

國 立 中 央 大 學

資 訊 工 程 學 系
碩 士 論 文

探索深度學習或簡易學習模型在點擊率預測任務
中的使用時機

Exploring the usage scenarios of deep learning or simple
learning models for click-through rate prediction

研 究 生：楊易哲

指 導 教 授：陳 弘 軒 博 士

中 華 民 國 一 百 零 九 年 六 月

國立中央大學圖書館學位論文授權書

填單日期：109 年 7 月 23 日

2019.9 版

| | | | |
|-------|-----------------------------|------|--|
| 授權人姓名 | 楊易哲 | 學號 | 107522101 |
| 系所名稱 | 資訊工程學系 | 學位類別 | <input checked="" type="checkbox"/> 碩士 <input type="checkbox"/> 博士 |
| 論文名稱 | 探索深度學習或簡易學習模型在點擊率預測任務中的使用時機 | 指導教授 | 陳弘軒 |

學位論文網路公開授權

授權本人撰寫之學位論文全文電子檔：

- 在「國立中央大學圖書館博碩士論文系統」.

() 同意立即網路公開

() 同意 於西元_____年_____月_____日網路公開

() 不同意網路公開，原因是：_____

- 在國家圖書館「臺灣博碩士論文知識加值系統」

() 同意立即網路公開

() 同意 於西元_____年_____月_____日網路公開

() 不同意網路公開，原因是：_____

依著作權法規定，非專屬、無償授權國立中央大學、台灣聯合大學系統與國家圖書館，不限地域、時間與次數，以文件、錄影帶、錄音帶、光碟、微縮、數位化或其他方式將上列授權標的基於非營利目的進行重製。

學位論文紙本延後公開申請（紙本學位論文立即公開者此欄免填）

本人撰寫之學位論文紙本因以下原因將延後公開

- 延後原因

() 已申請專利並檢附證明，專利申請案號：_____

() 準備以上列論文投稿期刊

() 涉國家機密

() 依法不得提供，請說明：_____

- 公開日期：西元_____年_____月_____日

*繳交教務處註冊組之紙本論文(送繳國家圖書館)若不立即公開，請加填「國家圖書館學位論文延後公開申請書」

研究生簽名：楊易哲

指導教授簽名：陳弘軒

國立中央大學碩士班研究生
論文指導教授推薦書

資訊工程學系碩士班 學系/研究所 楊易哲 研究生
所提之論文 探索深度學習或簡易學習模型在點擊率預測任務中
的使用時機
係由本人指導撰述，同意提付審查。

指導教授 203673 (簽章)
109 年 7 月 14 日

國立中央大學碩士班研究生
論文口試委員審定書

資訊工程學系碩士班 學系/研究所 楊易哲 研究生
所提之論文 探索深度學習或簡易學習模型在點擊率預測任務中
的使用時機

經由委員會審議，認定符合碩士資格標準。

學位考試委員會召集人

委

員

陳以勝
方紅民
孫弘仁

中華民國 109 年 7 月 7 日

探索深度學習或簡易學習模型在點擊率預測任務 中的使用時機

摘要

點擊率的預測在許多內容導向為主的資訊服務中一直有著非常重要的應用，這類服務如電子商務網站、影音串流平台與社群媒體網站，都會盡可能的將使用者會點擊的內容展示在最顯眼的位子，目的即是為了增加使用者使用服務的時間，使用者使用服務的時間增加自然能夠提升服務帶來的商業效益。

要如何找出使用者感興趣並且會點擊的內容一直以來都是推薦系統領域的研究重點，隨著近年來深度學習的興起與成功，已有許多國際大型公司將自身所提供的內容服務改以基於深度學習架構的推薦系統進行推薦，並且提出自行研發的深度學習模型。

我們發現這些成功應用深度學習的服務往往都有著國際級的超大型規模，相對的區域性中小型規模的服務就鮮少有看到成功使用深度學習的案例，這讓我們不禁懷疑時下流行的深度學習模型用於中小型服務的可行性，於是我們開始使用不同的深層與簡易模型對不同規模的資料進行實驗，我們發現深層模型的確不全然適用於中小規模的服務，但其與簡易模型一樣，有著一種在特定條件下逐漸準確的趨勢，也就是說不同的模型有著各自準確的時機，發現這點後，我們開始於不同時機選擇不同模型進行預測，最終提升了點擊率預測任務整體的準確性。

關鍵字：點擊率預測，推薦系統，深度學習，電子商務

Exploring the usage scenarios of deep learning or simple learning models for click-through rate prediction

Abstract

Click-through rate prediction has been an essential application in many content-oriented information services, such as e-commerce, video streaming platforms, and social media. These services display contents that users are likely to click in a prominent position. As a result, users may be attracted and spend more time on these services.

With the rise and success of deep learning in recent years, many large international companies have integrated their content services with recommendation systems based on the deep learning framework and proposed their deep learning models. However, it seems only the Internet giants reported successful stories on deep learning-based recommender systems. Consequently, we are suspicious of the feasibility of the deep learning models on small and medium-sized services, so we started experimenting with machine learning models with different complexity and datasets of different sizes. We found that deep learning models and simple models seem to appropriate in different cases. After discovering this, we proposed a model to select a recommendation algorithm based on the given scenario automatically. This selecting model improved the overall accuracy of the click-through rate prediction task.

Keywords: Click-Through Rate Prediction, Recommender System, Deep Learning, E-commerce

目錄

| | 頁次 |
|---------------------------------------|------------|
| 摘要 | iv |
| Abstract | v |
| 目錄 | vii |
| 圖目錄 | ix |
| 表目錄 | xi |
| | |
| 一、 緒論 | 1 |
| | |
| 二、 相關研究 | 3 |
| 2.1 推薦系統在使用者端的使用情境 | 3 |
| 2.2 推薦系統之架構 | 7 |
| 2.2.1 候選 (Candidate Generation)..... | 8 |
| 2.2.1.1 多路候選策略 | 9 |
| 2.2.1.2 基於 embedding 的候選方法 | 10 |
| | |
| 三、 研究方法與流程 | 13 |
| 3.1 候選 | 14 |
| 3.1.1 Word2vec..... | 14 |
| 3.1.2 Item2vec..... | 15 |
| 3.1.3 生成商品 embedding 之流程 | 16 |
| 3.1.4 透過計算商品相似度進行 Top-K 的候選 | 17 |

| | |
|--|-----------|
| 3.2 多模型排序 | 17 |
| 3.2.1 基於最近鄰居法 (k-nearest neighbors) 的排序 | 18 |
| 3.2.2 使用簡易神經網路的排序 | 18 |
| 3.2.3 使用 DIN(Deep Interest Network) 模型的排序..... | 20 |
| 3.2.4 使用 DIEN(Deep Interest Evolution Network) 模型 的排序 | 21 |
| 3.3 Switch..... | 23 |
| 四、 實驗結果與分析 | 26 |
| 4.1 資料集介紹 | 26 |
| 4.1.1 淘寶用戶行為資料集 | 26 |
| 4.1.2 台灣電商用戶行為資料集 | 27 |
| 4.2 實驗流程與細節 | 29 |
| 4.3 實驗結果 | 31 |
| 4.3.1 評量指標 | 31 |
| 4.3.2 排序模型實驗結果 | 32 |
| 4.3.3 Switch 模型實驗結果..... | 32 |
| 4.4 探討商品於訓練資料中出現次數與預測結果之關係 | 35 |
| 4.5 使用更多的資料去訓練 DIN 與 DIEN | 38 |
| 五、 結論與未來展望 | 40 |
| 5.1 結論 | 40 |
| 5.2 未來展望 | 40 |
| 參考文獻 | 41 |
| 附錄 A 實驗程式碼 | 45 |

圖目錄

| | 頁次 |
|--|----|
| 2.1 YouTube 首頁 | 4 |
| 2.2 YouTube 影片播放頁面 | 4 |
| 2.3 推薦循環流程 | 5 |
| 2.4 受瀏覽記錄啓發的推薦 | 6 |
| 2.5 推薦系統基本架構 | 7 |
| 2.6 細分為四階段的推薦系統架構 | 8 |
| 2.7 常見的多路候選策略 | 9 |
| 2.8 Deep Crossing、FNN、Wide&Deep 模型的 embedding 層 [25, 30, 7] | 10 |
| 2.9 YouTube 推薦系統的候選模型結構 [9] | 11 |
| 3.1 我們所提出的推薦系統架構 | 13 |
| 3.2 Word2Vec 的兩種模型：CBOW 與 Skip-gram | 15 |
| 3.3 將使用者的點擊記錄轉化為點擊序列 | 16 |
| 3.4 將商品點擊序列轉換為正負樣本資料 | 19 |
| 3.5 從訓練模型到產生出新排序的整體流程 | 19 |
| 3.6 從歷史瀏覽行為計算候選得分 [33] | 20 |
| 3.7 DIN 模型結構 [33] | 21 |
| 3.8 DIEN 模型結構 [32] | 22 |
| 3.9 於眾多排序結果中找出最佳排序模型 | 23 |

| | |
|---|----|
| 3.10 Tomek Links 演算法示意圖 [1] | 24 |
| 4.1 實驗流程圖 | 29 |
| 4.2 將資料依時間分段 | 30 |
| 4.3 淘寶資料集商品出現次數與排名關係圖 (kNN v.s. NN) | 36 |
| 4.4 台灣電商資料集商品出現次數與排名關係圖 (kNN v.s. NN) | 36 |
| 4.5 淘寶資料集商品出現次數與排名關係圖 (DIN v.s. DIEN) | 37 |
| 4.6 台灣電商資料集商品出現次數與排名關係圖 (DIN v.s. DIEN) | 37 |

表目錄

| | 頁次 |
|--|----|
| 4.1 淘寶資料集行為列表 | 27 |
| 4.2 淘寶資料集資料組織形式 | 27 |
| 4.3 淘寶資料集特徵統計值 | 27 |
| 4.4 台灣電商資料集資料組織形式 | 28 |
| 4.5 台灣電商資料集行為列表 | 28 |
| 4.6 台灣電商資料集頁面列表 | 28 |
| 4.7 台灣電商資料集特徵統計值 | 29 |
| 4.8 淘寶資料集排序實驗結果 | 32 |
| 4.9 台灣電商資料集排序實驗結果 | 32 |
| 4.10 淘寶資料集 Switch 實驗結果 | 33 |
| 4.11 台灣電商資料集 Switch 實驗結果 | 33 |
| 4.12 淘寶資料集排名前 20 Top 占比統計 | 34 |
| 4.13 台灣電商資料集排名前 20 Top 占比統計 | 34 |
| 4.14 兩資料集中不同 Switch 模型的效能指標 | 35 |
| 4.15 淘寶資料集 DIN 與 DIEN 增加訓練資料量的實驗結果 | 38 |
| 4.16 台灣電商資料集 DIN 與 DIEN 增加訓練資料量的實驗結果 | 38 |

一、緒論

點擊率 (Click-Through Rate) 的預測一直一來都是推薦系統的研究重點之一，其應用範圍之廣，不管是電子商務網站、影音串流平台或是社群媒體網站，都會透過推薦系統將使用者感興趣的內容依據點擊率的高低推薦給使用者，不同的使用者於不同的使用情境下會有不同的點擊傾向，如何準確的找出當下使用者會點擊的內容一直以來都是商業公司的研究重點，因為只要能夠提升使用者的點擊率，自然就能夠提升使用者使用服務的時間，使用者使用服務的時間增加，自然能夠提升公司的商業效益。

隨著近年來深度學習的興起，不少國際一線公司都將自己旗下服務的推薦系統改用深度學習的方式去進行建構，各種不同架構的點擊率預測模型也如雨後春筍般被提出，已有許多基於深度學習架構的點擊率預測模型 [33, 32, 10, 18]，被成功應用於大型服務網站，如淘寶網、Airbnb 與 YouTube 等等。

綜觀近年來這些成功的應用案例，我們發現其都有幾個共通點，像是用戶數量極多、內容多樣性極高與服務規模極大等等特徵，這讓我們不禁思考，這些成功被應用於大型服務的深度學習模型，如果將其應用於中小規模的服務是否也能有一樣的效果？根據這個疑問，我們開始了進一步的研究，過程中我們發現，當今中小規模服務如電子商務網站等等，大多都還是以傳統機器學習方法或是透過簡單的神經網路去進行推薦，鮮少有看到使用知名推薦模型的案例，為此，我們決定從頭開始建構一個以點擊率為導向的推薦系統，並且使用多種不同的點擊率預測模

型對不同規模的資料進行實驗，以驗證我們所提出的疑問。

在透過自行建構的推薦系統對不同的模型與資料進行實驗後，我們發現，在我們的實驗資料集中傳統機器學習方法在絕大多數案例都會優於深層與簡易學習模型，但在某些特別的時機與條件下，深層與簡易學習模型卻有著越來越準確的趨勢，發現這點後我們開始去尋找各個模型的使用時機，最後透過了於不同時機選擇不同模型的方法提升了整體推薦系統對點擊率預測的準確性。

本篇論文的具體貢獻總結如下：

- 我們建構了一個以點擊率為導向的推薦系統，並且使用了多種模型對不同規模的資料集進行實驗，最後發現那些成功應用於大型服務的深度學習模型並不全然適用於中小規模的服務。
- 我們發現了根據當下推薦任務環境的不同，不同的深層與簡易學習模型會有其適用與不適用的時機，並且在特定的條件下某些模型會有越來越準確的趨勢。
- 我們改進了傳統推薦系統的推薦方法，透過多個模型使用時機的選擇，提升了整體點擊率預測任務的準確度。

本篇論文在後續的結構如下：在第二章我們會介紹推薦系統於使用者端的運作情境與基本架構，在第三章我們會說明建構推薦系統的詳細方法與模型細節，第四章我們會介紹實驗所使用的資料集與探討實驗結果，最後於第五章我們會總結本研究至今為止的發現並且對未來可以改進的方向進行討論。

二、相關研究

近幾年在人工智慧的浪潮下，電腦視覺、自然語言處理、語音辨識等領域之研究皆有飛躍性的發展，時至今日相關研究數量仍然在以飛快的速度增長著。與上述三者相比，推薦系統的發展速度並不算太快，在應用實務上目前國內大多數中小型企業仍然是以使用傳統機器學習方法或者是基於規則 (Rule-based) 的方式進行推薦，不過隨著深度學習的興起與衆多新興技術的引入，我們可以觀察到推薦系統在國際一線科技公司所提供的服務，如，YouTube、淘寶網、Airbnb 已有明顯的技術發展趨勢 [9, 31, 33, 32, 10, 13, 12]。

本節將從推薦系統在使用者端的使用情境開始介紹，其中將會以目前應用市場較大的影音平台、電商平臺作為舉例，說明使用者接受推薦並且進行反饋的過程。在說明完使用者的使用情境後，接著我們將會針對推薦系統運作的架構與技術細節進行不同角度的歸納與介紹。

2.1 推薦系統在使用者端的使用情境

以全世界最大影音平台 YouTube 為例，其首頁 (圖2.1) 的眾多影片即為其推薦系統運作後的結果，當使用者點擊任意一部影片的瞬間，推薦系統即可收到點擊率 (Click-Through Rate) 的反饋資訊，其意義代表了系統推薦了多個項目，而使用者接受了其中之一的推薦。當使用者點擊進入影片播放頁面 (圖2.2) 後，其可以對影片按喜歡或者不喜歡並且在留言區域留言，這些資訊都會被系統紀錄，作為往後推薦系統運作時所

