Introduction to Data Science
Assignment #2 – Data Crawling

102062203 張偲宇

- **Usefulness**

最近一年內 Twitch 在 VOD 實況紀錄檔裡新增聊天重播的功能，讓使用者可以不用即時跟隨 Live 實況觀看，而使用者能在自己有空閒的時間隨時觀看 VOD 影片與留言重播，並且可以更容易地進行爬取留言資訊。

爬取到的 data 資訊包含**頻道名稱、時間戳記、使用者名稱與暱稱、留言文字、判斷是否為 MOD 或訂閱者**等，所有 data 資訊欄位皆為有意義的值，因此這個 dataset 是 clean 的，不會有 missing data 的情形產生。

輸出時利用在資料中間加入 TAB (\t)符號進行欄位分隔，可以匯入到 Excel（選擇以 TAB 分隔匯入、資料欄位設定皆為文字型態），整理成**清楚易讀、容易分享**的表格格式。

上述可以爬取到的留言資訊都有其重要性，後續可能可行的分析、用途如下列舉數項：

1. 根據 VOD 總留言量(包含訂閱與非訂閱的留言比例權重)與 VOD 時間長度、實況當時的尖峰/離峰時段的比較，或許可以量化該頻道的人氣指數。
2. 根據某使用者的留言量判斷該使用者在該頻道的活躍程度。
3. 根據 MOD 的留言量判斷該頻道的管理或宣傳互動程度。
4. 根據在某短時間區段內的留言量與留言內容判斷該 VOD 片段是否為 highlight，並且這些留言內容的意涵有很大機會是相同類似的，可以進行建立語意的 clustering/classification。
5. 在各熱門頻道的各 VOD 爬取留言資訊中的使用者名稱與暱稱，或許可以抓到頗大部分的 Twitch 使用者。

在 **VOD 的總留言量多或 VOD 時間長度長**的情況下，留言資訊的 **dataset 大多會頗大**，且**能在較短時間內爬完整個 dataset**。例如在 blusewilly_retry（魯蛋）頻道的某個 VOD 時間長度約為 3 小時 15 分，留言約為 26000 筆，利用程式約 4 分鐘爬完整個 dataset；在 asiagodtonegg3be0（統神）頻道的某個 VOD 時間長度約為 8 小時，留言約為 52000 筆，利用程式約 9 分鐘爬完整個 dataset。

