SAPIENZA
UNIVERSITÀ DI ROMA

From Scraping to Bot Detection: Navigating Mastodon and the Fediverse with APIs

Faculty of Information Engineering, Computer Science and Statistics
Bachelor's Degree in Computer Science

Nicolo Mariano Fragale
ID number 2017378

Advisor
Prof. Angelo Spognardi

Academic Year 2023/2024

From Scraping to Bot Detection: Navigating Mastodon and the Fediverse with APIs

Bachelor's Thesis. Sapienza University of Rome

This thesis has been typeset by LaTeX and the Sapthesis class.

Author's email: ZioNicky@protonmail.com

«Non è il fisico, ma l'allenamento
Non è l'esame, ma l'argomento
Non è la cima, ma il sentiero»

Abstract

Il Fediverso sta emergendo come un'alternativa decentralizzata e non commerciale ai social media tradizionali dando agli utenti più controllo e promuovendo l'impegno della comunità. Costruita su piattaforme open source e interconnessa tramite il protocollo ActivityPub, questa architettura consente trasparenza, autonomia degli utenti e governance etica tra migliaia di istanze indipendenti.

Tuttavia, la crescente presenza di bot automatizzati sta influenzando le conversazioni e la diffusione delle informazioni.

In risposta, questo documento presenta Mastoanalyzer, uno strumento versatile progettato per analizzare il comportamento degli utenti e rilevare l'automazione su Mastodon, una delle piattaforme di spicco del Fediverso. Questo strumento non solo affronta le sfide poste dalla crescente presenza di bot, ma aiuta anche a comprendere il comportamento umano e la dinamica digitale in contesti decentralizzati. Utilizza API REST, database MySQL e formattazione JSON, supportando la raccolta di dati scalabili e l'analisi a lungo termine. Il suo design modulare consente un adattamento alle varie esigenze di ricerca e l'integrazione del machine learning per il rilevamento dei bot in tempo reale.

Ciò che contraddistingue Mastoanalyzer non è solo la sua potenza tecnica, ma anche il suo impegno etico. Si presenta come un alleato per chiunque voglia capire e contribuire all'evoluzione delle reti sociali decentrate, favorendo una maggiore consapevolezza sull'impatto dell'automazione e sulla qualità dell'interazione online. Promuovendo un'analisi trasparente, questo strumento è una risorsa preziosa per comprendere meglio le dinamiche delle comunità digitali, supportando la gestione responsabile dei dati e rispettando l'autonomia di piattaforme federate come Mastodon.

Contents

Chapter 1

Il Fediverso

1.1 Social network decentralizzati

Online social networks (OSN) sono piattaforme digitali che consentono alle persone di connettersi, interagire e condividere contenuti con altre persone o gruppi.

Nel corso degli anni, gli OSN si sono evolute in una varietà di forme con funzioni specifiche a seconda del l'obiettivo e degli interessi degli utenti. Esempi includono:

- **Facebook e X**: Offrono una vasta gamma di condivisione di contenuti, inclusi testi, foto e video, e facilitano la comunicazione tra gli utenti.
- **Instagram, Pinterest e TikTok**: Si concentrano maggiormente sui contenuti visivi.
- **LinkedIn**: Collega professionisti, aziende e aspiranti lavoratori.
- **WhatsApp, Messenger e Telegram**: Forniscono servizi di messaggistica diretta.

Negli ultimi dieci anni, le OSN hanno vissuto una crescita esplosiva, diventando una parte centrale delle nostre vite digitali.

Questo aumento di popolarità ha spinto però i proprietari delle piattaforme a concentrarsi su nuovi obiettivi, in particolare sulla personalizzazione dei contenuti e sulle strategie pubblicitarie, con l'obiettivo di massimizzare il coinvolgimento e promuovere gli sforzi di marketing sociale.

Man mano che le OSN si sono evolute, le strategie di personalizzazione e monetizzazione sono state oggetto di scrutinio, suscitando dibattiti sull'autonomia degli utenti, la privacy dei dati e il controllo aziendale.

Queste questioni hanno alimentato l'interesse per lo sviluppo di nuovi paradigmi OSN, passando da modelli *centrati sull'azienda* a modelli *centrati sull'utente* che enfatizzano la proprietà, la trasparenza e il controllo degli utenti sui propri dati personali.

D'altra parte, le Reti Sociali Online Decentralizzate (DOSN) sono piattaforme sociali che operano in modo decentralizzato, evitando un sistema centralizzato

unico che funge da punto di accesso per tutte le interazioni, diversamente dalle reti sociali tradizionali come Facebook o Instagram. Le DOSN utilizzano una rete distribuita spesso basata su tecnologie come blockchain o peer-to-peer per garantire una maggiore autonomia degli utenti e un maggiore controllo sui dati personali. In questo modo, non esiste un server centrale con pieno controllo sull'uso dei dati, evitando fughe di dati massive o speculazioni sui dati degli utenti.

L'idea della decentralizzazione nei social network ha radici storiche nei movimenti del *Free/Libre and Open Source Software (FLOSS)* dei primi anni 2000, in cui sviluppatori e utenti immaginavano piattaforme libere dalla supervisione aziendale e guidate dagli interessi della comunità.

Le DOSN si basano su due principi fondamentali:

1. L'uso di **software open-source**, che consente agli individui di creare e gestire i propri server, contribuendo a prevenire la centralizzazione.

2. L'implementazione di **protocolli di comunicazione** che permettono un'interazione e uno scambio di dati fluido tra diversi server.

I nodi indipendenti aiutano a mantenere attiva la piattaforma e a mantenere il controllo sui propri dati, che non sono concentrati in un unico luogo vulnerabile a violazioni o abusi. Nelle OSN, la moderazione è centralizzata, portando a censure arbitrarie o restrizioni sulla libertà di espressione. Al contrario, nelle DOSN, la moderazione dipende dal server stesso, che può affidarsi a comunità decentralizzate o algoritmi, rendendo la censura selettiva ma più difficile da applicare per limitare contenuti inappropriati o dannosi. Grazie all'autonomia di ogni server, gli utenti possiedono i propri dati e ne decidono l'utilizzo, riducendo i rischi di violazioni della privacy, hacking massivi e tentativi di blocco o fallimento.

Tuttavia, le DOSN richiedono una certa familiarità con le tecnologie decentralizzate, rendendo queste piattaforme meno accessibili per gli utenti meno esperti. Senza una moderazione centralizzata, può essere difficile gestire contenuti inappropriati o illegali, il che comporta un rischio di spam o attività dannose.

I primi tentativi di creare piattaforme aperte e federate hanno attratto principalmente sviluppatori e sostenitori della privacy, ottenendo solo un interesse di nicchia. Tuttavia, il lancio di **Mastodon** nel 2016 ha segnato un punto di svolta, portando il networking decentralizzato nel mainstream e favorendo l'espansione di ciò che oggi chiamiamo il *Fediverso*. In questo modello federato, i server sono chiamati *istanze* e comunicano tra loro tramite un protocollo condiviso. Questa configurazione consente agli utenti registrati su un server di interagire senza problemi con utenti di altri server, proprio come i servizi di posta elettronica operano tra diversi fornitori. Ogni istanza si concentra tipicamente su una particolare comunità o interesse, che si tratti di pratiche condivise, ideologie o hobby. Questa struttura dimostra che le reti decentralizzate non sono solo tecnicamente fattibili, ma rispondono anche a un reale bisogno di spazi digitali indipendenti dove le comunità possono stabilire le proprie linee guida e norme.

Insieme, queste istanze formano una vasta rete sociale nota come Fediverso.

Grazie a questo sistema interconnesso, gli utenti possono seguire e interagire con persone su diverse piattaforme DOSN senza bisogno di account separati su

ciascuna, massimizzando l'interoperabilità e facilitando il trasferimento dei dati tra piattaforme diverse.

1.2 Uno sguardo al Fediverso

A seguito delle recenti crisi nelle principali piattaforme di social media come Twitter e Reddit, il **Fediverso** ha guadagnato notevole attenzione come una valida alternativa non commerciale. Individui, comunità e organizzazioni si stanno sempre più rivolgendosi al Fediverso per creare canali aperti di comunicazione pubblica, dialogo e dibattito al di fuori dell'ecosistema dei social media commerciali.

Il Fediverso è una rete decentralizzata composta da siti, app e servizi di social media open-source, per lo più senza scopo di lucro, che si interconnettono utilizzando il protocollo ActivityPub. Questo framework supporta un ecosistema diversificato di piattaforme collegate e indipendenti che promuovono un approccio più centrato sull'utente e guidato dalla comunità al networking sociale.

Il Fediverso si è rapidamente espanso fino a includere quasi 5.000 *istanze*, utilizzando diversi software come Friendica, Funkwhale, Lemmy, Hubzilla, Misskey, PeerTube, PixelFed e Pleroma. [1]

La popolarità del Fediverso è stata guidata da una combinazione di caratteristiche tecniche e cambiamenti nelle preferenze degli utenti. Dal punto di vista tecnico, il Fediverso si basa su protocolli aperti e una struttura della piattaforma trasparente, libera dal controllo aziendale. Socialmente, si allinea a una tendenza crescente degli utenti a cercare maggiore controllo e autonomia nelle loro interazioni sui social media, piuttosto che essere soggetti agli algoritmi e alle decisioni delle grandi aziende tecnologiche. Questa combinazione unica di empowerment degli utenti e architettura aperta rende il Fediverso un'alternativa attraente rispetto ai social network tradizionali e centralizzati.

Il Fediverso si distingue dai tradizionali social network online (OSN) grazie alla forte enfasi su **autonomia degli utenti** e **governance guidata dalla comunità**. Questo approccio consente agli utenti non solo di partecipare, ma anche di plasmare i loro spazi sociali in linea con i propri valori e le norme della comunità. Di conseguenza, il Fediverso non è solo una rete tecnica; incarna un movimento sociale dedicato all'autonomia digitale, alla governance etica e alla libertà di espressione. [2] La sua struttura flessibile consente a individui e comunità di costruire e gestire istanze indipendenti, creando spazi diversificati che supportano una gamma di interessi e ideologie promuovendo al contempo il controllo degli utenti e la trasparenza.

