基於自動分頁預測之大規模資料應用程式介面建置 - 以活動擷取爲例

吳承儒
資訊工程學系, 國立中央大學
sam19970416@gmail.com

張嘉惠
資訊工程學系, 國立中央大學
chia@csie.ncu.edu.tw

摘要—在傳統網頁擷取 **(Web Data Extraction)** 服務中，若碰到需要大量公告式資料 (如: 新聞、活動頁面等等) 的情況，往往會需要透過使用者手動在網頁擷取系統上做分頁標記，因此在遇到分頁資料量龐大的網站時，使用者會耗費大量的時間在 **“教導機器如何切換網頁”**，導致無法有效地進行大規模的資料擷取。本研究將會把這個問題轉換成序列標記 **(Sequence Labeling)** 問題，提供基於神經網路的序列標記方法 **- PRNSM**，並結合了大多數網頁標記研究不會使用的 **HTML Attribute** 資訊，將網頁中的分頁標記成 **“PAGE”**、**“NEXT”** 以及 **“OTHER”**，並在單一語言訓練、測試上面得到 **0.818** 的平均 **Macro F1**，另外我們也透過零樣本實驗展示模型在多語言的效能，在測試資料集 **DE, RU, ZH, JA, KO** 的零樣本實驗中達到了 **0.774** 的平均 **Macro F1**，最後我們將研究成果結合非監督式資料擷取系統 **(Unsupervised Data Extraction System)**，建立大規模自動化資料擷取系統，在大規模活動擷取的實際應用中，我們能從從 **402** 個網站中自動產生出 **196** 個資料 **API**，達到接近 **0.5** 的 **API** 建立率。

***Index Terms*—ETL**、分頁預測、序列標記、自動化爬蟲系統

I. 緒論

網頁抓取 (Web Scraping) 是指利用網頁 Agent 下載 HTML 並從網頁中擷取內容和資料的過程。網頁抓取用於各種依賴資料採集的應用，例如比價代購、天氣資料監測等。網頁抓取包括兩個任務：網頁抓取和網頁資料擷取 (Web Data Extraction)。因爲每個網站都有自己的訊息呈現方式，通常需要爲每個網站構建一個網頁抓取器。

在過去的二十年裡，網絡資料擷取技術已經從監督方法 [1], [2] 演變爲無監督方法 [3], [4]。然而，網頁獲取仍然依賴於使用者的參與來告訴系統要下載哪些頁面並編寫程式。例如，Import.io [5] 和 Dexi.io [6] 等現有工具允許使用者定義且建構 Web 擷取 Agent 的流程和規則，並指定從目標網站/資料源擷取哪些資料。

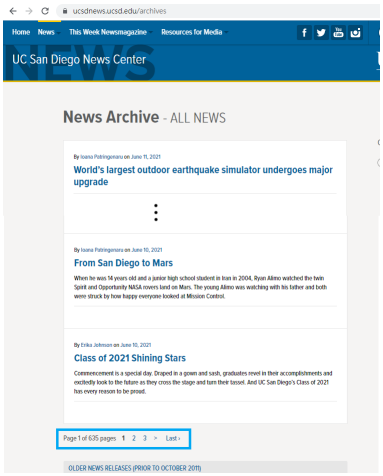


圖 1: 含有分頁標籤作爲網址切換的頁面

假設我們需要擷取來自 500 多個網站的最新公告的情況，如果要爲每個網站分別制定網頁抓取過程，這將會相當費時。我們是否有可能給定起始 URL 就可以自動化下載所有具有同樣樣板的網頁、再進行頁面資料擷取？答案是肯定的，因爲最新的新聞或公告通常顯示在具有分頁設計的列表中。如圖 1 所示，許多網站都包含一個最新的新聞頁面，並提供一個交互界面（稱爲分頁標籤）用於從一個頁面轉換到另一個頁面。

文獻上和自動分頁預測最相關的研究是 Wu 和 Sgro [7] 在 2016 年提出的美國專利 (US20160103799A1)，該專利採用監督模型自動檢測分頁，其專利內容包括識別分頁標籤的特徵、如何錄

製用於識別分頁標籤的資料、以及模型的訓練。

AutoPager [8] 則是一個 Python 的分頁標籤辨識套件，其使用基於線性鏈 CRF（條件隨機場）的序列標記模型，將每個可點擊的鏈結標記爲 PAGE、NEXT 鏈接或 OTHER。AutoPager 使用的特性包括鏈接模式 (Href link)、鏈結中查詢關鍵字 (Query key)、標籤屬性中的 Class 名稱 (Class Names) 和錨節點文字內容 (Text content)。

一旦可以檢測到分頁標籤，我們就可以將其與無監督的 Web 資料擷取相結合，以進行大規模資料收集。在本篇論文的最後，我們將以建立活動資料庫 [9], [10] 爲例，來說明我們如何結合上述自動分頁偵測模組，從活動來源網址來進行大規模活動活動擷取。

