個重要的研究領域。1998 年亞馬遜 (Amazon.com) 上線了基於物品的協同過濾演算法，將推薦系統推向服務千萬級使用者和處理百萬級商品的規模，並能產生質量良好的推薦。2003 年亞馬遜的 Linden 等人發表論文，公佈了基於物品的協同過濾演算法^[3]，據統計推薦系統的貢獻率在 20%~30% 之間^[4]。2005 年 Adomavicius 等人的綜述論文^[5] 將推薦系統分為 3 個主要類別，即基於內容的推薦、基於協同過濾的推薦和混合推薦的方法，並提出了未來可能的主要研究方向。2006 年 10 月，北美線上視訊服務提供商 Netflix 宣佈了一項競賽，任何人只要能夠將它現有電影推薦演算法 Cinematch 的預測準確度提高 10%，就能獲得 100 萬美元的獎金。該比賽在學術界和工業界引起了較大的關注，參賽者提出了若干推薦演算法，提高推薦準確度，極大地推動了推薦系統的發展。2007 年第一屆 ACM 推薦系統大會在美國舉行，到 2017 年已經是第 11 屆。這是推薦系統領域的頂級會議，提供了一個重要的國際論壇來展示推薦系統在不同領域的最近研究成果、系統和方法。2016 年，YouTube 發表論文^[6]，將深度神經網路應用推薦系統中，實現了從大規模可選的推薦內容中找到最有可能的推薦結果。近年來，推薦系統被廣泛的應用於電子商務推薦、個性化廣告推薦、新聞推薦等諸多領域。在課程上的推薦雖少，但許多知名的線上課程平台都有課程推薦的系統。



介面流程

頁面一

原先以 Adobe XD 設計的初稿	最終以 Android Studio 設計的 UI
	
說明	此頁面是告知學生此問卷的目的及使用方法
使用元件	<ul style="list-style-type: none"> (1) ToolBar (2) TextView (3) Material Floating Button

頁面二

原先以 Adobe XD 設計的初稿	最終以 Android Studio 設計的 UI
	
說明	<p>此頁面是讓學生選自己選修過的通識課，可透過分頁切換不同領域。原先設計是做 9 個同樣的頁面，而新的設計善用了 Android 分頁元件的功能，將其 9 個頁面整併成一頁</p>
使用元件	<div> <div>(1) ToolBar</div> <div>(2) Material Floating Button</div> <div>(3) TabLayout</div> <div>(4) ViewPager</div> <div>(5) ScrollView</div> <div>(6) RecyclerView</div> <div>(7) CardView</div> <div>(8) TextView</div> <div>(9) ImageView</div> </div>

原先以 Adobe XD 設計的初稿	最終以 Android Studio 設計的 UI
	
說明	此頁面是請學生針對他們剛選的課程去做評分
使用元件	<ul style="list-style-type: none"> (1) ToolBar (2) Material Floating Button (3) RecyclerView (4) CardView (5) TextView (6) ImageView (7) AppCompatRatingBar

頁面四

原先以 Adobe XD 設計的初稿	最終以 Android Studio 設計的 UI
	
說明	此頁面是講解接下來的頁面用處
使用元件	<ul style="list-style-type: none"> (1) ToolBar (2) TextView (3) Material Floating Button

原先以 Adobe XD 設計的初稿	最終以 Android Studio 設計的 UI						
							
說明	<p>此頁面會透過學生剛選的通識課透過一套深度學習推薦系統將新的課程及其他四門隨機產生的課程回傳給學生選取，其課程順序亦隨機排列。待學生選完後，會回傳至伺服器做準確度的採計及其他資料上的分析</p>						
使用元件	<table border="0"> <tr> <td>(1) ToolBar</td><td>(4) CardView</td></tr> <tr> <td>(2) Material Floating Button</td><td>(5) TextView</td></tr> <tr> <td>(3) RecyclerView</td><td>(6) ImageView</td></tr> </table>	(1) ToolBar	(4) CardView	(2) Material Floating Button	(5) TextView	(3) RecyclerView	(6) ImageView
(1) ToolBar	(4) CardView						
(2) Material Floating Button	(5) TextView						
(3) RecyclerView	(6) ImageView						

頁面六

原先以 Adobe XD 設計的初稿	最終以 Android Studio 設計的 UI
	
說明	此頁面顯示問卷填寫完畢，並會顯示一個短暫的 Toast 表示「資料已傳至伺服器供分析，謝謝！」
使用元件	<ul style="list-style-type: none"> (1) ToolBar (2) TextView (3) Material Floating Button (4) Toast

