Data Analysis and Crawler Application Implementation Based on Python

基於Python的資料分析和網路爬蟲應用

摘要：

- 在這個信息爆炸的時代，如何從各種雜亂無章的數據中高效地找到我們想要的數據，並批量從網絡中提取出來，成為了一個關鍵問題。而有時候自己沒有處理的數據可能會讓人迷惑，通過什麼樣的技術手段，如何把複雜的數據處理，最終變成一種直觀的數字，或者說人們可以直接從中提取信息的趨勢也是一個趨勢。在數據時代非常重要的研究課題。本課題將選擇Steam網絡遊戲平台作為研究對象。 Steam是美國 Valve 公司於 2003 年推出的在線遊戲零售平台。
- 基於Scrapy框架為Steam暢銷榜和發行商、開發商和商店頁面開發完整的爬蟲方法，在Steam平台頁面下爬取開發商和發行商的所有作品的各種數據。
- 基於爬取的數據，通過基礎數據分析，分析用戶最喜歡的暢銷遊戲類型、部分開發商和發行商的遊戲平台發布總數、好評比例等，通過數據分析過程。在決賽中得出結論和總結。總之，本文首先將探討如何開發一種具有可控、自動爬取能力的爬蟲，可以對特定目標進行爬取；然後使用Pandas庫和Matplotlib庫對爬取的數據進行分析和可視化，在數據分析和可視化過程中提取有用的信息，從而完成。