使用的歷史特徵訊息。

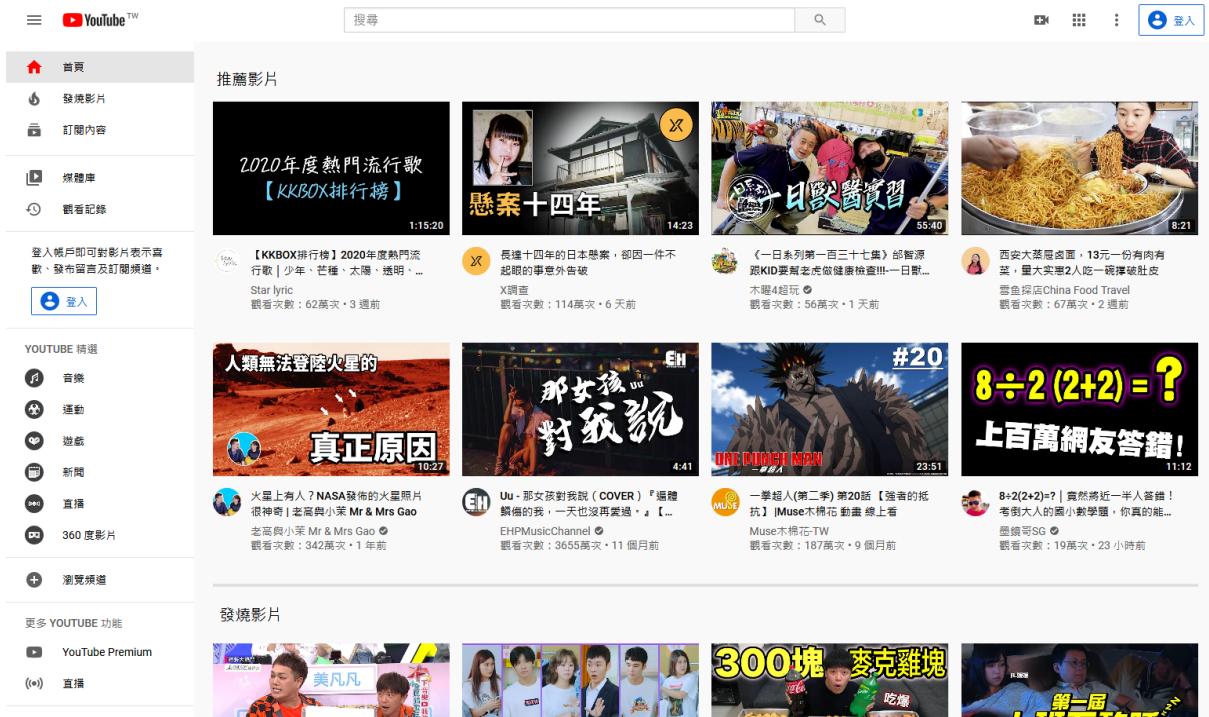


圖 2.1: YouTube 首頁



圖 2.2: YouTube 影片播放頁面

使用者接受推薦並且給予反饋，然後推薦系統又根據反饋進行推薦的循環行為，如圖2.3，即為目前主流推薦系統的運作模式。

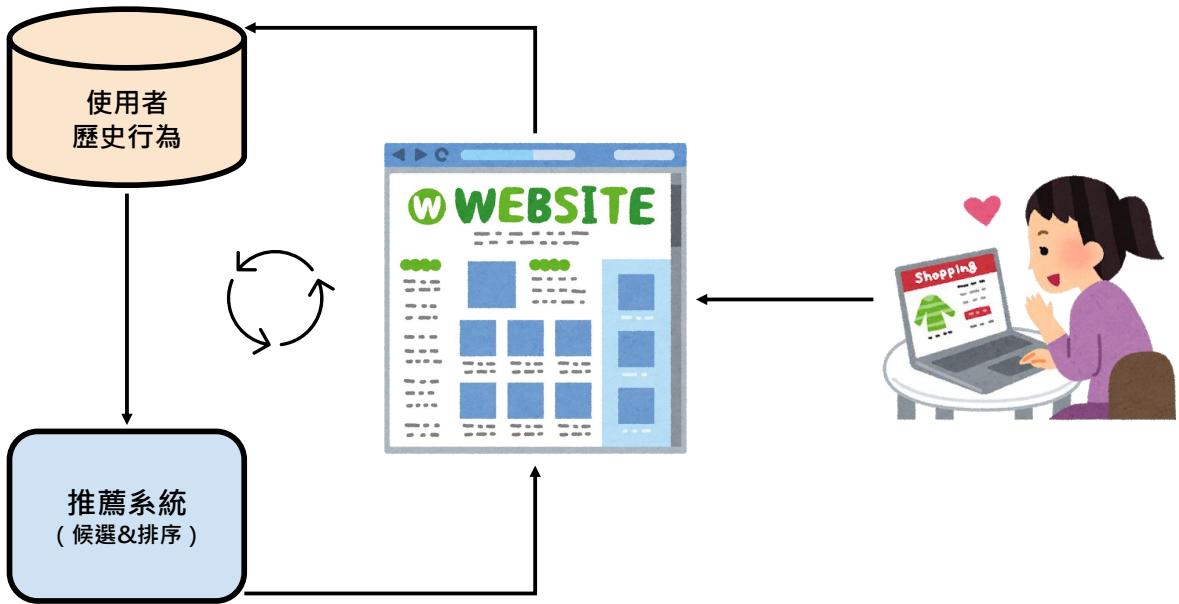


圖 2.3: 推薦循環流程

如同上述的循環模式，YouTube 的影片推薦理所當然的不會在點擊進入影片頁面後就結束，參照圖2.2右半部，我們可以發現在 YouTube 影片播放頁的畫面右方又推薦了一連串的影片，而且最頂部的影片還被預設為當下影片結束後接著播放的影片，此處即為推薦系統針對點擊率任務的一大應用，什麼樣的影片該在當下推薦給使用者？而在這些影片當中哪些影片該給予較前面的排序？這些都是在設計推薦系統時所需要思考的問題，好的推薦排序可以讓使用者直接點擊進入自己喜歡的影片，不需要有多餘的操作，如此一來即可增加使用者使用服務的時間，進而提高服務的收益。

與影音平台的推薦模式相比，電子商務平台也有著極其相似的推薦架構，以下我們以全球最大電子商務網站 Amazon 來做舉例。在有登入會員的情況下，進入網站首頁我們就能看到 Amazon 針對會員的個人化專屬推薦與最近檢視紀錄等一系列資訊，其中更有一個特別的區塊名為「受瀏覽記錄啟發」如圖2.4，當中有著衆多的推薦商品，這邊即為推薦系統根據使用者過往的瀏覽紀錄所進行的推薦。

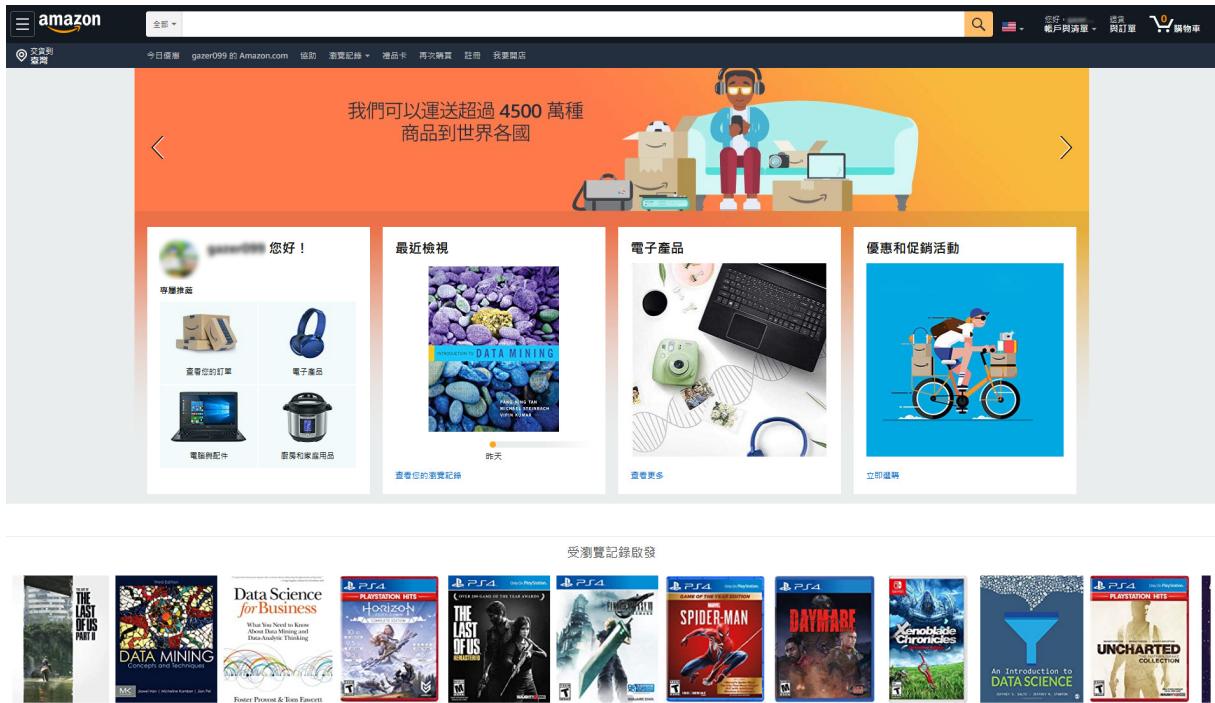


圖 2.4: 受瀏覽記錄啟發的推薦

如同先前所提到的推薦循環，在使用者點擊進入商品頁後推薦還會在持續著，我們能發現在商品頁中往往還會有著一系列的商品列表，這類列表通常都是「購買此商品的客戶還購買了」、「瀏覽本商品的客戶也瀏覽了」等等內容，這樣的推薦其實早已是傳統電商的標準功能，然而我們發現在國際大型電商中，其推薦往往不止於上述兩種依照統計而產生的列表，如 Amazon 有「受您最近的購物趨勢啟發」、淘寶有「猜你喜歡」等等，這類通常透過深度學習而產生的推薦比起傳統簡單的基於規則 (Rule-based) 的推薦方法更能貼近使用者興趣並且捕捉當下使用者真正想要的商品，對於增加點擊率的任務是有正向幫助的。

受限於畫面的大小，版面能顯示給使用者的推薦商品數量是有限的，如何讓使用者真正有興趣的商品顯示在最前方並且讓使用者的點擊率有所增長，即是目前推薦系統在電商平台上的重要挑戰，也是本研究探討的重點。

2.2 推薦系統之架構

在實務的應用上，推薦系統通常會有兩大運作階段，分別是候選 (Candidate Generation) 和排序 (Ranking)，如圖2.5所示。首先要進行的是候選，在此階段推薦系統主要進行的任務是依照當前的環境特徵，從海量的商品庫中快速的篩選出使用者感興趣並且有可能會點擊的項目，將其傳送給排序階段進行後續的操作。而在排序階段接收到候選所提供之一系列項目後，就會開始使用較多的商品特徵配合複雜的模型結構，針對當下環境狀況提供給使用者合適的個人化推薦。

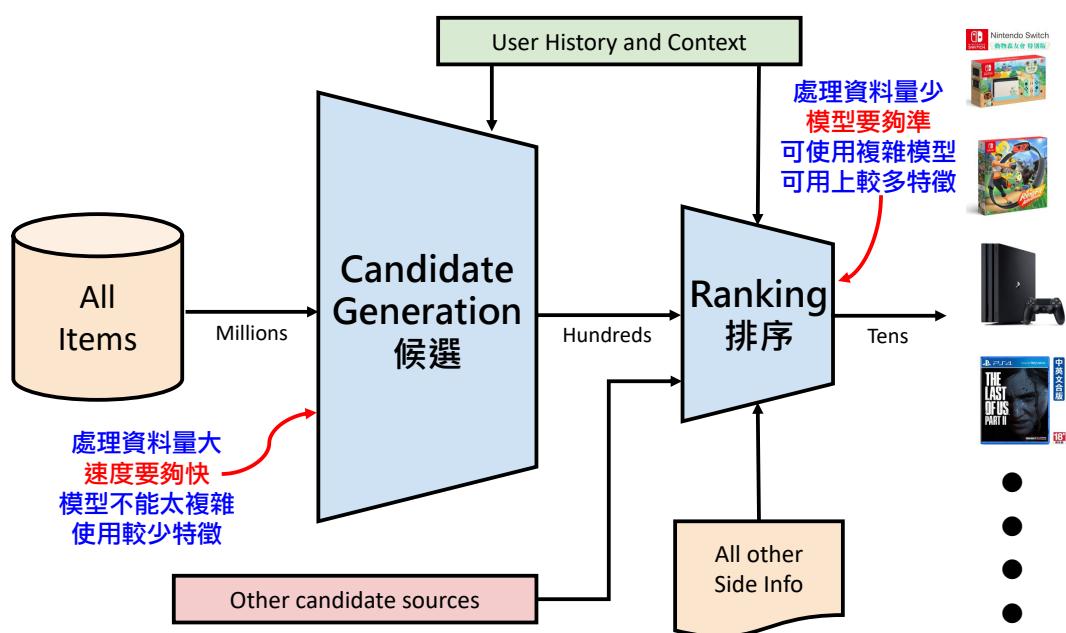


圖 2.5: 推薦系統基本架構

總體來說，候選階段強調的是快速，而排序階段強調的是精準，主要原因在於候選階段是從百萬級的資料中找出幾百個相關商品，這邊的量級相差之大，若在此處進行較精密的計算勢必會耗費較多時間，於是乎，如何在此階段快速的概略篩選出主要相關商品就成了一大議題。而在排序階段之所以強調精準，是因為前面所提到的版面大小與人眼一次所能接受的資訊量是有限的，若能將使用者真正有興趣的項目一開始就放在較高的序位的話，就能提高使用者的點擊率，進而提高商業效益。