Un aspetto chiave del funzionamento del Fediverso risiede nella sua base tecnica, che consente l'**interoperabilità** tra diverse piattaforme. Molti servizi del Fediverso,

[1] È difficile misurare la dimensione del Fediverso poiché non è centralizzato. Diversi progetti lo monitorano: https://fe-didb.org/, https://fediverse.party/en/fediverse/, https://the-federation.info/ e https://mastodon.social/@mastodon-usercount. Ognuno varia, poiché osserva insiemi diversi di server. Tutti riflettono una crescita esplosiva degli account durante il periodo 2022-23. Da allora, gli utenti sono diminuiti ma rimangono ben al di sopra dei livelli storici. Nell'ottobre 2021, il Fediverso contava circa 4,1 milioni di account [1]. Oggi sono circa 14 milioni.

[2] Art. 21, Costituzione Italiana: Tutti hanno diritto di manifestare liberamente il proprio pensiero con la parola, lo scritto e ogni altro mezzo di diffusione.

come Mastodon e PixelFed, operano utilizzando il **protocollo ActivityPub**, uno standard progettato per facilitare la comunicazione tra piattaforme di social media diverse.

Il Fediverso è anche una struttura sociale e politica in cui utenti e amministratori prendono decisioni consapevoli su con chi interagire o chi bloccare. Questo approccio consente alle comunità di **auto-regolarsi** in base ai propri standard e valori. Ad esempio, quando Gab.com, una piattaforma basata su ActivityPub nota per ospitare contenuti dell'alt-right, si è unita alla rete, molte comunità del Fediverso hanno scelto di bloccarla, riflettendo un impegno verso una moderazione umana e centrata sulla comunità, dando priorità ai valori comunitari e alla sicurezza degli utenti [2].

Gab.com è una piattaforma di social network lanciata nel 2016 con l'obiettivo dichiarato di promuovere la libertà di espressione, ma è stata controversa sin dall'inizio per i contenuti della sua comunità, che includevano discussioni estremiste, discorsi di odio e movimenti legati alla destra radicale. Nel 2019, ha deciso di adottare il software Mastodon, evitando la censura centralizzata e rimanendo online in modo autonomo. Questo gli ha permesso di mantenere le sue caratteristiche di contenuti non moderati e di attrarre molti utenti esclusi da altri social network a causa di discorsi estremisti o di odio. Tuttavia, la maggior parte delle istanze del Fediverso ha scelto di "de-federarsi" da Gab, impedendo l'interazione con utenti di altre istanze e quelli di Gab. Questa decisione è stata motivata dal desiderio di prevenire la diffusione di contenuti estremisti o di odio nella rete decentralizzata, mantenendo al contempo il principio della decentralizzazione. Questo caso ha evidenziato la difficoltà di bilanciare la libertà di espressione con la necessità di proteggere le piattaforme sociali da contenuti dannosi o estremisti e ha illustrato come ogni istanza sia libera di decidere con quali altre istanze federarsi o defederarsi.

Grazie a una serie di scelte sociali e pratiche fatte dai primi sviluppatori e amministratori di Mastodon, migliaia di istanze del Fediverso oggi utilizzano codici molto simili, vietando azioni come discorsi di odio e molestie e incoraggiando una moderazione locale, umana e decisioni collettive.

Oltre alle considerazioni sociali ed etiche, il Fediverso presenta anche unici **vantaggi tecnici** che attirano una gamma diversificata di utenti, inclusi individui attenti alla privacy, organizzazioni e giornalisti. Diversamente dagli OSN tradizionali, che spesso danno meno priorità ai link esterni o limitano la visibilità dei contenuti che conducono gli utenti fuori dalla piattaforma, l'interoperabilità del Fediverso supporta un'interazione fluida tra piattaforme. Questa apertura consente a giornalisti, ricercatori e creatori di contenuti di raggiungere e connettersi direttamente con il proprio pubblico senza essere limitati da algoritmi che potrebbero oscurare o penalizzare il loro lavoro. Di conseguenza, il Fediverso favorisce un ambiente trasparente e accessibile in cui le informazioni possono circolare liberamente e gli utenti possono interagire tra diverse reti, migliorando sia la visibilità che il coinvolgimento dei contenuti condivisi.

Tuttavia, non vi sono garanzie che questa rete federale, decentralizzata di server sociali locali e basata su un protocollo aperto, rimanga sostenibile, democraticamente controllata o etica. Il Fediverso presenta tre modalità chiave di governance, ciascuna con pratiche e necessità uniche:

- Il protocollo standard W3C ActivityPub, che ne mantiene ed estende lo standard.
- La manutenzione e lo sviluppo dei pacchetti software del Fediverso, come Mastodon, Pixelfed e Lemmy.
- La governance di ciascuna specifica istanza del Fediverso, che coinvolge la moderazione interna dei contenuti e le relazioni diplomatiche tra le istanze.

Poiché ogni istanza è libera di stabilire le proprie politiche e regole di moderazione, questo ultimo punto è un aspetto cruciale per garantire la funzionalità e la sicurezza. Ogni istanza è gestita da uno o più amministratori e moderatori che stabiliscono e applicano regole di moderazione bloccando, silenziando o avvisando chi le viola. Queste regole sono riportate nella policy dell'istanza, che definisce chiaramente quali contenuti sono consentiti e quali no. Alcune istanze sono molto permissive, mentre altre hanno regole severe contro discorsi di odio, pornografia o contenuti politici estremisti. Pertanto, quando gli utenti scelgono di unirsi a istanze che riflettono le loro opinioni e preferenze, sono moralmente obbligati ad accettare e rispettare questo insieme di regole.

Nonostante le sue potenzialità, il Fediverso deve affrontare diverse sfide significative. Come ecosistema decentralizzato, si basa su una sostanziale **collaborazione** tra le istanze, in particolare per quanto riguarda la **moderazione dei contenuti**. Poiché ogni istanza può stabilire le proprie linee guida e regole, mantenere una moderazione coerente attraverso la rete può risultare complesso e richiedere molte risorse. Un'altra sfida chiave è la **scoperta dei contenuti**. Senza algoritmi progettati per promuovere i contenuti, come accade negli OSN tradizionali, gli utenti potrebbero trovare difficile accedere a una vasta gamma di post, tendenze o comunità, limitando la portata e la visibilità del Fediverso per voci nuove o diverse. Questa assenza di promozione algoritmica rappresenta sia un vantaggio, in termini di autonomia degli utenti, sia una limitazione, in termini di scoperta dei contenuti.

1.2.1 Privacy e Sorveglianza

Il Fediverso si è evoluto, passando dal trattare la privacy e la sorveglianza come sfide puramente tecniche—focalizzate sulla crittografia e la messaggistica sicura—a considerarle anche questioni sociali. Nei primi anni, le comunità tecnologiche si concentravano principalmente sulla protezione della privacy attraverso misure di sicurezza robuste mirate a prevenire la sorveglianza da parte di governi o aziende. Tuttavia, con l'ascesa del Fediverso, il concetto di privacy si è ampliato. Ora include la gestione dei confini personali tra gli utenti, il controllo su chi può vedere i propri post e l'uso di strumenti come avvisi sui contenuti e moderazione per modellare le interazioni. Questo cambiamento è in parte guidato dalle esigenze di comunità marginalizzate, in particolare utenti queer, che affrontano preoccupazioni sulla privacy che vanno oltre la supervisione governativa o aziendale. Ad esempio, questi utenti spesso devono navigare relazioni personali o esplorare le proprie identità in modi che garantiscano la loro sicurezza. Di conseguenza, mentre il Fediverso continua a integrare misure tecniche per la privacy, enfatizza sempre più la creazione di uno spazio in cui le comunità possano stabilire le proprie regole e sentirsi sicure. Ad

esempio, molte istanze raccolgono solo le informazioni necessarie per il funzionamento della piattaforma, come i dati per la registrazione del profilo (nome utente, email). Altre dichiarano esplicitamente di non condividere i dati degli utenti con terze parti per scopi di marketing, pubblicità o analisi. Gli utenti hanno anche il diritto di richiedere la cancellazione dei propri dati in qualsiasi momento, inclusi i contenuti pubblicati, e la possibilità di eliminare definitivamente il proprio account e i dati associati. Alcune istanze adottano una politica di registri limitati, in cui i dati vengono eliminati dopo un certo periodo. Tuttavia, la cancellazione di contenuti federati su altre istanze non può sempre essere garantita.

Tenendo presente tutto ciò, il Fediverso offre un'alternativa alle piattaforme di social media aziendali, che spesso operano secondo il modello del *capitalismo della sorveglianza*, raccogliendo e vendendo i dati e le interazioni degli utenti. Sebbene queste piattaforme possano dare agli utenti l'illusione del controllo attraverso impostazioni sulla privacy, continuano a raccogliere dati in modi difficili da regolamentare. Al contrario, il Fediverso cerca di rompere questo modello, dando priorità alla trasparenza nella gestione dei dati e assegnando agli utenti un ruolo più attivo nel processo. Quando gli utenti si uniscono a un'*istanza*, sono generalmente informati su come funziona la federazione e su come i dati fluiscono attraverso la rete—un aspetto raro negli ambienti dei social media aziendali.

Tuttavia, il Fediverso non è privo di rischi per la privacy. Essendo costruito su una rete aperta e pubblica, può essere più facile per terze parti raccogliere dati da esso. Ciò potrebbe rappresentare una minaccia per gruppi come attivisti o persone esposte politicamente, che ripongono fiducia negli amministratori dei server per garantire rispetto e privacy. Tuttavia, ciò potrebbe non essere sufficiente per questioni più sensibili, come la privacy dei messaggi diretti, che potrebbe beneficiare di una crittografia più forte per ridurre i rischi legati alla visibilità dei dati.

In sostanza, il Fediverso sta sperimentando un approccio che considera la privacy una questione sociale e, mentre questa prospettiva è ancora in evoluzione, la conversazione in corso cerca di trovare un equilibrio tra soluzioni sociali e salvaguardie tecniche.

1.2.2 Sostenibilità e Finanziamento

Il Fediverso sfida l'idea comune che i social media siano "gratuiti". Diversamente dalle grandi piattaforme aziendali, che traggono profitto dai dati e dalle interazioni degli utenti senza compensare gli stessi, mantenere i server e sviluppare software nel Fediverso comporta costi reali. Sebbene alcuni progetti all'interno del Fediverso siano finanziati tramite donazioni e contributi volontari, questo modello di finanziamento rimane fragile. Progetti più grandi, come Mastodon, hanno iniziato a remunerare alcuni collaboratori, ma è necessario un supporto pubblico più consistente e stabile per garantire la sostenibilità a lungo termine delle piattaforme non commerciali.