總結本篇論文的貢獻如下：

- 據我們所知，這是第一個透過結合自動分頁識別和無監督網絡資料擷取系統來實現大規模資料擷取的工作。
- 我們介紹並比較了如何將 HTML 標籤拆分爲各種有用的特徵嵌入 (Feature Embedding)。
- 我們發佈了一個多語言分頁標籤預測模型 (PRNSM)，與 AugoPager 相比，將 Micro/Macro F1 提高了 5%。

II. 相關研究

現有市場上已有一些 Web Scraper 工具，爲用户提供方便快捷的資料擷取，如 Import.io [5] 和 Dexi.io [6]。Import.io 是市場上比較知名的服務商，主要優點在於全自動化的網頁資料擷取。用户只需要指定目標網站的 URL，系統就可以自動分析其背後可能的資料片段並加以對齊排列。如果擷取結果不好，用户也可以手動標註正確的資料。所以它可以看作是一個半監督的資料擷取系統。爲了處理多頁資料擷取，import.io 會檢測當前網頁的 URL 參數可能的組合，用户也可以通過 URL Generator 生成多個 URL。但這種方式僅支援 GET 類型的 HTTP 請求，對於 POST 類型的 HTTP 請求則無法處理。

針對分頁偵測，傳統的方式是透過 HTML 規則來判斷該網頁元件是否爲分頁標籤。除了基於規則的方法，CONNOTATE 的專利 [7] 則透過檢測 HTML 錨標籤的 Class 名稱是否有類似 “pagination” 的關鍵字來找尋分頁標籤。而這些方法可以有效地檢測出傳統網頁中分頁元素的位置。然而，隨著當前前端 Web 框架的快速發展，已經有許多機制被開發來防止這些分頁元素的自動檢測，例如採用 URL 參數的自動雜湊 (Auto-hashing)。

根據我們的調查，Autopager [8] 是第一個使用序列標籤進行分頁標籤識別的研究。它們保留所有 HTML 錨標籤，並根據標籤屬性（例如父標籤和子標籤的類名）、鏈接中的查詢字（例如查詢是否包含與 “分頁” 相關的關鍵字）和錨文本內容定義特徵當前、上一個和下一個錨節點預測四種標籤，包括 “PREV”、“PAGE”、“NEXT” 和 “Other”。基於設計的特徵，線性鏈 CRF（條件隨機場）模型可以解決部份前面提到的自動雜湊問題。但是，由於 CRF 仍然依賴於模型之前讀取的特徵和詞彙，因此該模型無法直接應用於使用不同語言或開發技術所撰寫的網頁。

在本篇論文中，我們嘗試利用神經嵌入 (Neural Embedding) 以更好地表示 HTML 標籤和路徑。雖然有一些使用神經嵌入來表示 HTML 標籤和路徑的方法，例如 [11], [12]，但大多數研究經常簡化標籤而忽略屬性和樣式。例如，Web2Text [11] 和 BoilerPlate [12] 使用前 50 個最常見的 Tag 標籤來建構用於 HTML 標籤路徑表示的標籤向量。

III. 自動分頁識神經序列模型

A. 問題定義

在 HTML 的前處理過程中，Autopager 只保留了含有 Href 屬性的錨節點，其做法會導致我們遺失掉潛在可點擊且不是以 Href 屬性的錨節點以及按鈕節點 (Button)，而我們在這部分則是同時保留了錨節點以及按鈕節點，讓我們能夠在日後更廣泛的應用中，處理那些利用按鈕節點且不含有 href 屬性的錨節點網頁。我們將每一個頁面以所有可點擊的錨節點以及按鈕節點序列來表示，每個頁面 $x = [x_1, x_2, \ldots, x_L]$ ，我們將每個 x_i 包含以下三種資訊:

- $ParentTag_i$: 目前節點在 DOM 結構中父節點的名稱。
- $AnchText_i$: 目前節點中的文字資訊。
- $AttCont_i$: 目前節點中的屬性資料 (如: Class 名稱、Href 中的查詢關鍵字)。

我們的目標是爲每個可點擊之節點預測分頁標籤，每個節點會被預測成 “PAGE”、“NEXT” 或是 “OTHER”。

$$y_i \in \{PAGE, NEXT, OTHER\} \quad (1)$$

如上所述，以前的分頁標籤辨識方法主要依賴於那些從可點擊節點中所擷取的特徵，尤其是 *Href* 連結中的查詢關鍵字和 *Class* 屬性中的類別名稱。舉例來說，對於以下的錨節點範例III-A，*Href* 連結中的查詢關鍵字 “page” 和 *Class* 中的名稱 “pagination” 都是分頁標籤辨識的良好指標。

NEXT

Autopager 針對每個錨節點擷取出了不同面向的啓發式特徵 (Heuristic features)，主要特徵圍繞於錨節點中的 Class、Href 以及文字節點內容，包括：

- **Href**: (1)Href 連結是否含有 “page”、“number”、“year” 等關鍵字? (2)Href 中的 Query keys
- **Class**: 目前節點、父親節點、以及孩子節點的 Class 名稱
- **Text**: (1) 目前的文字節點是否爲數字或是英文組成? (2) 目前文字節點中的內容以及相鄰文字節點中的內容