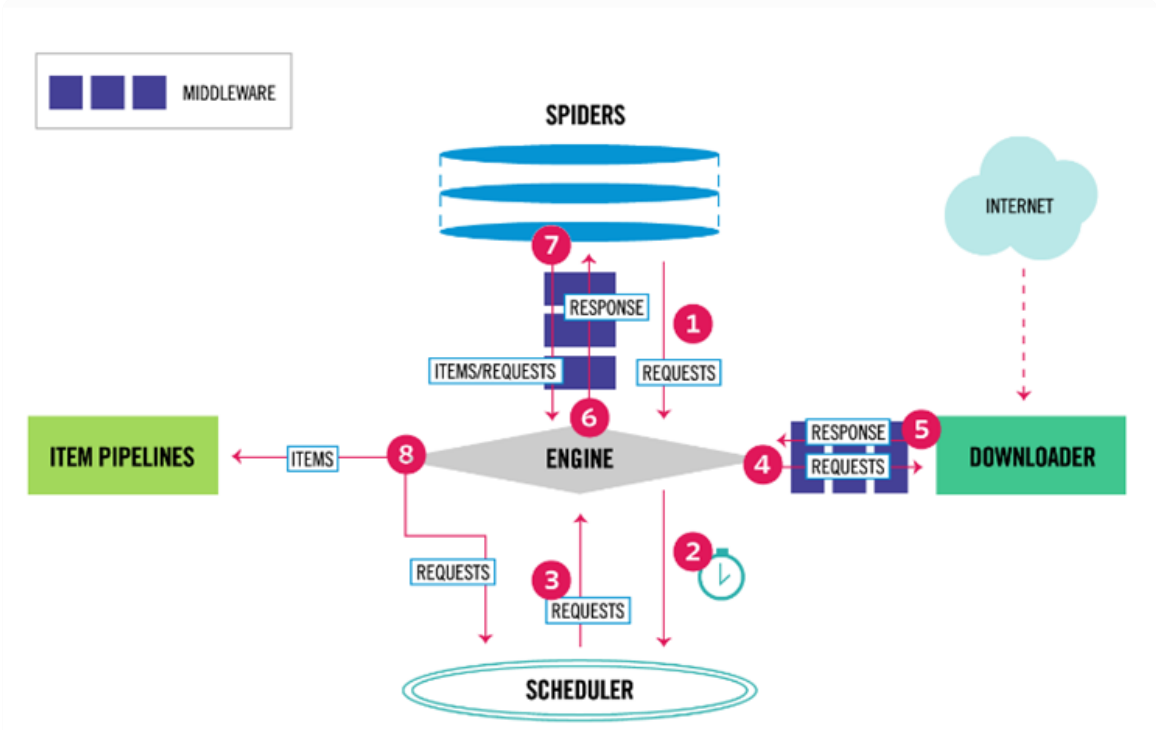
I. INTRODUCTION

- 大多數時候，數據都包含在互聯網中，數據龐大而雜亂無章。 使用傳統的人工手段很難將這些雜亂無章的數據從互聯網上分離出來，總結出有用的信息。

- 這就是為什麼我們需要利用計算機爬蟲的手段，對互聯網上存在的信息進行高效、自動的爬取，利用技術手段對數據進行分析，總結規律。
- 近幾十年來，隨著互聯網的飛速發展，在爬行的數量和難度、爬行種類不斷增加的同時。
- 現代網絡爬蟲一般都使用庫來開發，很多語言都有自己的爬蟲庫，在不同的爬取功能上各有優勢。 Python相關爬蟲廣泛應用於各個領域。 Python中常見的爬蟲框架有：scratch、Crawley、pysipider、cola、demiurge、robobrowser等，本文將重點研究基於python的爬蟲框架。

II. RELEVANT TECHNOLOGIES AND FRAMEWORKS

- 在爬蟲框架方面，本文主要選擇scrapy作為爬蟲的框架，並使用beautifulsoup作為本課題的爬蟲分析庫。 在scrapy的基礎上，加入了selenium，幫助爬取動態頁面。 scrapy框架的基本原理如圖1所示。



  - SPIDERS(爬蟲程式)：撰寫Python網頁爬蟲程式碼的地方，向ENGINE(引擎)發送網頁請求，以及將ENGINE(引擎)所接收的回應結果進行解析與爬取。

    - ENGINE(引擎)：Scrapy框架的核心模組，就像汽車的引擎一樣，負責控制各個模組、傳遞請求及資料。
    - SCHEDULER(調度器)：將ENGINE(引擎)所接收的SPIDERS(爬蟲程式)請求進行列隊，也就是排隊的意思，來調度請求的順序。
    - DOWNLOADER(下載器)：負責下載ENGINE(引擎)接收到SCHEDULER(調度器)調度請求的網頁HTML原始碼，提供回應結果給ENGINE(引擎)。
    - ITEM PIPELINE(資料模型管道)：將SPIDERS(爬蟲程式)所取得的資料進行後續處理，像是資料清理、存入資料庫(例：MySQL)或存入檔案文件(例：CSV、JSON)等。
    - 1.ENGINE(引擎)接收SPIDERS(爬蟲程式)所發送的一至多個請求。
    - 2.ENGINE(引擎)將請求傳遞給SCHEDULER(調度器)進行列隊。
    - 3.ENGINE(引擎)向SCHEDULER(調度器)提取下一個所要發送的請求。
    - 4.ENGINE(引擎)將請求傳遞給DOWNLOADER(下載器)。
    - 5.DOWNLOADER(下載器)將ENGINE(引擎)傳遞的請求網頁HTML原始碼下載下來，並且回應結果給ENGINE(引擎)。
    - 6.ENGINE(引擎)將回應的結果傳遞給SPIDERS(爬蟲程式)。
    - 7.SPIDERS(爬蟲程式)進行結果的解析及資料的爬取，組成ITEMS，傳遞給ENGINE(引擎)或發送新的請求。
    - 8.ENGINE(引擎)判斷SPIDERS(爬蟲程式)所傳遞過來的如果是ITEMS(資料)，就會傳遞給ITEM PIPELINES(資料模型管道)，進行後續的資料清理及儲存等。反之，如果傳遞過來的是新的請求，也就是相當於第一個步驟，接著，傳遞給SCHEDULER(調度器)，以此類推，重覆這樣的流程，直到SCHEDULER(調度器)沒有請求為止。
- Scrapy 框架在一般網絡爬蟲的應用中非常流行。 它的第一個版本是在 2008 年發布的，現在它作為一個爬蟲框架已經相當成熟了。