Alla sua base, il Fediverso è una raccolta di progetti di social media alternativi che, nonostante le differenze, condividono valori comuni legati alla libertà degli utenti, all'autogestione e alla privacy. Diversamente dalle piattaforme aziendali, dove i dati degli utenti vengono monetizzati e gli utenti hanno un controllo limitato su come vengono utilizzate le loro informazioni, il Fediverso enfatizza la partecipazione

attiva degli utenti. Essi sono incoraggiati a contribuire sia allo sviluppo del codice sia alla governance delle proprie comunità.

Le piattaforme all'interno del Fediverso supportano un'ampia gamma di modelli di utilizzo, dalle protezioni per la privacy per le comunità queer alle iniziative di sovversione politica, tutti adattabili alle esigenze uniche di ciascun gruppo. Questi modelli traggono le loro origini dalle prime comunità di software libero, in cui i primi utenti erano anche sviluppatori che modellavano attivamente il software. Tuttavia, con l'ascesa dell'industria tecnologica, il ruolo dell'utente è cambiato. Il Web 2.0 ha introdotto l'interattività ma ha anche dato origine al capitalismo della sorveglianza, in cui gli utenti sono diventati partecipanti passivi con un controllo limitato sui propri dati. Al contrario, il Fediverso mira a responsabilizzare gli utenti, permettendo loro non solo di partecipare, ma anche di contribuire e, in alcuni casi, gestire interi nodi della rete.

In sintesi, il Fediverso rappresenta uno spazio sperimentale per esplorare nuovi modelli di coinvolgimento degli utenti, promuovendo un approccio che bilancia partecipazione attiva e compensazione equa. Offre un'alternativa sia ai modelli di business sfruttativi delle piattaforme basate sui dati sia ai modelli non remunerati, a volte oppressivi, del lavoro gratuito ma intensivo. Questa evoluzione sta influenzando sia il panorama dei social media alternativi sia la cultura del software libero/open source (FLOSS), gettando le basi per un nuovo modo di pensare alla governance e al coinvolgimento degli utenti.

1.3 Mastodon

Mastodon è stato creato nel 2016 da Eugen Rochko, un programmatore tedesco, con l'obiettivo di fornire un'alternativa decentralizzata alle grandi piattaforme di social media esistenti, come Twitter e Facebook. È stato progettato principalmente per affrontare i problemi associati ai social media centralizzati, come il controllo sui dati e la moderazione dei contenuti, la censura da parte di entità centrali e la libertà di espressione, sfruttando la possibilità di creare organismi indipendenti.

Mastodon è una piattaforma di social media decentralizzata, open-source e federata che consente agli utenti di interagire, pubblicare contenuti e creare una rete sociale senza dipendere da un'unica entità centrale. Simile a Twitter, permette agli utenti di pubblicare brevi messaggi (fino a 500 caratteri) chiamati *toot*. La funzionalità analoga al retweet è denominata *boost*. Gli utenti possono anche condividere immagini, video e link, e interagire con altri utenti attraverso commenti, mi piace e condivisioni. Inoltre, Mastodon dà particolare rilievo alle *comunità di nicchia* e a una *moderazione dei contenuti* robusta, simile alle sub-community di Reddit, ma con ciascuna istanza che mantiene autonomia sulle proprie politiche e strategie di moderazione.

Gli utenti possono anche contrassegnare alcuni toot con un *content warning*, accompagnati da un testo *spoiler* che riassume il contenuto offuscato, permettendo ai lettori di decidere se accedere o meno a contenuti potenzialmente disturbanti.

Mastodon è una rete di server indipendenti noti come *istanze*, che possono variare notevolmente per dimensione, da grandi reti con milioni di account a piccole comunità,

persino dedicate a una singola organizzazione. Queste istanze possono federarsi o collegarsi, permettendo agli utenti di interagire tramite esse senza essere vincolati a un singolo server, creando così una rete globale e distribuita. Gli amministratori delle istanze possono, facoltativamente, chiudere le registrazioni alla loro istanza, creando una comunità privata o migliorando l'efficienza della moderazione. Hanno il controllo sulle politiche dei contenuti, stabilendo regole specifiche su ciò che è consentito o proibito. È importante sottolineare che le registrazioni chiuse non limitano le interazioni degli utenti, che rimangono possibili in tutta la rete grazie all'interoperabilità garantita dal protocollo ActivityPub.

Questo social gode di tutti i privilegi descritti per i DOSN nella sezione precedente: privacy e controllo dei dati, assenza di pubblicità, flessibilità nella moderazione e comunità personalizzabili. Tuttavia, presenta anche dei limiti: una curva di apprendimento ripida, moderazione complessa e incompatibilità tra le istanze.

Una caratteristica distintiva di Mastodon è l'uso del protocollo *ActivityPub*[3], che supporta comunicazioni sia client-to-server sia server-to-server. Attraverso questo protocollo e un *meccanismo di sottoscrizione*, gli utenti possono interagire facilmente, anche tra istanze diverse. Il sistema di follower unico di Mastodon struttura le sue timeline come segue:

- **Home Timeline**: Mostra i toot degli utenti seguiti.
- **Local Timeline**: Mostra i toot creati all'interno dell'istanza dell'utente.
- **Federated Timeline**: Contiene i toot pubblici di tutti gli utenti conosciuti dall'istanza, indipendentemente dall'origine dell'istanza.

La ricerca sui recenti DOSN open-source indica che Mastodon ha avuto un impatto unico all'interno del *Fediverso*, la raccolta di piattaforme social federate. In particolare, studi come [3, 4, 5, 6, 7] hanno esaminato vari aspetti di Mastodon. In particolare, [7] ha condotto un'analisi qualitativa tramite interviste con moderatori di istanze, evidenziando come la struttura federativa di Mastodon favorisca contenuti diversificati, autonomia comunitaria e una crescita orizzontale tra le istanze piuttosto che una crescita verticale al loro interno.

Da una prospettiva di *network science*, [5, 8] hanno condotto analisi fondamentali della rete di utenti di Mastodon, esplorando caratteristiche come la *distribuzione dei gradi*, la *chiusura triadica* e l'*assortatività*, confrontando questi metriche con quelle di Twitter. Le loro analisi hanno rilevato che Mastodon mostra una relazione più bilanciata tra follower e followee, con il 95% degli utenti che presenta una differenza tra follower e followee compresa nell'intervallo $(-250, 250)$. Inoltre, è stata osservata una minore presenza di bot social su Mastodon (circa il 5%) rispetto a Twitter (15%) come riportato in [9].

In termini di coefficiente di clustering, i valori di Mastodon si collocano tra quelli osservati su Facebook e Twitter. Inoltre, l'analisi dell'*assortatività dei gradi*—considerando metriche come Source In-Degree (SID), Source Out-Degree (SOD), Destination In-Degree (DID) e Destination Out-Degree (DOD)—ha rivelato schemi unici. La mancanza di correlazione tra (SOD, DOD), (SOD, DID) e (SID,

[3] `https://www.w3.org/TR/activitypub/`

DOD) suggerisce che gli utenti con alti numeri di follower si connettono a utenti con popolarità variabile, sia per chi seguono sia per chi li segue. In particolare, la correlazione negativa (-0.1) tra SID e DID indica che gli utenti popolari tendono a seguire account meno popolari.

Inoltre, [8] ha esplorato l'impatto della decentralizzazione sulle relazioni tra gli utenti, osservando che ogni istanza presenta caratteristiche strutturali uniche (es. distribuzione dei gradi e coefficiente di clustering) che influenzano le interazioni degli utenti all'interno dell'istanza.

1.4 Protocollo ActivityPub

Il **protocollo ActivityPub** è un protocollo decentralizzato per social network basato sul formato dati *ActivityStreams 2.0*. Esso offre sia una **API client-to-server**, che consente la creazione, modifica e cancellazione di contenuti, sia una **API server-to-server**, che facilita la consegna di notifiche e contenuti tra server diversi. Questo design permette l'interoperabilità tra servizi su server differenti, simile ai protocolli di posta elettronica, dove gli utenti possono comunicare senza difficoltà tra diversi provider.

In ActivityPub, gli utenti su un server possono interagire con utenti su altri server della rete. Ad esempio, se *utente1* si trova su *serverA* che utilizza Mastodon (una piattaforma di microblogging) e *utente2* è su *serverB* che utilizza Pixelfed (una piattaforma di condivisione di foto), entrambi possono seguirsi reciprocamente per visualizzare, mettere mi piace e commentare i post. Questa interazione è analoga al funzionamento dei protocolli email open-source, creando un'infrastruttura tecnica non centralizzata, distinta dal tradizionale modello di social media centralizzati [10].

1.4.1 Struttura del Protocollo

Il protocollo ActivityPub opera su due livelli distinti:

- **Protocollo di Federazione Server-to-Server**: Questo protocollo consente ai siti web decentralizzati di scambiarsi informazioni, facilitando la distribuzione delle attività tra utenti su server differenti. Questa interazione aiuta a creare un grafo sociale unificato e interconnesso, permettendo agli utenti di comunicare e condividere contenuti senza problemi tra istanze diverse.
- **Protocollo Client-to-Server**: Questo protocollo consente a un client, come un'applicazione mobile o un'interfaccia web, di agire per conto dell'utente. Offre la funzionalità di interagire con il feed sociale dell'utente e di svolgere attività come pubblicare, mettere mi piace, seguire e altro, assicurando che gli utenti possano partecipare pienamente al social network indipendentemente dalla piattaforma utilizzata.

1.4.2 Attori e Meccanismi di Messaggistica

In ActivityPub, un utente è rappresentato da un **attore**, un account su un server che interagisce all'interno della rete. Ogni attore ha i seguenti componenti:

- **Inbox**: Riceve messaggi da altri attori.
- **Outbox**: Invia messaggi agli altri [11].

Ad esempio, se *Alyssa* vuole comunicare con i suoi amici, i meccanismi di *inbox* e *outbox* facilitano queste interazioni:

- **POST a un'inbox**: Invia un messaggio alla inbox di un altro attore (server-to-server, federato).
- **GET da un'inbox**: Permette a un utente di recuperare i propri messaggi più recenti (client-to-server).
- **POST a un'outbox**: Invia messaggi alla rete più ampia (client-to-server).
- **GET da un'outbox**: Consente di visualizzare i messaggi pubblicati da un attore (client-to-server o server-to-server) [11].

Per garantire efficienza, i server distribuiscono tipicamente i messaggi alle inbox degli attori su altri server, creando un meccanismo di federazione robusto [11].