上述除了描述性特徵外，Autopager 皆用不同文字表示成不同 Tokens，並利用文字向量表示每一個 (1)Href 中的查詢關鍵字 (2)Class 名稱以及 (3) 文字節點內容。

但是，越來越多的網頁使用非傳統的頁面重新導向方法。三種常見的方法包括: (1) 用雜湊值替換所有分頁連結，(2) 或是使用 JavaScript 函數進行頁面跳轉 (而不是更改網址參數)，或 (3) 使用動態渲染方法替換頁面內容 (該方法通常用於單頁應用 (Single page applications)，尤其常見於動態編程式網站。

因此，基於規則的方法將無法再透過關鍵字去尋找分頁標籤。而對於 CRF 模型來說，在單一語言所訓練出的模型也會在多語言的應用中損失大量準確度。

B. 模型架構

爲了解決上述問題，我們提出了 Pagination recognition neural sequence model (如圖 2 所示)，利用 (1) Parent Tag Name，(2) Attribute embedding，(3) Text content embedding 以及 (4) Heuristic features 來表示每一個可點擊的節點 (Anchor/Button)。

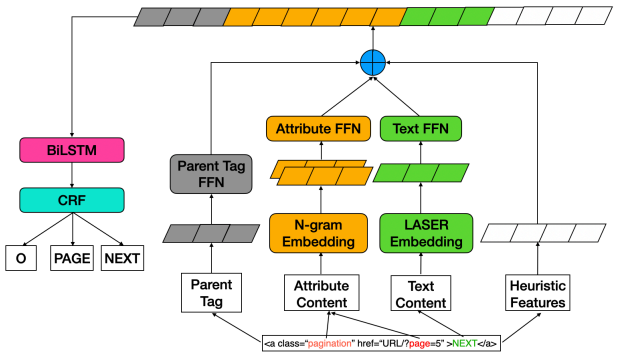


圖 2: Pagination Recognition Neural Sequence Model

1) 父節點表示法: 在我們的研究中，我們參考了 [11] 中表示節點資訊的方法。利用向量 $d(=30)$ 來表達在訓練資料中最常見的 d 個父節點名稱，並在向量中多增加一維度來表達其他的父節點名稱。因此，每個父節點名稱都可以表示成 $d+1$ 的 one-hot encoding 向量。對於每個父節點 one-hot encoding 向量$\mathbf{v_i}$，我們映射到 m 維度的全連接網路來得到 $D \in R^{m\times(d+1)}$ 的父節點向量。

$$PT_i = D * \mathbf{v_i} \quad (2)$$

2) 網頁屬性嵌入: 大多數接受網頁作爲輸入的神經序列標記模型 (例如 Web2Text、BoilerPlate) 只考慮標籤路徑 (Tag Path) 和文字節點內容，而忽略節點中的 Attribute 內容。然而，在傳統分頁辨識方法中，Href 連接和 Class 名稱才是主要成功的兩個關鍵特徵。雖然在不同網站中程式設計師給予 Class 名稱可能不太一樣，但是考量到會有類似的命名規則。因此在我們提出的神經序列標記模型中，我們將檢查神經網路是否可以從 Href 連接以及 Class 名稱中擷取重要特徵。由於每個分頁都有自己的頁碼，因此我們不考慮 Href 中的查詢值 (Query value)。

爲了對應不同程式設計師對 Class 名稱取名的不同，我們使用 N-gram 嵌入 (N-gram Embedding) 來處理，也就是文中提到的屬性嵌入 (Attribute Embedding)。誠如上述，每個節點中的屬性內容 $AttrCont_i$ 會被分成 Class 名稱以及 Query keys，因此我們可以表示爲 $AttrCont_i = [CN_i, QK_i]$。對於每個節點來說，含有多個 Class 名稱以及 Query keys 是正常的現象。因此我們可以利用 CN_i 表示 Class 名稱的聯集，接下來我們透過 N-gram 嵌入表示每一個 Class 名稱 $c_j^i \in CN_i$，如圖 3 所示。最後我們透過 Average

pooling 層針對所有 Class 名稱的 N-gram 嵌入做平均得到 $Ngram_C(c_j^i)$。

$$ClassEmb(CN_i) = AvgPooling(Ngram_C(c_1^i), ...), \quad (3)$$

同樣地，我們利用 N-gram 嵌入對 Query keys 取得 Query 嵌入:

$$QueryEmb(QK_i) = AvgPooling(Ngram_Q(q_1^i), ...), \quad (4)$$

QK_i 表示節點中 Href 連結的 Query keys 聯集，而對於每個 Query keys $q_j^i \in QK_i$，我們透過 N-gram 嵌入層得到 $Ngram_Q(q_j^i)$。