- 在數據分析方面，項目主要使用Python的pandas庫和Matplotlib可視化庫對爬取的數據進行基礎數據分析和數據可視化。

III. DESIGN OF CRAWLER

- 基本思路是從熱賣頁面開始，遍歷列表，爬取並存儲，然後進入每個鏈接讀取產品的子頁面，抓取所有需要的信息並記錄下來，並傳遞給下一個模組爬取廠商頁面，爬取廠商基本信息和廠商遊戲商店鏈接，最後通過遊戲商店鏈接爬取各個廠商旗下所有遊戲的基本信息。
- 整個設計的難點在於商店頁面的一部分是Ajax動態異步加載頁面。
- 如果只用scrapy做靜態頁面爬取，有些數據是加載不出來的，所以還需要selenium來模擬用戶操作來實現動態頁面爬取。 而且steam的動態廠商頁面還處於測試階段，並不是所有廠商都有動態加載首頁，所以如何判斷廠商的頁面是否是動態頁面也需要注意。
- Scrapy 爬蟲架構本身主要由Items、spiders、Pipeline 、middlewares組成。
    - Items主要用於定義要爬取的Items
    - spider負責定義整個爬取過程和爬取的手段。
    - Pipeline 負責一些基本的操作，比如數據的清理和保存。 您可以在此處定義輸出爬網結果。
    - middlewares可以負責為 Scrapy 和其他插件或架構橋接服務。
- 此外，Scrapy 爬蟲架構提供了 Settings 文件，用戶可以根據需要控制 cookie 的使用、爬取速度限制、聲明管道中添加的項目等。
- item design：在這個項目設計過程中，分別定義了devitem開發者的項目、pubitem發布者的項目和gamitem遊戲的項目。
    - 此處定義的需要爬取的項目也是根據要執行的最終數據分析的要求來確定的。 例如，開發者和發行商的 Nam 供應商名稱和遊戲列表的 Gam_title 可以設置為 Index 作為查找特定數據的唯一標識符。

    - PUBITEM 出版商信息表
    - DEVITEM 開發人員信息表
    - GAMITEM 遊戲信息表
- Spider design
    - 蜘蛛設計是這個項目的重點。 它定義瞭如何獲取項目的實體。 無論是如何從初始廠商動態頁面上的廠商產品商店獲取信息，還是如何從最後一個商店的靜態頁面或暢銷榜上的遊戲列表獲取信息，都將定義在這個文件。
    - 本項目中spider主要設置在spider類的count start_Requests方法、top_sell_Parse方法、store_Parse方法、DP_Parse方法和Gam_parse方法中實現了上述抓取物品的功能。
    - start_requests 方法負責讀取 Spider 類中定義的 start_URL 熱門列表 url，並將其 Request 傳遞給下一個 top_sell_PARS 方法。
    - top_sell_parse 為top lists爬取方法，主要負責爬top lists list，主要使用beautifulsoup解析sell like hot cakes search_resultsRows的訪問列表在內容中的鏈接中的錨標籤，並將請求的鏈接列表傳遞給 下一種解析方式，可以修改。 Find_all 字段的limit of sell like hot cakes top 多少調查控制，在這個程序中定義了15 的限制，即調查廠商命中列表中的Top15。
    - store_parse 接收到上述方法的請求，解析並爬取暢銷榜中每件商品的店鋪頁面。該方法可以對爬取的商店頁面中的開發者頁面和發布者頁面進行分類。由於開發者和發布者的鏈接信息存儲在頁面 details_block 的 A TAB 中，因此需要從獲取的 URL 中確定開發者和發布者這兩個詞，以確定爬取目標是開發者頁面還是發布者頁面。
    - 最後，該方法循環獲取Top_dev_list開發者列表、Top_pub_list發布者列表，輸出獲取的結果，並通過傳遞一個標誌告知後面的DP_parse供應商頁面store_parse方法返回的請求是抓取開發者頁