1.4.3 Oggetti e Attività

In ActivityPub, gli elementi centrali includono gli **oggetti**, che rappresentano vari tipi di contenuti o azioni (ad esempio, post, commenti o mi piace). Questi oggetti possono essere "incapsulati" in **attività** e sono spesso organizzati in **collezioni** (o flussi di oggetti), che a loro volta sono sottoclassi di oggetti. Per mantenere la sicurezza, i server che ricevono contenuti devono validare questi oggetti per prevenire attacchi di spoofing, verificando le firme digitali e assicurandosi che l'oggetto provenga dal server dichiarato.

Ogni oggetto è assegnato a un **identificativo unico** (tipicamente un URI HTTPS), che lo rende accessibile in tutta la rete. Ad esempio:

- **Oggetti Pubblici**: Recuperabili tramite una richiesta HTTP GET sul loro URI unico.
- **Oggetti Privati**: Possono restituire codici di errore (es. 403 o 404) per limitare l'accesso.

Se un oggetto viene creato tramite comunicazioni server-to-server, l'ID è obbligatorio; nelle interazioni client-to-server, il server assegna un ID se non ne è specificato uno.

1.4.4 Tipi di Attori e Flessibilità

ActivityPub supporta diversi tipi di attori definiti in ActivityStreams:

- **Person**: Rappresenta un utente umano, spesso associato a un profilo personale.
- **Group**: Rappresenta un collettivo di utenti.

- **Organization**: Rappresenta un'entità organizzativa.
- **Service**: Rappresenta un servizio o una piattaforma automatizzata.

Inoltre, la flessibilità di ActivityPub consente agli attori di essere tipi di oggetti personalizzati, come profili o tipi estesi di ActivityStreams, permettendo interazioni complesse all'interno della rete:

- **Profili Multipli**: Un singolo utente può avere più attori (ad esempio, account separati per uso personale e professionale).
- **Bot e Processi Automatizzati**: Gli attori possono rappresentare bot che interagiscono automaticamente.

Questa flessibilità consente ad ActivityPub di supportare non solo utenti individuali, ma anche sistemi più complessi come comunità, organizzazioni e software automatizzati. Attraverso ActivityPub, gli attori possono svolgere azioni come pubblicare, mettere mi piace, seguire e commentare, e comunicare sia all'interno sia tra server, rendendo possibili interazioni decentralizzate [11].

Chapter 2

Social Bots Detection

Ci sono prove crescenti che sempre più contenuti sui social media vengano creati da entità automatizzate conosciute come **social bot** o **account Sybil**. Studi recenti hanno evidenziato casi in cui questi social bot imitano il comportamento umano per influenzare le conversazioni, manipolare la popolarità degli utenti, inondare le piattaforme di contenuti irrilevanti, diffondere disinformazione e persino impegnarsi in propaganda e reclutamento per attività terroristiche. I social bot (abbreviazione di *software robots*) o software automatizzati, operano tramite algoritmi che vanno da semplici script a intelligenze artificiali complesse, impegnandosi in attività o conversazioni simili a quelle umane online [12]. Un esempio significativo di bot è il **chatbot**, un algoritmo progettato specificamente per interagire in conversazioni con un essere umano, un'idea originariamente proposta da *Alan Turing* negli anni '50 [13]. L'obiettivo di creare un algoritmo informatico capace di superare il *test di Turing* ha alimentato la ricerca sull'intelligenza artificiale per decenni. Questa ambizione è evidente in iniziative come il **Premio Loebner**, che premia i progressi nell'elaborazione del linguaggio naturale [14]. I social bot benigni aiutano ad automatizzare la condivisione di contenuti, l'aggregazione di notizie, il servizio clienti e l'assistenza, aiutando così le aziende a gestire la loro presenza sui social media e a migliorare l'engagement degli utenti [15, 16, 17]. Il livello di sofisticazione dei bot varia ampiamente. Alcuni social bot sono relativamente semplici, limitandosi ad aggregare informazioni da fonti di notizie, aggiornamenti meteo o post di blog e ripubblicare questo contenuto sui social network. Altri, tuttavia, possono essere altamente sofisticati, capaci di infiltrarsi nelle conversazioni umane. Queste capacità hanno sia vantaggi che svantaggi per gli utenti delle *Reti Sociali Online (OSN)* e possono essere impiegate per scopi positivi o negativi. Dal lato positivo, i bot possono essere progettati con buone intenzioni. Ad esempio, possono essere utilizzati per proteggere l'anonimato degli utenti o per automatizzare e svolgere compiti molto più velocemente di quanto potrebbero fare gli esseri umani manualmente. Tuttavia, una significativa capacità malevola dei social bot è il loro potenziale di diffondere rapidamente disinformazione, manipolare il sentiment pubblico, promuovere teorie del complotto online ed esporre informazioni private, come numeri di telefono e indirizzi. Questo può portare a un'erosione della fiducia nelle piattaforme di social media. Un'altra preoccupazione è **l'astroturfing**, in cui un gruppo aziendale organizza una campagna di supporto popolare apparente. Si tratta di una protesta,

ma in realtà è tutta orchestrata da una parte per promuovere il proprio obiettivo sfruttando la percezione pubblica per far sembrare che ci sia un ampio consenso su una determinata causa, quando in realtà tale consenso è artificiale e creato ad hoc. La domanda principale è: come possiamo rilevare efficacemente attività dannose nelle Reti Sociali Online (OSN)? In generale, le tecniche di rilevamento possono essere classificate in tre categorie:

1. **Sistemi di rilevamento dei bot basati sulla topologia della rete sociale**: Questi sistemi analizzano le informazioni basate sulla struttura della rete.
2. **Sistemi di Crowdsourcing**: Questi sistemi utilizzano il crowdsourcing per analizzare i post e i profili degli utenti.
3. **Sistemi basati sulle caratteristiche**: Questi sistemi sfruttano i metodi di apprendimento automatico.

La sofisticazione dei social bot è progredita da interazioni basate su regole e parole chiave a modelli che incorporano l'apprendimento automatico e l'elaborazione del linguaggio naturale, permettendo scambi più realistici simili a quelli umani [18, 19]. Ad esempio, alcuni bot utilizzano modelli **basati su catene di Markov** [1] per generare testo, imitano contenuti prodotti dagli esseri umani [20, 21].

2.1 Evoluzione delle Tecniche di Rilevamento dei Social Bot

Per rilevare efficacemente i social bot, è necessario un approccio completo che tenga conto della vasta gamma di tattiche e comportamenti che questi bot impiegano. Poiché i bot possono essere progettati per imitare le interazioni umane, diffondere disinformazione o impersonare utenti reali, è essenziale utilizzare una varietà di metodi di rilevamento. Combinando diverse tecniche, possiamo migliorare l'accuratezza e l'affidabilità, rendendo più facile individuare i bot che altrimenti potrebbero passare inosservati. In questa sezione, esamineremo più da vicino diversi metodi di rilevamento e come la loro integrazione possa portare a risultati migliori, facendo anche riferimento alle ricerche chiave che hanno plasmato questi approcci.

Modello Basato su Euristiche

Nelle fasi iniziali del rilevamento dei social bot, i ricercatori si affidavano principalmente a euristiche semplici e a sistemi basati su regole per identificare i bot. Queste tecniche si concentravano sull'analisi di modelli e caratteristiche facilmente osservabili, indicative di comportamenti da bot. Le euristiche comunemente utilizzate includevano:

[1] Questo principio è conosciuto come la proprietà di Markov o memoria zero. Un modello a catena di Markov è un tipo di modello matematico utilizzato per descrivere sistemi che evolvono nel tempo, in cui il prossimo stato del sistema dipende solo dallo stato attuale, non da come si è arrivati a quello stato.

1. **Attività dell'account:** I bot spesso mostrano livelli di messaggistica ad alta frequenza e attività complessive più elevati rispetto agli utenti umani. Analizzando metriche come il numero di post, i ricercatori potevano identificare modelli tipici dei bot.

2. **Metadati dell'account:** Alcuni metadati del profilo, come la data di creazione dell'account, il numero di follower e il rapporto tra il numero di persone seguite e i follower, possono anche segnalare un comportamento da bot.

3. **Caratteristiche basate sul contenuto:** I bot spesso generano contenuti che sono ripetitivi, contengono parole chiave specifiche o derivano da un set limitato di fonti. Analizzando la diversità dei contenuti, la presenza di parole chiave e la distribuzione delle fonti, i ricercatori potevano identificare potenziali bot [22, 23].

4. **Caratteristiche basate sulla rete:** I bot mostrano spesso modelli di rete distintivi, come la formazione di gruppi strettamente connessi o la presenza di poche relazioni reciproche. Analizzando la struttura delle reti di follower e amici, i ricercatori potevano rilevare account sospetti di bot [24].

Sebbene questi primi metodi di rilevamento abbiano gettato le basi per l'identificazione dei social bot, faticavano a tenere il passo con la crescente sofisticazione del comportamento dei bot. I bot più avanzati potevano facilmente eludere questi metodi basati su euristiche, imitare il comportamento umano, modificare i loro modelli di attività o generare contenuti diversificati [19]. Inoltre, la dipendenza da regole e caratteristiche create manualmente rendeva questi metodi soggetti a falsi positivi e falsi negativi. Gli utenti legittimi potevano manifestare comportamenti simili a quelli dei bot, mentre i bot potevano imitare azioni umane. Questa limitazione ha portato allo sviluppo di tecniche più avanzate che integrano **apprendimento automatico** e **elaborazione del linguaggio naturale** per migliorare l'accuratezza e l'adattabilità del rilevamento [25, 12]. Oggi, molti social bot sono in grado di generare automaticamente risposte tramite algoritmi avanzati di *elaborazione del linguaggio naturale*. Questi bot possono creare risposte contestualmente rilevanti e spesso includono riferimenti a contenuti multimediali, come immagini o video, nonché link a risorse esterne. Ciò li rende altamente efficaci nell'integrarsi nelle discussioni online, rendendo difficile per gli utenti distinguere tra interazioni automatizzate e umane. Oltre a queste capacità comunicative, altri bot si concentrano su attività più malevoli, come alterare l'identità di individui legittimi. Alcuni di questi bot agiscono come **ladri di identità**, adottando leggere varianti di nomi utente reali per ingannare gli altri facendogli credere che siano utenti legittimi. Questi bot possono quindi rubare informazioni personali, comprese foto, link e persino messaggi privati, compromettendo ulteriormente la sicurezza degli individui sulle piattaforme sociali. Inoltre, i bot più sofisticati vengono sviluppati con la capacità di *clonare* il comportamento degli utenti legittimi. Questi bot possono interagire con gli amici di un utente, pubblicare contenuti pertinenti e contestualmente coerenti, e replicare i modelli temporali degli utenti reali. In questo modo, creano l'illusione di autenticità, rendendo ancora più difficile per gli utenti e i sistemi automatizzati rilevarli. In alcuni casi, questi bot possono stabilire una presenza sui social media che è quasi

indistinguibile da quella di una persona reale, contribuendo alla crescente complessità del rilevamento dei bot e alle sfide relative alla sicurezza informatica nelle reti sociali.