N-gram 嵌入的關鍵想法是我們可以透過將每個 Class 名稱以及 Query keys 分解爲 2-gram、3-gram 和 4-gram，以得到更多含義。最後，我們將平均 Class 嵌入以及平均 Query 嵌入合併後得到 $AttrCont_i$ 。

$$AC_i = ClassEmb(CN_i) \oplus QueryEmb(QK_i) \quad (5)$$

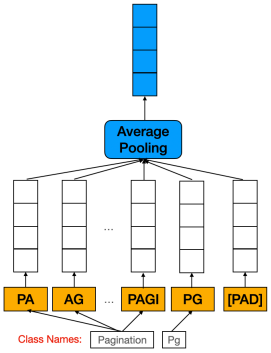


圖 3: Attribute Embedding

3) 文字內容嵌入: 現在有許多預訓練的多語言嵌入模型 (BERT [13], LASER [14])，在我們提出的方法中，我們在零樣本實驗中測試了 BERT 以及 LASER，最後選擇使用預訓練的 LASER 來預測文本嵌入 (Sentence embedding)，它允許將不同語言的文本內容映射到同一個向量空間，這大大提高了我們在多語言零樣本應用 (Zero-shot applications) 中的準確性。對於節點中的文本內容，我們利用 $LASER(AnchText_i)$ 表示第 i 個節點的文本內容嵌入。

$$TC_i = LASER(AnchText_i) \quad (6)$$

C. Heuristic Features

我們根據 Autopager 設計八個特徵，包括 (1)Class 屬性中是否含有 disabled 類別？(2) *Href* 連結是否包含 “page”？(3) *Href* 連結是否包含 “數字”？(4) *Href* 連接是否包含 “年份”？(5) *Href* 連結中的查詢鍵數是否等於 0、1 或 2？(6) 文字節點內容是否包含字母或數字？(7) 當前文字節點之前的節點是否有 *Href* 連結？(8) 當前文字節點之後的節點是否有 *Href* 連結？

1) BiLSTM 序列表示層: 對於每個可點擊節點 x_i，我們連接該節點中的 Parent tag vector、Text content embedding、Attribute embedding 以及 Heuristic feature vector 來得到最終的節點表示 $Rep(x_i)$：

$$Rep(x_i) = TP_i \oplus AC_i \oplus TC_i \oplus HC_i \quad (7)$$

每個節點表示接下來會透過雙向長短期記憶網路來獲取節點的上下文關係。在我們給定的節點表示 $Rep(x_1), Rep(x_2), ..., Rep(x_n)$，向前的長短期記憶網路以及向後的長短期記憶網路將產生以下的輸出:

$$\begin{aligned} \overrightarrow{h}_i &= \overrightarrow{LSTM}(\overrightarrow{h}_{i-1}, Rep(x_i)) \\ \overleftarrow{h}_i &= \overleftarrow{LSTM}(\overleftarrow{h}_{i+1}, Rep(x_i)) \\ h_i &= \overrightarrow{h}_i \oplus \overleftarrow{h}_i \end{aligned} \quad (8)$$

2) 標記預測層: 經過 Encoding 層的輸出後，我們在雙向長短期記憶中的輸出隱藏狀態 $H(x) = \{h_1, h_2...h_L\}$ 接上一層 CRF。假設對應的分頁標籤 $y = (y_1, y_2...y_L)$，則每個標記 $y_i \in \{OTHER|PAGE|NEXT\}$，在 CRF 的推理層會利用等式9預測序列標籤 $\hat{y}$。

$$\begin{aligned} p(y|x) &= \frac{1}{Z(H(x))} exp\{\sum_{t=1}^{L} T[y_{t-1}, y_t] + \sum_{t=1}^{L} U(h_t, y_t)\} \\ Z(H(x)) &= \sum_{y'} exp\{\sum_{t=1}^{L} T[y'_{t-1}, y'_t] + \sum_{t=1}^{L} U(h_t, y'_t)\} \end{aligned} \quad (9)$$

上述算式中，T 代表了 3×3 的狀態轉移矩陣 (State transition matrix)，U 函數是將輸入向量 h_t 與對應標記 y_t 的權重向量作內積，而 Z 則代表了所有標籤序列 y' 潛藏函數值的總和。

3) 訓練目標: 最終，我們利用訓練演算法最小化 Negative log likelihood (等式10) 來得到最佳參數。

$$-\log p(y|x) = Z_{log}(H(x)) - (\sum_{t=1}^{L} T[y_{t-1}, y_t] + \sum_{t=1}^{L} U(h_t, y_t)) \quad (10)$$

IV. 實驗分析

A. 資料集

由於目前在學術界還沒有一個可用的資料集，因此我們從 Amazon Global Top Website [15] 手動搜集

表 I: 從 Amazon Global Top Sites 搜集來的訓練以及測試資料集

Dataset	Type	Pages	Label Ratio			#Labels per Page		
			PAGE	NEXT	nonLabel	# PAGE	# NEXT	# Nodes
EN	Dev	164	65.20%	59.75%	26.82%	10.0	1.4	242.7
	Test	49	34.69%	46.94%	53.06%	7.4	1.3	459.7
DE	Test	20	60.00%	55.00%	30.00%	11.6	2.0	237.6
RU	Test	21	38.10%	33.33%	61.90%	4.6	1.0	484.4
ZH	Test	44	54.55%	45.45%	45.45%	11.5	1.2	180.3
JA	Test	23	26.09%	34.78%	34.78%	8.2	1.4	401.6
KO	Test	24	25.00%	20.83%	75.00%	10.0	1.0	484.4