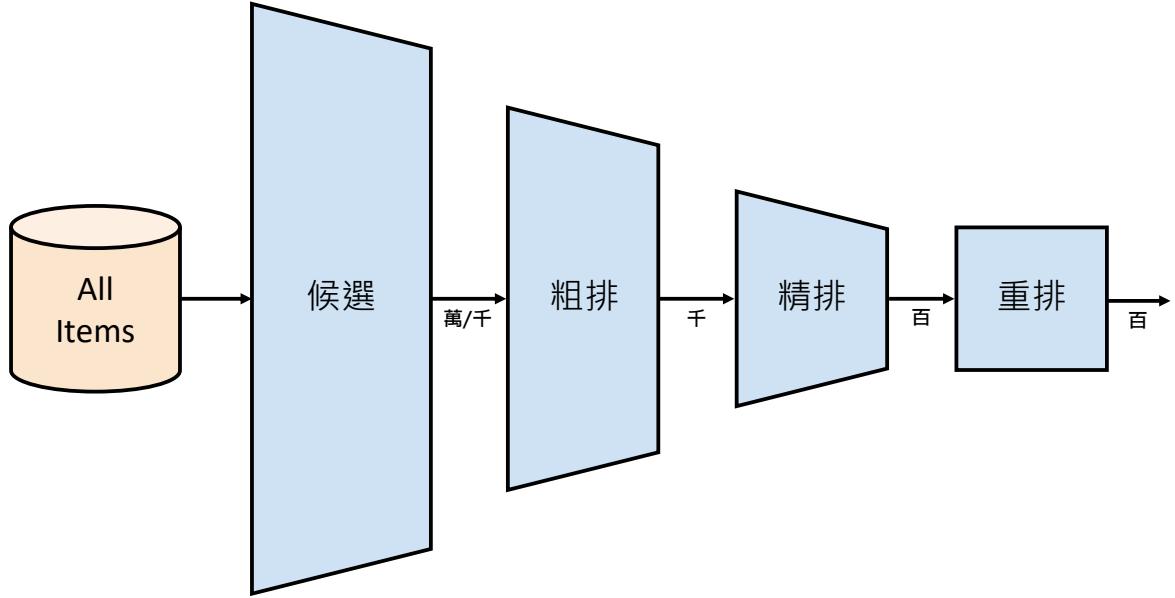


圖 2.6: 細分為四階段的推薦系統架構

若我們以更加實務精細的角度去建構推薦系統，其將會有四個階段，如圖2.6所示，分別為：候選、粗排、精排和重排。候選所進行的任務如前段所述，但有時候候選出來的商品數量還是太大，如果就這樣將其結果傳送至最終排序將會導致排序時間增加，這時可以選擇性的對候選結果進行一個粗略排序以減少往下一階段傳送的商品數量。精排階段則為將接收到的項目用各種方法進行精準的個人化排序。而最後重排的階段往往會是套用各式各樣的商業策略，例如，去除已讀過的項目、同質性過高的商品與提高主打商品的序位等等。

不論以簡單或精細的形式去建構推薦系統，其最終都還是不會脫離候選 (Candidate Generation) 和排序 (Ranking) 兩個環節，接下來我們將會對候選階段進行詳細介紹。

2.2.1 候選 (Candidate Generation)

推薦系統的候選階段對整體系統來說是一個非常關鍵的環節，在此階段中我們需要其能在有限的運算時間內找出使用者高機率會點擊的項

目，也就是要求「低計算時間」、「高候選率」，但這兩項指標其實是互相抵觸的，因為要求低計算時間的話候選策略勢必不能太複雜，而要求高候選率的話勢必又要花時間仔細找出排序模型需要的項目。在權衡這兩項要求後，目前業界主要使用的候選方法有兩種，分別為(1)多路候選策略、(2)基於embedding的候選方法。

2.2.1.1 多路候選策略

「多路候選策略」指的是採用不同的策略、使用不同的特徵或簡單模型，分別候選一部分的候選項目，然後再把這些候選項目混和在一起提供給後續排序模型使用的方法。

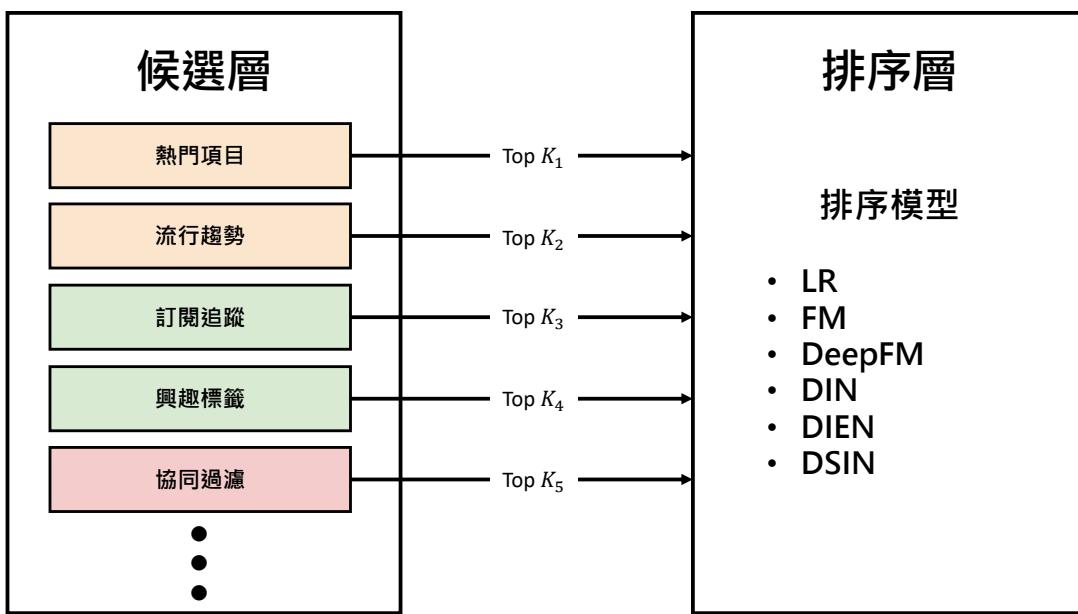


圖 2.7: 常見的多路候選策略

如圖2.7所示，常見的多路候選策略通常會有熱門項目、流行趨勢、訂閱追蹤、興趣標籤、協同過濾等等，其中「熱門項目、流行趨勢」這類的候選路線就屬於簡單的候選策略，而「訂閱追蹤、興趣標籤」則是具個人化特徵的候選，最後「協同過濾」[11, 17] 即是高效率簡單模型的應用。在多路候選策略中每一路都會拉回 K 個候選項目，根據策略的不

同，每一路拉回的候選項目數量也可能不同，這邊的 K 是一個超參數 (hyperparameter)，具體的數值需要經過離線評估與線上 A/B 測試才能界定出合理的範圍。

2.2.1.2 基於 embedding 的候選方法

所謂的「embedding」基本上就是用一個低維稠密的向量去表示一個物件 (object)，這邊所謂的物件可以是一個詞、一個使用者、一個商品或是一部影片等等。透過 embedding 的技術，不但可以向量化表達物件的特徵，還可以去計算物件與物件之間的關係，在當今的深度學習框架中 embedding 的技術已經應用的非常廣泛，可說其為深度學習的基礎核心也不為過。

在推薦系統運作之初，那些商品、影片等要被推薦的物件其原始特徵通常都是非常高維且稀疏的，這樣的高維稀疏特徵自然不適合直接使用深度學習方法去進行推薦。為了解決推薦任務在實務上原始輸入資料通常都高維且稀疏的問題，許多的深度學習推薦模型，如 Deep Crossing[25]、FNN[30]、Wide&Deep[7] 都會在輸入層到全連接層之間加入一個「embedding 層」(如圖2.8中紅框所示)，目標就是要將高維稀疏的原始特徵映射到低維稠密的向量之中。

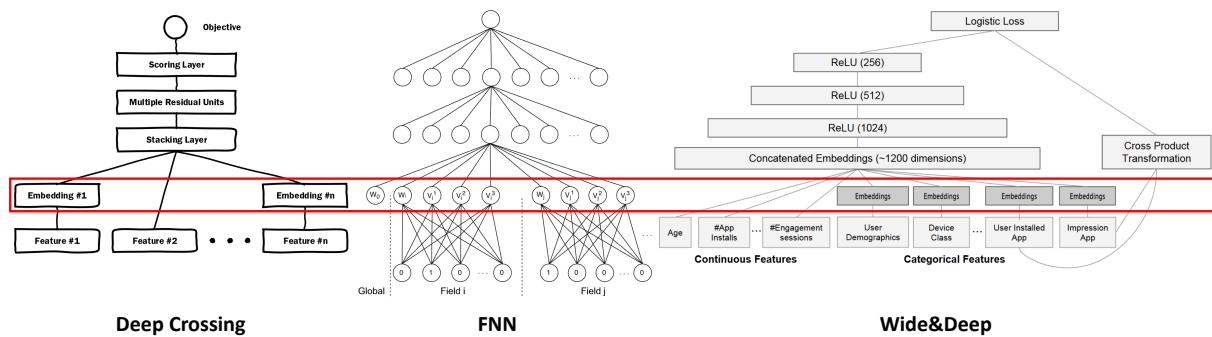


圖 2.8: Deep Crossing、FNN、Wide&Deep 模型的 embedding 層 [25, 30, 7]

將 embedding 層整合進整個深度學習網路在理論上自然是最佳的作法，因為這麼做的話，神經網路在進行權重更新時，梯度自然能反向傳播

(Back-propagation) 到最底部的 embedding 層去，使得每次的訓練都能讓物件的 embedding 被更新，層層更新後的 embedding 自然能更準確的表達其特徵。但是這樣做缺點是顯而易見的，由於 embedding 層的輸入維度通常都很大，導致該層的訓練參數量過多，會進而拖慢整個深度學習網路的訓練收斂速度，所以在某些要求速度的實務應用上通常會放棄這類端到端 (end-to-end) 的訓練方法，改以使用預訓練好的物件 embedding 去進行後續操作。

在有了物件 embedding 這種高表達能力的特徵之後，我們即可利用其向量的相似性，開始進行推薦流程的候選階段，生成推薦的候選清單。在此我們以 YouTube 推薦系統的候選模型為例進行介紹。

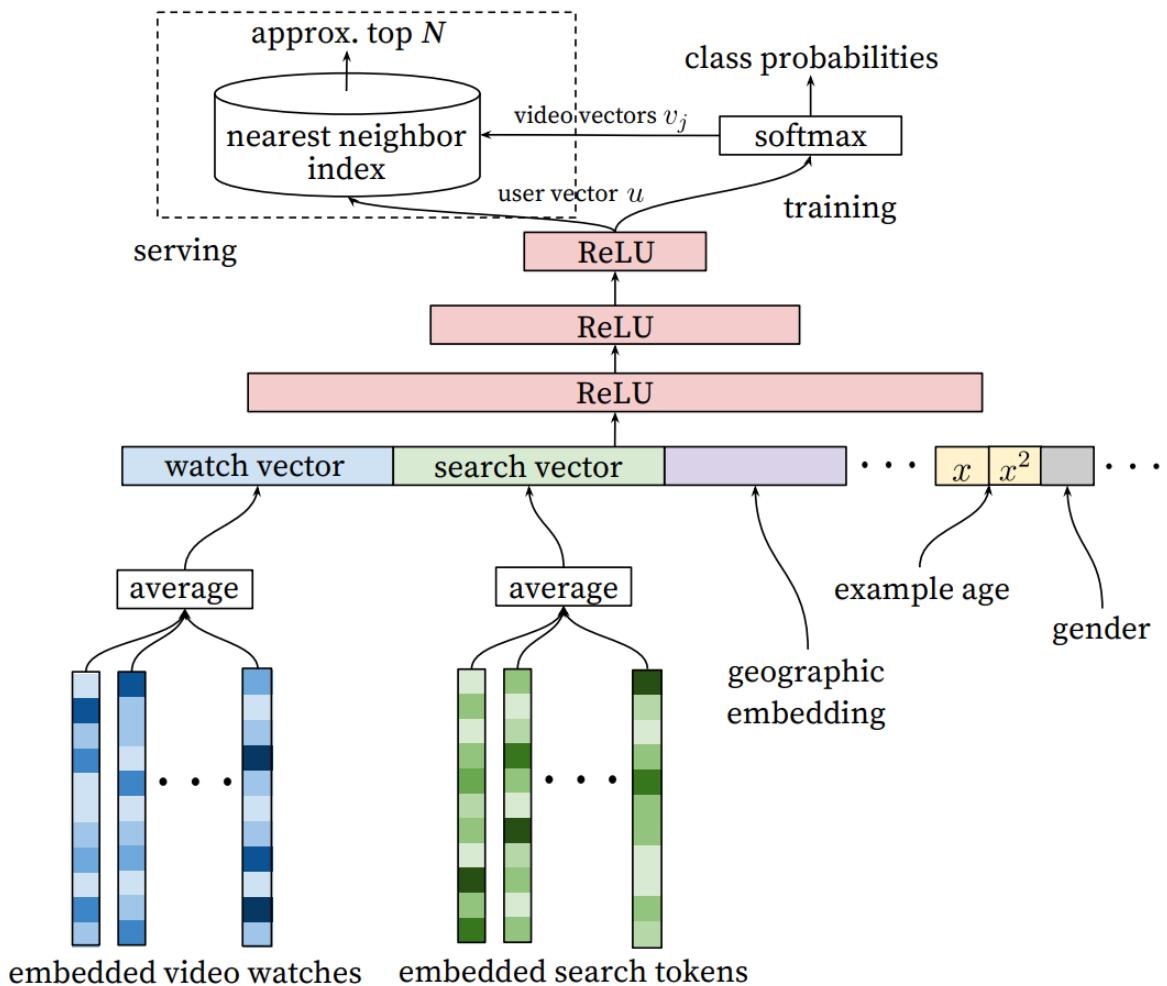


圖 2.9: YouTube 推薦系統的候選模型結構 [9]

圖2.9為 YouTube 推薦系統的候選模型結構，其最底部輸入層的輸入

特徵分別為，使用者的觀看紀錄 embedding、搜尋紀錄 embedding、地理特徵 embedding 和使用者基本資訊等，該模型把候選任務建構成一個「超大規模多分類問題」，預測目標為使用者會不會看某部影片，其中經過三層 ReLU 全連接層後可得到使用者 embedding，最後輸出層會使用前一層輸出的使用者 embedding 透過 softmax 得到使用者觀看每一部影片的機率分布，由於輸出的機率分布向量每一維即對應一部影片，也就是說 softmax 出來的行向量就是影片的 embedding。

在訓練完上述的候選模型後，進行實際候選預測之前，我們不需要再重新部署整個深度學習網路來完成從原始特徵向量到最終輸出的過程，只需要將在訓練完成後得到的使用者 embedding 與影片 embedding 存入資料庫之中，再透過兩者的內積運算後重排序的方法即可從整個候選序列中取出需要的 Top- K 個物件，這樣的過程即為一個經典的基於 embedding 的候選方法。

不過在真實世界的推薦場景中，候選物件通常都是百萬量級，即使是進行像內積這樣時間複雜度為 $O(n)$ 的運算也會耗費許多時間，所以在大規模應用實務上，embedding 的運算通常會使用局部敏感雜湊 (Locality-Sensitive Hashing)[26] 或是透過多 GPU 運算 [15] 的方式減少運算所需時間。

三、研究方法與流程

在本章節中，我們將會從頭開始建構一個推薦系統，並且在排序階段應用不同的深度學習架構與簡易模型去實作不同的排序方法。不同於過往的研究與標準的推薦系統推薦流程，我們發現在不同的條件下，不同的排序方法所產生的排序結果會互有優缺，也就是說複雜的深度學習模型與簡單架構的模型所產生的排序都會有其準確的運作情境，因此我們決定在現行的推薦系統架構中新增一個階段，選擇 (Switch)，此階段的目標為依據當下的環境特徵，於眾多的排序結果中選擇出最優者進行推薦。