問題排除

目前我們進度大概是維持在每周 implement 一個 feature 這樣的速率，從 12/27 建立起最初的 commit 後記載了共 5 個問題、實際解決了 4 個，詳情如下：

1. 取得 API 回傳的推薦課程

在經過資料的搜索及比較後決定使用 Retrofit 2.0，理由為較輕量、易寫、及完全支援 POST 請求。在處理 API 伺服器的非同步請求時，為了不需等待收到回傳資料而凍住 UI，應將 Retrofit 搭配著 Kotlin Coroutine 使用才是最佳實務。

```
fab.setOnClickListener {  
    val request = APIRequest(  
        ...使用者所評分的資料...  
    )  
    modifyRatingPage(request)  
    navController.navigate(R.id.action_surveyingPage_to_ratingPage)  
}
```

SurveyPage.kt

```
private fun modifyRatingPage(request: APIRequest) {  
    scope.launch {  
        ...發送 API 請求及接收回應...  
    }  
}
```

SurveyPage.kt

2. 將回傳的 API response 填到推薦清單

+

3. 將資料在 fragment 之間互傳

在 API 請求及回應的處理上，我們也是經比較後決定用 Moshi 這個函式庫來處理，這兩個函式庫作者是同一個人，作者在網路上列出了 10 項為何要選 Moshi 而非 GSON^[1]，其他參考網站也提供了很好的見解^[2]。另外，在 fragment 之間分享資料這件事困擾我們很久，是唯一打破了我們速率的困難點，拖了大概兩天半才找到合適的方法(也就是用 SharedViewModel)^[3]並實作它。 *1 New Feature/Day*

將學生的評分資料利用 Retrofit 2.0 傳送至 API 伺服器並取得回應，利用 Moshi 將回應 body 內的 json 轉成 Kotlin 類別，儲存在 SharedViewModel 裡

```
object ServiceBuilder {  
    ...  
    private val apiClient: Retrofit = Retrofit.Builder()  
        .baseUrl(...API 端點...)  
        .client(getHttpClient())  
        .addConverterFactory(MoshiConverterFactory.create())  
        .build()  
    ...  
}
```

利用 Moshi 處理請求及回應的 json

ServiceBuilder.kt

```
private fun modifyRatingPage(request: APIRequest) {  
    ...  
    val result: APIResponse  
    try {  
        val retrofit = ServiceBuilder.buildService(APIClient::class.java)  
        val response = retrofit.getRecommendedCourses(request)  
        result = response.body()!!  
        model = ViewModelProvider(requireActivity()).get(SharedViewModel::class.java)  
        model.saveResponse(result) → 將 API 回應儲存到 ShareViewModel  
    } catch (e: Exception) {  
        ...  
    }  
    ...  
}
```

SurveyingPage.kt

將儲存在 SharedViewModel 裡的 API 回應內容取出來， 並更新推薦頁面 CardView 上的課程資訊

ListAdapter 第一個參數需要是 List<Course>故需要轉換資料型態

取出儲存在 ShareViewModel
裡的 API 回應

```
val model = ViewModelProvider(requireActivity()).get(SharedViewModel::class.java)
val madapter = ListAdapter(apiResponse2Course(model.response.answers!),
    ...要 inflate 的 Layout 檔...,
    ...標題 TextView ID...,
    ...授課教師 TextView ID...,
    ...該 Layout 的 CardView ID... )
```