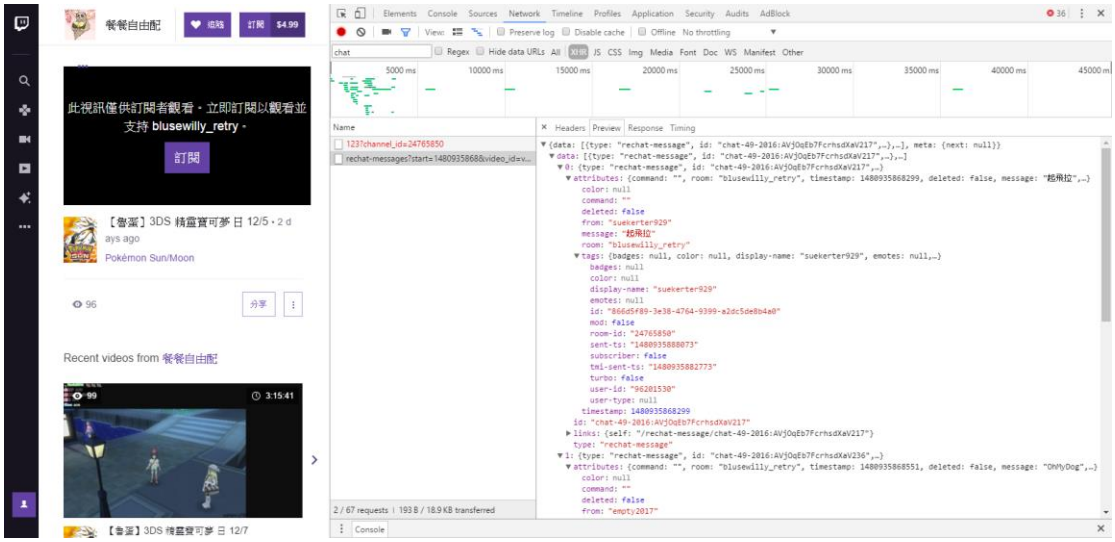
- **Uniqueness**

目前尚未在網路上 google 搜尋到有完整 twitch VOD 所有留言資訊的相關 dataset。

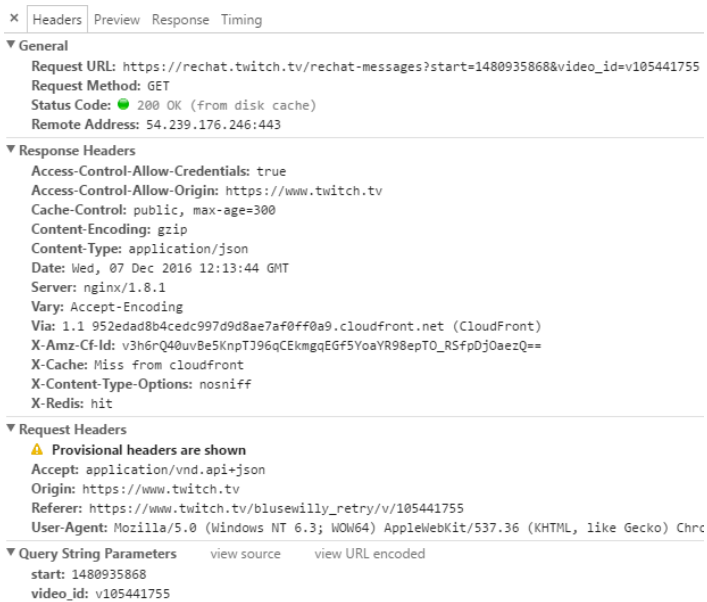
自己利用下列的 Technical depth 所提到的技巧爬出給定 VOD 的所有留言資訊，並整理成易讀的 Excel 表格格式，如最下方連結的 dataset 所示，往後若要進行分析即能較容易地抓出特定的 feature 資訊。

- **<u>Technical depth (including problems encountered & solved)</u>**

由於 Twitch API (api.twitch.tv/)好像沒有可以直接爬取 VOD 留言重播的資訊，為了要找出瀏覽器是如何 send request 給 Twitch server 而獲得含有留言資訊的 response，因此在有 VOD 的網頁上按下 F12 進行快速查看網頁元件資訊，其中在 Network 的 XHR 裡可以搜尋到**含有”chat”字眼的 request/response**，在 Preview 時顯示的 json 格式內容可以更確定該 request 就是留言重播的 response 資訊，如下圖。



切換到 Headers 裡可以查看 **Request URL** 的 domain name 網址模樣，如下圖。根據 Request URL 的 Query String Parameters，我們可以**設定 start 與 video_id 的值進行 query** 得到特定 VOD 與特定時間點的留言資訊，並且**不受限於使用者的身分權限**，任何人都可以爬取 VOD 留言資訊。但是在 VOD 影片的部分，擁有者可以設定限制能觀看使用者的身分，若設定為僅有訂閱者能觀看，則無法輕易抓取 VOD 影片檔案。



在 python 實作部分，**利用 requests + json 方式進行爬取與 decode json 包裝的階層式留言資訊**，爬取時雖然知道可以設定 start 與 video_id 的值進行 query，但仍不知道 VOD 時間範圍內的值，包含 VOD 起始與最後的 start 值（start 值為某個時間點的 UNIX timestamp 值），**故意將 start 值設為 0 時**（start 值為 0 代表此時間點至少在 2000 年以前），**會導致 server response 回傳 error 的訊息**，查看其中的 detail 資訊：”0 is not between {min_start} and {max_start} ”，其中{min_start}與{max_start}的值即分別為 **VOD 起始與最後的 timestamp 值**，因此將 start 值依序設為 VOD 時間範圍內的每個 timestamp 值時，即能爬取該 timestamp 的留言資訊。雖然能獲得每個 timestamp 的所有留言資訊，但太多次數、頻繁 request/response 會需要極長的爬取時間。