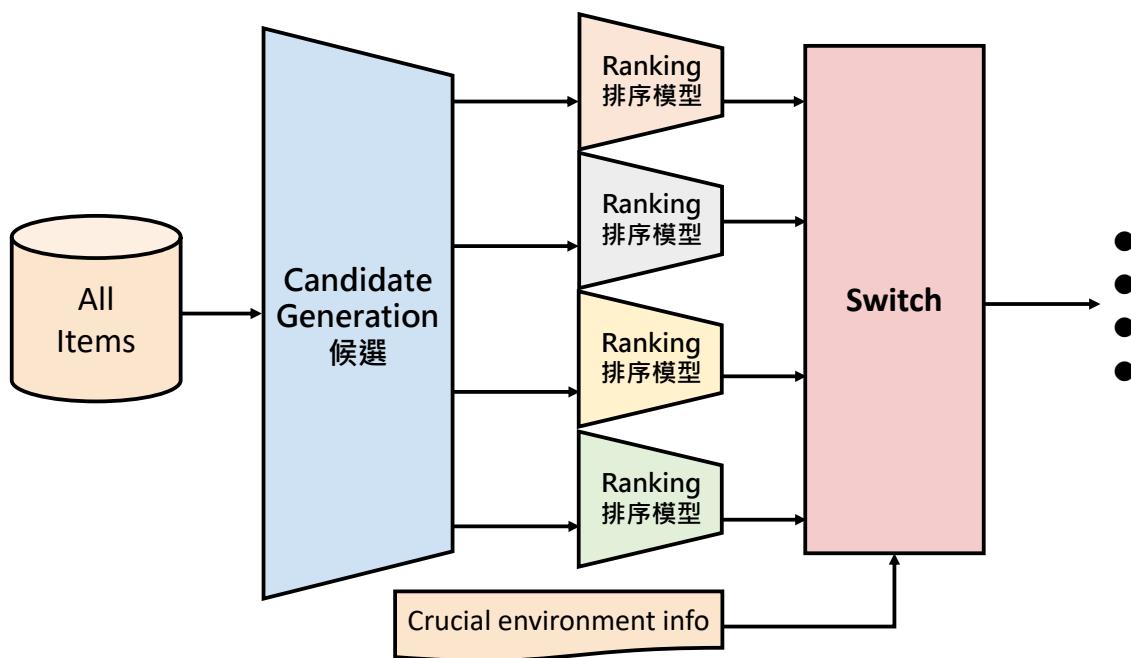


圖 3.1: 我們所提出的推薦系統架構

我們所提出的新型推薦系統架構如圖3.1所示，整體流程依序為 (1)

候選、(2) 多模型排序、(3) 選擇。接下來我們將依序針對此三階段的實作進行介紹。

3.1 候選

在最初的候選階段，我們選擇使用基於 embedding 的候選方法，在此我們的首要目標為產生每件商品的 embedding，接著才會去進行 $\text{Top} - K$ 的候選，在具體描述生成商品 embedding 的流程之前，我們將先針對 Word2vec[20, 21, 24]、Item2vec[5] 等經典 embedding 生成方法進行介紹。

3.1.1 Word2vec

Word2vec 是 Google 於 2013 年所提出的將文字 embedding 的方法，其主要是透過學習大量文本資料的方式，將字詞用數學向量的方式去表示他們的語意，透過這樣的方法相似語意的單字在同一個空間中會有較近的距離。

Word2Vec 模型中，主要有 CBOW 與 Skip-gram 兩種模型。從直觀上來理解，Skip-gram 是給定輸入字詞後，來預測上下文；CBOW 則是給定上下文，來預測輸入的字詞，如圖3.2所示。

在訓練 Word2Vec 模型時需要一組句子組成的語料庫。假設一個長度為 T 的句子為 $w_1, w_2 \dots w_T$ ，為了生成模型訓練樣本我們會選擇一個長度為 $2c + 1$ (目標詞前後各 c 個詞) 的滑動窗口，將窗口由左至右滑動，每移動一次，窗口中的詞組即成了一個訓練樣本。在有了訓練樣本後我們就可以開始定義優化目標，既然每個詞 w_t 都決定了相鄰詞 w_{t+j} ，基於最大似然估計 (Maximum likelihood estimation) 的方法，希望所有樣本的條件機率 $p(w_{t+j} | w_t)$ 之積最大，這裡使用對數機率，我們就可以得到 Word2Vec 的目標函數 (式3.1)，透過求解目標函數的過程我們即可得到

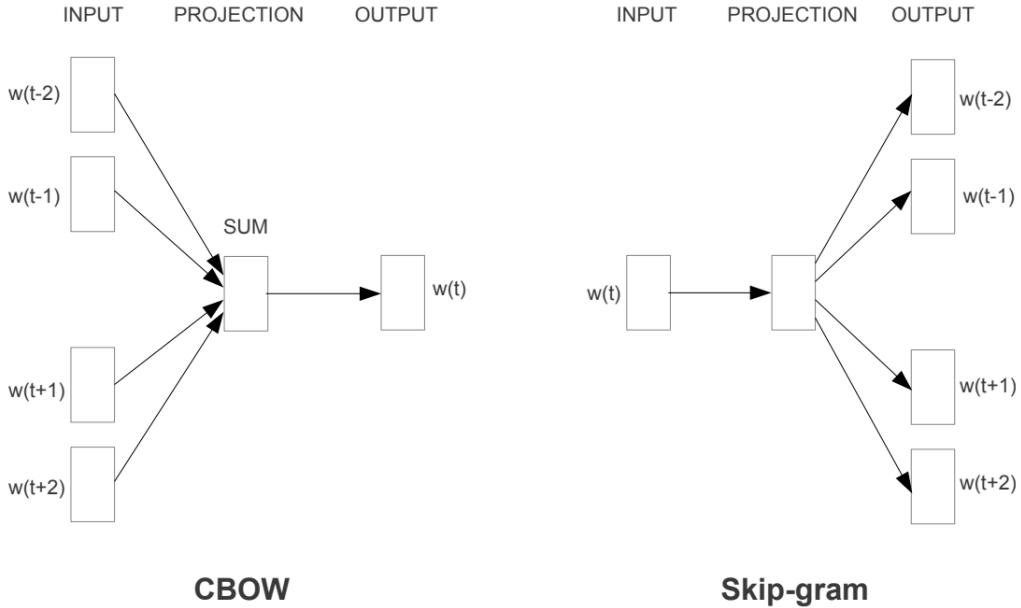


圖 3.2: Word2Vec 的兩種模型: CBOW 與 Skip-gram

語料庫中各個詞的 embedding 向量。

$$\frac{1}{T} \sum_{t=1}^T \sum_{-c \leq j \leq c, j \neq 0} \log p(w_{t+j} | w_t) \quad (3.1)$$

3.1.2 Item2vec

有了 Word2vec 的概念後，我們就能夠做進一步的推廣，既然可以對詞序列中的詞進行 embedding，那麼自然也可以將類似的作法套用到電商網站的使用者商品點擊序列中，對序列中的商品進行 embedding。基於這樣的概念，微軟於 2016 年提出 Item2vec，其原理為將使用者點擊的商品集合等價於 Word2vec 的詞序列，假設長度為 K 的使用者商品點擊紀錄為 $w_1, w_2 \dots w_K$ ，其目標函數即會為：

$$\frac{1}{K} \sum_{i=1}^K \sum_{j \neq i}^K \log p(w_j | w_i) \quad (3.2)$$

該方法不同於 Word2vec 的地方在於其忽略了序列中商品的空間關係 (Spatial Information)，也就是捨棄了時間窗口 (Window) 的概念，對

於出現在同一序列中的商品兩兩都視為相關並且會去計算兩者條件機率的合，而不會只計算整個窗口內的條件機率。

3.1.3 生成商品 embedding 之流程

有了前述的 embedding 生成方法後我們就可以開始進行商品 embedding 從無到有的生成工作，在此我們以 Word2vec 的概念進行說明。



| 使用者點擊紀錄 | | | 使用者點擊序列 | |
|---------|------------|------|---------|------------------|
| User | Click-item | Time | User | Clickstream |
| Joel | 主機板A | 7:00 | Joel | 主機板A、主機板B、顯示卡、滑鼠 |
| Joel | 主機板B | 7:10 | Joel | 鍵盤、耳機 |
| Ellie | 貓糧 | 7:20 | Ellie | 貓糧、貓砂 |
| Joel | 顯示卡 | 7:30 | Ellie | 泡麵A、泡麵B、泡麵C、餅乾 |
| Ellie | 貓砂 | 7:40 | | |
| Joel | 滑鼠 | 7:50 | | |
| Ellie | 泡麵A | 8:00 | | |
| Ellie | 泡麵B | 8:10 | | |
| Joel | 鍵盤 | 8:20 | | |
| Ellie | 泡麵C | 8:30 | | |
| Ellie | 餅乾 | 8:40 | | |
| Joel | 耳機 | 8:50 | | |

圖 3.3: 將使用者的點擊記錄轉化為點擊序列

如圖3.3所示，首先我們需要將過往的使用者商品點擊紀錄轉化為一筆一筆的商品點擊序列，也就是相當於產生語料庫 (Corpus) 的概念，在此即會有衆多的序列轉化策略，其中最典型簡單的方法就是將同個帳號的使用者在指定時間內點擊的所有商品當成一個序列，這樣做的好處是簡單快速不用做複雜的處理，但其缺點就是每個使用者的使用習慣不同，同一段時間內有些使用者可能都在看同一類的商品，但有些使用者可能已經從寵物飼料看到進口泡麵了，若差異太大的商品都在同一序列中的話勢必會對訓練出的商品 embedding 造成影響，所以該用什麼樣的策略

將使用者商品點擊紀錄轉化成商品序列、該用什麼樣的參數去訓練生成 embedding 都需要依照真實的任務情境與資料的狀況進行調整，如此才能確保 embedding 的品質，以增進後續推薦操作的準確度。

3.1.4 透過計算商品相似度進行 Top-K 的候選

在訓練出所有商品的 embedding 後，我們就可以開始進行具體的候選策略。本研究的實驗推薦情境為，當使用者點擊進入某項商品頁面後，頁面周圍該持續推薦的項目。在此場景下我們選擇的方法是以當下使用者瀏覽的商品做為目標，使用餘弦相似度 (Cosine Similarity) 如式3.3，去計算所有商品與其之相似性 [27]，最後取出與當下瀏覽商品最相似的 99 項商品做為要候選的推薦項目。

$$\text{similarity} = \cos(\theta) = \frac{A \cdot B}{\|A\| \|B\|} = \frac{\sum_{i=1}^n A_i \times B_i}{\sqrt{\sum_{i=1}^n (A_i)^2} \times \sqrt{\sum_{i=1}^n (B_i)^2}} \quad (3.3)$$

有了候選階段取得的 99 項商品後，我們就可以將其傳往下一階段「排序」，透過不同的排序方法進而得到不同的推薦序列。

3.2 多模型排序

在排序階段我們的目標為，將這些候選的商品中最有可能被使用者點擊到的商品排序到較前面的位子。而在不同的推薦環境下，不同的排序模型都會產生不一樣的排序結果。

最終我們選擇了四種方法作為我們推薦系統使用的排序模型，這四種方法分別是：

1. 基於最近鄰居法 (k-nearest neighbors)[3] 的排序

2. 使用簡易神經網路的排序
3. 使用 DIN(Deep Interest Network) 模型 [33] 的排序
4. 使用 DIEN(Deep Interest Evolution Network) 模型 [32] 的排序

接下來我們將會針對這四種排序方法進行介紹。

3.2.1 基於最近鄰居法 (**k-nearest neighbors**) 的排序

由於我們在前一階段「候選」時所使用的策略為單路候選，即是只使用了商品的 embedding 特徵去進行 Top – 99 的相似度搜尋。不同於前段2.2.1.1所介紹的多路候選，其候選的商品混雜了多類別特徵，我們使用的單路候選策略所產生的原始商品排序，本質上即是依照當下瀏覽商品的 embedding 去進行的最近鄰居法搜尋。

3.2.2 使用簡易神經網路的排序

在有了候選的商品後，我們嘗試訓練一個全連接 (Fully-Connected) 架構的簡易神經網路模型 NN 去預測每項候選商品會被使用者點擊的機率，在依照點擊機率的高低去重排商品序列，以達到排序越高的商品使用者點擊率越高的目的。

具體來說，我們會先從歷史資料中整理出使用者的商品點擊序列 (Clickstream)，在整理出商品點擊序列後我們會將其兩兩分組如圖3.4，每一組商品對 (pair) 即為神經網路訓練點擊率的正樣本資料 (positive samples)，而負樣本資料 (negative samples) 的商品對我們則是依據商品 w_i 的出現頻率 $f(w_i)$ 搭配 Word2vec 中常使用的訓練方法「負採樣 (Negative Sampling)」去進行生成，負採樣的具體機率公式如下：

$$P(w_i) = \frac{f(w_i)^{3/4}}{\sum_{j=0}^n (f(w_j)^{3/4})} \quad (3.4)$$

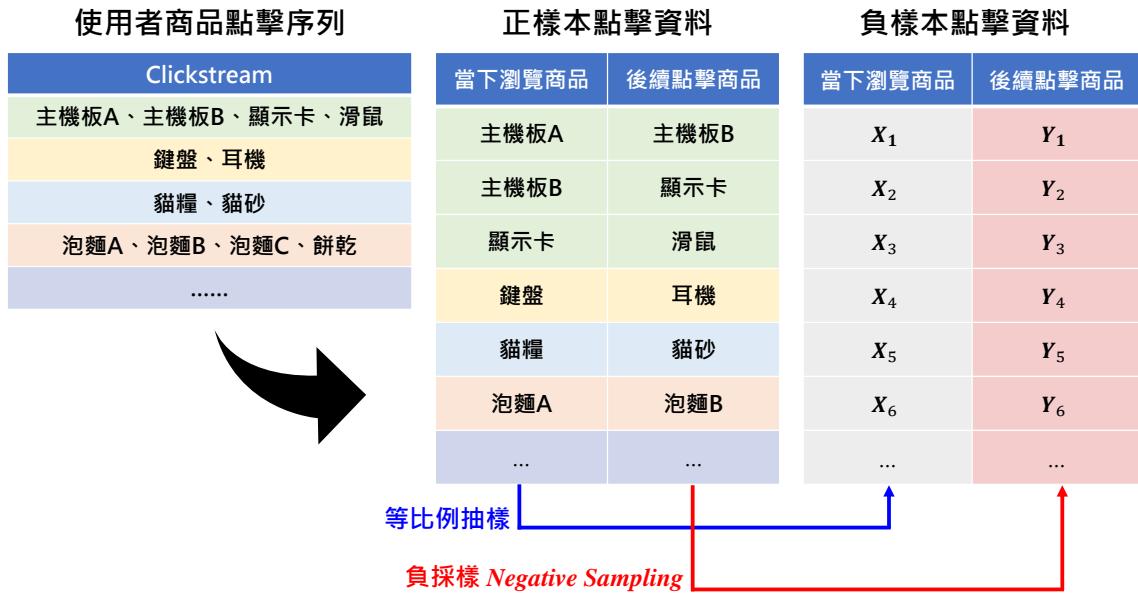


圖 3.4: 將商品點擊序列轉換為正負樣本資料

有了正負樣本的訓練資料後，我們即可開始針對點擊率去訓練神經網路模型，最後我們會用訓練好的神經網路模型去預測所有當下商品對應其候選商品的點擊機率，並且依據預測出的點擊率高低去調整商品在序列中的位子，進而產生新的排序。從訓練模型到產生出新排序的整體流程如圖3.5所示。

