Distributed Search Engine Web Crawler

Motivation

- 蒐集大量資料是現在大數據和 AI 的基礎，例如: BERT GPT3 等現今主流 AI 模型都是透過大量文本訓練而來，如何有效率且保證品質的在網路上蒐集資料是重要的議題。

- 此外，每個網站的 html 架構都有所不同，針對每個網站編寫不同的 parser rule 顯然是非常耗費人力的。

- 因此，本組這次的 Project 著重在以下兩個部分：
  - 高效率的 ，其中特別注意 URL 分配和效能
  - 設計 main content extractor，盡量適用於各種網站

Difference between Normal Crawler and Search Engine Spider

	Normal Crawler	Search Engine Spider
目的	針對特定網站，例如：Ptt / Dcard	以數個網站頁起點，將所有爬過網頁上的 URL 放入 url queue 中
防 ban	透過切換IP / 切換 user-agent 等方法減緩	可透過平衡分配URL給各個裝置來減緩
目標數量	目標數量通常較少	目標數量非常多，計算 url 數量時無法使用一般的 open addressing hash 或 chaining hash

Environment

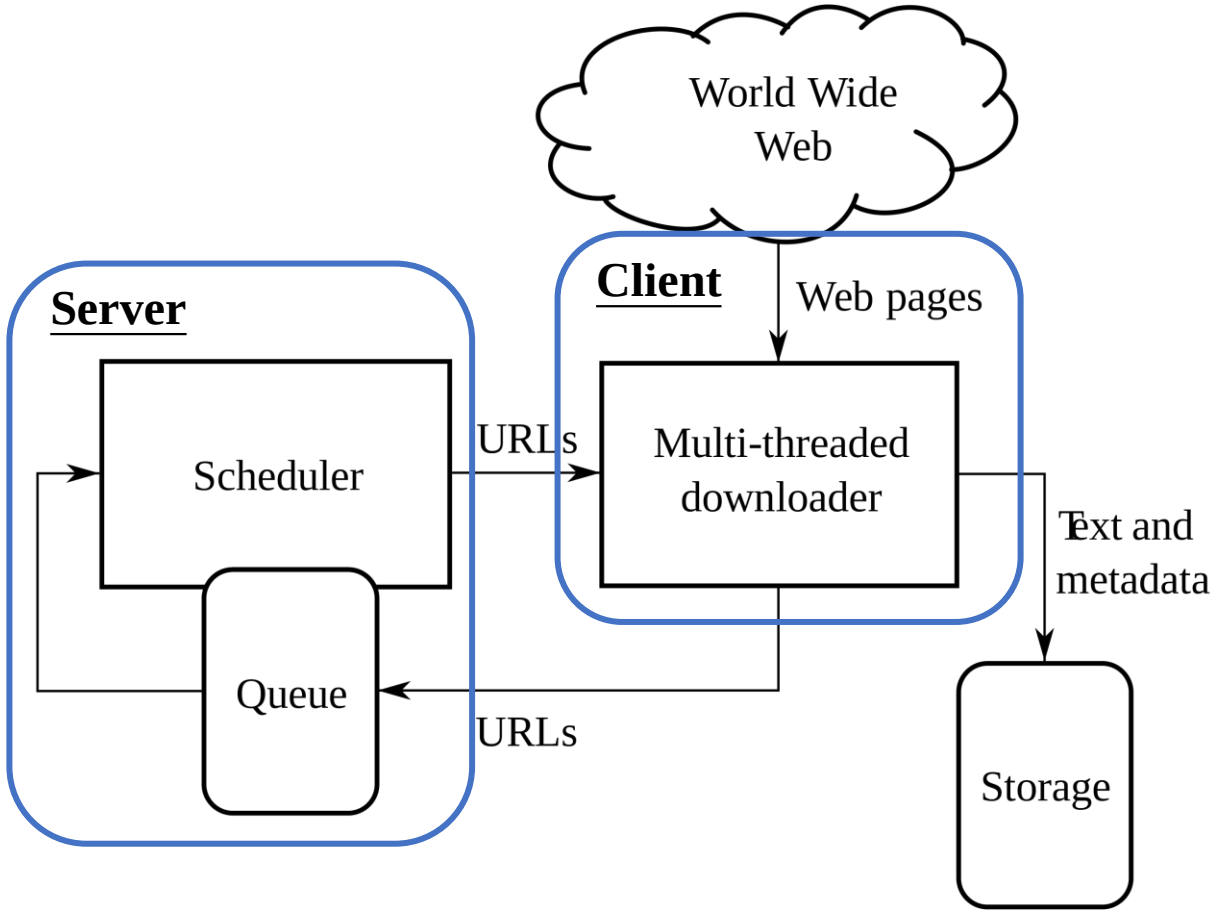
PC x 3

- Intel I9-9900K
- RAM 62 GB
- Ubuntu 18.04

Programming

- Python
- C++

Server-Client Search Engine Spider Architecture



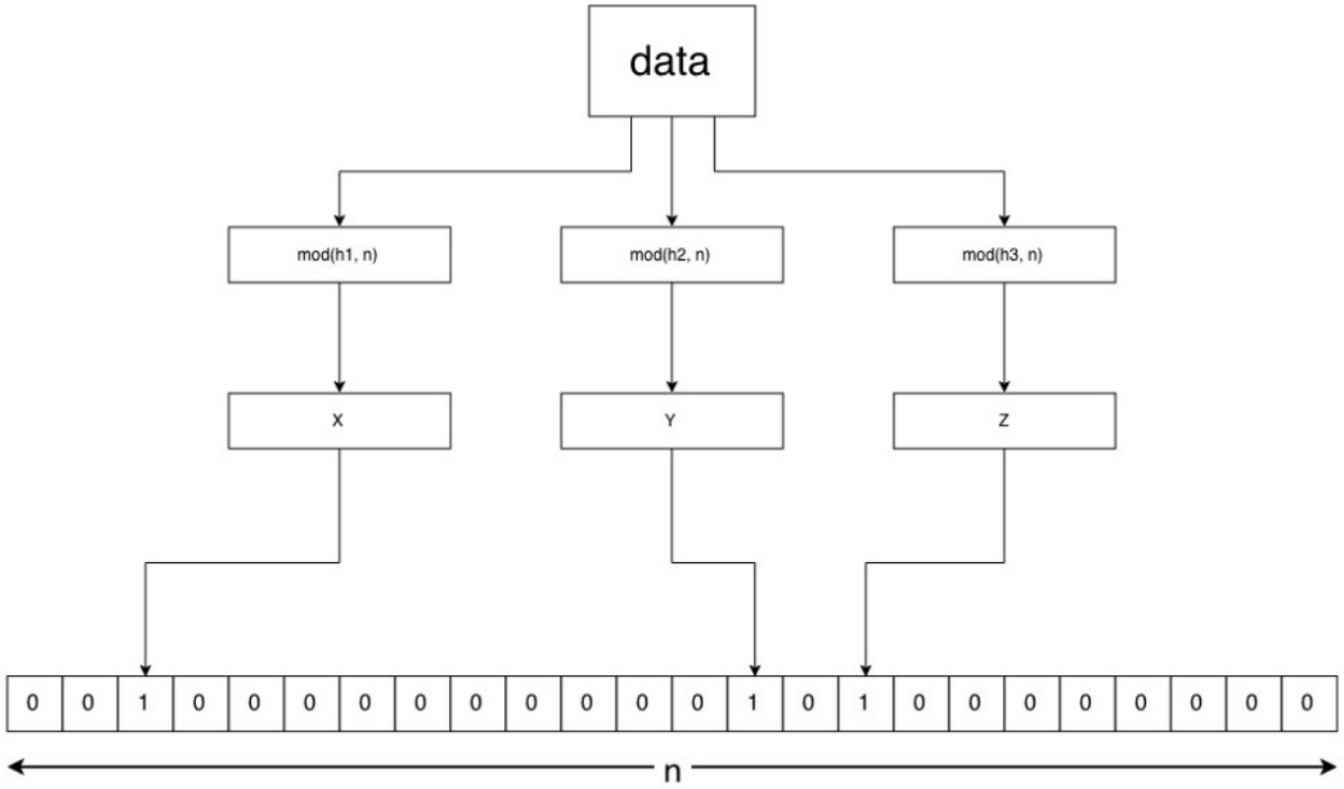
Server Side

URL Queue

- 檢查一個網站是否被爬取過，若沒有則放入 URL Scheduler
- Hash Table 會隨著資料量增大而造成大量記憶體使用
- 使用 Bloom Filter 來檢查一個網站是否被爬取過

