應用強化式學習探勘活動來源網頁

廖于晴
中央大學資訊工程系
sunnyliao29@gmail.com

張嘉惠
中央大學資訊工程系
chia@csie.ncu.edu.tw

摘要—隨著交通的便利性提升，旅行不再只是爲了觀光或攝影，更多的是希望可以參加當地的活動，深入體驗當地的文化。大多數的活動發起人，例如政府、企業和組織，都會在其網站上更新活動資訊，因此如何有效率的找到任何網站中的活動來源的所在頁面的任務，即稱爲活動來源頁面的探勘。在本文中，我們使用了一個深度強化學習模型來訓練我們的爬蟲代理人 **(Agent)**，並使用兩個階段來訓練我們的爬蟲，分別是預訓練和微調。在預訓練階段，模型使用有限的標記數據進行訓練，其中每個 **Episode** 都有固定的時間步長，在微調階段，代理使用未標記的數據與基於活動來源頁面的分類器爲獎勵系統進行訓練。我們的爬蟲代理人在這邊會通過一個自適應閾值學習是否繼續探索，因此在微調的階段時，每個 **Episode** 的步數是變動的。最後，我們的爬蟲代理人在眞實的數據集上以 **1.3** 的單位成本（每個事件源頁面的點擊次數）實現了 **74%** 的投資回報率（即準確度）。

Index Terms—活動網頁搜尋, 活動來源網頁的探勘, 強化式學習, 活動來源頁面的分類

I. 緒論

對於大多數人來說，尋找當地活動是旅行或搬到新城市時的常見需求，然而，當地的活動通常散佈在各網站中不易搜尋。過去，企業和政府組織通過電視或平面媒體宣傳他們的活動，而後隨著智慧型手機和社交網絡的普及，Eventbrite、Meetup、Allevents.in、Ticketmaster、Stubhub、CityTalk、ACCUPASS 等商業付費票務網站成爲了新的活動資訊來源，但由於許多地方活動都是以社區爲基礎的，例如校園音樂會、亦或是當地的文化活動，此類的活動資訊比較不會在活動社交網站 (Event Based Social Networks, EBSN) 或是售票網站中進行廣告，因此除非使用者知道關鍵字，否則活動消息不易傳播出來。

有鑑於活動搜索在網絡上的重要性的提升，Google 從 2017 年開始與多個 EBSNs（如 Eventbrite、Meetup、Allevents.in、Ticketmaster、Stubhub 等）合作，將其內容授權予 Google 搜尋引擎中顯示。Google Map 也於 2019 年推出了新的活動功能，使一些企業和活動管理者能夠通過其 Android 應用程序上的“*Contribution Tab*”創建與特定地點相關聯的活動 [1]，展示了事件與地點兩者彼此之間相關聯的重要性。

相較於被動等待用户通過活動共享平台或基於活動的社交網絡貢獻活動，另一種方法是主動從 Web 收集和擷取活動。例如，Google 的研究團隊（Foley et al. [1]）即採用遠程監督，藉由公共網絡語料庫 ClueWeb12[2] 中 **Schema.org** 的 217,000 個特定活動，透過將網頁中的區域劃分爲不同種類，並藉由系統構建了一個多類分類器，該分類器會基於 13 個特徵去學習四種類型的權重，分別是“What”、“Where”、“When”和“Other”。假設一個活動必須包含所有三種類型，他們則會將各個類型的分數相乘以確定該區域是否包含一個活動。換句話說，活動擷取模型目標在於希望可以應用於 Web 上的所有頁面，因此導致效率不高。

爲了加快這一過程，Wang et al. [2] 添加了一個額外的步驟來在活動提取之前預測網頁是否包含任何活動資訊。同時，他們還採用了網絡資訊自動擷取的技術 [3]–[5] 來處理多活動公告的頁面，提高活動提取的準確性。然而，這種方法仍然需要在整個網域中去進行爬行，因此效率也不高。

在本文中，我們希望以目標導向的方式去進行活動頁面的搜尋，在進行活動偵測和擷取之前，有效率地收集可能的活動網頁 URL。我們提出三個步驟的活動網頁收集方式，包括：活動來源頁面的搜尋、資料

[1]https:// www.searchenginejournal.com/ google-maps-rolls-out-new-public-events-feature/300151/

[2]ClueWeb12: https://lemurproject.org/clueweb12/

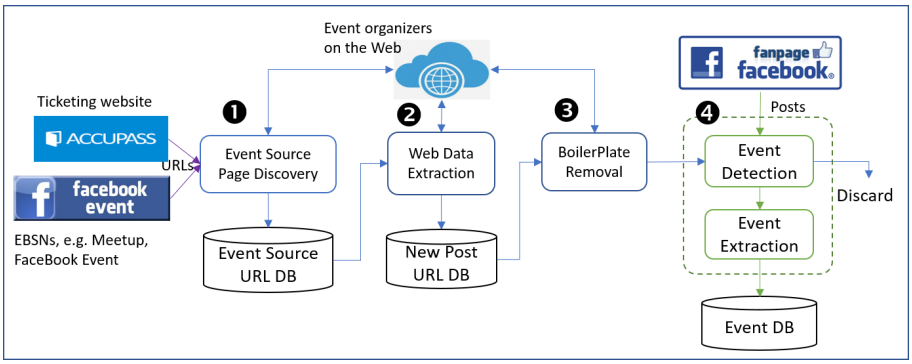

圖 1. 建立活動資料庫的流程總覽

集擷取、以及模板移除，如圖 1，第一個步驟是活動來源頁面的探勘，我們會將潛力活動主辦者的網站當作起始頁面，任務是在此網站上找到發布活動的頁面，換句話說，目標是活動發佈集合地而不是單個活動。