2.1.1 Basato su Approcci Multimodali

Gli approcci multimodali, che combinano diversi tipi di dati come **testo**, **immagini** e **caratteristiche di rete**, offrono un modo più completo e affidabile per comprendere il comportamento degli utenti, rendendo più facile il rilevamento dei social bot. Utilizzando una varietà di fonti di informazioni, questi metodi possono individuare modelli complessi e sottili nei dati che potrebbero sfuggire agli approcci che si basano su un solo tipo di dato. Ad esempio, l'analisi del **testo** può rivelare modelli linguistici tipici dei bot, mentre l'analisi delle **immagini** potrebbe aiutare a individuare foto false o riutilizzate, e lo studio delle **connessioni di rete** può evidenziare modelli di relazione insoliti, come connessioni unilaterali o rapidi aumenti dei follower. Recentemente, c'è stato un crescente interesse nell'utilizzare dati multimodali per il rilevamento dei social bot. Questo approccio consente di ottenere una visione più completa del comportamento degli utenti su diversi aspetti, rendendo più facile identificare i bot che si mimetizzano imitandone le attività umane. Combinando dati provenienti da più fonti, si possono scoprire comportamenti da bot che sarebbero più difficili da individuare con un solo metodo. Poiché i bot continuano a evolversi e a diventare sempre più sofisticati, le tecniche multimodali sono considerate uno strumento chiave per migliorare l'accuratezza del rilevamento e affrontare queste sfide. Una direzione promettente è l'integrazione di **testo** e **analisi delle immagini**. Ad esempio, [26] ha proposto un approccio multimodale per il rilevamento delle notizie false, modellando congiuntamente informazioni testuali e visive. Il loro approccio utilizzava una rete di attenzione gerarchica per catturare le dipendenze tra le caratteristiche testuali e visive, consentendo al modello di identificare i bot che diffondono informazioni fuorvianti sia tramite testo che contenuti visivi. Similmente, [27] ha proposto un modello che combinava segnali testuali e visivi per rilevare i social bot su Twitter, dimostrando l'efficacia degli approcci multimodali nel catturare informazioni complementari. Un'altra direzione per gli approcci multimodali è l'incorporazione delle **caratteristiche di rete**. Il rilevamento dei social bot può trarre beneficio dall'analisi delle interazioni degli utenti e delle strutture delle reti, che possono rivelare modelli di interazione sospetti o strutture di rete indicative di attività di bot coordinate [28, 29, 30]. [30] ha dimostrato il valore delle caratteristiche temporali proponendo un metodo che le combinava con le caratteristiche basate sul contenuto per rilevare campagne di influenza coordinate sulle piattaforme sociali. Gli approcci multimodali possono anche essere estesi ad altri tipi di dati, come **audio** o **video**, per migliorare ulteriormente le capacità di rilevamento dei social bot. Ad esempio, l'analisi dell'audio o del video può essere utilizzata per identificare bot che generano o distribuiscono contenuti *deepfake*, una minaccia significativa per le piattaforme online [31]. Combinando diversi tipi di dati, come **testo**, **immagini** e **caratteristiche di rete**, i metodi di rilevamento dei social bot diventano più accurati e robusti, aiutando a stare al passo con la natura in continua evoluzione del comportamento dei bot. Poiché i social bot stanno diventando sempre più sofisticati, è importante che i metodi di rilevamento siano flessibili e in grado di adattarsi a questi cambiamenti. Gli approcci multimodali, che raccolgono una varietà di fonti di

dati, sono particolarmente utili perché consentono ai ricercatori di vedere un quadro più ampio dell'attività degli utenti. In questo modo, è possibile individuare modelli più sottili e complessi che potrebbero sfuggire a semplici approcci basati su un singolo tipo di dato. In poche parole, i metodi multimodali hanno un grande potenziale per migliorare il rilevamento dei social bot. Esaminando diversi livelli di dati, i ricercatori possono identificare indizi che altrimenti potrebbero essere trascurati. Che si tratti del **testo** scritto dalle persone, delle **immagini** che condividono o delle loro interazioni con altri nella loro rete, ogni pezzo di informazione aggiunge un ulteriore strato di comprensione. Quando tutti questi pezzi vengono combinati, creano un quadro più completo e accurato dell'attività online, il che aiuta a identificare i bot in modo affidabile. Questo è cruciale per prevenire gli effetti negativi che i bot possono avere sulle piattaforme online, come la diffusione di disinformazione o l'erosione della fiducia degli utenti. Alla fine, gli approcci multimodali rappresentano un passo fondamentale per lo sviluppo di sistemi di rilevamento più adattabili, completi ed efficaci, per affrontare le sfide poste dai social bot.

2.1.2 Basato su Modelli NLP

Una delle strategie più utilizzate nel rilevamento dei social bot è la combinazione di machine learning e natural language processing (NLP). Gli algoritmi di machine learning sono particolarmente bravi nell'individuare modelli in grandi set di dati, aiutando a identificare comportamenti tipici dei bot. Nel contempo, l'NLP analizza il linguaggio utilizzato dai bot, consentendo il rilevamento di frasi innaturali o ripetitive indicative di automazione. Combinando queste tecniche, si ottiene un potente insieme di strumenti per riconoscere i bot, basandosi sia sulle loro azioni che sul loro stile comunicativo. Integrando algoritmi di apprendimento supervisionato e non supervisionato, nonché metodi di natural language processing, i ricercatori possono analizzare i contenuti generati dai social bot e identificare modelli e caratteristiche che li differenziano dagli utenti umani [12]. Ad esempio, [32] ha proposto un metodo basato sul deep learning che combinava reti neurali convoluzionali (CNN) e reti neurali ricorrenti (RNN) per analizzare contenuti e caratteristiche degli account al fine di rilevare i bot, ottenendo elevata accuratezza e richiamo. L'analisi della rete è un'altra tecnica cruciale per il rilevamento dei social bot. Consiste nello studio delle strutture delle reti sociali e dei modelli di interazione degli account bot. Esaminando come i bot si connettono con altri utenti e si comportano all'interno di una rete, i ricercatori possono identificare modelli insoliti tipici degli account automatizzati. Ad esempio, i bot potrebbero formare gruppi strettamente connessi, impegnarsi in relazioni unidirezionali o mostrare una crescita rapida e innaturale dei follower. Questo tipo di analisi fornisce informazioni preziose sul comportamento dei bot ed è spesso utilizzato insieme ad altri metodi di rilevamento per ottenere risultati più accurati. Esaminando questi modelli, i ricercatori possono identificare comportamenti anomali e scoprire attività coordinate, come campagne di astroturfing e sforzi di disinformazione [22]. [33] ha introdotto un framework cross-platform chiamato FakeNewsNet per studiare la diffusione delle fake news e il ruolo dei social bot nella sua propagazione. Raccogliendo un ampio set di dati di articoli di notizie etichettati come falsi o veri e analizzando l'attività dei social bot attraverso le piattaforme, hanno identificato modelli di diffusione delle fake news e le caratteristiche dei bot coinvolti

nella loro diffusione. In sintesi, combinare una serie di tecniche di rilevamento come machine learning, natural language processing, analisi della rete, analisi temporale e analisi cross-platform può migliorare significativamente l'accuratezza e l'affidabilità del rilevamento dei social bot. Ognuna di queste metodologie porta punti di forza unici, consentendo un approccio più completo. Il machine learning e l'NLP aiutano ad analizzare i contenuti e i modelli linguistici, l'analisi della rete svela strutture di interazione insolite, l'analisi temporale traccia le attività nel tempo e l'analisi cross-platform identifica i bot che operano su diverse piattaforme sociali. Insieme, queste tecniche creano un sistema più robusto in grado di rilevare anche i social bot più sofisticati.

2.1.3 Basato su Modello di Crowdsourcing

[34] ha esplorato l'idea di coinvolgere gli esseri umani nel rilevamento dei social bot, suggerendo che il crowdsourcing del compito a grandi gruppi di lavoratori potrebbe essere una soluzione efficace. Come prova del concetto, hanno sviluppato una piattaforma di *Online Social Turing Test*, basata sull'assunzione che gli esseri umani siano ancora migliori dei macchinari nel rilevare i bot. Gli esseri umani sono in grado di cogliere sottili segnali nelle conversazioni, come sarcasmo o linguaggio persuasivo, e riconoscere schemi emergenti o anomalie nelle interazioni sociali, competenze che le macchine non hanno ancora replicato pienamente. Per testare il loro approccio, gli autori hanno utilizzato dati provenienti da Facebook e Renren (una popolare rete sociale cinese) e hanno chiesto a entrambi i valutatori esperti e ai lavoratori online di cercare di rilevare gli account bot basandosi esclusivamente sulle informazioni del profilo. I risultati hanno mostrato che, sebbene l'accuratezza del rilevamento dei lavoratori assunti sia diminuita nel tempo, è rimasta sufficientemente alta da poter essere utilizzata in un sistema di voto di maggioranza: ogni profilo veniva mostrato a più lavoratori, e l'opinione della maggioranza determinava il verdetto finale. Questa strategia ha portato a una bassissima percentuale di falsi positivi, un aspetto particolarmente importante per i fornitori di servizi che desiderano ridurre al minimo gli errori. Nonostante i risultati promettenti, il comportamento dei bot è diventato sempre più sofisticato. I bot ora possono costruire reti sociali realistiche e generare contenuti che imitano l'attività umana, inclusi schemi temporali naturali nei loro post. Man mano che i sistemi di rilevamento migliorano, possiamo aspettarci una continua "corsa agli armamenti", simile a quella che abbiamo visto con lo spam in passato [35]. Una delle principali sfide, tuttavia, è la necessità di una grande quantità di dati etichettati per addestrare i modelli di machine learning. In un ambiente così dinamico, tecniche come l'apprendimento attivo—dove il sistema può adattarsi e apprendere dai nuovi dati in tempo reale—potrebbero essere utili per affrontare la natura in continua evoluzione dei social bot.