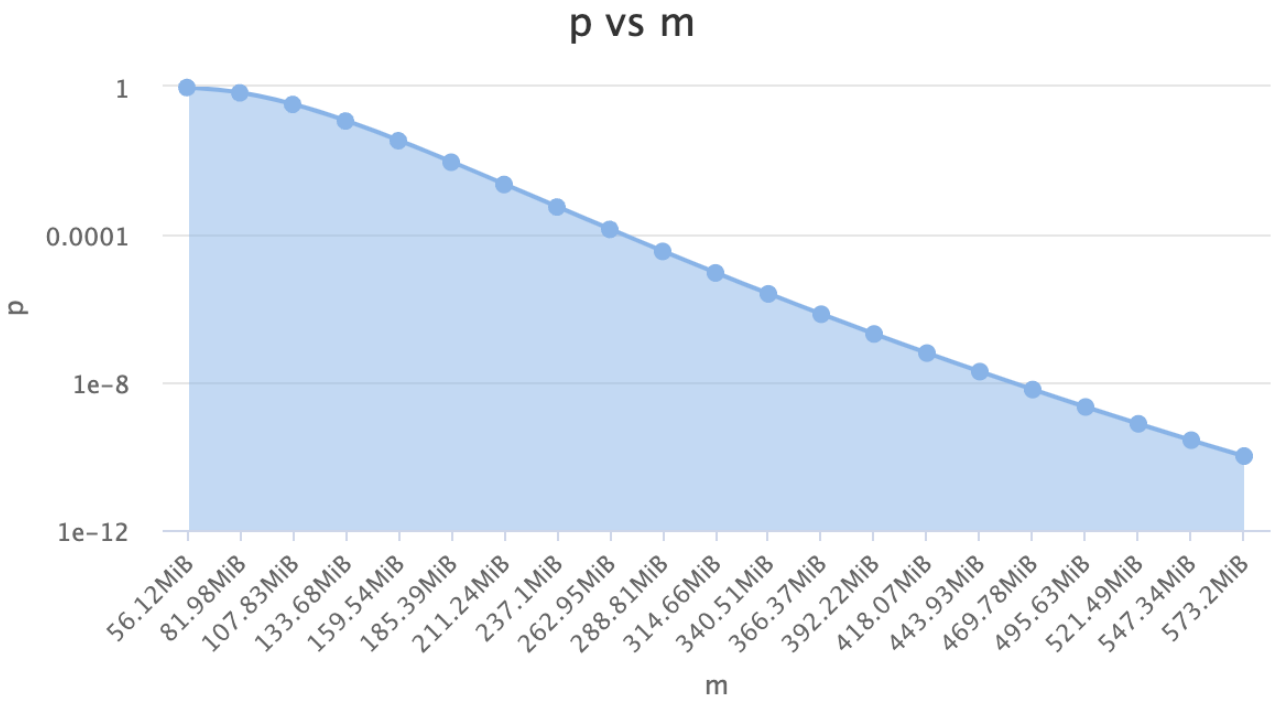
Bloom Filter

- 測試一個元素是否存在特定集合中，比起 hash table ，Bloom Filter 的空間複雜度有巨大的優勢
- 有可能發生 false positive



Bloom Filter 缺點

- False Positive
  - 如果 Bloom Filter 回傳沒有：代表資料 **一定沒有** 在 Bloom Filter 中
  - 如果 Bloom Filter 回傳有：代表資料 **可能有** 在 Bloom Filter 中，不是一定在 Bloom Filter 中
- 無法刪除資料



URL Scheduler

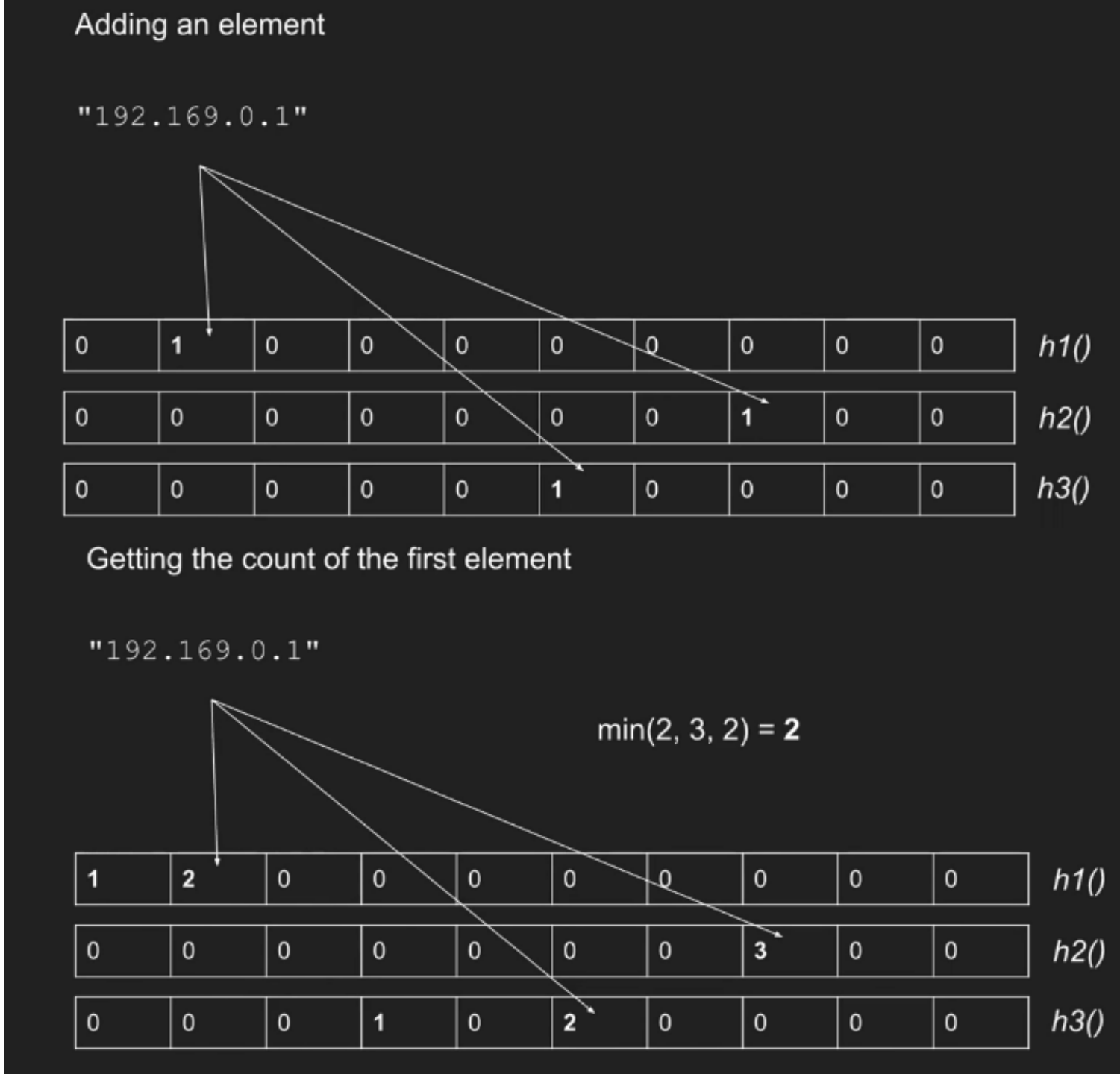
- 透過計算每個 client url domain 爬取次數，分配 URL 給各個 client
- 由 C 個 Min Priority Queue 和 Domain Counter 組成，C 是 client 數量
- 以下方法來平衡各個 ip