當我們獲得了活動來源頁面的 URL 後，我們就可以應用無監督式網頁資料集擷取 (第二步驟) 技術，例如 IEPAD [6] 和 DEPTA [7]，爲每個活動來源進行多筆資料擷取技術，獲取最新活動消息的 URL。相較於大範圍從整個 Web 搜尋新的活動網頁相比，這種方式的優勢在於我們可以從源頭上監控所有活動的發布狀態，效率很高。此外，我們可以通過無監督的數據提取技術快速的篩選網頁中的最新消息頁面的所有發文的 URL，儘管並非最新消息中的每個文章都包含活動，但後續可以再通過活動偵測模組 (第四步驟) 去過濾非活動的消息。不過，由於活動偵測和擷取的模型都只接受以自然語言編寫的文本，因此在應用活動偵測模塊之前，我們再應用模板移除 [8](第三步驟) 來提取每個消息的主要內容。

在本文中，我們主要關注活動來源頁面的探勘 (Event Source Discovery)，我們提出了一個兩階段強化式學習的框架來訓練我們的爬蟲代理人。在第一階段，基於有限的標記數據設計固定的報酬回饋方式，並使用每一輪 (Episode) 固定步數的方式，讓我們的爬蟲代理人與環境交互來進行訓練。在第二階段，模型使用未標記的網站進行微調，並採用一個活動來源頁面的分類器給予報酬。第二階段中每一輪的嘗試步伐數由自適應閾值控制，也就是由爬蟲代理人自行判定是否繼續探索。我們所提出的爬蟲代理人在眞實世界的資料集上以 1.3 單位成本（每個活動來源頁面的點擊次數）實現了 74% 的投資回報率（即準確度）。

II. 相關研究

活動來源網頁的探勘這個任務類似於焦點式爬蟲的問題，目的皆是希望可以找到與目標主題相關的網頁。我們希望爬蟲代理人可以學習評估每一個連結的分數，並根據此估計數值選擇最有希望的連結進行點擊。最早將強化式學習應用於爬蟲的任務是 Grigoriadis 和 Paliouras (SETN 2004) [9]。在他們的設定中，每個狀態（網頁）由詞頻最高的 500 個關鍵字所組成的 500 維二進制值的特徵向量表示，並訓練神經網絡來估計進行估計。

爲了減少狀態-動作空間大小，Partalas 等人 [10] 提出了一種新穎的自適應焦點式爬蟲，它採用強化式學習來選擇合適的分類器，進而評估網頁中的每一個連結，因爲分類器的引入可以大大減少強化式學習方法的搜索空間，將問題簡單化。爲了提高爬行效率，Han 等人 [11] 提出了一種考慮錨文字和 Page Rank 狀態和動作表示方法，爲了減少狀態-動作空間，他們通過從頁面和連結中提取特徵，提出了狀態和動作的廣義表示，而演算法的部分，前面所提到的 Grigoriadis 和 Paliouras (SETN 2004) 是使用的 Temporal-Difference 演算法，Han 等人則是選擇 SARSA 作爲他們的演算法，同時，他們還研究了一種同步方法，重新計算前沿所有鏈接的分數，以及一種只計算當前頁面所有連結分數的異步方法。Mishra [12] 等人提出了在線學徒評論家 (Apprentice-Critic) 自適應焦點式爬蟲，這個方法也是基於強化式學習來選擇合適的分類器，不過和我們的實驗最大的不同在於代理人的更新方式，Mishra 等人是用 batch 的方式，而我們在每一個 step 都會不斷的調整。