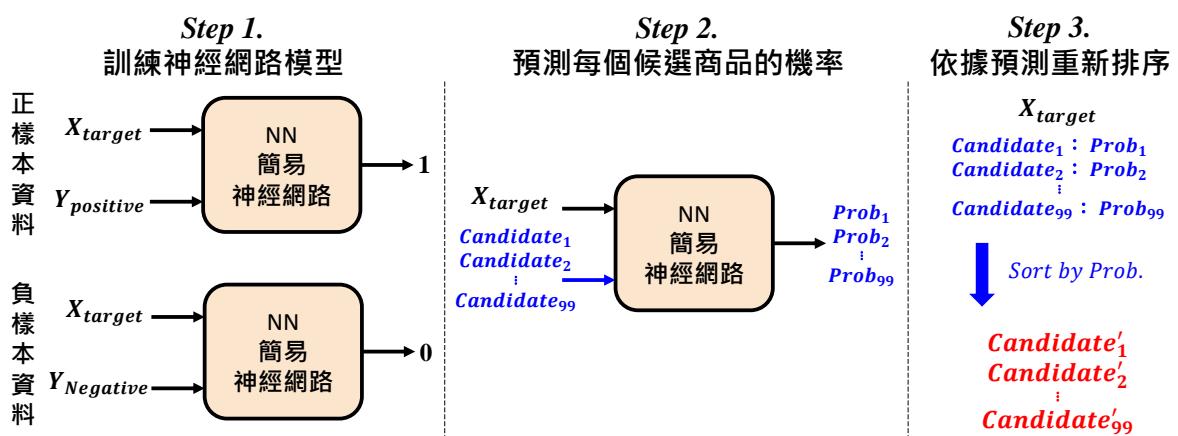


圖 3.5: 從訓練模型到產生出新排序的整體流程

3.2.3 使用 DIN(Deep Interest Network) 模型的排序

深度興趣網路 (Deep Interest Network, DIN) 是由阿里巴巴集團旗下阿里媽媽精準定向廣告算法團隊於 KDD2018 所提出的針對電商場景深入理解用戶興趣的預估模型。



圖 3.6: 從歷史瀏覽行為計算候選得分 [33]

此模型的核心思想為，利用使用者的歷史瀏覽資料，透過 Attention 機制 [4] 去計算候選商品的得分。這邊以原始論文中的例子來做說明，如圖3.6所示，假設今天要計算某位使用者對於一件候選大衣的點擊率，在他的歷史瀏覽紀錄中，瀏覽過一個杯子和瀏覽過另一件大衣這兩個行為相比，肯定是瀏覽過另一件大衣參考價值高，也就是說模型在預測的時候，對使用者不同行為的注意力是不一樣的，相關的歷史行為看重一些，不相關的歷史行為甚至可以忽略。

$$V_u = f(V_a) = \sum_{i=1}^N w_i * V_i = \sum_{i=1}^N g(V_i, V_a) * V_i \quad (3.5)$$

DIN 模型的注意力形式化表達如式3.5所示。 V_u 是用戶的 embedding 向量， V_a 是候選商品的 embedding 向量， V_i 是用戶 u 的第 i 次行為的 embedding 向量，因為這裡用戶的行為就是瀏覽商品，所以行為的 embedding 的向量就是那次瀏覽的商品的 embedding 向量。

因為加入了注意力機制， V_u 從過去 V_i 的加和變成了的 V_i 加權和， V_i

的權重 w_i 就由 V_i 與 V_a 的關係決定，也就是式3.5中的 $g(V_i, V_a)$ ，即「注意力得分」。 $g(V_i, V_a)$ 函數其實就是一個注意力激活單元 (activation unit)，其本質上也是一個小神經網路，具體結構如圖3.7右上角的 Activation unit 所示。

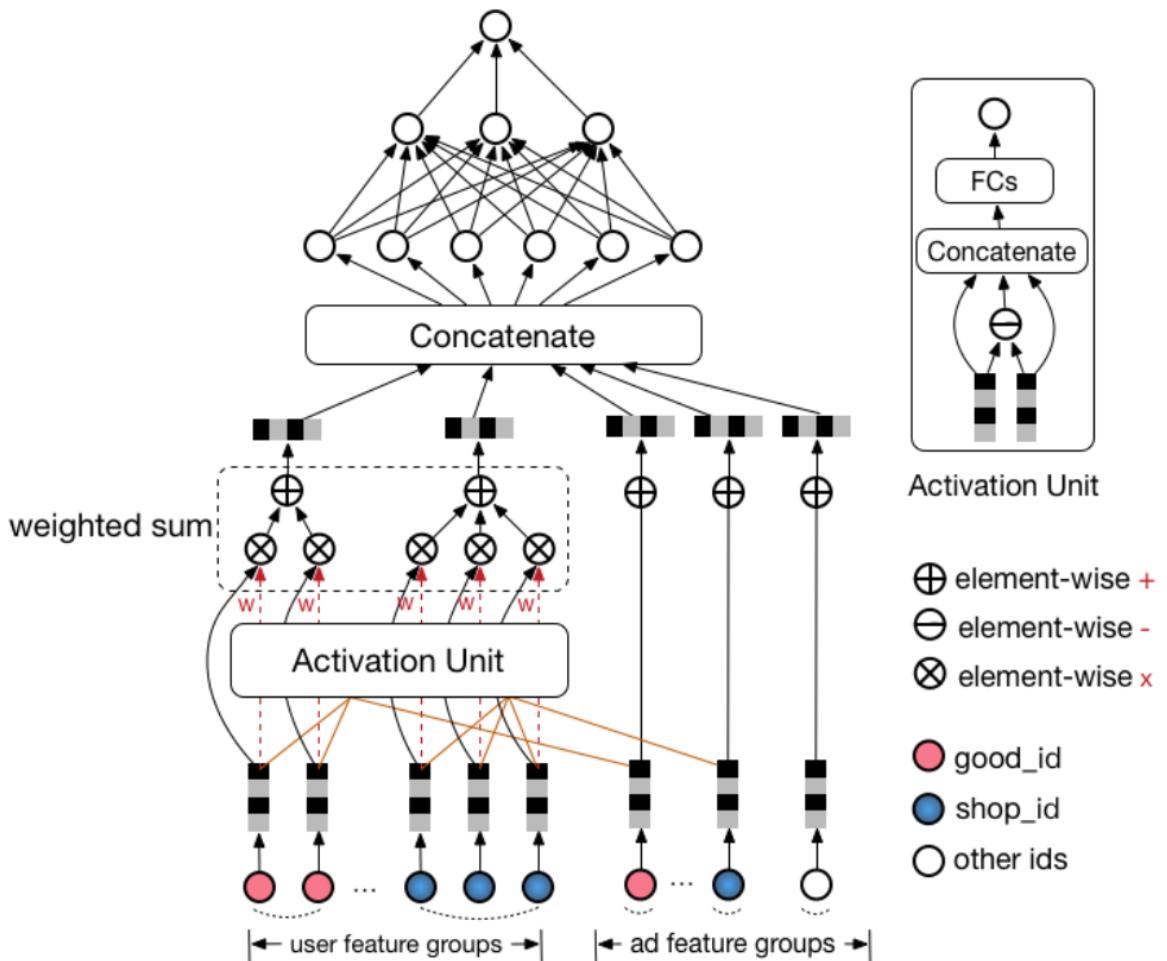


圖 3.7: DIN 模型結構 [33]

3.2.4 使用 DIEN(Deep Interest Evolution Network) 模型的排序

用戶興趣演化模型 (Deep Interest Evolution Network, DIEN) 是阿里巴巴團隊於 2019 年提出的 DIN 模型演化版本，該模型的創新點在於其使用了時間序列模型去模擬了用戶興趣演化的過程。

過往的模型如 DIN 等，都鮮少對序列演化訊息進行推薦，其主要都只是用歷史序列中不同行為對候選商品的重要度進行評分，這樣的評分法是與時間序列無關的。

這邊以一個簡單的例子來說明序列訊息的價值，假設上週一位使用者在挑選一雙運動鞋，那麼該位使用者上週的行為序列都會集中在運動鞋這類商品上，但他購買完運動鞋後，這週的興趣可能會變成購買運動耳機。使用序列模型去捕捉使用者興趣的話，我們就能得知用戶近期購買運動耳機的機率明顯會高於再買一雙運動鞋，某種意義上就能夠建立使用者從運動鞋到運動耳機的興趣轉移率。如果這類轉移率在全局統計意義足夠高的話，那麼使用者購買運動鞋時，推薦運動耳機也會變成不錯的選項，也就是說兩者的用戶群體可能是一致的。

為了達成上述使用序列訊息的構想，阿里巴巴對 DIN 模型進行了改進，建構了 DIEN 模型結構，如圖3.8所示。

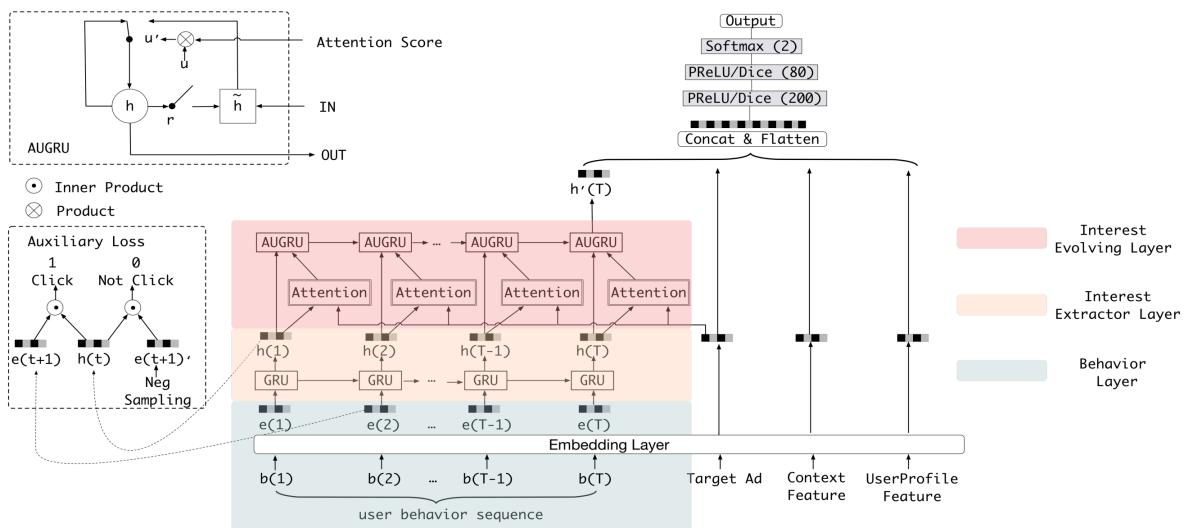


圖 3.8: DIEN 模型結構 [32]

圖中彩色的部分即為「興趣演化網路」，是一種產生使用者興趣 embedding 的方法，網路共分成三層，從下而上分別是：

1. 行為序列層：主要是把原始的商品 id 行為序列轉換為 embedding 行為序列。

2. 興趣抽取層：主要是使用 GRU 結構模擬使用者興趣轉移的過程，抽取使用者興趣。
3. 興趣進化層：主要是透過興趣抽取層產生的基礎加上注意力機制，模擬與候選商品相關的興趣演化過程。

「興趣演化網路」的最終輸出為用戶興趣向量 $h'(T)$ ，其會與其他特徵共同進入全連接層，最後於輸出層輸出使用者對於候選商品的候選得分。

3.3 Switch

在不同的排序模型產生出多種不一樣的商品排序後，我們的推薦系統即可進入最後一個階段 Switch。在這個階段我們的終極目標為在多種不一樣的商品推薦排序中，依據當下的環境特徵，選擇出一個會將使用者接下來想點擊的商品放在高位的推薦排序，如圖3.9所示。

The diagram shows a comparison of four recommendation models (kNN, NN, DIN, DIEN) across five items. A red arrow points from a box containing 'real-world task environment features' to a dashed red box labeled 'Best Model'.

| 真實任務情境 | | kNN | NN | DIN | DIEN |
|--------|--------|-----------|-----------|-----------|-----------|
| 當下瀏覽商品 | 後續點擊商品 | | | | |
| 主機板A | 主機板B | 1 主機板C | 1 主機板B | 1 滑鼠 | 1 主機板C |
| | | 2 主機板B | 2 鍵盤 | 2 主機板C | 2 顯示卡 |
| | | 3 滑鼠 | 3 滑鼠 | 3 顯示卡 | 3 鍵盤 |
| | | 4 鍵盤 | 4 主機板C | 4 主機板B | 4 滑鼠 |
| | | 5 顯示卡 | 5 顯示卡 | 5 鍵盤 | 5 主機板B |
| | | ... | ... | ... | ... |

當下任務環境特徵

- 使用者資訊
- 任務時間
- 當下商品出現頻率
- 之前瀏覽過的商品

找出最佳排序模型

Best Model

圖 3.9: 於衆多排序結果中找出最佳排序模型

我們將上述的「依據特徵，從多種不同的排序中選出一種做為推薦」的任務視為一種分類問題，在開始嘗試去訓練分類器之前，我們發現在

實際資料中的類別分布狀況非常的不平衡，也就是說在大多數的案例中都是某幾種模型排序表現得比較好，而其餘的模型排序只有在少數的案例中勝出。

這樣的數據不平衡問題會對分類器的建模造成極大影響，所以在訓練分類器之前我們決定先對資料進行一些採樣處理。面對於數據不平衡的資料通常會有兩種處理方式，一種是將多數樣本中不具代表性的資料去除，以免造成雜訊，稱為欠採樣 (Under-sampling)，而另一種則是將少數樣本用某種方式重複抽樣或合成新樣本，稱為過採樣 (Over-sampling)。在經過多次實驗後，我們發現在我們的實驗數據中使用屬於欠採樣的 Tomek Links 演算法 [29] 能夠有效提升分類器的準確性。

Tomek Links 演算法會針對資料中所有樣本進行遍歷，假設今天有兩個樣本點 (X, Y) 分別屬於不同的類別，也就是一個為多數樣本一個為少數樣本，我們可以去計算他們之間的距離 $d(X, Y)$ ，此時如果我們找不到第三個樣本點 Z ，使得任一樣本點的距離 $d(X, Z) \leq d(Y, Z) \leq d(X, Y)$ 比剛剛算出來的 $d(X, Y)$ 還小的話，我們就稱這兩個樣本點 (X, Y) 為 Tomek Link。如圖3.10所示，該演算法的關鍵思路在於，找出資料邊界中那些鑑別度不高的樣本，將其視為雜訊於以剔除，使得剔除後的類別樣本點能更好被區分開來。