當Server 接收到新的 Url
• domain counter 取得該 url 的 domain 在各個 client 中已經爬取 k 次 • 將 url 放到給最少次 client 的 priority queue，權重是 k，該 client 的 domain counter + 1

當client 請求分配 n 個 urls
• 從 priority queue pop n 個 urls

Count Min Sketch

- 用來計算各個已爬取過 URL 的 domain 數量
- 組成
  - Hash function k 個
  - k * m 的 int array



Count Min Sketch 缺點

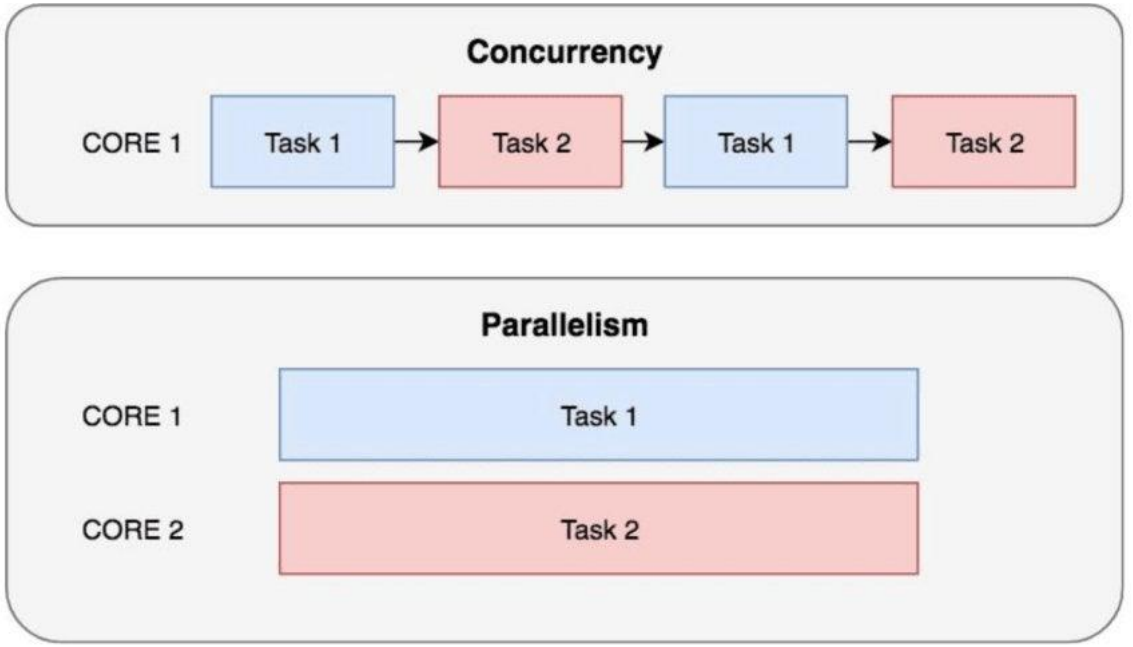
- 所得到的次數是有可能高估
  - 若有兩筆data 經過 hash + mod 後是一樣的，就會導致高估