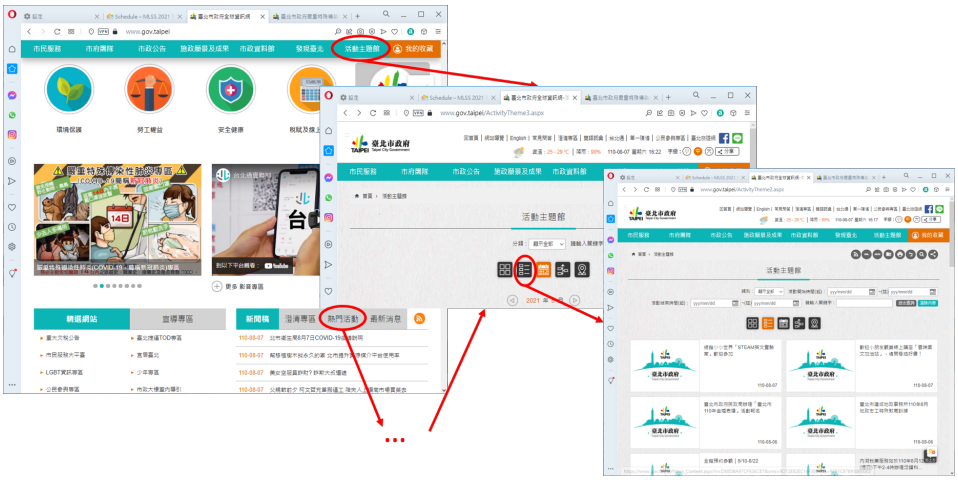

圖 2. 活動來源網頁探勘範例: 以台北市府網站爲例

III. 問題定義

由於我們的主要目的是從任何給定的起始頁面中發現活動來源頁面，本質上是一個搜尋問題。假設爬蟲代理人從起始頁面 URL 開始進行探索，通過預測每個錨節點 (anchor link) 連結到活動消息發佈網頁的機率，決定該要點擊的錨結點。爬蟲代理人的目標是有效率地找到提供最新消息或是活動發佈網頁，即活動來源網頁。舉例來說，圖2中顯示從台北市政府網站開始搜尋活動公告來源網頁，紅色圈圈及箭頭分別代表錨結點以及點擊動作。圖2中顯示有兩個錨節點可以到達活動發佈網頁，但是兩條路徑所使用的點擊成本並不相同，爬蟲代理人的目標不僅要找到活動來源網頁，也要控制點擊成本。

從強化式學習的角度來看，每次點擊 a 爬蟲代理人會從目前狀態 s 遷移至另一個狀態 s'，而環境則返回該次所選動作的獎勵 r，強化式學習的目標即在學習如何選擇動作以最大化累積的環境獎勵。這個過程通常被抽象爲馬爾可夫決策過程（Markov decision processes，MDP）。簡單來說，強化學習問題由狀態 (S)、動作 (A)、狀態轉移函數 (T:S×A→S)、以及獎勵函數 (R:S×A→R) 所構成。將爬蟲代理人與環境互動每個時間點的狀態、動作以及獎勵 (<s, a, r>) 串起來，即爲稱之爲一輪 (episode)，可以 τ 來表示: $\tau = (<s_0, a_0, r_0>, <s_1, a_1, r_1>, ..., <s_T, a_T, r_T>)$。爬蟲代理人的目標即爲最大化累積獎勵 $R(\tau)$:

$$R(\tau) = \sum_{t=0}^{T} r_t, \tag{1}$$

對網頁爬蟲來說，每個狀態都有不同數量尚未嘗試的錨節點可供選擇。假設 $successors(s_t)$ 是目前所在頁面 s_t 中所有的對外錨節點的集合，$frontier(s_t)$ 爲前一時間點尚未探索的鏈結 ($frontier(s_0)=\emptyset$)，因此爬蟲代理人在時間 t 可以選擇的候選動作集合爲 $successors(s_t)$ 以及 $frontier(s_t)$ 的聯集，也就是說 $Action(s_t) = frontier(s_t) \cup successors(s_t)$。若爬蟲代理人在時間 t 選擇了動作 $a_t \in Action(s_t)$，則 $frontier(s_{t+1}) = Action(s_t) - \{a_t\}$，亦即將其他未點擊的動作保留在下次候選動作集中。假設爬蟲代理人從起始頁面 s_0 開始進行探索尋找活動來源頁面，$a_0 \in successors(s_0)$，每個錨節點僅有錨節點文字內容等有限資訊，爬蟲代理人可以選擇一個節點進行點擊並跳轉至下一頁，或選擇不進行點擊結束一輪互動。

IV. 研究方法

爬蟲代理人的工作是在時間點 t 時從候選動作集 a_t 中選擇最有機率找到活動來源頁面的錨鏈結來點擊。我們假設 π_θ 爲我們爬蟲代理人的評估函數，我們會用 π_θ 去評估每個一個候選動作的分數，如式子 2所示:

$$a_t = \underset{a \in Action(s_t)}{\arg\max}\ \pi_\theta(a) \tag{2}$$

強化式學習的特點在於尋找“利用已有知識”以及“探索未知領域”兩者之間的平衡，爬蟲代理人會有 $1-\epsilon$ 的機率去點擊由 π_θ 評分中最高分的錨節點，以及 ϵ 的機率去隨機點擊動作集中的錨節點。

在這邊 ϵ 的設置的方法是採用變動的數值，而不是固定常數。由於代理人在早期沒有得到充分的訓練，

ϵ=0.9，也就是讓他可以高度自由的去探索頁面，但在之後隨著 epochs 數量的增加，我們會減少 ϵ 直到 $\epsilon = 0.1$，具體來說，ϵ 每兩個 epoch 減少 0.1。

A. 獎勵函數

在環境獎勵上，理想上我們希望環境對活動來源頁面給予 +1 回饋，並給予其他頁面負的回饋。因此我們也爲此訓練一個分類器擔任此項任務。我們事先人工標記了從一組種子網站 *Seed* 中連結到的網頁，並判斷每個網頁是否爲活動來源網頁。不過爲了加快訓練過程，我們使用兩步驟的架構來訓練我們的爬蟲代理人，分別爲預訓練階段和微調階段。在預訓練階段是在前述已標記數據上，也就是種子網站 *Seed* 上訓練的；而微調階段則是在未標記數據上訓練的。