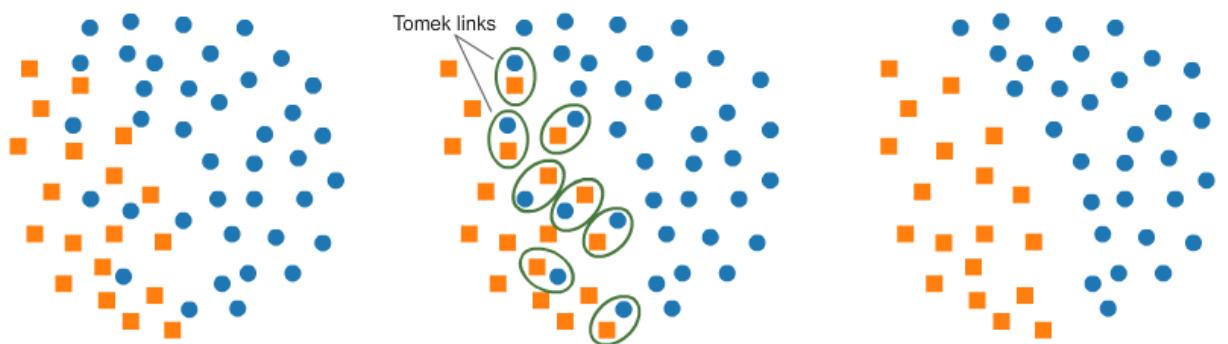


圖 3.10: Tomek Links 演算法示意圖 [1]

在透過 Tomek Links 演算法對樣本進行欠採樣處理後，我們即可開始對我們的 Switch 分類器進行訓練。回顧我們的目標「依據當下的環境特徵，選擇出最好的排序」，這邊如何選擇當下推薦環境中的特徵，作

為分類器訓練時所需的參數是影響整個分類結果好壞的關鍵，在我們的推薦情境下，能使用的特徵通常有使用者編號 (uid)、當下瀏覽商品 (iid)、當下瀏覽商品的類別 (cid)、當下瀏覽商品的出現頻率 (iid_freq)、使用者過往點擊的商品序列 (hist_iid) 與使用者過往點擊的商品類別序列 (hist_cid) 等等。

經過多次實驗後最終我們選擇使用「當下瀏覽商品 (iid) 的 embedding」與「當下瀏覽商品出現頻率 (iid_freq)」作為我們訓練分類器所使用的特徵，這樣的選擇其實並無絕對，主要還是要依據當下的推薦場景與資料的特性去進行實驗與測試才能找出最佳的特徵。

最後，具體的分類器模型我們則是同樣嘗試使用簡易神經網路還有 XGBoost[6] 去進行實作，其在最終的預測結果上都有不錯的表現。

四、 實驗結果與分析

在本章節中，我們會先介紹我們實驗所使用的資料集，接著說明實驗的流程與詳細的操作方法，然後再依據不同的資料集分別展示其實驗結果，最後我們會針對實驗結果中所發現的現象進行進一步的討論。

4.1 資料集介紹

在本研究中我們選擇了兩種資料集進行實驗與比較，分別是中國大型電商集團阿里巴巴旗下「淘寶網」的用戶行為資料集 [28] 與台灣某中小型電商的用戶行為資料集。在此之所以選擇這兩種資料集的原因是因為，我們希望能夠探討目前主流的深度學習架構是否不管是應用於大型電商或中小型電商之中都能有不錯的表現，而不會受限於中小型電商商品量較少、用戶使用率較低等資料特性而影響效能。

4.1.1 淘寶用戶行為資料集

我們所使用的淘寶用戶行為資料集其來源為阿里雲的天池大數據競賽平台。該資料集主要是用於隱式反饋推薦問題的研究，其包含了 2017 年 11 月 25 日至 2017 年 12 月 3 日，共 9 天之間約一百萬位隨機用戶的所有使用行為，行為類型主要可以分成四種，如表4.1所示。

而資料集的組織形式則和 MovieLens-20M[14] 類似，即資料中的每一列都會對應一筆用戶行為，每筆用戶行為都是由用戶 id、商品 id、商品類別 id、行為類型和時間戳記所組成，並以逗號分隔。關於資料集中

每一種特徵的詳細敘述如表4.2所示。最後，表4.3列出了該資料集中各項特徵的統計數值。

表 4.1: 淘寶資料集行為列表

| 行為類型 | 說明 |
|------|-----------------|
| pv | 商品詳情頁 pv, 等價於點擊 |
| buy | 商品購買 |
| cart | 將商品加入購物車 |
| fav | 收藏商品 |

表 4.2: 淘寶資料集資料組織形式

| 特徵名稱 | 說明 |
|---------|--------------------------------------|
| 用戶 id | 整數型別，序列化後的用戶 id |
| 商品 id | 整數型別，序列化後的商品 id |
| 商品類型 id | 整數型別，序列化後的商品所屬類別 id |
| 行為類型 | 字串型別，包括 ('pv', 'buy', 'cart', 'fav') |
| 時間戳記 | 行為發生的時間 |

表 4.3: 淘寶資料集特徵統計值

| 維度 | 數量 |
|--------|-------------|
| 用戶數量 | 987,994 |
| 商品數量 | 4,162,024 |
| 商品類別數量 | 9,439 |
| 所有行為數量 | 100,150,807 |

4.1.2 台灣電商用戶行為資料集

在台灣電商資料集這邊，我們則是擁有 2019 年 3 月 1 日至 2019 年 6 月 30 日之間，共 122 天的資料，與淘寶資料集的組織形式相似，該資料集中每一列資料都對應了一筆用戶行為，但不同的是台灣電商資料集中的每列資料都比起淘寶資料集還多紀錄了瀏覽器 id(viewer id)、購物階段 id(session id) 與瀏覽頁面種類 (page type) 這三個特徵。

表4.4列出了台灣電商資料集中的每項特徵的說明，而表4.5與表4.6則分別列出了行為與頁面的類型，最後表4.7則統計了該資料集中各項特徵的統計數值。

表 4.4: 台灣電商資料集資料組織形式

| 特徵名稱 | 說明 |
|-------------|-------------------------|
| user_id | hex 字串，序列化後的用戶 id |
| viewer_id | 字串編碼，序列化後的瀏覽器 id |
| session_id | 字串編碼，序列化後的使用階段 id |
| event_name | 字串型別，行為類型 |
| page_type | 字串型別，頁面類型 |
| category_id | hex 字串，序列化後的商品所屬類別 id |
| product_id | 串列型別，內容元素為 hex 編碼之商品 id |
| time_local | 行為發生的時間 |

表 4.5: 台灣電商資料集行為列表

| 行為類型 | 說明 |
|-------------|----------|
| pageload | 載入頁面 |
| cartadd | 將商品加入購物車 |
| cartremove | 將商品移出購物車 |
| favoriteadd | 收藏商品 |

表 4.6: 台灣電商資料集頁面列表

| 頁面類型 | 說明 |
|----------------|-------|
| product-detail | 商品詳情頁 |
| category | 商品類別頁 |
| shopping-cart | 購物車頁面 |
| checkout | 結帳頁面 |
| portal | 首頁 |

表 4.7: 台灣電商資料集特徵統計值

| 維度 | 數量 |
|--------|------------|
| 用戶數量 | 294,937 |
| 商品數量 | 626,311 |
| 商品類別數量 | 43,558 |
| 所有行為數量 | 38,786,948 |

4.2 實驗流程與細節

本研究全程使用 Python 3.7 作為實驗的開發環境，圖4.1展示了整體的實驗流程。

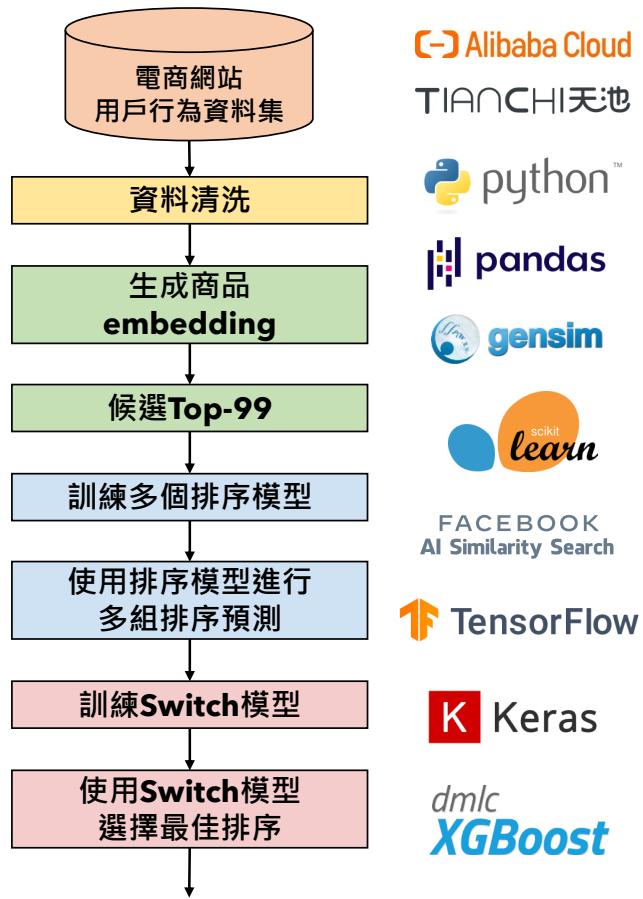


圖 4.1: 實驗流程圖

在一開始我們需要先針對資料集中的資料進行清洗，由於用戶行為資料的部分來源為使用者裝置的 log 資訊，我們發現當使用者的裝置異

當時往往都會有錯誤的時間戳記發生，例如未來時間等等，為了避免干擾我們在一開始時就會先將這類資料進行剔除。然而除了上述異常資料外，在用戶行為資料中另一個常發生的現象就是爬蟲機器人，由於電商網站中的商品價格經常是同業競爭的指標，所以在用戶行為資料中不乏會有爬蟲機器人的存在，其行為特徵即為在短時間內瀏覽非常大量的商品，這類非真實用戶的行為我們也會在資料清洗階段進行剔除。

在將資料清洗完畢後，我們會先將資料分成三個區段 Part A、Part B 與 Part C，以方便後續的操作與描述。兩資料集於此三階段的長度如圖4.2分別所示。

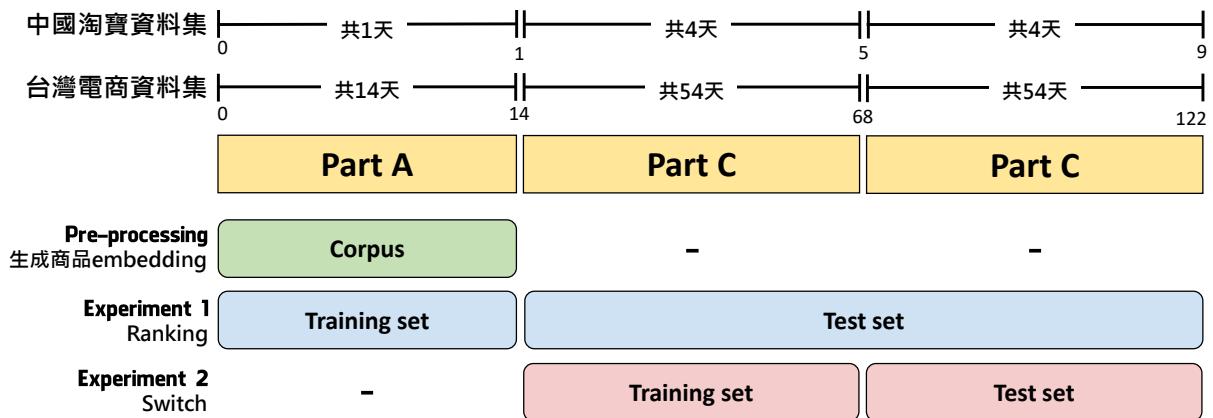


圖 4.2: 將資料依時間分段

在有了乾淨的資料後我們就會開始進行商品 embedding 的生成工作，在這步驟我們主要會使用基於 Python 的開源套件庫 Gensim[23] 去進行實作。由於我們的推薦情境為「當使用者點擊進入某項商品頁面後，頁面周圍該持續推薦的項目」，所以我們主要會使用資料集中用戶行為類型為「pv」或「product-detail」的資料去進行商品的 embedding 生成，而我們用來生成商品 embedding 的訓練資料範圍即為圖4.2所示的 Part A 區段。

在生成每項商品的 embedding 之後，我們就會開始進行候選的動作，由於資料集中的商品數量衆多，過往在使用 scikit-learn[22] 對每件商品的 embedding 進行相似度計算時往往會花費大量的時間，為了降低時間花

費，我們在該步驟改使用了業界中常用的 FAISS (Facebook AI Similarity Search)[15] 工具，透過近似最近鄰居搜尋 (Approximate Nearest-Neighbor Search) 的方法來加速我們整體的候選速度。

在排序模型部份，我們的簡易神經網路模型主要是使用 Keras[8] 框架基於 TensorFlow[19] 後端進行搭建，而 DIN 與 DIEN 模型則分別取自原始論文 [33] 所提供的程式碼¹ 與阿里巴巴釋出的開源深度學習框架 X-DeepLearning[2] 進行實作，在訓練排序模型所用的訓練資料範圍同生成商品 embedding 階段，皆為圖4.2中 Part A 區段，而 Part B 與 Part C 區段則為測試資料，有各個模型預測出的排序結果。

最後在 Switch 部份，我們先使用了 Imbalanced-learn[16] 套件庫去實作 Tomek Links 演算法對訓練資料進行欠採樣，然後再使用簡易神經網路和 XGBoost 來訓練分類器，在此階段 Part B 與 Part C 已有排序模型的預測結果，分類器會使用 Part B 作為訓練資料，並用 Part C 做為測試資料去預測分類的結果。

4.3 實驗結果

4.3.1 評量指標

在實驗結果的評量指標上我們使用的是「平均排名」，也就是說在每一個點擊推薦任務中，各個排序模型都會將候選得到的 99 項商品，依據其預測之點擊率從高排序到低，假設在某任務中某模型將 X 商品排序於第一位，而使用者真的於該任務中點擊了 X ，則該模型於該任務中的商品推薦排名即為 1，將每一個排序模型於所有任務的排序結果取平均值即為此指標。

¹Github repository: <https://github.com/zhougr1993/DeepInterestNetwork>，我們使用的版本於 commit: 634310a

4.3.2 排序模型實驗結果

表4.8與表4.9分別展示了四個排序模型於「淘寶資料集」與「台灣電商資料集」的排序實驗結果(圖4.2淺藍色Test set區段)，從實驗結果中我們可以發現，無論在哪個資料集中，kNN與NN這兩個簡易模型的表現都優於複雜的DIN與DIEN模型，而DIN與DIEN這兩個複雜模型相比，DIN則略優於DIEN一些。

表 4.8: 淘寶資料集排序實驗結果

| | kNN | NN | DIN | DIEN |
|--------|---------------|--------|--------|--------|
| 平均排名 | 30.023 | 31.933 | 47.801 | 50.171 |
| 標準差 | 28.040 | 27.834 | 27.998 | 28.826 |
| 最小值 | 1 | 1 | 1 | 1 |
| 第一四分位數 | 6 | 8 | 24 | 25 |
| 第二四分位數 | 20 | 24 | 46 | 50 |
| 第三四分位數 | 49 | 51 | 71 | 75 |
| 最大值 | 99 | 99 | 99 | 99 |