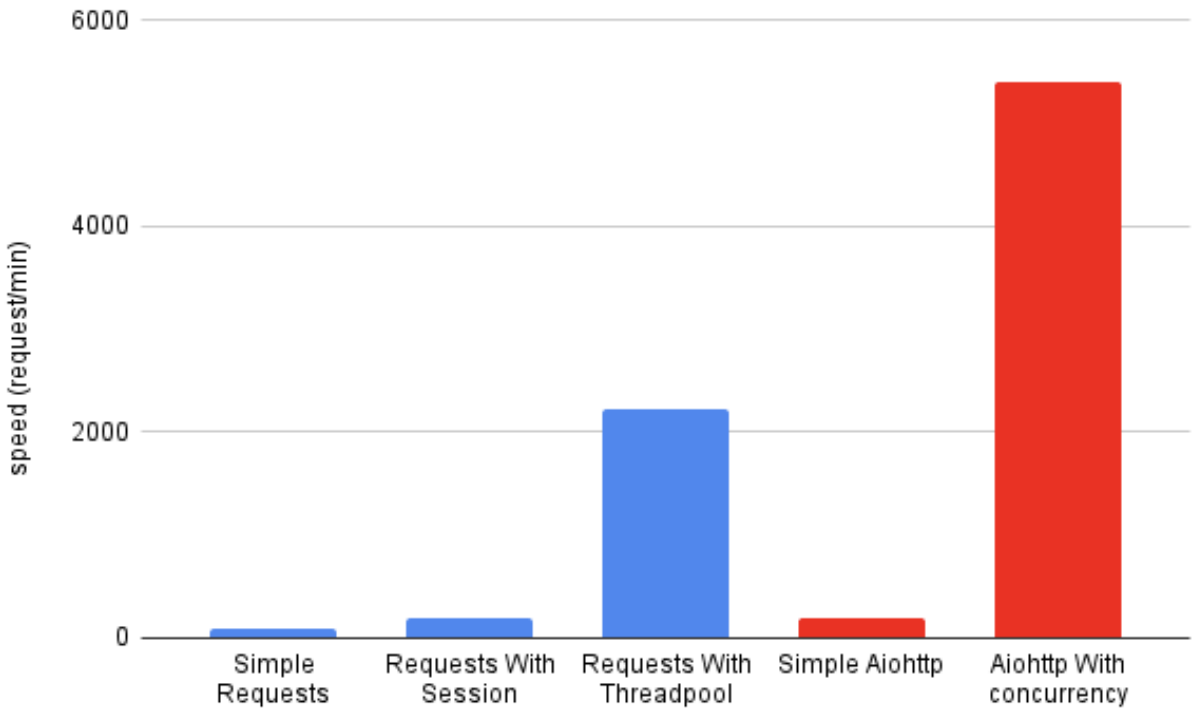
- 對於低頻率元素較不準確

Client Side

Crawler

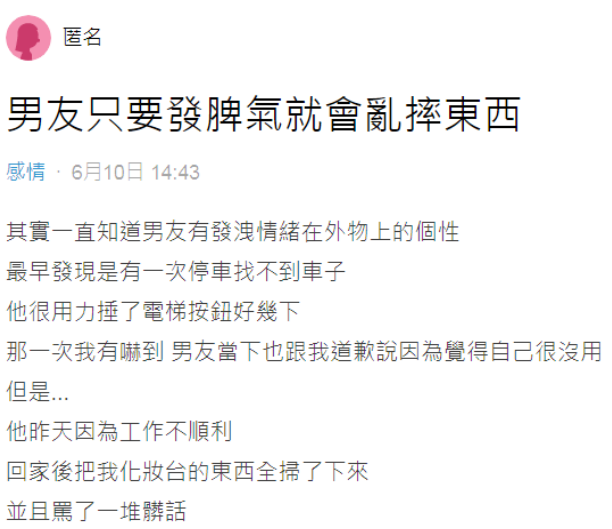
- 考量爬蟲是極端 IO bound，使用 aiohttp 實現異步爬蟲：
  - Asyncio 版的 requests
  - 相較於 mutithread requests，提升2~3倍速度，同時需要記憶體更少





Main Content Extractor

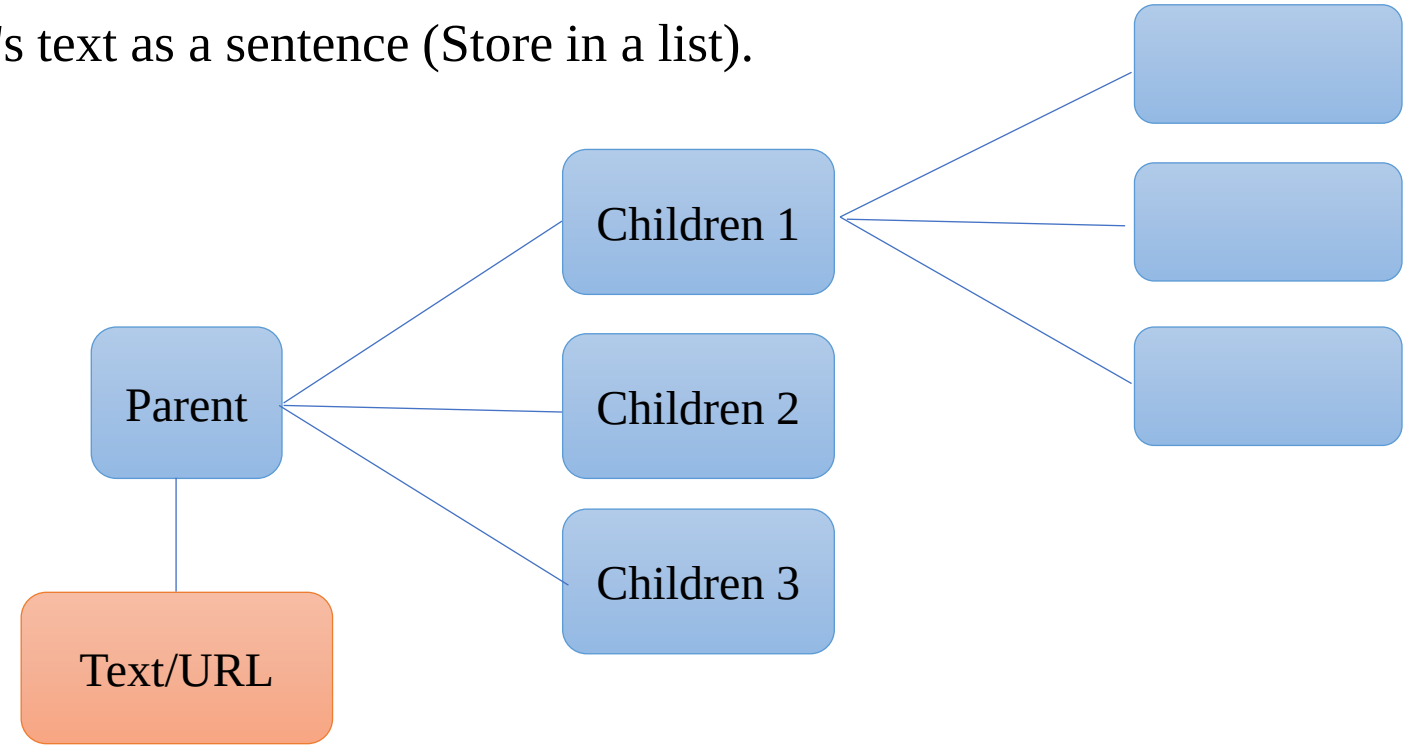
- 從爬取的html程式碼當中找到文章的重點內容 (summarization)
- 用統計的方法做 summarization (Deep learning model inference speed is too slow for crawling)