根據不同的階段，我們使用了不同的函式當作報酬函式。在預訓練階段，根據我們標記的資料，我們的報酬函式如下式 3：

$$r_t(a_t) = \begin{cases} 1, if\ a_t\ is\ event\ source \\ max(-0.1, -0.1 - \lfloor \frac{epoch}{2} \rfloor * 0.1), o/w \end{cases} \tag{3}$$

在微調的階段中，我們使用未標記的資料，因此報酬函式是由二元分類器所給予的，報酬函式如式子 4，在這邊我們所使用的活動來源網頁的二元分類器是一個準確度與召回率各自爲 76% 跟 73% 的分類器，模型架構將在第 E 小節描述。

$$r_t(a_t) = \phi_\omega(a_t) \tag{4}$$

B. 固定步伐與自調適步伐

在預訓練階段我們將最多步伐 T 設置爲 3，以節省訓練時間。而微調階段或是測試階段則是透過閾值的設置，讓爬蟲代理人決定是否繼續去探勘這個網站。閾值設計如公式 5，這式子中所使用是我們預訓練資料集裡的起始頁面。

$$\eta_\theta = \frac{1}{|Seed|} \sum_{p \in Seed} \frac{1}{Z(\tau_p)} \sum_{t=0}^{T-1} \pi_\theta(a_t\ in\ \tau_p) * w(a_t) \tag{5}$$

其中 $Z(\tau_p) = \sum_{t=0}^{T-1} w(a_t)$，而 $w(a_t)$ 則代表權重。我們使用預訓練過程最後一次 epoch 所搜集的互動過程 $\tau_p, p \in Seed$ 來計算點擊閾值，主要是因爲通過在這些頁面上進行多次重複的訓練已能準確的點擊活動來源頁面。在預訓練階段 $w(a) = 1,\ \forall a \in \mathcal{A}$，而微調階段則是 $w(a) = \phi_\omega(a)$，其中 $\phi_\omega(a_t)$ 爲活動來源頁面的二元分類器。

C. 演算法

Algorithm 1及 2 分別是我們的預訓練及微調階段的演算法。在預訓練階段中，我們將在一組起始頁 *Seed* 上重複進行 50 回合的訓練，而在微調步驟中，我們針對每一起始頁面只會進行一次的訓練，不會進行重複的訓練且是採用變動步伐的方式去收集 eqisode。

Algorithm 1 預訓練階段演算法

Input Starting pages $p_1, p_2, \ldots$
Randomly Initialize π_θ with weight θ

repeat
  episodes = []
  for each page p_i **do**
    τ = CollectEpisode(p_i, π_θ)
    episodes.add(τ)
  end for
  Update π_θ with mini-batch 3 from episodes by minimizing the loss function
until epoch=50

Algorithm 2 微調階段演算法

Input Starting pages $p_1, p_2, \ldots$ and agent π_θ

episodes = []
for each page p_i **do**
  τ = CollectEpisode(p_i, π_θ)
  episodes.add(τ)
end for
Update π_θ with mini-batch 10 from episodes by minimizing the loss function

微調和預訓練兩階段有兩個差別：在預訓練步驟中，我們會重複的在同一組起始頁面進行運練，而微調時一個頁面只會進行一次的訓練。在預訓練時，我們將 Batch size 設定爲 3，而在微調的步驟中，我們將 Batch size 設定爲 10，主要是因爲微調的時候使用分類器作爲獎勵函式是相對不穩定的，因此我們設置了比預訓練步驟更大的批次大小。

Algorithm 3 爲與環境互動收集 episode 的演算法。第 1 行所設置的 maxsteps 是指我們希望在此起始頁上查找的深度，我們使用閾值來決定在每個時間點中是否要進行下一回合的點擊，如果所有可選動作的

Algorithm 3 CollectEpisode

Input Starting page p, π_θ
Output episode τ
Initialize τ= [], $s_0 = p$, Actions = $\emptyset$

1: **for** $T = 0, 1, 2, \ldots,$ *maxsteps* **do**
2: Use π_θ to compute each anchor node n in s_T
3: Action = Action $\cup$ successor(s_T)
4: **if** $max_{n \in Action(s_T)} \pi_\theta(n) \leq \eta$ **then**
5: break, quit the episode
6: return τ
7: **end if**
8: $a_T = \begin{cases} \arg\max_{n \in Action} \pi_\theta(n), & prob.\ 1-\epsilon \\ a\ random\ node\ In\ Action, & prob.\ \epsilon \end{cases}$
9: $\tau = \tau \cup (s_T, a_T, r(a_T))$
10: set $s_{T+1} = a_T$
11: **end for**
12: return τ

表 I
預訓練、微調階段參數設定

Training	Pre-Training	Fine-Tuning
Maxsteps	3(固定)	3(變動)
Threhold	η=0	Eq.(5)
Reward	Eq. (3)	Eq. (4)
Epoch	50	1
Batch Size	3	10
ϵ-greedy	0.9 to 0.1	0.1
Evaluation	**Pre-Training**	**Fine-Tuning**
Episode	maxsteps=10, η=Eq.(5), ϵ=0	
Epoch, Batch Size	Epoch=1, Batch Size=1	