表 4.9: 台灣電商資料集排序實驗結果

| | kNN | NN | DIN | DIEN |
|--------|--------|---------------|--------|--------|
| 平均排名 | 20.314 | 20.231 | 44.337 | 46.766 |
| 標準差 | 24.190 | 22.446 | 27.139 | 30.257 |
| 最小值 | 1 | 1 | 1 | 1 |
| 第一四分位數 | 3 | 3 | 20 | 17 |
| 第二四分位數 | 9 | 11 | 42 | 48 |
| 第三四分位數 | 29 | 29 | 67 | 73 |
| 最大值 | 99 | 99 | 99 | 99 |

4.3.3 Switch 模型實驗結果

有了四個模型的排序結果後，我們將其部分資料用於訓練分類器，目標為透過分類器去尋找出各個模型於不同任務中的使用時機，也就是說

預測出每個任務中排名表現較佳的模型，並將該模型的排序作為該任務的推薦結果。

每個任務都經過這樣的選擇後即可產生出新的排序，我們在此稱其為 Switch 排序，而透過這樣的選擇方法產生的排序會有其排名的上限，也就是每個任務都選擇到最佳模型的排序結果，我們將此上限指標稱為 Oracle，表4.10與表4.11分別展示了兩資料集的 Switch 排序結果(圖4.2淺紅色 Test set 區段)與 Oracle 指標，我們可以發現在兩個資料集中，Switch 排序結果都能夠比原始四個排序結果擁有更佳的排名表現。

表 4.10: 淘寶資料集 Switch 實驗結果

| | Ranking | | | | Switch | | | Oracle |
|--------|---------|-------|-------|-------|--------------|---------|--|--------|
| | kNN | NN | DIN | DIEN | NN | XGBoost | | |
| 平均排名 | 30.08 | 32.00 | 47.66 | 50.02 | 27.75 | 27.94 | | 12.54 |
| 標準差 | 28.00 | 27.84 | 27.97 | 28.79 | 26.96 | 27.20 | | 13.28 |
| 最小值 | 1 | 1 | 1 | 1 | 1 | 1 | | 1 |
| 第一四分位數 | 6 | 8 | 24 | 25 | 5 | 5 | | 3 |
| 第二四分位數 | 20 | 24 | 46 | 50 | 18 | 18 | | 8 |
| 第三四分位數 | 49 | 51 | 71 | 75 | 44 | 45 | | 18 |
| 最大值 | 99 | 99 | 99 | 99 | 99 | 99 | | 91 |

表 4.11: 台灣電商資料集 Switch 實驗結果

| | Ranking | | | | Switch | | | Oracle |
|--------|---------|-------|-------|-------|--------|--------------|--|--------|
| | kNN | NN | DIN | DIEN | NN | XGBoost | | |
| 平均排名 | 20.29 | 20.62 | 44.65 | 46.63 | 19.72 | 19.23 | | 8.92 |
| 標準差 | 24.08 | 22.63 | 27.13 | 30.25 | 23.57 | 23.19 | | 11.14 |
| 最小值 | 1 | 1 | 1 | 1 | 1 | 1 | | 1 |
| 第一四分位數 | 3 | 4 | 21 | 16 | 3 | 3 | | 2 |
| 第二四分位數 | 10 | 11 | 43 | 47 | 9 | 9 | | 4 |
| 第三四分位數 | 29 | 30 | 67 | 73 | 28 | 27 | | 11 |
| 最大值 | 99 | 99 | 99 | 99 | 99 | 99 | | 96 |

除了平均排名外，我們還分析了各個排序方法在兩資料集中預測結果在 Top-5、Top-10、Top-15 與 Top-20 中所占的比例，如表4.12與表4.13所示，我們可以從分析結果發現在兩資料集中無論哪一種 Switch 方法，其在前面排名的占比都比最佳的排序模型所佔的比例高，在此可見我們提出的 Switch 方法的有效性。

表 4.12: 淘寶資料集排名前 20 Top 占比統計

| | Ranking | | | | Switch | | | Oracle |
|----------|---------|-----|-----|------|--------|---------|-----|--------|
| | kNN | NN | DIN | DIEN | NN | XGBoost | | |
| Top - 5 | 24% | 20% | 5% | 6% | 26% | 26% | 41% | |
| Top - 10 | 35% | 30% | 10% | 10% | 38% | 38% | 58% | |
| Top - 15 | 43% | 39% | 15% | 15% | 47% | 47% | 71% | |
| Top - 20 | 50% | 46% | 21% | 20% | 54% | 54% | 79% | |

表 4.13: 台灣電商資料集排名前 20 Top 占比統計

| | Ranking | | | | Switch | | | Oracle |
|----------|---------|-----|-----|------|--------|---------|-----|--------|
| | kNN | NN | DIN | DIEN | NN | XGBoost | | |
| Top - 5 | 38% | 33% | 5% | 8% | 39% | 40% | 56% | |
| Top - 10 | 52% | 48% | 12% | 17% | 53% | 54% | 73% | |
| Top - 15 | 61% | 58% | 18% | 24% | 62% | 63% | 82% | |
| Top - 20 | 67% | 65% | 25% | 29% | 68% | 69% | 88% | |

最後，我們訂出了一個 Switch 效能指標去評斷在不同的資料集與排序模型中，Switch 排序結果的效能，具體公式如式4.1所示，表4.14展示了該指標於兩資料集的數值。

$$\frac{\text{最佳排序模型的平均排名} - \text{Switch的平均排名}}{\text{最佳排序模型的平均排名} - \text{Oracle的平均排名}} \times 100\% \quad (4.1)$$

表 4.14: 兩資料集中不同 Switch 模型的效能指標

| | NN | XGBoost |
|---------|---------------|--------------|
| 淘寶資料集 | 13.28% | 12.20% |
| 台灣電商資料集 | 5.01% | 9.32% |

4.4 探討商品於訓練資料中出現次數與預測結果之關係

根據前段所展示的實驗結果，我們可以發現 kNN 與 NN 這兩個模型的排名表現是非常相近的，雖然在大多數的情況下兩者相比都是 kNN 略為勝出，但在此我們不禁好奇，在本研究的實驗下簡易神經網路 NN 與 kNN 這樣傳統的機器學習演算法相比，真的在每個任務情境下都略輸於 kNN 嗎？是否有 NN 表現較佳的任務情境？為了驗證這個問題我們開始針對實驗結果進行分析，過程中我們探討了任務發生的時間、目標使用者過往的點擊情形與任務當下的商品類別等等衆多因素，最後發現不管是在哪個資料集「任務當下商品於訓練資料中出現的次數」都是一個影響 NN 排名表現的重要因素。

如圖4.3與圖4.4所示，橫軸為當下商品於訓練資料中的出現次數，而縱軸則是當下商品於訓練資料中出現此次數的任務平均排名，我們可以發現在商品出現次數較低的時候 NN 在淘寶資料集的排名表現都是略輸的，而在台灣電商資料集這邊兩者則是有著差不多的結果。但是隨著商品出現次數的增加，我們可以發現到 kNN 的排序表現越來越差，不同的是 NN 的表現卻是越來越好，這樣的現象在兩個資料集中都有發生，因此我們能夠推斷我們的簡易神經網路排序模型 NN 對於當下商品於訓練資料中出現次數較多的任務情境會有著較佳的推薦效果。

最後，我們將同樣的分析方法應用於 DIN 與 DIEN 中，發現在不同的資料集下這兩個模型的確也有著表現有越來越佳或越來越差的現象，只是其相對於 kNN 與 NN 來說表現較沒那麼明顯，如圖4.5與圖4.6所示。

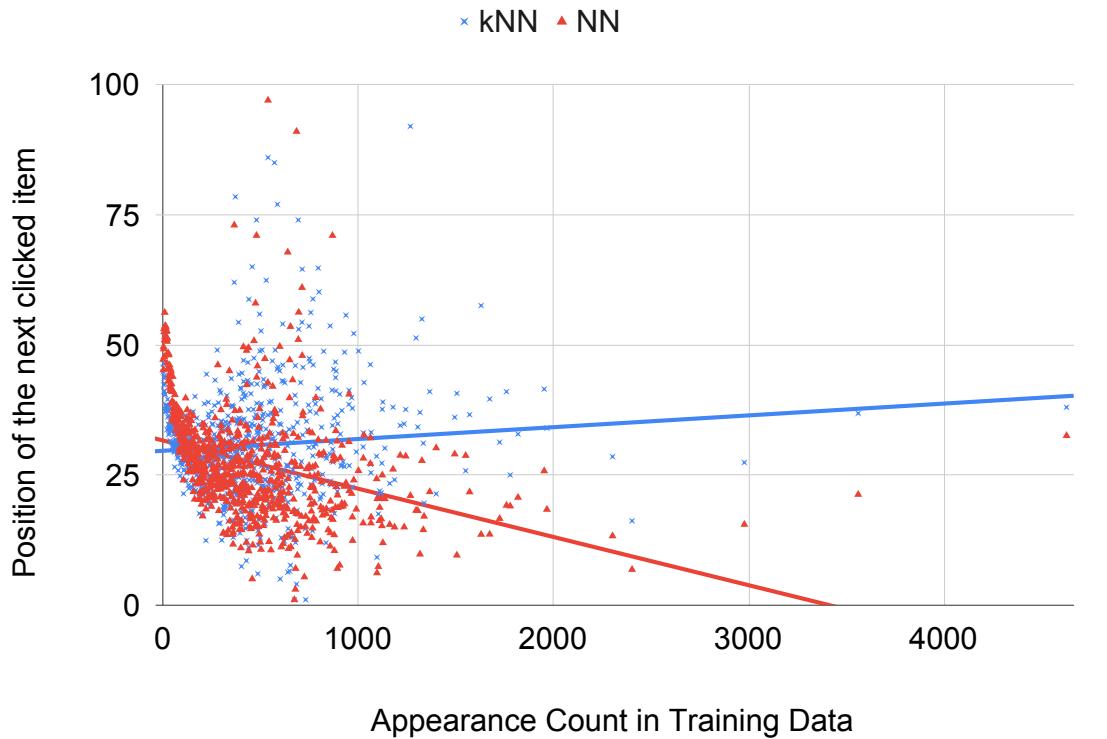


圖 4.3: 淘寶資料集商品出現次數與排名關係圖 (kNN v.s. NN)

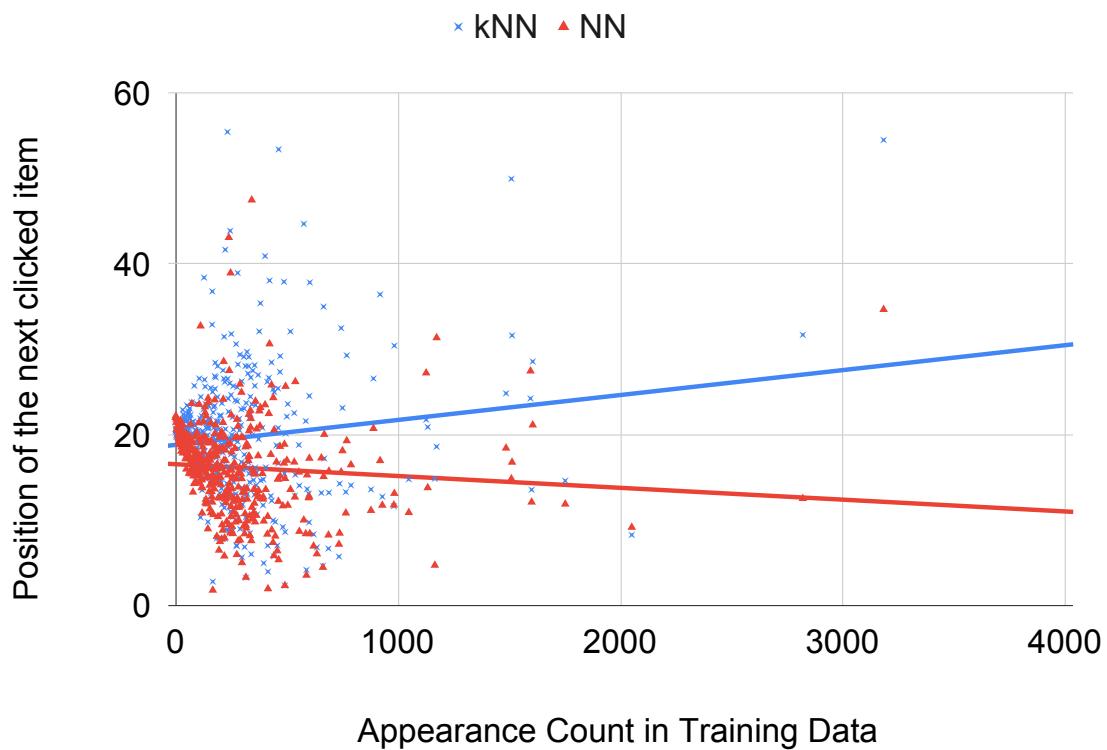


圖 4.4: 台灣電商資料集商品出現次數與排名關係圖 (kNN v.s. NN)

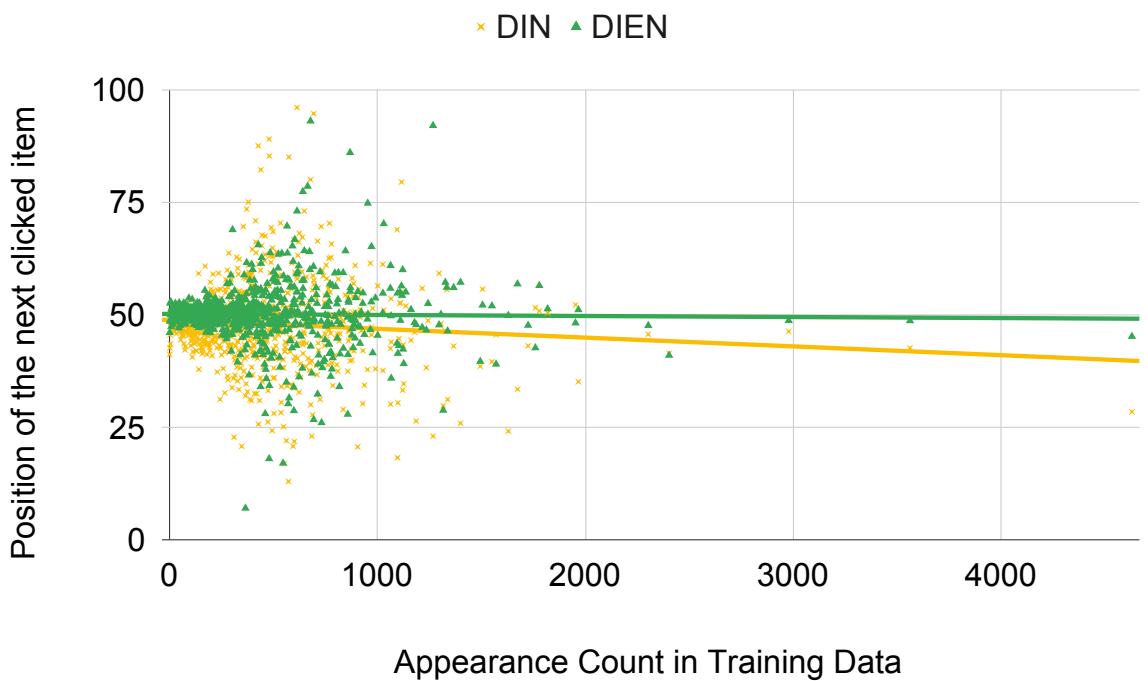


圖 4.5: 淘寶資料集商品出現次數與排名關係圖 (DIN v.s. DIEN)