此段程式碼負責將 UI 填到畫面上

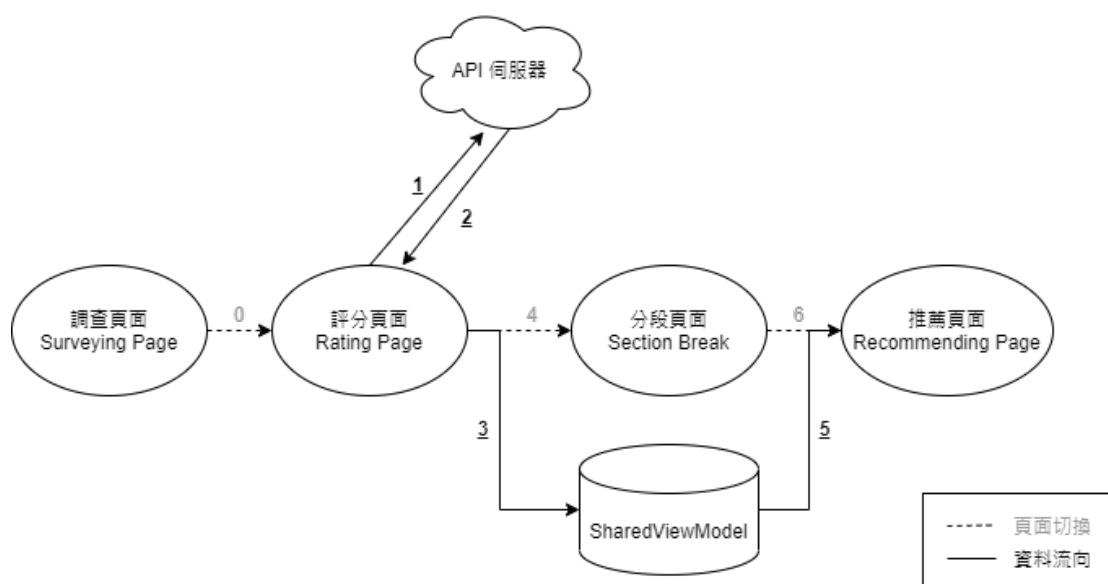
RecommendingPage.kt

```
private fun apiResponse2Course(courses: List<String>): List<Course> {
    val res = mutableListOf<Course>()
    for(course_name in courses) {
        val course = Course(
            title = course_name,
            instructor = course2instructor(course_name),
            isSelected = false
        )
        res.add(course)
    }
    return res.toList()
}
```

此段程式碼負責將 List<String>轉成 List<Course>

RecommendingPage.kt

流程圖如下：



4. 將 RecyclerView 的每個 Item 可點選

經一番查找才知道，原來 Google 在 v7 的小工具裡將原有的 ListView 取代成更加彈性的 RecyclerView，不過也同是將原有的 ItemOnClick 方法拿掉，也就是說使用者必須自己去實現 RecyclerView 的點選事件。這個問題我們現在是暫時在其 ViewHolder 裡將 onClickListener 加在每一個 CardView 上，不過我們認為這也是一個暫時解而已。目前只實現了點選功能，尚無在切換分頁後能保留選項以及在按下一頁時能將所選之選項收集起來的功能。

5. 每個分頁都加上 RecyclerView

因為我們有 11 個學門也就是有 11 個分頁，為避免專案檔案過大，我們盡可能的讓 layout 可以重複立用，讓每一個分頁帶入不同資料但是 inflate 相同 layout。

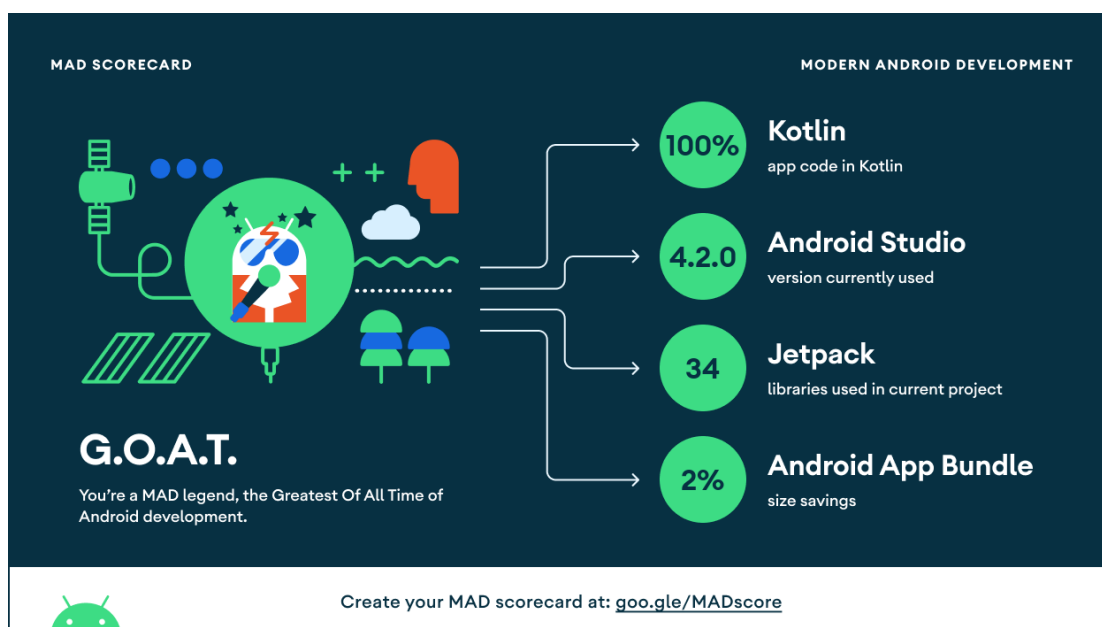
統一 RecyclerView 的 Adapter 來達到重複利用性

```
class ListAdapter(private val list: List<Course>,
                  @LayoutRes Layout: Int,
                  @IdRes Title: Int,
                  @IdRes Instructor: Int,
                  @IdRes Card: Int)
    : RecyclerView.Adapter<CourseViewHolder>() {
    ...
}
```

