Scraping Discord

Francesco Pinsone

Indice

.

1 Introduzione

L'obiettivo è quello di estrarre dati sensibili su possibili minacce informatiche da Discord. La suite di script implementati quindi ha l'obiettivo di effettuare un'attività di scraping su Discord, per fare questo ci sono diverse strade percorribili. Discord mette a disposizione una sua libreria Python con la quale è possibile collegarsi ad un bot, che deve essere prima accettato all'interno del server e del canale nel quale si vuole fare scraping. Un altro modo è quello di utilizzare la libreria Selenium per automatizzare il browser ed estrarre i dati. In input quindi il codice avrà bisogno, oltre che delle credenziali di accesso al proprio account Discord, di una serie di server e canali target (dei quali ovviamente l'utente deve essere membro). In uscita si ottiene invece che i dati estratti vengono organizzati in oggetti JSON che vengono poi salvati su un apposito database MongoDB.

2 Discord

Discord è una piattaforma di comunicazione sviluppata inizialmente per videogiocatori, ma che oggi viene utilizzata da una vasta gamma di comunità per chattare, effettuare chiamate vocali e videochiamate. Gli utenti possono creare comunità virtuali organizzate in **server**. Ogni server rappresenta uno spazio dedicato a un determinato gruppo, tema o interesse, e può essere personalizzato con ruoli e permessi specifici per gli utenti.

Un server Discord è suddiviso in **canali**, che possono essere di due tipi principali:

- **Canali testuali**: spazi in cui gli utenti comunicano attraverso messaggi scritti. Qui è possibile condividere link, immagini, file e utilizzare bot per automatizzare attività specifiche.
- **Canali vocali**: spazi in cui gli utenti possono parlare in tempo reale attraverso chiamate vocali, con la possibilità di avviare videochiamate o condividere lo schermo.

La struttura a server e canali di Discord è altamente flessibile. Gli amministratori di un server possono creare diversi canali per organizzare meglio le discussioni o le attività del gruppo. Inoltre, essi possono assegnare *ruoli* con permessi specifici agli utenti, facilitando la gestione del server.

Discord è noto anche per l'integrazione con bot personalizzabili, che permettono di automatizzare funzioni come la moderazione delle conversazioni, l'invio di notifiche o l'integrazione con contenuti esterni, come aggiornamenti da social media o videogiochi. Grazie alla sua struttura modulare e alle numerose funzionalità, Discord è diventato una piattaforma versatile per la collaborazione online e per la costruzione di comunità, andando oltre il mondo del gaming.

3 Scraping con Bot

Questo script utilizza un bot di Discord per effettuare lo *scraping* dei messaggi in un canale specifico. Per poterlo eseguire, è necessario che il bot sia stato preventivamente aggiunto al server e al canale di interesse e disponga dei permessi appropriati per accedere ai messaggi. Lo script inizia configurando un client Discord con i permessi necessari per leggere i messaggi, utilizzando `discord.Intents`. Dopo aver stabilito la connessione al server e al canale specificati, il bot accede alla cronologia dei messaggi nel canale, recuperando fino a 100 messaggi recenti.

All'interno dello script è possibile definire una lista di parole chiave (`keywords`), che il bot utilizza per filtrare i messaggi estratti: solo quelli contenenti le parole chiave specificate saranno inclusi nel risultato. Il bot scorre la cronologia del canale e confronta ogni messaggio con le parole chiave, memorizzando solo quelli rilevanti. I messaggi selezionati vengono infine visualizzati nel log, includendo l'autore e il contenuto di ciascun messaggio.

Questo metodo consente di eseguire una raccolta selettiva di informazioni da Discord in base a parole chiave specifiche, sfruttando l'architettura di permessi e canali di Discord. Tuttavia, a causa delle limitazioni sui bot, i dati estratti in questo script non vengono salvati in un database, ma restano disponibili nella sessione di log corrente.
Per una descrizione più approfondita del codice sorgente si rimanda alla documentazione dettagliata.

4 Scraping con Selenium

Per effettuare lo scraping su piattaforme che non offrono un'API di facile accesso o che limitano il prelievo dei dati, come Discord, è possibile utilizzare *Selenium*, una libreria Python che consente di automatizzare le interazioni con un browser web. Con Selenium, è possibile simulare l'accesso manuale, automatizzare l'inserimento delle credenziali, e navigare tra le pagine o sezioni della piattaforma d'interesse. In questo contesto, Selenium utilizza un `webdriver` per aprire un browser reale o simulato e interagire con gli elementi della pagina tramite selettori CSS o identificatori specifici.