評估值都小於閾值，我們就會結束互動並返回所收集的 episode(第 4-6 行)；否則我們繼續互動，將有 $1-\epsilon$ 的機率點擊得分最高的錨節點，及 ϵ 的機率點擊隨機一個錨節點 (第 8 行)。

預訓練步驟和微調步驟與訓練階段和評估階段之間有許多不同的參數設置。表I 顯示了預訓練、和微調階段之間參數的設定，以及評估階段的不同設置。

D. 模型設計

我們使用深度神經網絡來設計爬蟲代理人模型，並將其結合多任務學習，讓我們的代理人可以更深入的學習到特徵表示，圖 3是我們的模型架構。我們從網頁 p 中提取所有錨節點 $n_0, ...n_k$ 並將它們定義爲 $AN(p) = \{n_0, n_1, ..., n_k\}$，我們從每個節點中提取兩個特徵分別是錨文字與 HTML 的標籤路徑 n_i=<$AnchorText_i, TagPath_i$>，其中錨文字是指包含外部連結的節點中的的文字，標籤路徑則是指將 HTML 檔案中建構成 DOM Tree 後一路從節點追朔到根節點的最短路徑。

架構圖 3的左側，我們主要用來處理錨文字，我們稱之爲 Anchor-Net，在 Anchor-Net(參數 θ_a) 中，我們將錨文字一個一個放入預訓練的 BERT 模型中，然後將錨文字轉換成向量表示，因爲在這裡我們使用的是預訓練的 BERT，所以我們通過多層感知器對 BERT 所輸出的向量進行了微調。圖 3右側主要用來處理標籤路徑，稱之爲 Tag-Net(參數 θ_t)，在 Tag-Net 中，我們將頁面中的所有標籤路徑放入我們的模型中，然後使用詞袋模型將它們轉換成向量，在這邊我們使用 BiLSTM 讓我們的網絡可以考慮上一個或下一個節點的資訊。

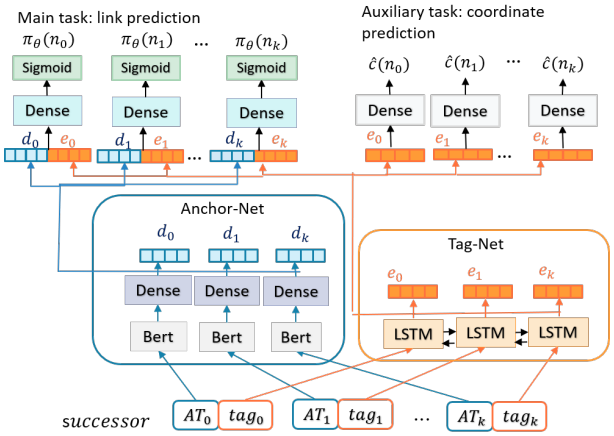

圖 3. 爬蟲代理人模型架構

在這邊我們使用了兩個任務來訓練代理人模型：模型的主要任務 $\pi_\theta(a)$ 是結合來自 Tag-Net 和 Anchor-Net 的表示向量，再透過多層感知器去微調結果，最後使用 sigmoid 函數輸出該節點 a 帶領我們到活動來源頁面的機率，再從中選取最大機率的錨節點 (式子 2)。

輔助任務中所要預測的節點坐標是指瀏覽器打開的節點坐標 (x,y) 和尺寸 (width,height)，亦即 $coord(n_i) = (x, y, width, height)$。我們這裡使用的輔助任務不僅要考慮單一節點坐標，還要計算這個節點和前後節點的座標，換言之，我們使用 12 維的向量來表示坐標特徵。

$$c(n_i) = \{coord(n_{i-1}), coord(n_i), coord(n_{i+1})\} \quad (6)$$

對於輔助任務，我們使用均方誤差作爲我們的目標函數，即式子 7。

$$L_c = \frac{1}{|\tau|}\sum_{\tau}\sum_{t=0}^{T}\sum_{n_i \in AN(a_t)} mse(c(n_i), \hat{c}(n_i)) \quad (7)$$

我們使用策略梯度 (Policy Gradient) 作爲我們的強化式學習演算法，最大化報酬的期望值 $R(\tau)$(式子1)，因此主任務的目標函數爲式子 8。結合輔助任務，整體目標函數爲 L(式子 9)。

$$L_a = -\frac{1}{|\tau|}\sum_{\tau} R(\tau) * P(\tau|\theta) \quad (8)$$

$$L = \alpha * L_c + (1-\alpha) * L_a \quad (9)$$

其中 $P(\tau|\theta)$ 爲了計算上的方便，我們使用「Pseudo Loss」的概念來當作我們的目標函式，也就是取 $\log P(\tau|\theta)$，因此在最佳化計算 Gradient 時可以省略其他環境無關的狀態轉換機率，只考慮每一個動作的機率:

$$\nabla \log P(\tau|\theta) = \sum_{t=0}^{T} \log \left[\underset{a \in Action(s_t)}{softmax}\ \pi_\theta(a_t|s_t) \right] \quad (10)$$