https://www.dcard.tw/f/relationship/p/239135882

Travel recurrently

- Start from root, travel its children
- If node.tag in tag ['a', 'p', 'h1', 'h2', 'h3', 'h4', 'h5', 'span', 'div']: store their texts and travel their children
- If node.tag == 'a': store the url
- View each node's text as a sentence (Store in a list).



Method 1: Naïve word frequency computing

- Jieba word segmentation
- Compute how many times a word appears in the whole paragraph as the frequency score
- Score a sentence by summing all of its words' frequency score
  (If the score is larger than mean, then the score equals to mean*(mean/original_score))
- Pick the top 20 sentences

Advantages: Fast implementation. Usually get the main content.

Disadvantages: Depend too much on word length. Influenced by the most frequent words.

{'註': 1, '冊': 1, ' ': 0.02069973953904838, '/': 1, '登入': 1, '下載': 1, 'App': 1, '所有': 1, '看': 2.773765098232483, '板': 2, '即時': 1, '熱門': 2, '看板': 1, '好物': 1, '研究室': 1, '女': 0.3962521568903547, '匿名': 2.773765098232483, '男友': 1.1095060392929932, '只要': 1, '發脾氣': 1, '就': 1.2327844881033259, '會': 2.773765098232483, '亂': 2, '摔': 1.5850086275614188, '東西': 1.008641853902721, '感情': 2.773765098232483, '6': 3, '月': 2, '10': 3, '日': 1, '14': 2, ':': 2, '43': 2, '其實': 1, '一直': 1, '知道': 1.8491767321549888, '有': 1.008641853902721, '發洩': 1, '情緒': 2.773765098232483, '在外': 1, '物上': 1, '的': 0.18491767321549887,…...}

Method 2: TFIDF

- Jieba word segmentation
- Compute TF-IDF scores for each word
- Score a sentence by summing all of its words' TF-IDF scores
- Pick the top 20 sentences

Advantages: Better catch the importance of each word

Disadvantages: A little slower. May not catch the main content

TF: term frequency
一個字在該句出現的次數/該句的總字數

IDF: inverse document frequency
log (總句數/一個字在幾個句子裡出現過)

TFIDF = TF × IDF

TF

男友	0.125
只要	0.125
發脾氣	0.125
就	0.125
會	0.125
亂	0.125
摔	0.125
東西	0.125

IDF

男友	4.505963535904644
只要	6.703188113240863
發脾氣	6.703188113240863
就	4.505963535904644
會	5.316893752120972
亂	6.010040932680917
摔	4.911428644012807
東西	4.505963535904644

TFIDF

男友	0.5632454419880805
只要	0.8378985141551079
發脾氣	0.8378985141551079
就	0.5632454419880805
會	0.6646117190151215
亂	0.7512551165851147
摔	0.6139285805016009
東西	0.5632454419880805

Example: Dcard (Naïve word frequency)