表 II: 英文資料集的五次平均表現

Metric	Model	Page			Next			Avg.
	(25 epochs)	P	R	F1	P	R	F1	F1
Micro	AutoPager	0.822	0.877	0.844	0.757	0.788	0.758	0.801
	PRNSM	0.824	0.932	0.874	0.733	0.869	0.794	0.834
Macro	AutoPager	0.645	0.663	0.646	0.683	0.778	0.727	0.687
	PRNSM	0.808	0.922	0.861	0.737	0.815	0.77	0.818

並標記資料。在資料蒐集的部分，我們將會從美國、德國、俄國、中國、日本以及韓國蒐集最常瀏覽的網頁，用來做後續零樣本跨語言的實驗。而在資料標記的部分，對於可點擊的錨節點以及按鈕節點來說，有些並不是葉節點 (Leaf node)，例如 <a> 標籤中間可能包括如 、 等等子節點，因此我們將擷取整塊錨節點作爲節點，而不是傳統的葉節點，爲了加速資料標記的流程，我們採用 SelectorGadget [16] 此套應用插件，透過點擊網頁中的分頁標籤來直接取得錨節點或是按鈕節點的 DOM 路徑，而我們將所有分頁標記成 <PAGE>，並把下一頁標記成 <NEXT>，其餘節點將標記成 <Other>。過程中我們也會尋找不含有任何分頁標籤的網頁納入資料集當中，用來衡量我們是否能夠正確判斷網頁爲單頁網站。我們總共從 US Top Website 標記了 164 個 Train Pages 以及 49 個 Test Pages，另收集 Global Top Site 中來自德國、俄國、中國、日本、韓國共 132 個零樣本測試頁面。表 I 顯示了各個語言的測試資料集的統計數據，尤其是這些網站包含分頁及下一頁的比例，兩者都未包含的情況占約 1/4 到 3/4。

1) 評估指標*:* 我們使用 Micro 和 Macro F1 分數進行模型評估。Micro F1 計算所有測試頁面中每個標籤的 F1，而 Macro F1 則爲每個頁面計算 F1 後，再取平均值。我們認爲 Macro F1 在分頁標籤識別上面更能符合實際的應用情況。因此在選擇最佳模型時，我們就會以平均 Macro F1 爲評量指標。

B. 實驗結果

在本篇論文中，我們使用 Autopager 作爲我們的基準模型。除了前面提到的 8 個啓發式特徵之外，還包括來自當前、上一個和下一個節點的類屬性、查詢詞和文本內容這五個特徵，而詞彙量大小則分別爲 299、3823、9791、5684 以及 5153。

對於提出的神經序列標記模型，我們對類屬性和查詢鍵的 N-gram 嵌入使用 32 維度，而對於預訓練的 LASER 嵌入則使用 1024 維度。另外對於 Parent Tag 編碼，我們只考慮前 30 個最常見出現的 Parent Tag 名稱。訓練 epoch 的數量則固定爲 25。對於每個實驗，我們進行了 5 次實驗並取平均值以獲得平均效能。

1) 英文資料集表現*:* 首先我們在英文訓練集上做訓練，並在英文測試集上做測試，首先我們利用衡量總體節點預測分數的 Micro F1 做衡量，如表 II 所示，與 CRFSuite 相比，我們所提出的模型在 PAGE 的 F1 上面提高了 3%，並在 NEXT 的 F1 上面提高了 3.6%，並在平均 F1 分數提高了 3.3%。接下來我們透過 Macro F1 衡量頁面中的節點預測能力，如表 II 所示，因爲評估方式改從節點層級拉高到頁面層集，因此會讓模型的效能下降，CRFSuite 在 PAGE 的預測部分較 Micro F1 低了接近 20%，而我們所提出的模型則是下降了 1.8%，並在平均 F1 上面還是保有領先 CRFSuite 的優勢。

爲了證明每個輸入特徵的有效性，我們對父標籤、N-gram 屬性嵌入、文本內容嵌入和啓發式特徵進行了消融實驗。實驗結果如表 III 所示。

表 III: 在 EN 資料集上不同特徵的消融實驗

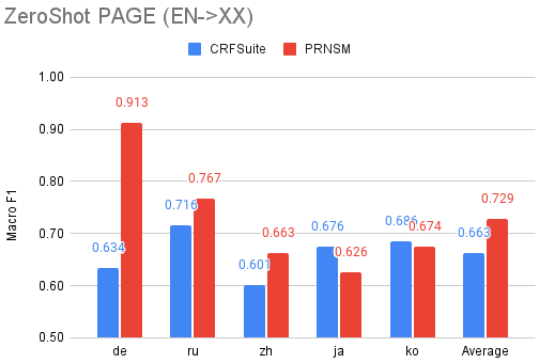
Model	Page	Next	Macro
PRNSM	0.861±2.8%	0.774±5.3%	0.818±3.5%
w/o Parent Tag	0.848±3.5%	0.715±4.1%	0.782±3.5%
w/o Text Cont.	0.851±2.4%	0.696±6.4%	0.773±7.4%
w/o Query Keys	0.840±5.7%	0.761±2.8%	0.800±3.3%
w/o Class Attr	0.816±4.8%	0.732±5.0%	0.774±4.3%
w/o Ngram Attr	0.856±2.5%	0.728±5.7%	0.792±3.3%
w/o Heuristic F.	0.764±3.6%	0.806±5.5%	0.800±4.5%