E. 活動來源網頁評分器

針對爬蟲代理人點擊的鏈結下載的網頁，我們設計一個活動來源網頁評分器來做爲強化式學習的報酬機制。雖然深度學習可以自動擷取對分類任務最重要的特徵，但是當訓練資料不足時，適當的輸入資料過濾可以幫助模型聚焦在重要的內容，比起使用所有葉節點的資訊時效能更好。本文使用了四個特徵，分別是最大資料區塊、次大資料區塊中、網頁標題與中間資料區塊文字。其中所使用的最大資料區塊的內文和次大資料區塊內文的擷取方法主要是參考單頁非監督式資料集擷取技術 MDR [13]，應用 HTML 標籤樹和字串比較的演算法去找出網頁中所有資料區塊/資料集，每個區塊中均包含多個資料記錄，也就是資料區塊 $R_i = (r_1^i, r_2^i, ...r_m^i), 1 \le i \le n$。假設資料區塊 $R_1, R_2, ...R_n$ 根據資料紀錄的數量與資料紀錄中的節點數量去進行排序，最後我們會依據排序的結果去取出最大資料區塊的內文、及次大資料區塊的內文。

關於中間區塊的內容。根據標籤的出現順序 $t_0^p, t_1^p, ..., t_x^p$，我們使用最中間的 50% 標籤中的內文放入模型。模型架構如圖 4 中。我們使用 BERT 對我們的四個文本特徵進行編碼，然後將它們連接起來並使用 MLP 來調整向量表示，最後將它們放入一個 sigmoid 函數中輸出此頁面爲活動來源頁面的機率。在 1863 筆訓練網頁與 583 筆測試網頁中，樣本分佈如表 II，我們提出的模型效能爲 83% 正確率、76% 的精確率與 73% 的召回率。相對使用完整網頁的效能:76% 的正確率、64% 的 Precision 與 62% 的 Recall，效能改善許多。

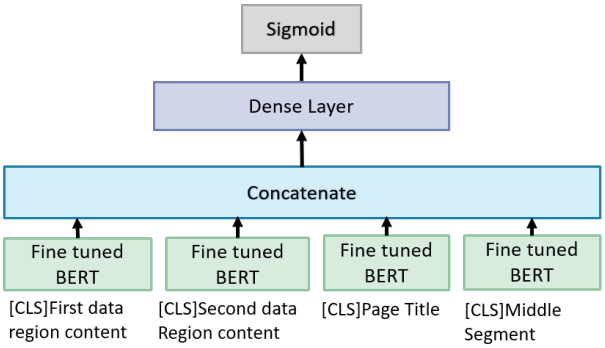

圖 4. 活動來源頁面的分類器架構

表 II
活動來源網頁評分器的資料集, 單位: 網頁

訓練集		測試集	
正樣本	負樣本	正樣本	負樣本
630	1233	193	390
1863		583	

V. 實驗

我們收集起始頁面的方式是取來自 FB Event 中的活動貼文，並從貼文中擷取出其中的 URL，當作我們的起始頁面。我們從 FB 貼文中一共收集了 864 個起始頁，以人工的方式從中找到了 160 個可以再一步之內帶領我們走到活動來源頁面的種子 URL，我們將起始頁面拆分爲訓練資料 (40 URLs)、驗證資料 (20 URLs) 和測試資料 (100 URLs)。剩餘 704 個起始網頁，再拆分爲 300 微調資料集和 404 測試資料。

評估階段使用兩種測試資料，第一個測試資料是從已標記的活動來源網站拆分出來的 100 個測試起始頁，我們將其命名爲 Test Event Source Website (TESW)，第二個測試資料集則是混合 TESW 和眞實世界的 404 測試起始 URL，命名爲混合資料集 (Mixed Dataset)。前者每一個 始 URL 都可到達某個活動來源網頁，後者則符合眞實世界的分佈情況，起始頁面

可能到達活動來源網頁，也可能無法到達活動來源網頁。表 III 表示每個資料集中的起始網頁個數如下。

表 III
活動來源網頁探勘任務中所使用的資料集

Dataset	訓練集	驗證集	TESW	RWW
Seed URLs	40	20	100	404
Event source pages	156	256	1757	

我們使用的三個評估指標，分別爲準確率 (Precision, 以網頁爲單位)，召回率 (Recall, 以網站爲單位) 及每頁的單位成本 (Unit Cost, Precision 的倒數)。

$$Precision = \frac{\#of\ event\ source\ pages}{\#\ of\ clicks} \tag{11}$$

$$Recall = \frac{\#\ of\ episodes\ reaching\ event\ source\ pages}{|Seeds|} \tag{12}$$

$$UnitCost = \frac{\Sigma_\tau Cost(\tau)}{\#\ of\ event\ source\ pages} \tag{13}$$

在 Recall 的評估上我們以網站爲單位，原因是我們無法標記所有網頁是否是活動來源網頁。而在 Mixed 資料集，我們僅能以準確率及單位成本來評估。

A. 預訓練模型效能評估

圖 5顯示預訓練模型在 TESW 資料集中的 Recall 效能。由於評估過程採用變動步伐，我們也同時計算爬蟲代理程式執行第一、第二、及第三個動作時的 Recall 值。可以看到隨著預訓練回合 (epoch) 的增加，模型在第一個動作時的 Recall 大幅的從 0.22 提升到 0.75。