2.1.4 L'Importanza dell'Aggiornamento

Per affrontare in modo efficace il paesaggio in continua evoluzione dei social bot, è cruciale adottare un approccio proattivo durante lo sviluppo e il deployment dei modelli di rilevamento. Ciò comporta l'aggiornamento e il perfezionamento continuo di questi modelli per adattarsi alle tattiche e alle strategie in evoluzione utilizzate

dagli operatori dei bot. Ad esempio, [36] ha dimostrato l'importanza di utilizzare set di dati diversificati e recenti per l'addestramento e la valutazione dei modelli di rilevamento dei social bot. Hanno condotto un confronto sistematico tra diversi set di dati e hanno scoperto che le prestazioni dei modelli addestrati su set di dati più vecchi erano significativamente inferiori rispetto a quelli addestrati su dati più recenti e diversificati. Adattarsi ai cambiamenti nelle piattaforme dei social media è un altro aspetto chiave per mantenere modelli di rilevamento efficaci. Queste piattaforme sono in costante evoluzione, con aggiornamenti alle API, alle interfacce utente e nuove funzionalità che possono influenzare il comportamento dei bot o come i sistemi di rilevamento interagiscono con la piattaforma. Di conseguenza, è essenziale regolarmente adeguare i modelli di rilevamento per tenere conto di questi cambiamenti e dei nuovi schemi di comportamento dei bot che potrebbero introdurre. [37] ha esaminato l'impatto dei limiti di velocità delle API di Twitter sul rilevamento dei social bot e ha scoperto che le restrizioni imposte riducevano l'accuratezza di alcune caratteristiche di rilevamento, evidenziando la necessità di adattare i modelli per accomodare i cambiamenti delle piattaforme. La regolazione dei parametri del modello è un passaggio critico per garantire l'efficacia continua dei modelli di rilevamento. Valutare regolarmente come questi modelli performano e aggiustare i loro parametri per migliorare l'accuratezza, il richiamo e altri metriche rilevanti può portare a risultati più affidabili. Questo processo potrebbe includere esperimentazioni con diversi set di caratteristiche, algoritmi o valori dei parametri, nonché la validazione incrociata per testare la robustezza dei modelli in diverse condizioni. Incorporare loop di feedback è un'altra pratica essenziale per mantenere e migliorare i modelli di rilevamento. Creando meccanismi per raccogliere feedback dagli utenti sull'accuratezza del rilevamento dei bot, i modelli possono essere continuamente perfezionati e migliorati. Ciò potrebbe coinvolgere il crowdsourcing, la validazione da parte di esperti o altre forme di coinvolgimento degli utenti per raccogliere intuizioni preziose e valutare le prestazioni reali dei metodi di rilevamento. Tali feedback aiutano a garantire che i sistemi di rilevamento evolvano parallelamente alle tattiche utilizzate dai bot, portando a una mitigazione più efficace dei problemi legati ai bot. Ad esempio, [38] ha sviluppato un sistema chiamato Botometer, che combinava il rilevamento dei bot basato sul machine learning con il feedback degli utenti per migliorare le sue prestazioni. Il sistema permetteva agli utenti di segnalare falsi positivi e falsi negativi, e gli input venivano utilizzati per perfezionare i modelli di rilevamento sottostanti. In conclusione, l'aggiornamento e il perfezionamento regolare dei modelli di rilevamento dei social bot è essenziale per rimanere al passo con il paesaggio in continua evoluzione dei social bot. Implementando strategie come l'aggiornamento dei dati di addestramento, l'adattamento ai cambiamenti delle piattaforme, la regolazione dei parametri del modello e l'incorporazione di loop di feedback, i ricercatori e i professionisti possono creare sistemi di rilevamento più efficaci e robusti. Questi sforzi aiuteranno infine a proteggere gli ecosistemi di informazioni online e a migliorare il rilevamento e la mitigazione delle attività dei social bot.

2.2 Altri progetti di ricerca

2.2.1 TwiBot20

TwiBot20 è un innovativo dataset creato per il rilevamento dei bot su Twitter, incorporando informazioni semantiche, proprietarie e relative al vicinato degli utenti per sfruttare la diversità della Twittersfera. Il dataset include informazioni come tweet recenti, caratteristiche degli utenti (ad esempio, numero di follower) e le relazioni di follow tra gli utenti. Gli utenti sono stati selezionati utilizzando una ricerca in ampiezza (Breadth-First Search, BFS) per generare un grafo diretto, con 40 utenti "seme" provenienti da vari settori, tra cui politica, affari, intrattenimento e sport, scelti per rappresentare una vasta gamma di interessi. Le informazioni semantiche (tweet recenti), le informazioni di proprietà (dati numerici e categoriali sugli utenti) e le informazioni relative al vicinato (relazioni di follow) sono state raccolte direttamente dall'API di Twitter. Per etichettare gli utenti come bot o umani, è stato impiegato un processo di crowdsourcing. Annotatori esperti su Twitter hanno valutato gli utenti sulla base di criteri specifici di comportamento automatizzato. Attraverso l'analisi dei dati, TwiBot20 ha rivelato differenze interessanti tra bot e utenti umani. Ad esempio, gli utenti umani tendono a seguire utenti reputati o "famosi", mentre i bot generalmente generano meno tweet rispetto agli utenti autentici. TwiBot20 ha anche messo in evidenza schemi unici nei bot, come l'uso di stringhe casuali nei nomi utente, distinguendoli ulteriormente dagli account umani.

2.2.2 TwiBot22

TwiBot-22 rappresenta un avanzamento significativo nel rilevamento dei bot su Twitter, distinguendosi per le sue dimensioni, la struttura del grafo eterogeneo e il robusto framework di valutazione. Essendo il più grande dataset del suo genere, TwiBot-22 è cinque volte più grande rispetto ai precedenti benchmark e utilizza grafi eterogenei per rappresentare la complessa rete di Twitter. Questa struttura più ampia e completa migliora la capacità del dataset di catturare accuratamente comportamenti anomali dei bot. TwiBot-22 include quattro tipi di entità (ad esempio, utenti, tweet, hashtag) e quattordici tipi di relazioni (ad esempio, follow, retweet). Il dataset contiene oltre 92 milioni di nodi e più di 170 milioni di archi, rendendolo il benchmark più grande disponibile per il rilevamento dei bot su Twitter. È stata utilizzata una strategia di ricerca in ampiezza (BFS) per la raccolta dei dati, partendo da "utenti seme" e ampliando attraverso le relazioni di follow. Oltre alla rete di follow, sono state raccolte anche altre entità e relazioni, come tweet, liste e hashtag, risultando in una struttura di grafo eterogenea. Per l'annotazione dei dati, TwiBot-22 ha fatto affidamento su 1.000 esperti di rilevamento dei bot piuttosto che sul crowdsourcing, come nel caso di TwiBot-20. Etichette probabilistiche generate con il sistema Snorkel sono state utilizzate per addestrare un classificatore MLP, ottenendo un'accuratezza del 90,5% nelle annotazioni di test. Questo approccio ha migliorato la qualità e la coerenza delle etichette, offrendo uno standard di affidabilità più elevato rispetto ai metodi precedenti.

2.2.3 Fox8

Questo studio analizza una botnet sociale su Twitter chiamata "fox8", composta da 1.140 account, alcuni dei quali sono parzialmente controllati dall'automazione, interagendo principalmente tramite retweet generati da modelli linguistici avanzati (LLM) come ChatGPT. Questi bot sono stati utilizzati per promuovere siti web sospetti e diffondere contenuti dannosi. La botnet fox8 è stata scoperta analizzando i tweet contenenti la frase "as an AI language model", che ha rivelato modelli di comportamento automatizzati. I ricercatori hanno raccolto 12.226 tweet da 9.112 account unici tra il 1° ottobre 2022 e il 23 aprile 2023. Dopo una revisione manuale, il 76% di questi account è stato classificato come umani che interagivano con contenuti generati da ChatGPT, mentre gli account rimanenti erano bot che utilizzavano LLM. Durante l'annotazione, i ricercatori hanno identificato modelli che collegano alcuni di questi bot a tre siti web sospetti: fox8.news (un sito di imitazione), cryptnomics.org e globaleconomics.news. Questo ha portato all'identificazione di una botnet di 1.140 account probabilmente utilizzando ChatGPT per la generazione di contenuti. Gli account bot di Fox8 hanno una media di 74,0 follower (SD=36,7), 140,4 amici (SD=236,6) e 149,6 tweet (SD=178,8). Questi bot interagiscono in tre modi principali: seguendo, retwittando e rispondendo. La rete di follower è densa, con una media di in-degree di 13,7 e out-degree di 13,4, mentre le reti di risposta e retweet sono più rare, con in-degree e out-degree molto più bassi. Un ulteriore analisi ha confrontato il comportamento di risposta dei bot fox8 con 7.972 account umani non associati alla botnet. I risultati hanno mostrato che i bot fox8 hanno una probabilità di risposta dello 0,2% tra loro, rispetto allo 0,016% nel gruppo di controllo, indicando interazioni più forti all'interno del gruppo di bot.

Rilevamento

La botnet fox8 è stata rilevata utilizzando il seguente approccio:

1. È stato utilizzato il dataset "fox8-23", contenente tweet sia di bot che di account umani.
2. I tweet di ciascun account sono stati valutati. Un punteggio elevato suggerisce che l'account potrebbe essere un bot.
3. È stata analizzata la deviazione standard dei punteggi dei tweet. Una bassa deviazione indica che l'account potrebbe essere un bot, poiché i bot tendono a generare contenuti coerenti.
4. È stato eseguito un test su un campione casuale di tweet pubblicati tra l'8 e il 14 maggio 2023.

Il test ha rivelato numerosi falsi positivi, soprattutto tra gli account con pochi tweet brevi. Ad esempio, frasi come "thank you" e "amen" sono state erroneamente classificate come generate da LLM. **Punteggio Elevato:** Un punteggio elevato indica una probabilità alta che il testo sia stato generato da un LLM, sulla base di caratteristiche come un linguaggio sofisticato, coerenza tematica e sintassi tipica degli LLM. **Deviazione Standard:** Questa misura analizza la variazione del contenuto

dei tweet di un account. I bot tendono a generare contenuti uniformi, risultando in una deviazione standard più bassa, mentre gli utenti umani producono contenuti più diversificati.

2.3 Casestudies: Rilevamento dei Bot Sociali nelle Applicazioni del Mondo Reale

In questa sezione esploriamo una serie di studi di caso che evidenziano le applicazioni reali delle tecniche di rilevamento dei bot sociali in vari settori. Questi esempi sottolineano il ruolo cruciale del rilevamento dei bot sociali nell'affrontare sfide significative come l'interferenza elettorale, la manipolazione politica, le campagne di disinformazione, le notizie false e le truffe finanziarie, inclusa la manipolazione delle criptovalute. Esaminando questi studi di caso, l'obiettivo è fornire approfondimenti sulle implicazioni pratiche del rilevamento dei bot sociali e sottolineare la necessità di un continuo sviluppo e ricerca in questo settore.