- Ngram attribute embedding: 在沒有 Attribute embedding 的情況下，Macro F1 下降了 2.6% 。單純只使用 Query embedding，模型效能會下降 1.8 %，如果使用 Class embedding 的話，模型效能則可以提升至 0.8 Macro F1。
- Parent tag embedding: 去掉 Parent tag name vector 後，模型效能下降 3.6% Macro F1
- Text content embedding: 去掉 LASER sentence embedding 後，模型性能下降 4.5% Macro F1
- Heuristic features: 去掉 Heuristic features 後，模型性能下降 1.8% Macro F1，有趣的是雖然 Page 的預測下降了近 10%，但是 Next 的預測則是有了 4% 的提升。

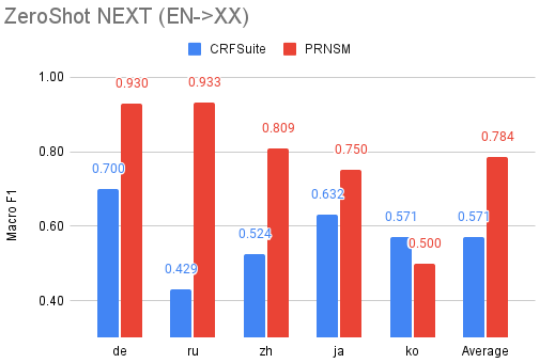
2) 零樣本跨語言實驗*:* 接下來，我們對 EN 以外的五種語言（DE、RU、ZH、JA、KO）進行零樣本實驗，以測試模型在跨語言資料集中的表現。圖 4a 和圖 4b 表明，在 PAGE 預測方面，除了 JA 和 KO 測試集之外，所提出的方法對大多數語言都有顯著改進，尤其是在 DE 集上。性能增幅最大（從 0.634 到 0.913）。總體而言，F1 平均增長了 6.6%（從 0.663 到 0.729）。對 NEXT 預測的改進更加顯著。五個資料集上的平均 Macro F1 相差超過 21.3%（從 0.571 到 0.784）。

3) 不同的 *Embedding* 對於模型的效果*:*

- **Text Content Embedding** 圖 5 展示了不同預訓練多語言句子嵌入方法的零樣本實驗。LASER 嵌入在平均 F1 分數上比 BERT 高 1.5%，因此我們選擇 LASER 作爲我們的文本嵌入方法。
- **Char-CNN Attribute Embedding** 爲了直接處理 Attribute 內容中的訊息，我們還嘗試了幾種



(a) PAGE 標籤辨識的 Macro F1



(b) NEXT 標籤辨識的 Macro F1

圖 4: 不同語言的零樣本實驗結果

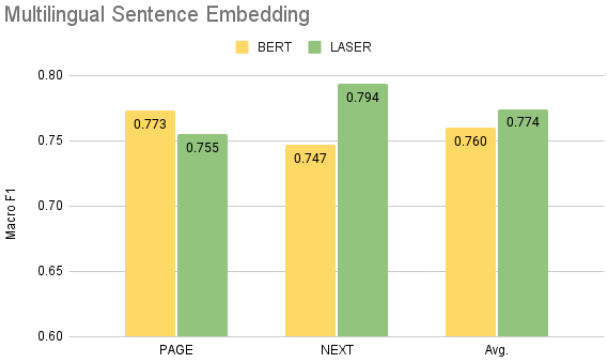


圖 5: 不同預訓練語言模型的效能比較

不使用對 Attribute 進行人工特徵擷取的方法。我們按照 [17] 將 Attribute 的全部內容轉換成字符級別的 tokens，放入 Char-CNNs 層得到 Char-CNN 屬性嵌入（參數見表 IV ）。另外，我們也嘗試通過 char token 的 Embedding 層直接生成屬性 embedding。我們將上述方法與我們提出的

n-gram 嵌入進行比較。如圖6所示，Char-CNN 在 PAGE 標記上實現了接近 N-gram 的結果，但在 NEXT 預測中則落後 N-gram 方法，整體說來 N-gram 生成的屬性 embedding 得到了最好的平均 Macro F1，而無需對屬性內容進行任何特徵選擇。

表 IV: Char-CNN 參數

Layer	Filter Num	Filter Size	Pooling Size
1	512	3	3
2	512	4	3
3	512	5	3
4	512	6	3