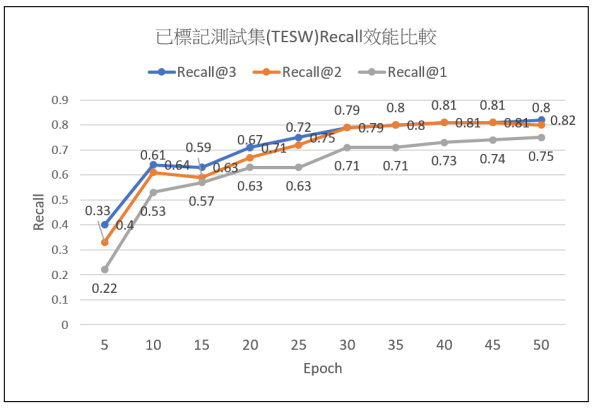

圖 5. 預訓練階段所得模型在 TESW 資料集 Recall 效能改變

圖 6則比較評估過程中採用固定步伐 (T=1, 2, 3) 或變動步伐訓練，所得模型在 Mixed 資料集的準確率及單位成本。如圖所示，評估階段採取固定步伐時，

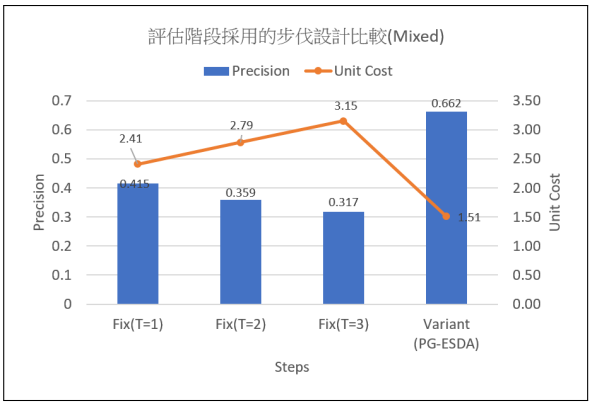

圖 6. 評估階段採用固定步伐與變動步伐的模型在 Mixed 資料集中的效能

隨著 T 增加其準確率下降 (0.415 下降至 0.317)、且單位成本也增加 (2.41 提升至 3.15)。相對的變動步伐的模型準確率 (0.662) 相對較高、且單位成本也較小 (1.51)。

圖 7則比較了不同報酬給予方式的差異。從圖 7來看，無論使用標記資料或是二元分類器或作爲獎勵函數，隨著 epoch 的增加，模型在第一個點擊動作的準確率效能也會跟著變好。但我們也發現，使用二元分類器作爲獎勵函數 (0.797)，最終模型效能比使用標記資料還要差的 (0.5)。

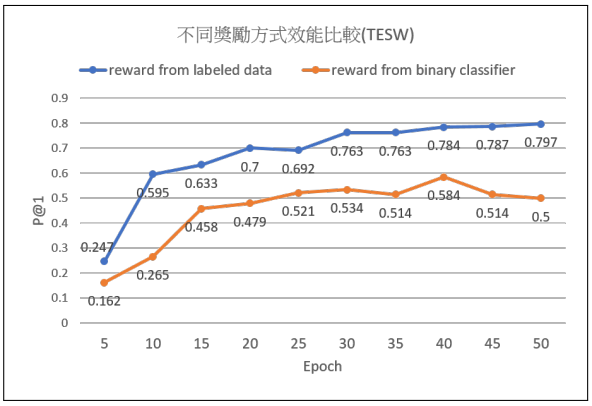

圖 7. 不同的報酬給予方式比較 (TESW)