面還是發布者頁面。在DP_parse方法中，DP即Developer & Publisher，負責爬取開發者和發布者頁面，返回開發者和發布者信息。 dp_parse首先使用if語句判斷傳遞給他的請求，決定實例化開發者信息對像還是發布者信息對象，因為開發者頁面既有靜態頁面也有發布者動態頁面，在判斷傳入對像是開發者還是發布者，dp_parse方法會繼續分析傳入的對象連接，檢查是否包含'/search/field是動態頁面還是靜態頁面，判斷是否使用Selenium爬取動態頁面。

- Steam商業動態頁面會將平台上發布的各類產品的商業總數寫在網頁上，你要抓取的產品列表在頁面加載開始時只會加載10個產品，所以需要Selenium on web頁面模擬人工下拉操作，對這些項進行加載，但每次下拉只會在原來的基礎上在此加載10個條目，需要讀取爬取項目的總項數，除以數量超過10個就記下來，所以可以根據條目總數下拉列表設置個數，如果總數不超過11個，直接閱讀列表。 Beautifulsoup 在關閉瀏覽器之前讀取加載的動態頁面並爬取 url 列表。
- 瀏覽器設置為不加載圖片和CSS模式以節省系統資源，實現更高效的爬蟲。每個物品的具體存儲連接在物品之間都存在一個錨標籤，使用循環讀取這些連接到定義的link_list列表中，完成列表爬取，但有時在詞條和圖片的條目中可能包含一個標籤，並且會指向同一個頁面，如果是直接的話
- 應用程序可能會導致重複爬取，使用if not on the list 來加重語句循環。
- gam_parse 方法從 DP_PARSE 方法傳入的每個商家頁面接收所有遊戲商店的鏈接，從每個供應商處爬取最終的遊戲商店頁面，並返回 Items 中定義的每個遊戲的基本信息實體，例如游戲名稱、媒體分數，播放器媒體聲譽，等等。在爬取過程中，完成了一些

沒有標註遊戲類型或開發者的老遊戲，或者一些沒有評級和評價的數據。那是蜘蛛的末日。

- Pipeline design
  - 在 Scrapy 中，管道是處理捕獲的 Item 的管道。一般負責完成爬取數據的清理和保存操作。本爬蟲項目中的Pipeline主要負責Item的輸出和CSV文件的創建和寫入。 DevItem_to_CSV、PubItem_to_CSV 和 GamItem_to_CSV 類在 Pipeline 中定義。他們負責將開發者、發行商和遊戲信息項導出到同一文件夾中的不同 CSV 文件中。所有三個類都被設計為與類中的方法相似。
  - init方法中的初始化定義了文件的輸出目錄，使用w+file的寫入方式，每次覆蓋用rewrite現有文件的輸出，調用write row(write)函數到文件的第一行文件到項目的每一列，在process_Item方法中定義了特定Item的輸出，使用if實例判斷是否輸出Item，因為每個Item輸出的文件不同，不同的Item入口也不相同，所以需要用於判斷。最後，當蜘蛛關閉時關閉文件。除了 Items 輸出文件的管道外，還定義了管道和數據分析管道用於輸出文件夾檢測。輸出文件夾檢測管道會在前一個文件的輸出管道之前執行，檢測CSV文件的OUTPUT文件夾是否存在。如果該文件夾不存在，則會自行創建。數據分析管道負責在Spider完成爬取後對保存的數據進行數據分析，最後將文件寫入CSV文件。這裡只列出流水線中數據分析方法的定義，具體數據分析方法的代碼和解釋將在本章之後進行。
  - 實例化數據分析方法類，調用數據分析方法中的CSV_reader方法開始數據分析。 Pipeline寫好後，resettings文件聲明每個Pipeline類，並設置每個類的運行優先級，完成文件在Pipeline中的輸出類的定義。
- HeaderRdm design
  - 值得注意的是，網站在進行大量爬取時可能會懷疑受到攻擊並要求提供 Headers。 Headers 請求是在普通瀏覽器中瀏覽 Web 時將