在觀察每個 timestamp 的留言資訊後，發現若 start 值設為第(30k)個 timestamp 時（k = 0, 1, 2, …），該 timestamp 會獲得從該 timestamp 開始到（該 timestamp + 29）範圍 30 秒內**新**的留言資訊；若 start 值設為第(30k + i)個 timestamp 時（$1 \leq i \leq 29$），則該 timestamp 會獲得從該 timestamp 開始到（該 timestamp + (29 - i)）範圍(30 - i)秒內**舊**的留言資訊。由上可知，每 30 個 timestamp 才會更新出新的留言資訊，因此只要**每 30 個 timestamp 進行爬取留言資訊即可**抓出完整的 dataset，減少總共爬取的時間長度。

在 decode json 包裝的階層式留言資訊時，可以利用 key 值去找到對應的 value 值，將 json 結構一層一層拆開並整理取得有用的資料，並根據 Excel 的 TAB 分隔格式，將資料依序輸出到檔案中。

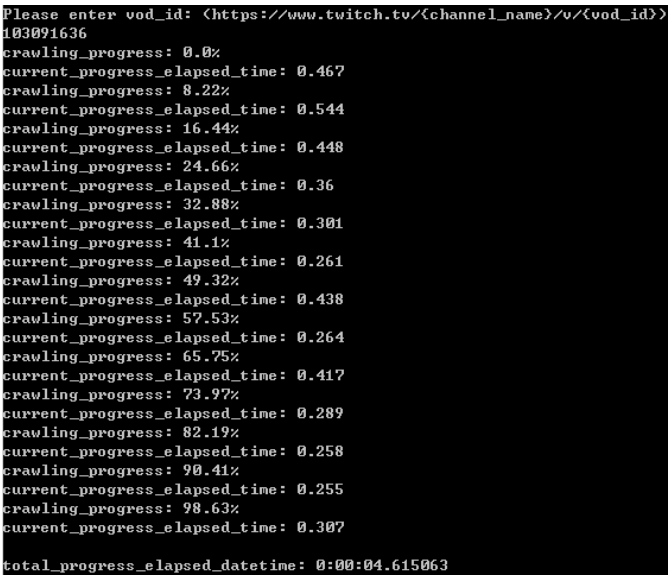
輸出的檔案內容常常會有部分亂碼產生，起初 google 搜尋關於 file、string/bytes、encode/decode 等方面的解法，經過測試後仍舊同樣的亂碼結果，後來用 json request header encoding 的關鍵字搜尋才發現真正的問題點與找到正確的解法，**得到 response 時必要將 encoding 設為 UTF-8**，如此 response.text 的結果即能正常編碼，**寫檔案時亦要將 encoding 設為 UTF-8**，檔案內容才能正常、無亂碼產生。

- **<u>Repeatability</u>**

使用方法如下：

1. 於 command line 輸入 ”**python twitch_rechat.py** ” 執行程式。
2. 根據 cmd 畫面顯示的提示範例：
   ” Please enter vod_id: (https://www.twitch.tv/{channel_name}/v/{vod_id}) ”
   輸入 VOD 的 ID 編號即可完整抓取該 VOD 的所有留言紀錄。

執行過程中 cmd 螢幕會顯示目前進度、每次 send request & receive response 所需的時間與總執行時間，如下圖。



- **URL linking to my crawled dataset**

https://docs.google.com/spreadsheets/d/12etiCzG95lU8MNNVnxxe3mgQzItgvYxmK_xIbPJTsIQ/edit?usp=sharing