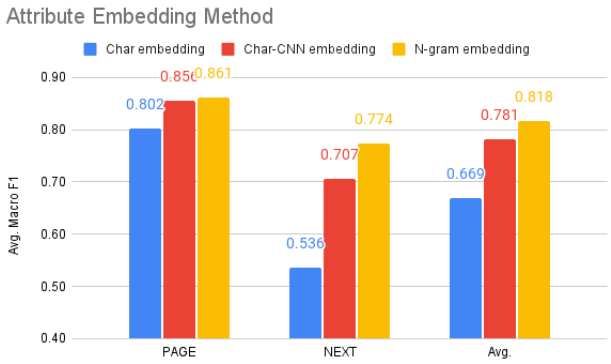


圖 6: 不同屬性嵌入方法比較

V. 案例分析：大規模活動擷取

正如介紹中所提到的，網頁抓取工具通常爲用户提供一個界面來指定從哪裡獲取網頁。當我們需要監控大量資料源時，爲每個網站一一創建資料 API 是不切實際的。本節演示我們如何將分頁標籤識別模型與 Web 資料擷取技術相結合來建構大型活動擷取系統。

如圖 7 所示，給定來自市政府、大學、活動組織等任何活動組織者的活動來源頁面，我們首先應用分頁識別模型來確定給定的 URL 是否包含分頁元素。對於每個標記爲 PAGE 或 NEXT 的錨點或按鈕節點，我們將自動下載並使用多頁擷取方法 (如 DCADE [18]) 進行無監督的網絡資料擷取。如果沒有找到 PAGE 或 NEXT 節點，則使用單頁擷取方法 (如 MDR [19]) 進行資料擷取。

因爲一個活動來源頁面通常包含眾多鏈結多個活動消息，而活動發佈通常聚集在記錄集中的區域。此

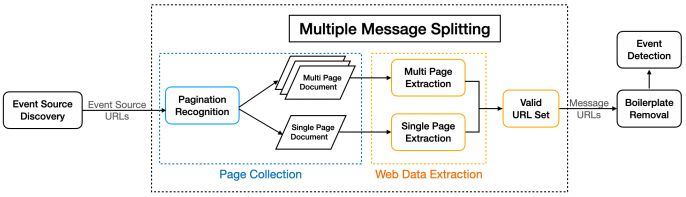


圖 7: 自動擷取活動的系統流程圖

外，此類活動列表通常具有鏈接到詳細活動信息的錨點。因此，我們透過資料擷取，找出資料集的區域，並且僅留下具有有效 URL 的資料集。這種設計是因爲在我們進行活動擷取之前，還有一些諸如樣板移除和活動檢測等步驟。因此，即使存在不是活動消息的 URL，我們仍然可以通過樣板移除和活動檢測將它們從資料中過濾掉。

A. 已標記資料集

作爲演示，我們收集了 100 個潛在的活動來源 URL，並手動將收集到的 URL 中的每個錨點和按鈕節點標記爲 PAGE、NEXT 或 NONE。這些帶有 PAGE/NEXT 標籤的活動來源 URL 的百分比分別爲 43% 和 38%。總體來說，有 44% 的活動源 URL 帶有分頁標籤。爲了展示大規模的網站 scraping 效能，我們另外準備了 402 活動來源網頁進行最新發佈消息網址的擷取。

我們在表 V 和表 VI 中展示了分頁標籤識別模型的性能。性能與前節的實驗結果相似，在節點方面的評估，PAGE 和 NEXT 分別爲 0.857 和 0.773 micro F1。而在頁面級評估方面，多頁面識別性能達到 0.818 Macro F1。

表 V: 節點層級的分頁標籤辨識結果

Label	Detected	Correct	Truth	P	R	F1
PAGE	259	213	238	0.822	0.895	0.857
NEXT	50	34	38	0.680	0.895	0.773

表 VI: 頁面層級的分頁標籤辨識結果

Type	Detected	Correct	Truth	P	R	F1
Single	51	49	57	0.961	0.860	0.907
Multi	49	41	43	0.837	0.953	0.891

表VII顯示了網頁資料擷取的結果，即最終活動消息擷取的性能。MDR 成功生成了 51 個資料集合，經

過濾留下 40 個資料集合，總共擷取出了 509 個 URL。另外，在分頁標籤辨識模型的幫助下，我們能夠額外處理多頁活動來源，並應用 DCADE 生成額外的 37 個多頁擷取器，其中包含 15 個對齊的資料集合，最後經過網頁過濾得到 14 個資料集合，共擷取出的 URL 數量爲 1,120。如果我們能夠技術提高資料擷取的準確度，就可以繼續增加我們在活動 URL 擷取的數量。

表 VII: 針對 100 個已標記的活動來源測試資料結果

Type	ETL	#RecSets	Filtered*	#URLs
Single	51	51	40	509
Multi	37	15	14	1120

*# of RecordSets containing legal URLs

B. 大規模活未標記資料集

表 VIII: 大規模活動來源擷取結果

Type	Detected	ETL	RecSets	Filtered*	#URLs
Single	206	202	202	156	2,616
Multi	196	150	45	40	2,158

*# of Recordsets containing legal URLs

在大規模活動擷取實驗中，我們準備了 402 個可能包含活動來源網頁，實驗結果如表 VIII所示，其中沒有偵測到分頁的活動來源網頁共有 206 個，透過單頁非監督式資料集擷取 MDR，可以成功的擷取到 202 個資料集合，經過有效 URL 過濾最後取得 2,616 個潛力活動 URL；有偵測到分頁結構的網頁共有 196 個網頁，經過 DCADE 僅建立 150 擷取器，但是其中包含 45 個資料集合，最後可以多取得 2,158 個潛在活動 URL。