圖 8顯示了我們有無使用多任務學習的模型在 Mixed 資料集中的效能，從圖中可看出，使用多任務學習可以讓我們的模型學習的更好。

B. 微調訓練模型效能評估

上面提到的所有表格或圖表都是運用只經過預訓練的爬蟲代理人進行評估，接下來我們會用預訓練的代理人結合微調步驟來評估效能。

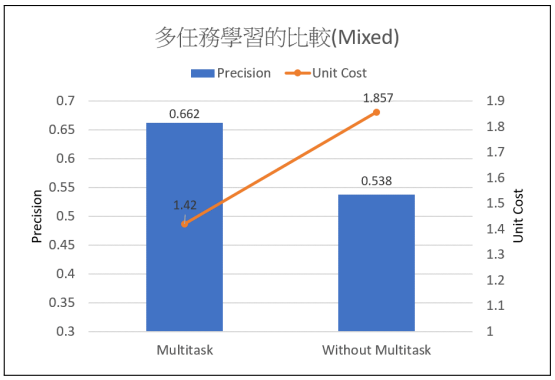

圖 8. 比較有無多任務學習對效能的影響

從圖 9顯示我們對預訓練模型進行微調，並在 TESW 資料集中進行模型評估，一共進行了五次實驗的結果。P@1 的表現都比初始預訓練模型都有提升。

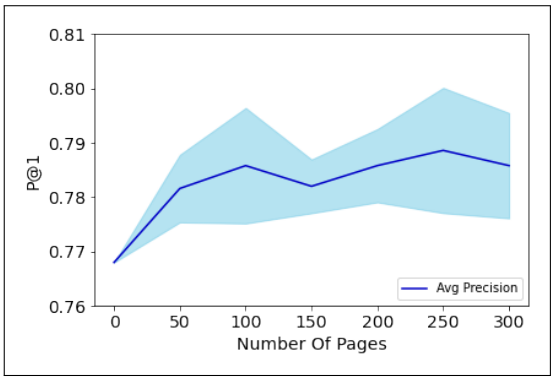

圖 9. 在微調步驟的學習曲線

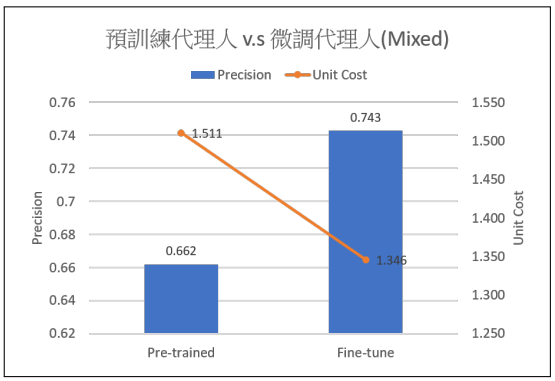

圖 10. 比較預訓練階段與微調階段的差異

另外我們也在 Mixed 資料集進行了實驗，如圖 10所示。經過微調之後，爬蟲代理人模型的準確度從 0.662 提升至 0.743，而單位成本從 1.511 降至 1.346。

VI. 結論

在本篇論文中，我們提出了一個訓練爬蟲代理人的方式，可以以更低的成本和更高的準確度，從起始頁面去收集活動來源頁面，並使用閾值讓我們的代理人可以更有效率。我們將搜索問題建構爲 MDP，並使用策略梯度算法來解決這個問題，我們的爬蟲代理人是使用深度神經網絡並將其結合多任務學習，讓我們的模型可以更進一步的學習向量的表示。在我們的任務中，因爲標記資料量的限制，因此我們提出了兩步驟架構來訓練代理人，而兩個步驟之間最大的區別則是報酬函數，第一步的報酬函數基於我們的標記資料，第二步的報酬函數由一個活動來源頁面的分類器給予。從實驗的結果可以看出，我們使用多任務學習提升了 11% 的準確率，透過微調，其準確度更是比預訓練還要提升了 8%。

誌謝

本文爲科技部專題計畫之部分研究成果——EventGo! 社群媒體貼文中探索城市的活動事件動態與活動熱門度預測之研究（計畫編號:MOST-109-2221-E-008-060-MY3）。

參考文獻

[1] J. Foley, M. Bendersky, and V. Josifovski, “Learning to extract local events from the web,” in *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, no. 09-13 in SIGIR15, (Santiago, Chile), pp. 423–432, ACM, August 2015.

[2] Q. Wang, B. Kanagal, V. Garg, and D. Sivakumar, “Constructing a comprehensive events database from the web,” in *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, CIKM ’ 19, (New York, NY, USA), p. 229—238, Association for Computing Machinery, 2019.

[3] A. Laender, B. Ribeiro-Neto, A. da Silva, and J. Teixeira, “A brief survey of web data extraction tools,” *SIGMOD Record*, vol. 31, no. 2, pp. 84–93, 2002.

[4] C.-H. Chang, M. Kayed, M. Girgis, and K. Shaalan, “A survey of web information extraction systems,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 18, no. 10, pp. 1411–1428, 2006.

[5] E. Ferrara, P. De Meo, G. Fiumara, and R. Baumgartner, “Web data extraction, applications and techniques: A survey,” *Knowledge-Based Systems*, vol. 70, p. 301—323, Nov 2014.

[6] C.-H. Chang and S.-C. Lui, “Iepad: Information extraction based on pattern discovery,” in *Proceedings of the 10th International Conference on World Wide Web*, WWW ’01, (New York, NY, USA), p. 681—688, Association for Computing Machinery, 2001.

[7] A. Manjaramkar and R. L. Lokhande, "Depta: An efficient technique for web data extraction and alignment," in *2016 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pp. 2307–2310, 2016.

[8] T. Vogels, O.-E. Ganea, and C. Eickhoff, "Web2text: Deep structured boilerplate removal," in *European Conference on Information Retrieval*, (Grenoble, France), pp. 167–179, Springer, Springer, 2018.

[9] A. M. Grigoriadis and G. Paliouras, "Focused crawling using temporal difference-learning," in *SETN*, 2004.

[10] I. Partalas, G. Paliouras, and I. Vlahavas, "Reinforcement learning with classifier selection for focused crawling," in *ECAI*, 2008.

[11] M. Han, P.-H. Wuillemin, and P. Senellart, "Focused crawling through reinforcement learning," in *Proceddings of the 18th International Conference on Web Engineering*, ICWE 2018, pp. 261–27, 2018.

[12] P. R. Asitang Mishra, Chris Mattmann and W. Burke, "Roach: Online apprentice critic focused crawling via css cues and reinforcement," in *Proceedings of the 14th International Workshop on Mining and Learning with Graphs (MLG)*, 2018.

[13] B. Liu, R. Grossman, and Y. Zhai, "Mining data records in web pages," KDD '03, (New York, NY, USA), p. 601—606, Association for Computing Machinery, 2003.