傳遞到 Web 瀏覽器和系統的數據。 有時服務器使用它來確定它是否是腳本操作。

    - 另外，由於有時在訪問特定頁面時需要輸入年齡，輸入後這些年齡數據會保存在cookies中。 雖然不是每個頁面都這樣，但是如果在爬蟲過程中遇到這樣的頁面，就會報錯，導致無法爬取。 因此，HeaderRdm方法提供了請求頭和cookie參數的列表，並使用Python提供的隨機庫，使用headers['user-agent']=Random。 選擇（USER_AGENTS）語句每次調用該類中的方法時返回一個隨機的請求頭，從而達到一定的反爬蟲目的。

- Crawl results
    - 本程序編寫了一個名為start的py文件，通過調用CmdLine庫的函數直接在IDE中運行Scrapy程序，避免了很多麻煩。 根據 Scrapy 返回日誌，共取了 724 頁，最快的速度是每分鐘 96 頁，總共 17 分鐘，平均每分鐘 40 頁。 由於部分廠商規模小，沒有廠商頁面，所以follower數據為none。 另外，由於部分遊戲發佈時間較早，商店頁面上的媒體當時沒有評級，所以部分遊戲也被評為無。 具體結果見下圖。

IV. DATA ANALYSIS

- 本程序的數據分析功能是利用pandas庫和Matplotlib可視化庫讀取保存在CSV文件中的數據，對爬取的數據進行基本的數據分析和可視化操作。
- 該功能通過CSV_reader、dir_finder、get_date、datanls_init實現，由四個模塊和幾個數據分析模塊組成。 get_Date 獲取當前系統時間，以字符串形式返回月、月、日的組合時間。 dir_finder方法主要用於檢測項目目錄下是否有Datanls數據分析結果輸出文件夾。 如果使用if isdir方法判斷目錄是否已經創建，則使用MKDIR創建目錄。 在此過程中，調用 get_above。

- date 方法獲取當前系統時間，並使用返回的日期字符串創建相關目錄。 csv_reader方法首先調用dir_finder驗證輸出目錄是否存在。 然後我用 csv.reader 方法從前面爬蟲輸出的三個文件中讀取數據，並作為參數傳給dataframe傳給datanls_init初始化模塊。
- datanls_init模塊主要負責初始化各個數據分析模塊，在數據分析前對各個信息進行匯總，模塊會不斷地將各個數據以參數的形式傳遞給各個數據分析方法。
- 數據分析可視化模塊包含多個子模塊，包括Ave_score_C廠商作品平均玩家好比/媒體平均分分析模塊、DP_scorer_C廠商作品玩家評價比例分析模塊、Pt_flw_C廠商粉絲排名模塊、Pt_ sum_C分佈排名模塊，sum_tpye_所有餡餅廠商的遊戲類型占比分析模塊。
- 前四個模塊可以根據flag開發者或發布者的輸入參數輸出不同的圖表。以下是部分分析結果。
- Top seller list vendors focus on the visualization module
  - 最暢銷的銷售商關注可視化模塊
- Top seller list manufacturers game total type proportion visualization module
  - 暢銷榜廠商遊戲總類型比例可視化模塊
- The top 5 analysis visualization modules in the list of Top seller list
  - 暢銷榜榜單前5名的分析可視化模塊
- The visualization module of the percentage of word-of-mouth comments of game players from top seller manufacturers
  - 暢銷廠商遊戲玩家口碑比例可視化模塊
- The average critical rating of game players /the average media score of visualization module
  - 遊戲玩家平均評分/可視化模塊平均媒體評分

V. SUMMARY

- 本文通過爬取Steam網絡遊戲商城熱銷頁面的過程，探討了使用Selenium庫和Python Scrapy框架對動態頁面進行數據爬取和基礎數據分析的過程，並對最終的數據進行分析。
- 在我看來，無論是爬蟲還是數據分析的各個模塊都有很好的擴展性。在爬蟲反爬蟲方面，Selenium 本身俱有非常好的爬蟲反爬蟲能力。 如果想進一步做爬蟲反爬蟲，還可以擴展多個cookie，甚至設置代理IP池等等。 數據分析部分還可以使用一些更複雜的計算模塊，比如記錄日常數據，對爬取的數據進行趨勢分析等。