2.3.1 Manipolazione Politica

I bot sociali sono stati ampiamente utilizzati per manipolare l'opinione pubblica e interferire nelle elezioni, diffondendo propaganda politica, false informazioni e contenuti polarizzanti. Le elezioni presidenziali statunitensi del 2016 sono un esempio ben noto in cui i bot sociali hanno svolto un ruolo significativo nell'amplificare e diffondere contenuti politicizzati, influenzando le dinamiche delle elezioni [39, 40, 41, 42]. I ricercatori hanno sviluppato varie tecniche di rilevamento dei bot sociali mirate specificamente ai bot politici, con l'obiettivo di ridurre al minimo la loro influenza sul discorso pubblico e sui processi democratici [43, 44]. Un'analisi approfondita delle elezioni presidenziali statunitensi del 2016 [40] ha rivelato che i bot sociali erano responsabili della generazione e della diffusione di una porzione significativa dei contenuti legati alle elezioni sulle piattaforme sociali, in particolare su Twitter. I bot sono stati trovati responsabili di circa un quinto di tutte le conversazioni relative alle elezioni, con una visibile inclinazione verso determinati argomenti politici e candidati. Applicando algoritmi di machine learning per analizzare i modelli comportamentali, Bessi e Ferrara sono stati in grado di scoprire la presenza diffusa di bot e valutare il loro potenziale impatto sulle elezioni. Questi risultati sono stati successivamente collegati a un'operazione sponsorizzata dallo stato condotta dalla Russia per interferire nelle elezioni statunitensi [41, 45, 46]. [47] ha proposto un metodo per rilevare i bot sociali nel contesto delle discussioni politiche su Twitter, focalizzandosi sul referendum catalano per l'indipendenza del 2017. Hanno utilizzato una combinazione di tecniche di apprendimento non supervisionato, come il clustering e la riduzione dimensionale, per identificare gruppi di utenti simili e rilevare i bot in base ai loro modelli comportamentali. Esaminando i contenuti creati da questi bot, i ricercatori [47] sono stati in grado di rivelare campagne di disinformazione coordinate e comprendere le tattiche utilizzate dagli operatori dei bot durante le elezioni. In conclusione, i bot sociali hanno avuto un ruolo determinante nel plasmare l'opinione pubblica e interferire nelle elezioni in vari paesi. La ricerca sul rilevamento e la comprensione di come operano i bot politici ha fornito importanti spunti sulle

loro strategie e sull'impatto potenziale sui processi democratici. Sviluppare metodi efficaci per rilevare e contrastare questi bot può aiutare a limitare la loro influenza sulle discussioni pubbliche e proteggere l'equità delle elezioni.

2.3.2 Campagna di Disinformazione

La diffusione di disinformazione e fake news sui social media è diventata una preoccupazione crescente, poiché può alimentare la disinformazione, approfondire la polarizzazione e minare la fiducia nei media e nelle istituzioni. Rilevare i bot sociali che partecipano a queste campagne di disinformazione è essenziale per ridurre la diffusione di fake news e preservare l'integrità delle informazioni online. In uno studio di [48], gli autori hanno proposto un modello ibrido di deep learning chiamato CSI (Capture, Score, and Integrate) per rilevare le fake news e i bot sociali responsabili della loro diffusione. Combinando caratteristiche sia del contenuto che delle strutture delle reti sociali, il loro modello ha ottenuto un'alta precisione nel rilevare le fake news e i bot che le diffondevano. Ciò evidenzia il potenziale delle tecniche basate sull'IA nel contrastare le sfide legate alla disinformazione. In un altro studio, [49] ha esplorato la diffusione di notizie vere e false online e ha scoperto che le informazioni false si diffondevano più rapidamente e ampiamente rispetto alle informazioni vere, in parte a causa dell'attività dei bot sociali. Hanno utilizzato diverse tecniche di machine learning per modellare come le notizie si diffondono e identificare le caratteristiche che distinguono le informazioni vere da quelle false. I loro risultati hanno sottolineato l'importanza di un rilevamento efficace dei bot sociali e di contromisure per frenare la diffusione della disinformazione. Affrontare il problema della disinformazione e delle fake news richiede un approccio globale, che comprenda lo sviluppo di metodi di rilevamento in grado di identificare e mitigare l'impatto dei bot che diffondono informazioni false. Utilizzando tecniche avanzate di machine learning e elaborazione del linguaggio naturale, i ricercatori possono ottenere una comprensione più profonda delle strategie e delle tattiche delle campagne di disinformazione, permettendo loro di progettare interventi che proteggano l'integrità delle fonti di informazioni online.

2.3.3 Frodi Finanziarie

L'ascesa dei bot sociali sulle piattaforme di social media ha portato a una nuova ondata di frodi finanziarie e manipolazioni delle criptovalute, lasciando molti utenti ignari vulnerabili alla truffa digitale. Questi bot, spesso definiti "lupi travestiti da agnelli", possono imitare i comportamenti umani in modo così convincente che possono diffondere rapidamente informazioni false o addirittura eseguire schemi fraudolenti come il pump-and-dump, il phishing o manipolare i prezzi delle azioni e delle criptovalute. I bot sociali nel settore finanziario sono particolarmente problematici a causa della loro capacità di operare su larga scala e con una velocità allarmante. Possono inondare le piattaforme con notizie false, convincere gli utenti a investire in asset senza valore, o gonfiare artificialmente il prezzo delle criptovalute, il tutto mentre restano inosservati dai metodi tradizionali di rilevamento. Tuttavia, ci sono segnali di speranza all'orizzonte. Sebbene rilevare questi bot non sia affatto facile, poiché essi si evolvono continuamente per eludere i metodi tradizionali di rilevamento, la ricerca sta facendo progressi per sviluppare strumenti più sofisticati.

Vari studi hanno proposto metodi per tracciare il comportamento dei bot attraverso l'analisi delle reti, gli algoritmi di machine learning e il rilevamento di anomalie comportamentali—mirando essenzialmente a cogliere i bot in flagrante prima che possano causare danni ai sistemi finanziari. Incorporare queste tecniche in strategie anti-frode più ampie potrebbe fornire una difesa multilivello, dando agli utenti e alle piattaforme finanziarie la capacità di identificare meglio questi disturbatori digitali prima che causino danni maggiori. Ad esempio, [50] ha sviluppato un framework basato sui dati per identificare gli schemi pump-and-dump nei mercati delle criptovalute. Raccogliendo e analizzando milioni di messaggi da piattaforme come Twitter, Telegram e Discord, il loro modello è stato in grado di svelare meccanismi come pump-and-dump e schemi Ponzi, rivelando attività associate a bot sospetti coinvolti in frodi legate alle criptovalute. Per ridurre l'impatto dei bot sociali nelle frodi finanziarie e nella manipolazione delle criptovalute, è essenziale che i ricercatori sviluppino nuovi metodi di rilevamento in grado di stare al passo con le tattiche in continua evoluzione utilizzate da questi bot. Le potenziali direzioni di ricerca includono l'incorporazione di algoritmi di deep learning e reinforcement learning, nonché lo sviluppo di modelli che possano analizzare fonti di dati multimodali, come testi, immagini e video [51]. Il rilevamento dei bot sociali è diventato un campo di ricerca sempre più importante, poiché i bot maligni continuano a perturbare gli ecosistemi informativi online e a creare sfide significative in vari settori. In questa sezione finale, riassumiamo le principali sfide e opportunità nel rilevamento dei bot sociali, esploriamo il futuro del rilevamento nell'era dei modelli avanzati di IA come ChatGPT e offriamo considerazioni finali insieme a potenziali direzioni per la ricerca futura.

Chapter 3

Strumenti e tecnologie utilizzati

3.1 Web scraping

La principale tecnica utilizzata dal nostro strumento per recuperare i dati è il web scraping; questo processo prevede l'estrazione automatica di dati da siti web, finalizzata alla raccolta di informazioni specifiche da una o più pagine per scopi come l'analisi dei dati, la creazione di dataset o il monitoraggio di determinati contenuti. Per ottenere ciò, gli script o i programmi noti come scrapers inviano richieste ai siti web per estrarre i dati desiderati (come testo, immagini o altri contenuti strutturati) e salvarli in un formato riutilizzabile. Ecco i principali passaggi del processo di scraping nel nostro progetto:

- **Invio della richiesta**: Lo scraper accede a una o più pagine web inviando richieste HTTP, simulando il comportamento di un utente o di un browser normale. Nel nostro progetto, l'accesso alle informazioni è facilitato dall'uso delle API fornite dal server, che consentono richieste di dati specifici in modo strutturato e ottimizzato, riducendo il rischio di errori e sovraccarichi.
- **Ricezione del contenuto**: La risposta alle richieste inviate arriva in formato JSON, un formato leggero e facilmente leggibile sia dai computer che dagli esseri umani. Ogni risposta JSON include i campi di dati di interesse, facilitando una gestione rapida e precisa delle informazioni.
- **Estrazione dei dati**: Vengono identificati e raccolti solo i dati specificamente necessari per l'analisi, mentre il contenuto non rilevante viene ignorato. In questa fase, lo scraper filtra accuratamente il contenuto JSON ricevuto, selezionando i campi e gli attributi mirati.
- **Organizzazione e salvataggio**: Una volta estratti, i dati vengono organizzati in un formato strutturato, come un database SQL, che consente l'analisi successiva. Questo è un passaggio cruciale per garantire che i dati siano pronti per essere elaborati, visualizzati o esportati in strumenti di analisi dei dati.

Il web scraping è una tecnica potente, ma richiede il rispetto delle politiche di utilizzo dei siti web e delle leggi sulla protezione dei dati. Alcuni siti vietano

esplicitamente lo scraping nei loro termini di servizio o tramite il file `robots.txt`, che fornisce istruzioni ai bot su quali pagine siano accessibili e quali no. Nel nostro caso, abbiamo rispettato il limite del server, che consente un massimo di 300 richieste ogni 5 minuti. Questo limite è stato gestito inserendo pause automatiche nello script e ruotando i server proxy.

Uno degli strumenti principali utilizzati è la libreria `BeautifulSoup` per Python, che facilita il parsing del contenuto HTML. Sebbene la maggior parte dei dati nel nostro progetto provenga da API JSON, `BeautifulSoup` è stata utilizzata in situazioni in cui era necessario accedere a contenuti disponibili solo in HTML, come i post pubblicati dagli utenti e le descrizioni dei loro profili.