- 其實一直知道男友有發洩情緒在外物上的個性 最早發現是有一次停車找不到車子 他很用力捶了電梯按鈕好幾下 那一次我有嚇到 男友當下也跟我道歉說因為覺得自己很沒用 但是... 他昨天因為工作不順利 回家後把我化妝台的東西全掃了下來 並且罵了一堆髒話
- 電話被丟下來的時候差點打到我的腳 這次離的好近 我的腦袋空白了片刻 我請他立刻離開我家不然就報警。 冷冷地說完就是一陣的飆淚
- 他除了道歉還是道歉 不斷的死纏爛打要我原諒他 我已經把他的聯絡資訊都封鎖了 最近有恐怖情人的新聞看了真的毛骨悚然 幸好現在居家上班基本上不會出門 但真的好怕要踏出房門的那天
- 妳最近要找個可以信任的人，隨時報平安哦！
- 工作不順利就砸妳的東西還罵妳?? 到底跟妳有什麼關係啊 快跑吧，這次砸東西下次不知道會不會砸妳 請注意自己的安全，可以買個防狼噴霧放包包以防萬一
- 如果你是一個人住建議搬家 這種人就是會埋伏在妳家門口的那種
- 妳最近要找個可以信任的人，隨時報平安哦！
- 如果你是一個人住建議搬家 這種人就是會埋伏在妳家門口的那種

https://www.dcard.tw/f/relationship/p/239135882

Example: Dcard (TFIDF)

- 其實一直知道男友有發洩情緒在外物上的個性 最早發現是有一次停車找不到車子 他很用力捶了電梯按鈕好幾下 那一次我有嚇到 男友當下也跟我道歉說因為覺得自己很沒用 但是... 他昨天因為工作不順利 回家後把我化妝台的東西全掃了下來 並且罵了一堆髒話
- 他除了道歉還是道歉 不斷的死纏爛打要我原諒他 我已經把他的聯絡資訊都封鎖了 最近有恐怖情人的新聞看了真的毛骨悚然 幸好現在居家上班基本上不會出門 但真的好怕要踏出房門的那天
- 工作不順利就砸妳的東西還罵妳?? 到底跟妳有什麼關係啊 快跑吧，這次砸東西下次不知道會不會砸妳 請注意自己的安全，可以買個防狼噴霧放包包以防萬一
- 這種人真的要不得! 好險你夠果決! PS:好市多衛生紙 & 小蜜瓶好熟悉的身影XD
- 趕快分手吧…這個性太差了 你能想像跟他結婚後的日子嗎
- 是我的話絕對噴爆他 保養品跟吹風機很貴 叫他賠你一罐小蜜瓶
- 我媽都說男生情緒失控摔東西 之後就會摔妳了
- 每天都要跟朋友們報平安喔！ 然後趕快把他東西放出家門，最好放一個監視器隨時可以看，如果他對你怎麼樣直接報警。

https://www.dcard.tw/f/relationship/p/239135882

Example: PTT (Naïve word frequency)

- 而是他們能封鎖那隻球隊贏球的命脈 以下是賽提今年季後賽每輪的DRTG 首輪-籃網
- KD不是被守死了嗎，為何籃網反而是從賽提手下拿到最多的分數 因為這就是賽提精心設計的防守策略 有每輪都跟到的話 就會知道賽提防守策略常常逐場改變 第一輪面對籃網，策略就是
- 封鎖KD 給KD滿滿的身體對抗，降低他個人的破壞力 KD的命中率整體命中率是
- 0.386 這是繼他第一年打季後賽之後最差的一次 但反效果就是籃網的其他人獲得極大的空檔 全隊整體命中率 0.503 三分命中率
- 寧願鎖死KD給其他人空檔，因為KD打不好你們就贏不了球（籃網季賽 就因為KD缺賽連敗十多場）第二輪打公鹿，第一場的策略是
- ※ 發信站: 批踢踢實業坊(ptt.cc), 來自: 36.236.15.247 (臺灣)
- : 這可以稍微解釋嘴綠的表現，完全不給嘴綠Roll in
- : 所以我說不要看誰盯誰，中鋒才不容易被換出去

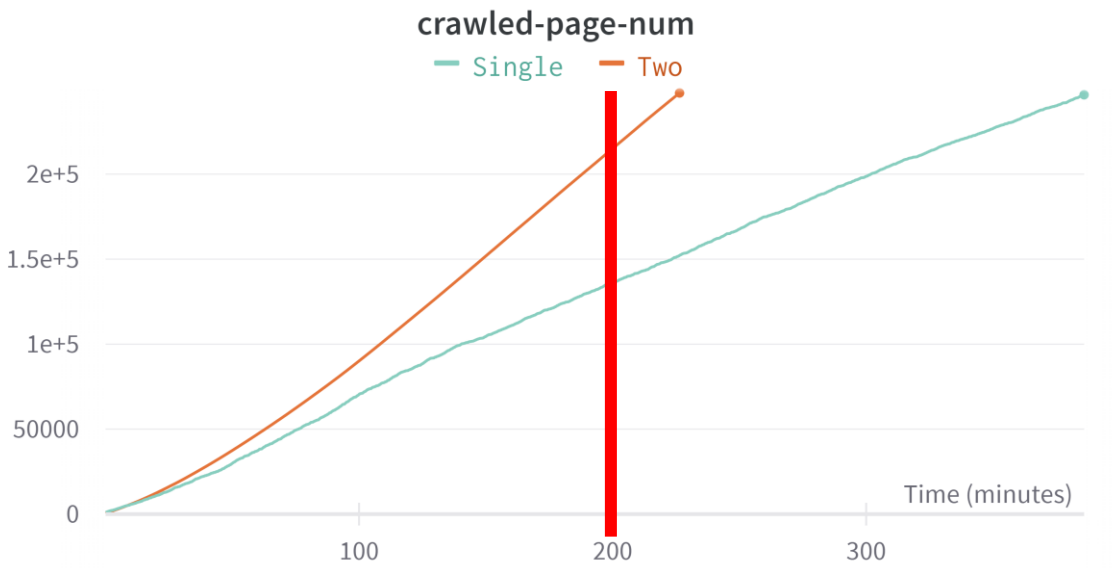
https://www.ptt.cc/bbs/NBA/M.1654956930.A.A87.html