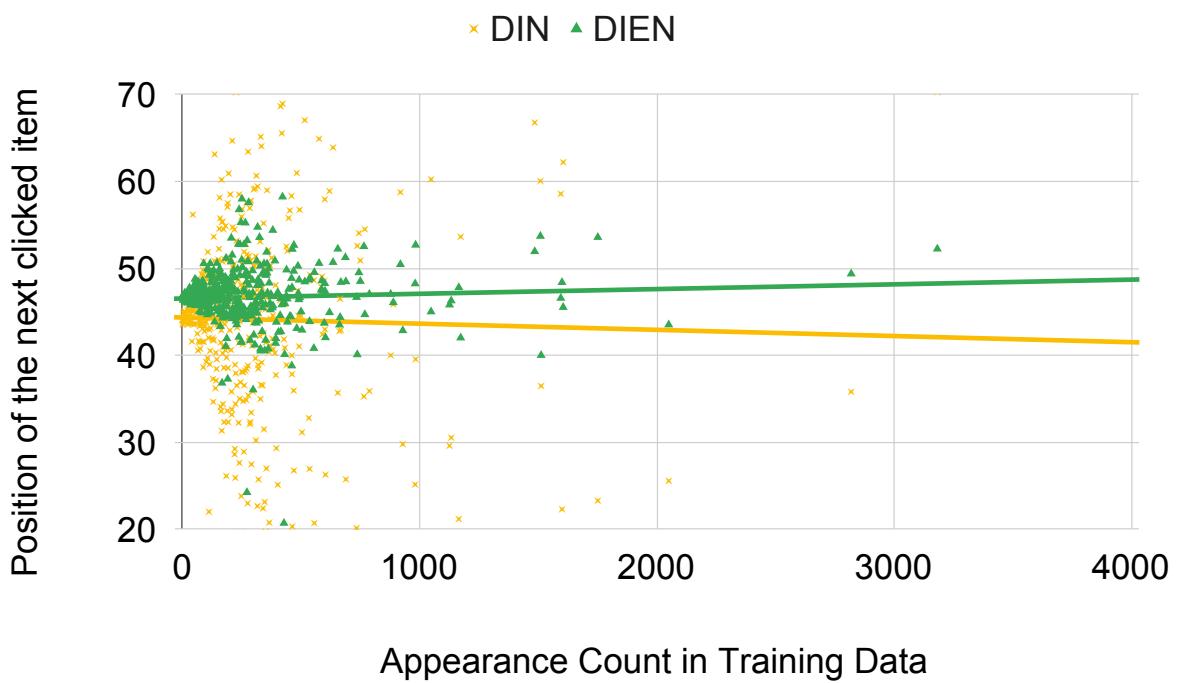


圖 4.6: 台灣電商資料集商品出現次數與排名關係圖 (DIN v.s. DIEN)

4.5 使用更多的資料去訓練 DIN 與 DIEN

在我們的實驗結果中，阿里巴巴所提出的 DIN 與 DIEN 模型其排名表現都沒有特別亮眼，在此我們認為可能的原因是因為訓練資料量不夠多，由於 DIN 與 DIEN 這類深度學習模型的目標是透過使用者的歷史行為對候選商品進行捕捉，其不像 kNN 與 NN 模型主要是對任務情境去進行學習，所以需要更多的使用者行為資料，為了驗證此想法我們嘗試增加了 DIN 與 DIEN 模型於兩資料集所用的訓練資料量並重新進行實驗，最終的實驗結果如表4.15與表4.16所示。

表 4.15: 淘寶資料集 DIN 與 DIEN 增加訓練資料量的實驗結果

| 訓練資料量 | DIN | | DIEN | |
|--------|-------|--------------|-------|-------|
| | 1 天 | 4 天 | 1 天 | 4 天 |
| 平均排名 | 47.66 | 33.42 | 50.02 | 51.80 |
| 標準差 | 27.97 | 27.38 | 28.79 | 28.58 |
| 最小值 | 1 | 1 | 1 | 1 |
| 第一四分位數 | 24 | 10 | 25 | 27 |
| 第二四分位數 | 46 | 26 | 50 | 53 |
| 第三四分位數 | 71 | 52 | 75 | 77 |
| 最大值 | 99 | 99 | 99 | 99 |

表 4.16: 台灣電商資料集 DIN 與 DIEN 增加訓練資料量的實驗結果

| 訓練資料量 | DIN | | DIEN | |
|--------|-------|--------------|-------|-------|
| | 14 天 | 61 天 | 14 天 | 61 天 |
| 平均排名 | 44.65 | 42.49 | 46.63 | 47.14 |
| 標準差 | 27.13 | 25.54 | 30.25 | 28.85 |
| 最小值 | 1 | 1 | 1 | 1 |
| 第一四分位數 | 21 | 21 | 16 | 23 |
| 第二四分位數 | 43 | 40 | 47 | 42 |
| 第三四分位數 | 67 | 62 | 73 | 73 |
| 最大值 | 99 | 99 | 99 | 99 |

依據實驗結果我們可以發現，DIN 模型在增加訓練資料量後，於兩個資料集中的表現皆有所提升，可見對於 DIN 模型來說，擁有足夠的訓練資料是提升其準確率的關鍵。至於另一方面，我們可以發現 DIEN 模型就算於兩個資料集中都增加了訓練資料量，其實驗結果仍然沒有比較好，對於這個現象我們推測可能的原因是，如同前段3.2.4所述，DIEN 模型的目標是透過時間序列資料對使用者的興趣演化進行預測，因此我們認為這種方法是需要在擁有長期穩定用戶資料的條件下才能發揮他的預測效果。

五、結論與未來展望

5.1 結論

透過不同規模的資料集與多個模型的實驗，我們發現近年來成功應用於大型服務的複雜點擊率預測模型並不全然適用於中小規模的服務，對於資料量較少的中小型服務來說，應用簡易的模型反倒就能有不錯的預測效果。

而在衆多模型的排序實驗結果中，我們發現「任務當下商品於訓練資料中出現的次數」是一個影響預測結果好壞的關鍵，透過「尋找在不同的推薦情境下各個模型的使用時機」這個概念，我們提出了多模型選擇的推薦系統架構，最終實驗結果證明透過這樣的方法能夠有效的提升整體點擊率預測的準確度。

5.2 未來展望

在本篇研究中我們使用的「淘寶用戶行為資料集」雖然為大規模服務，但其資料天數較少，多少還是會影響到需要較大資料量的複雜模型，在往後的研究，我們會試著尋找更完善的資料集，並且嘗試使用更多的點擊率預測模型進行實驗，希望透過完善的資料能讓我們更加掌握各種模型的使用時機，以增進整體預測的效能。

參考文獻

- [1] Rafael Alencar. 2017. Resampling strategies for imbalanced datasets. Kaggle. (2017). <https://www.kaggle.com/rafjaa/resampling-strategies-for-imbalanced-datasets>.
- [2] Alibaba. 2018. X-deeplearning. Github. (2018). <https://github.com/alibaba/x-deeplearning>.
- [3] Naomi S Altman. 1992. An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, 46, 3, 175–185. DOI: 10.2307 / 2685209.
- [4] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. (2014). arXiv: 1409 . 0473 [cs . CL].
- [5] Oren Barkan and Noam Koenigstein. 2016. Item2vec: neural item embedding for collaborative filtering. *2016 IEEE 26th International Workshop on Machine Learning for Signal Processing (MLSP)*, (September 2016). DOI: 10 . 1109/mlsp . 2016 . 7738886.
- [6] Tianqi Chen and Carlos Guestrin. 2016. Xgboost: a scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (KDD ’ 16). Association for Computing Machinery, San Francisco, California, USA, 785–794. ISBN: 9781450342322. DOI: 10 . 1145 / 2939672 . 2939785.
- [7] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, Rohan Anil, Zakaria Haque, Lichan Hong, Vihan Jain, Xiaobing Liu, and Hemal Shah. 2016. Wide & deep learning for recommender systems. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems* (DLRS 2016). Association for Computing Machinery, Boston, MA, USA, 7–10. ISBN: 9781450347952. DOI: 10 . 1145 / 2988450 . 2988454.
- [8] François Chollet et al. 2015. Keras. (2015). <https://keras.io>.

- [9] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems* (RecSys '16). Association for Computing Machinery, Boston, Massachusetts, USA, 191–198. ISBN: 9781450340359. DOI: 10.1145/2959100.2959190.
- [10] Yufei Feng, Fuyu Lv, Weichen Shen, Menghan Wang, Fei Sun, Yu Zhu, and Keping Yang. 2019. Deep session interest network for click-through rate prediction. *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, (August 2019). DOI: 10.24963/ijcai.2019/319.
- [11] David Goldberg, David Nichols, Brian M. Oki, and Douglas Terry. 1992. Using collaborative filtering to weave an information tapestry. *Commun. ACM*, 35, 12, (December 1992), 61–70. ISSN: 0001-0782. DOI: 10.1145/138859.138867.
- [12] Mihajlo Grbovic and Haibin Cheng. 2018. Real-time personalization using embeddings for search ranking at airbnb. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (KDD '18). Association for Computing Machinery, London, United Kingdom, 311–320. ISBN: 9781450355520. DOI: 10.1145/3219819.3219885.
- [13] Long Guo, Lifeng Hua, Rongfei Jia, Binqiang Zhao, Xiaobo Wang, and Bin Cui. 2019. Buying or browsing?: predicting real-time purchasing intent using attention-based deep network with multiple behavior. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (KDD '19). Association for Computing Machinery, Anchorage, AK, USA, 1984–1992. ISBN: 9781450362016. DOI: 10.1145/3292500.3330670.
- [14] F. Maxwell Harper and Joseph A. Konstan. 2015. The movielens datasets: history and context. *ACM Trans. Interact. Intell. Syst.*, 5, 4, Article 19, (December 2015), 19 pages. ISSN: 2160-6455. DOI: 10.1145/2827872.
- [15] Jeff Johnson, Matthijs Douze, and Herve Jegou. 2019. Billion-scale similarity search with gpus. *IEEE Transactions on Big Data*. ISSN: 2372-2096. DOI: 10.1109/tbdata.2019.2921572.
- [16] Guillaume Lemaître, Fernando Nogueira, and Christos K. Aridas. 2017. Imbalanced-learn: a python toolbox to tackle the curse of imbalanced datasets in machine learning. *Journal of Machine Learning Research*, 18, 17, 1–5. ISSN: 1532-4435. <http://jmlr.org/papers/v18/16-365.html>.
- [17] G. Linden, B. Smith, and J. York. 2003. Amazon.com recommendations: item-to-item collaborative filtering. *IEEE Internet Computing*, 7, 1, 76–80. DOI: 10.1109/MIC.2003.1167344.

- [18] Jiaqi Ma, Zhe Zhao, Xinyang Yi, Jilin Chen, Lichan Hong, and Ed H. Chi. 2018. Modeling task relationships in multi-task learning with multi-gate mixture-of-experts. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (KDD ’18). Association for Computing Machinery, London, United Kingdom, 1930–1939. ISBN: 9781450355520. DOI: 10.1145/3219819.3220007.
- [19] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: large-scale machine learning on heterogeneous systems. (2015). <https://www.tensorflow.org/>.
- [20] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. (2013). arXiv: 1301.3781 [cs.CL].
- [21] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2* (NIPS’13). Curran Associates Inc., Lake Tahoe, Nevada, 3111–3119.
- [22] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. 2011. Scikit-learn: machine learning in python. *Journal of Machine Learning Research*, 12, 85, 2825–2830. ISSN: 1532-4435. <http://jmlr.org/papers/v12/pedregosa11a.html>.
- [23] Radim Řehůřek and Petr Sojka. 2010. Software framework for topic modelling with large corpora. In *Proceedings of LREC 2010 workshop New Challenges for NLP Frameworks*. University of Malta, Valletta, Malta, 46–50. ISBN: 2-9517408-6-7.
- [24] Xin Rong. 2014. Word2vec parameter learning explained. (2014). arXiv: 1411.2738 [cs.CL].
- [25] Ying Shan, T. Ryan Hoens, Jian Jiao, Haijing Wang, Dong Yu, and JC Mao. 2016. Deep crossing: web-scale modeling without manually crafted combinatorial features. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (KDD ’16). Association for Computing

- Machinery, San Francisco, California, USA, 255–262. ISBN: 9781450342322. DOI: 10.1145/2939672.2939704.
- [26] M. Slaney and M. Casey. 2008. Locality-sensitive hashing for finding nearest neighbors [lecture notes]. *IEEE Signal Processing Magazine*, 25, 2, 128–131. DOI: 10.1109/MSP.2007.914237.
- [27] 2006. *Introduction to data mining*. (1st edition). Addison-Wesley. Chapter 8, 500. ISBN: 0321321367.
- [28] TIANCHI. 2018. User behavior data from taobao for recommendation. Website. (May 2018). <https://tianchi.aliyun.com/dataset/dataDetail?dataId=649>.
- [29] I. Tomek. 1976. Two modifications of cnn. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-6, 11, 769–772. DOI: 10.1109/TSMC.1976.4309452.
- [30] Weinan Zhang, Tianming Du, and Jun Wang. 2016. Deep learning over multi-field categorical data. *Advances in Information Retrieval*, 45–57. ISSN: 1611-3349. DOI: 10.1007/978-3-319-30671-1_4.
- [31] Zhe Zhao, Lichan Hong, Li Wei, Jilin Chen, Aniruddh Nath, Shawn Andrews, Aditee Kumthekar, Maheswaran Sathiamoorthy, Xinyang Yi, and Ed Chi. 2019. Recommending what video to watch next: a multitask ranking system. In *Proceedings of the 13th ACM Conference on Recommender Systems* (RecSys ’19). Association for Computing Machinery, Copenhagen, Denmark, 43–51. ISBN: 9781450362436. DOI: 10.1145/3298689.3346997.
- [32] Guorui Zhou, Na Mou, Ying Fan, Qi Pi, Weijie Bian, Chang Zhou, Xiaoqiang Zhu, and Kun Gai. 2019. Deep interest evolution network for click-through rate prediction. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33, (July 2019), 5941–5948. ISSN: 2159-5399. DOI: 10.1609/aaai.v33i01.33015941.
- [33] Guorui Zhou, Xiaoqiang Zhu, Chenru Song, Ying Fan, Han Zhu, Xiao Ma, Yanghui Yan, Junqi Jin, Han Li, and Kun Gai. 2018. Deep interest network for click-through rate prediction. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (KDD ’18). Association for Computing Machinery, London, United Kingdom, 1059–1068. ISBN: 9781450355520. DOI: 10.1145/3219819.3219823.

附錄 A 實驗程式碼

本研究的實驗程式碼公開於 Github 軟體原始碼代管服務平台
連結網址: <https://github.com/ncu-dart/Rec-Switch>
詳細操作方法請參閱 README.md 文件