ListAdapter.kt

小巧思

1. 我們在做完應用程式後有稍微重構了整個專案的程式碼並使用 Google 提供的「現代 Android 開發技能評分」(Modern Android Development (MAD) Skills) 做了評分，其結果為最高等的「G.O.A.T.」(Greatest Of All Time)，表示我們在撰寫程式碼的過程中有遵守著 Google 建議在 Android 開發上的最佳實務，亦有善用了 Android Jetpack 函式庫（我們使用了 36 個），下圖為證：



2. 為了美化其他 UI，我們還將其應用程式添加了 App 的 Logo，讓程式安裝起來有圖案可以看，下圖為證：



心得與結論

從大一到大四，接觸了很多的電腦程式編程，從最初的 C 語言、C++ 以及 VB，到後來的 Python、Creda，也學習了 Neural Network，但是手機程式的編寫的卻是第一次接觸。通過這門課讓我對 Kotlin 也有了些許瞭解。它是一種相容 Java 的語言，並且比 Java 更安全，能夠靜態檢測常見的陷阱。它也比 Java 更簡潔，通過支持 variable type inference，higher-order functions(closures)，extension functions，mixins and first-class delegation 等實現。同時它還比最成熟的競爭對手 Scala 語言更加簡單，讓我們這些初學者更加容易上手。

我們想到製作這份報告的原因就是想到了自己以往的幾年選課的不容易，喜歡的課程以及老師的評價都需要去 Dcard 上一一尋找，這樣不僅費時費力，而且也並不一定能夠找到自己心儀的課程。同時自己是否修過這門課也還需要上學校的網站上去查詢。那我們就想到製作這樣一款 APP，能夠同時相容以上的功能，讓選課簡單化。我們從一開始的毫無眉目直到漸漸設計出來這款簡單的 APP 也花費了不少的時間和精力，大家齊心協力，找資料、設計介面、考慮語言邏輯，最終得到了我們心血的結晶。很感謝老師的諄

諄教誨，也很感謝同學的積極幫忙，我們的程式終於順利完成了，
接下來的就是寫報告收尾了，看著自己的勞動成果寫起報告來特別
起勁、特別有靈感。很感謝這次的課程設計，它使我更加深刻地體
會到多尋找並學習

資料的重要性，只有掌握了一定量的專業知識才能得心應手地
解決諸多問題；另外，做任何事都要有耐心，不要一遇到困難就退
縮；在學習和工作中要時刻謹記團結二字，它好比通向成功的鋪路
石，不可或缺。

未來展望

此次期末作品僅做出了一個能結合深度學習並適時更改題目的問卷，未來的目標是實現在動機所述，「若有一推薦功能像現今電商平台，可以根據過去購物過的經驗及評分推薦新商品那樣，能夠推薦課程提供選擇多樣性及便利性，肯定會大受學生喜愛。」我們有參考了其他學校的非校方課程平台，發現很多學校都有自己的課程平台，不論是在上面互相交流課程內容及資訊或是做課表的安排等等。

現階段已將問卷的網頁版及 Android 應用程式版開發出來，未來若考慮將其專案擴大亦有先前開發經驗，相信會迅速上手。或許在畢業之前，能將其平台開發出來供後續的淡江師生們使用，將適合課程媒合給適合的學生。

參考資料

- [1] R/androiddev - Why use Moshi over Gson? (2017, April 29). Retrieved January 11, 2021, from https://www.reddit.com/r/androiddev/comments/684flw/why_use_moshi_over_gson/
- [2] M. N. (2020, October 03). Goodbye Gson, Hello Moshi. Retrieved January 11, 2021, from <https://proandroiddev.com/goodbye-gson-hello-moshi-4e591116231e>.
- [3] ViewModel Overview ; ; Android Developers. (n.d.). Retrieved January 11, 2021, from <https://developer.android.com/topic/libraries/architecture/viewmodel>

還有許多其他參考資料，未能一一貼上來。