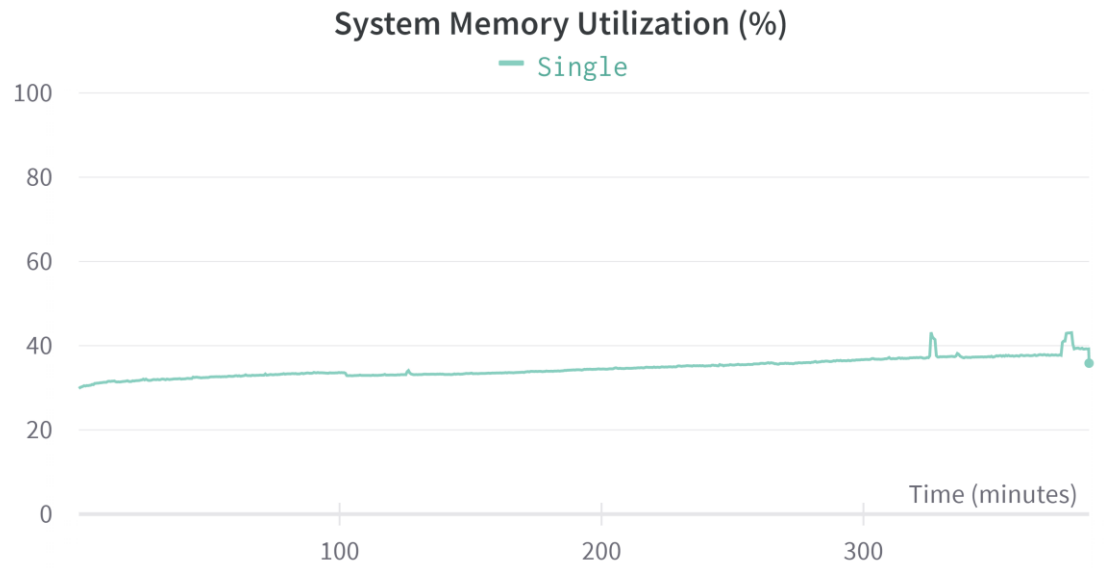
Example: PTT (TFIDF)

- 寧願鎖死KD給其他人空檔，因為KD打不好你們就贏不了球（籃網季賽 就因為KD缺賽連敗十多場）第二輪打公鹿，第一場的策略是
- 包夾字母 結果字母哥傳出了12次的助攻，公鹿全隊投出了12顆三分，公鹿帶走了第一場勝利 第二場開始調整對策，讓GW和活佛去對位字母 使用1盯1的防守，封鎖字母與隊友的連線 從第二場開始計算，字母只傳出場均5.8次的助攻，整整少了第一場快6次 字母的出手由第一場的25次提升至29次
- 這六場內字母場均得了37.6分，但公鹿全隊也僅僅只得了97.2分 面對公鹿，目標就是
- 以大家才會說蜜豆湯在這輪到底多重要 第三輪熱火，
- 對手的進攻主軸是DHO起手創造三分機會 賽提採用第一線防守者繞掩護阻止第一拍三分出手 中鋒沉退放給你投中距離，不放禁區 熱火也陷入嚴重打鐵，阿爹隱形的畫面 而熱火可以跟賽提鏖戰七場的主因在於他們利用防守製造快攻的得分，以及吉巴無雙級的演 出 吉巴不只透過快攻搶分，連三分球都投進不少
- : U文推 咖哩吸包夾製造局部多打少的戰術這輪幾乎廢一
- : 成的（單指短擋拆後找不到隊友空檔的問題，命中率慘
- : 禁區擠滿人放你投外線還進不了那是沒辦法玩耶

https://www.ptt.cc/bbs/NBA/M.1654956930.A.A87.html

Experiments





Future Works

- 考量不同網站 ban 爬蟲的原則
- 針對不同網站調整出適合的 parser rule
- 重新存取問題，檢查網頁是否更新

Code

- https://github.com/aqweteddy/ServerClientSearchEngineCrawler

Reference

- 中正大學吳昇教授<網際網路資料檢索授課內容>授課內容
- Bloom Filter https://medium.com/@Kadai/%E8%B3%87%E6%96%99%E7%B5%90%E6%A7%8B%E5%A4%A7%E4%BE%BF%E7%95%B6-bloom-filter-58b0320a346d
- Count Min Sketech https://titanssword.github.io/2018-02-23-Bloom%20Filter%20and%20Count-Min%20Sketch.html
- Asyncio https://dev.to/welldone2094/async-programming-in-python-with-asyncio-12dl