VI. 結論

在本文中，我們通過提出用於分頁識別的神經序列標記模型 PRNSM 來解決大規模資料 API 創建的問題。我們構建了一個包含來自 Alexa Top Sites 的多語言訓練/測試資料的資料集。該模型具有多語言句子表示和 N-gram 屬性嵌入。實驗結果表明，我們的模型比 Autopager 有 11% 的 Macro F1 改進。同時，我們展示了如何將分頁識別有效地應用於現實案例進行大規模活動源網站抓取。如第 5 節所示，分頁識別解決了多頁面收集的問題，從而可以應用多種頁面資料擷取方法在短時間內擷取潛在的活動 URL。活動擷取的分頁識別結果也證明了我們的模型在零樣本應用中有一定效能。

誌謝

本文爲科技部專題計畫 “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）之部分研究成果。

參考文獻

[1] G. Gottlob, C. Koch, R. Baumgartner, M. Herzog, and S. Flesca, “The lixto data extraction project: back and forth between theory and practice,” in *Proceedings of the twenty-third ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, ser. PODS ’04. New York: ACM, 2004, pp. 1–12. [Online]. Available: https://dl.acm.org/doi/10.1145/1055558.1055560

[2] T. Furche, G. Gottlob, G. Grasso, X. Guo, G. Orsi, C. Schallhart, and C. Wang, “Diadem: thousands of websites to a single database,” in *Proceedings of the VLDB Endowment*, ser. 14, vol. 7, 2014, pp. 1845–1856. [Online]. Available: https://dl.acm.org/doi/10.14778/2733085.2733091

[3] C.-H. Chang, M. Kayed, M. R. Girgis, and K. F. Shaalan, “A survey of web information extraction systems,” *IEEE transactions on knowledge and data engineering*, vol. 18, no. 10, pp. 1411–1428, Oct. 2006. [Online]. Available: https://dl.acm.org/doi/10.1109/TKDE.2006.152

[4] E. Ferrara, P. D. Meo, G. Fiumara, and R. Baumgartner, “Web data extraction, applications and techniques: A survey,” *Knowledge-Based Systems*, vol. 70, pp. 301–323, Nov. 2014. [Online]. Available: https://dl.acm.org/doi/abs/10.1016/j.knosys.2014.07.007

[5] Import.io, “Import.io,” https://www.import.io/product/, 2012.

[6] Dexi.io, “Dexi.io,” https://www.dexi.io/, 2015.

[7] T. Wu and V. Sgro, “Methods and systems for automated detection of pagination,” 2016, uS20160103799A1.

[8] Mikhail Korobov and Iván de Prado and Mark E. Haase, “Autopager: Detect and classify pagination links,” https://github.com/TeamHG-Memex/autopager, 2020.

[9] 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*, ser. SIGIR15, no. 09-13. Santiago, Chile: ACM, August 2015, pp. 423–432. [Online]. Available: https://dl.acm.org/citation.cfm?doid=2766462.2767739

[10] 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*, ser. CIKM ’ 19. New York, NY, USA: Association for Computing Machinery, 2019, p. 229—238. [Online]. Available: https://dl.acm.org/doi/10.1145/3357384.3357986

[11] T. Vogels, O.-E. Ganea, and C. Eickhoff, “Web2text: Deep structured boilerplate removal,” in *Advances in Information Retrieval*, ser. ECIR. Springer, 2018, pp. 167–179.

[12] J. Leonhardt, A. Anand, and M. Khosla, “Boilerplate removal using a neural sequence labeling model,” in *Companion Proceedings of the Web Conference 2020*, ser. WWW ’20. New York, NY, USA: Association for Computing Machinery, 2020, p. 226—229. [Online]. Available: https://dl.acm.org/10.1145/3366424.3383547

[13] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, ser. NAACL. Minneapolis, Minnesota: Association for Computational Linguistics, 2019, p. 4171—4186.

[14] M. Artetxe and H. Schwenk, "Massively multilingual sentence embeddings for zero-shot cross-lingual transfer and beyond," in *Transactions of the Association for Computational Linguistics*, ser. TACL, 2018, pp. 597–610.

[15] Amazon, "Alexa global top sites," https://www.alexa.com/topsites.

[16] Andrew Cantino, "Selector gadget," https://github.com/cantino/selectorgadget.

[17] X. Zhang, J. Zhao, and Y. LeCun, "Character-level convolutional networks for text classification," in *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, ser. NIPS'15. Cambridge, MA, USA: MIT Press, 2015, p. 649—657.

[18] O. Y. Yuliana and C.-H. Chang, "Dcade: divide and conquer alignment with dynamic encoding for full page data extraction," *Applied Intelligence*, pp. 1–25, Jul. 2019. [Online]. Available: https://link.springer.com/article/10.1007/s10489-019-01499-0

[19] B. Liu, R. Grossman, and Y. Zhai, "Mining data records in web pages," in *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*. New York: ACM, 2003, pp. 601–606. [Online]. Available: https://dl.acm.org/doi/10.1145/956750.956826