Una volta configurato, Selenium può riprodurre azioni come il caricamento dei messaggi scorrendo all'interno di una finestra di conversazione, azione necessaria per visualizzare ed estrarre dati non caricati immediatamente. Attraverso funzioni di attesa, l'estrazione può gestire dinamiche temporali della piattaforma, come i tempi di caricamento dei contenuti.

I dati raccolti vengono quindi trasformati tramite un'analisi HTML con *BeautifulSoup* o un'altra libreria di parsing, che permette di estrarre informazioni strutturate dai contenuti grezzi. Successivamente, queste informazioni possono essere salvate in un database per un'analisi o per una consultazione

successiva. Utilizzare Selenium per lo scraping è vantaggioso per recuperare informazioni in contesti dove gli strumenti convenzionali non sono applicabili, consentendo di replicare in modo programmatico le interazioni dell'utente.
Per una descrizione più approfondita del codice sorgente si rimanda alla documentazione dettagliata.

5 Scraping con Selenium & IA

Un'altra funzionalità implementata nel codice è quella di poter svolgere lo scraping avvalendosi dell'intelligenza artificiale. A tal fine è stato sviluppato un apposito script che segue gli stessi obiettivi descritti finora, ma con un approccio diverso rispetto a quello tradizionale: invece di utilizzare `BeautifulSoup` per analizzare manualmente l'intero HTML estratto, sfrutta il Large Language Model (LLM) Ollama per il parsing e l'estrazione delle informazioni rilevanti dai risultati di ricerca.

Questo consente di ottenere diversi vantaggi:

- **Flessibilità**: Un LLM come Ollama è in grado di comprendere testi complessi e non strutturati, consentendo di estrarre informazioni in modo più intuitivo e con meno codice specifico rispetto al parsing manuale.
- **Adattabilità**: L'uso di un prompt consente di modificare facilmente i criteri di estrazione, adattandosi a diverse esigenze di analisi senza dover riscrivere il codice per la manipolazione del DOM.
- **Riduzione della complessità del codice**: Utilizzando un LLM, il codice per l'analisi e l'estrazione dei dati si riduce significativamente, poiché non è necessario scrivere regole dettagliate per navigare e filtrare l'HTML.

Per una descrizione più approfondita del codice sorgente si rimanda alla documentazione dettagliata.

6 MongoDB

MongoDB è un database NoSQL orientato ai documenti, progettato per gestire grandi quantità di dati non strutturati o semi-strutturati. A differenza dei database relazionali tradizionali (RDBMS), MongoDB non utilizza tabelle e schemi rigidi, ma archivia i dati in documenti JSON-like chiamati *BSON* (Binary JSON), che permettono una maggiore flessibilità nella gestione delle informazioni.
MongoDB è organizzato in:

- **Database**: è l'unità più grande che contiene una collezione di dati correlati.

- **Collezioni**: ogni database può contenere una o più collezioni, l'equivalente delle tabelle in un database relazionale. Tuttavia, una collezione non ha uno schema fisso.

- **Documenti**: all'interno di una collezione, i dati vengono memorizzati sotto forma di documenti BSON. Ogni documento è una struttura JSON-like che contiene campi chiave-valore. Questa flessibilità permette che i documenti all'interno della stessa collezione possano avere campi differenti.

MongoDB supporta le operazioni CRUD (Create, Read, Update, Delete) e permette la memorizzazione di dati complessi, come array o documenti nidificati. È particolarmente adatto per applicazioni che devono gestire dati eterogenei o in rapido cambiamento, come il web scraping o i big data.
Nel codice Python, MongoDB viene utilizzato tramite la libreria `pymongo`, che consente di interagire facilmente con il database.

7 Risultati & Struttura DB

Una volta eseguita la ricerca, effettuato lo scraping e il parsing dei dati estratti questi ultimi vengono salvati su MongoDB. Prima di essere salvati, i risultati vengono organizzati, durante il processo di parsing, in oggetti JSON che poi potranno essere archiviati direttamente.
Il formato dei documenti salvati è il seguente:

```
{
    "author": "string",
    "date": "string",
    "content": "string",
    "channel_name": "string"
}
```

Esempio:

```
{
    "author": "Paolo Rossi",
    "date": "2020-11-17T21:53:29.957Z",
    "content": "Buongiorno",
    "channel_name": "generale"
}
```

Il database nel quale vengono salvati questi documenti è strutturato in varie collection. In particolare è presente una collection per ognuno dei server sui quali viene svolta l'attività di scraping dei dati. Questo, al contrario di quanto viene fatto con il nome del canale, consente di non specificare il nome del server di appartenenza del messaggio.

8 Repository Github

Di seguito il link alla Repository Github.