Grazie al web scraping, è stato possibile raccogliere rapidamente e con precisione grandi quantità di dati senza inserimento manuale. Questo approccio ha permesso la creazione di dataset aggiornati e coerenti, riducendo il margine di errore umano e accelerando il processo di analisi.

3.2 Application Programming Interfaces (APIs)

Abbiamo utilizzato le **Application Programming Interfaces (APIs)** per interrogare il server Mastodon. Le API sono interfacce che consentono a diverse applicazioni software di comunicare tra loro. Un'API definisce un insieme di regole e protocolli che specificano come un'applicazione può richiedere e ricevere dati o servizi da un'altra applicazione, senza la necessità di comprendere i dettagli interni della sua implementazione.

Pensate a un'API come a un cameriere in un ristorante:

- **Richiesta**: Voi (l'applicazione client) fate un ordine al cameriere (API), scegliendo dal menu (specifica dell'API).
- **Intermediario**: Il cameriere prende il vostro ordine e lo invia alla cucina (server dell'applicazione), senza che voi dobbiate entrare in cucina o capire come il piatto viene preparato.
- **Risposta**: La cucina prepara il piatto e il cameriere lo porta a voi.

Questo è esattamente il ruolo di un'API tra due applicazioni: l'applicazione client invia una richiesta all'API, che poi restituisce i dati o i servizi richiesti.

Esistono diversi tipi di API a seconda del loro scopo e funzionamento:

- **Web API**: Il tipo più comune, che consente alle applicazioni di comunicare tramite Internet (ad esempio, REST, SOAP).
- **System API**: Permette ai programmi di interagire con il sistema operativo (ad esempio, Windows API).
- **Library API**: Utilizzata per accedere a funzionalità specifiche di una libreria o framework (ad esempio, la libreria grafica OpenGL).
- **Class or Module API**: Utilizzata all'interno del codice di un programma per accedere alle funzionalità di una specifica classe o modulo.

Le API sono una funzionalità utile poiché consentono di riutilizzare servizi e funzionalità esistenti senza doverli ricreare; questo permette a diversi sistemi di connettersi e integrarsi facilmente all'interno dello stesso contesto.

3.2.1 REST API di Mastodon

Nell'ecosistema di Mastodon, le **REST API (Representational State Transfer)** forniscono un modo per accedere ai dati e alle funzionalità della piattaforma in modo strutturato, utilizzando **HTTP** per le richieste e **JSON** per i dati inviati e ricevuti. Come spiegato nella documentazione:

> Mastodon fornisce accesso ai suoi dati tramite una REST API. REST sta per REpresentational State Transfer, ma per i nostri scopi, consideratelo semplicemente come un modo per inviare e ricevere informazioni su varie risorse in base alla richiesta. Le REST API di Mastodon utilizzano HTTP per le richieste e JSON per i payload.

La logica delle REST API si basa su un insieme di endpoint specifici accessibili tramite metodi HTTP, come indicato nella documentazione di Mastodon:

- **GET**: Recupera informazioni su una risorsa. Ad esempio, è possibile utilizzare GET per visualizzare un post o ottenere i dettagli di un utente.
- **POST**: Invia informazioni al server per creare una nuova risorsa, come un nuovo post.
- **PUT o PATCH**: Aggiorna una risorsa esistente, ad esempio modificando il contenuto di un post o aggiornando il profilo di un utente.
- **DELETE**: Rimuove una risorsa specifica, come eliminare un post o un commento.

Esempi di una richiesta GET su Mastodon potrebbero essere:

- **GET /api/v1/statuses**: Per recuperare i post pubblici.
- **GET /api/v1/accounts/:id**: Per ottenere i dettagli di un account specifico, dato il suo ID.
- **GET /api/v1/statuses/:id**: Per visualizzare un post specifico, dato il suo ID.

Le REST API di Mastodon offrono numerosi endpoint per interagire con la piattaforma:

- Recupero di post pubblici.
- Aggiornamento dei profili utente.
- Visualizzazione delle notifiche.

- Creazione o eliminazione di post.

Le richieste API possono includere parametri aggiuntivi, come query string, dati di modulo o JSON, che permettono di filtrare o personalizzare la risposta. Per impostazione predefinita, ogni richiesta restituisce 20 oggetti, ma con il parametro `limit`, è possibile ottenere fino a 40 oggetti. Molti altri parametri possono essere utilizzati come filtri.

Le risposte delle API includono un corpo in formato JSON e codici di stato HTTP, come spiegato di seguito:

- **200 OK**: La richiesta è stata gestita con successo.
- **4xx Client Error**: La richiesta contiene un errore, come ad esempio:
  - **401 Unauthorized**: L'utente non è autorizzato ad accedere alla risorsa.
  - **404 Not Found**: La risorsa richiesta non esiste.
  - **422 Unprocessable**: La richiesta è valida ma non può essere elaborata.
- **5xx Server Error**: Indica un errore del server, come ad esempio:
  - **503 Unavailable**: Il server non è attualmente disponibile.

Alcuni metodi delle API di Mastodon richiedono l'autenticazione tramite **OAuth**, un protocollo che consente di effettuare richieste sicure per conto di un utente. Per utilizzare le API protette da OAuth, è necessario includere il `access_token` nell'header HTTP della richiesta:

Authorization: Bearer <access_token>

3.2.2 Rate Limits

Mastodon impone limiti sul numero di richieste che un utente o una macchina può effettuare in un determinato periodo di tempo per prevenire sovraccarichi del server e garantire che la piattaforma rimanga reattiva per tutti gli utenti.

I principali limiti sono:

- **Limiti per account**:
  - Ogni metodo API può essere chiamato 300 volte ogni 5 minuti per account.
  - I caricamenti di file multimediali (`POST /api/v1/media`) sono limitati a 30 richieste ogni 30 minuti.
  - L'eliminazione dei post (`DELETE /api/v1/statuses/:id`) è limitata a 30 richieste ogni 30 minuti.
- **Limiti per IP**:
  - Ogni metodo API può essere chiamato 300 volte ogni 5 minuti per indirizzo IP.

Inoltre, le risposte delle API includono header che forniscono informazioni sullo stato dei limiti:

- **X-RateLimit-Limit**: Numero massimo di richieste consentite nel periodo di tempo.
- **X-RateLimit-Remaining**: Numero di richieste rimanenti prima di raggiungere il limite.
- **X-RateLimit-Reset**: Timestamp che indica quando il limite verrà reimpostato.

3.3 Data Annotation and MySQL

Un dataset annotato costruito su MySQL è stato utilizzato per registrare i dati raccolti dallo strumento.[1] Ogni annotazione arricchisce i dati grezzi (ad esempio, il testo di un post) con dettagli contestuali che possono essere utilizzati per eseguire analisi più avanzate. Ad esempio, ciò consente di analizzare come gli account bot evolvano nel tempo osservando i cambiamenti nel loro comportamento in relazione all'evoluzione della piattaforma.

Nel nostro caso, stiamo raccogliendo post da Mastodon insieme a dettagli aggiuntivi come la data di creazione, le risposte ricevute, il numero di reblog e altro. Questi dati e le relative annotazioni sono archiviati nel database MySQL. Ogni post è un record che contiene diverse colonne per ciascuna annotazione (ad esempio, "responses", "media", "language").

3.4 Data Annotation and MySQL

Come parte di questa tesi, il dataset annotato che stiamo costruendo non è solo un semplice repository di dati sociali, ma un dataset specializzato per la *rilevazione dei bot* all'interno della piattaforma Mastodon. In altre parole, stiamo creando un dataset che non solo raccoglie informazioni sugli utenti e i loro post, ma include anche annotazioni che ci permetteranno di distinguere tra comportamenti umani e quelli indicativi di attività di bot.

MySQL gioca un ruolo cruciale nella gestione e nell'analisi di questo dataset annotato. La struttura del database consente la raccolta e la memorizzazione efficienti dei dati sugli utenti e sui post, nonché delle annotazioni specifiche essenziali per la rilevazione dei bot. Ogni post è rappresentato da un record che include colonne per vari tipi di annotazioni (ad esempio, `replies_count`, `reblogs_count`, `favourites_count`), permettendo query avanzate per cercare e identificare comportamenti sospetti. Inoltre, l'uso di chiavi primarie (`post_id`, `account_id`) permette di stabilire relazioni tra utenti e i loro post, facilitando la raccolta di tutte le informazioni pertinenti per ogni account.

[1] Un dataset annotato è un insieme di dati arricchito con informazioni aggiuntive (le "annotazioni") che descrivono, etichettano o categorizzano i dati stessi.

Un esempio di query utile per rilevare i bot potrebbe coinvolgere la ricerca di account con un numero anomalo di post in un breve intervallo di tempo o un alto numero di post che non ricevono interazioni (senza like, senza risposte, senza reblog). Le annotazioni dei post, come il campo `created_at` e i conteggi delle interazioni, aiutano a costruire questi schemi di comportamento.

3.5 Proxy Server

Come descritto nella sezione successiva, i server proxy sono stati un componente fondamentale del nostro progetto. Prima di entrare nei dettagli sui loro utilizzi specifici, comprendiamo innanzitutto cosa sono e quali sono le loro funzioni.

Un proxy è un server intermedio che agisce come un "ponte" tra un client e il server finale, inoltrando le richieste e le risposte tra i due.

Esso svolge molteplici funzioni:

- **Miglioramento della Sicurezza**: Fornisce protezione contro l'accesso diretto e consente l'anonimato.
- **Caching e Ottimizzazione delle Prestazioni**: Riduce la latenza grazie alla cache intermedia.
- **Controllo degli Accessi e Filtraggio**: Monitora e limita l'accesso a siti specifici.
- **Distribuzione del Carico**: Ottimizza la gestione del traffico sui server di backend.

Esistono diversi tipi di proxy:

- **Proxy HTTP**: Gestisce le richieste e le risposte HTTP, tipicamente sulle porte standard (come 80 per HTTP e 443 per HTTPS).
- **Proxy HTTPS (SSL Proxy)**: Gestisce il traffico HTTPS, crittografato con SSL/TLS.
- **Proxy SOCKS**: Gestisce qualsiasi tipo di traffico (HTTP, FTP, SMTP e altri) ed è spesso utilizzato per bypassare restrizioni di rete. SOCKS è un protocollo di livello inferiore che opera al livello di sessione.[2]
- **Proxy Trasparente**: Reindirizza il traffico senza che l'utente sia consapevole della sua esistenza ed è comunemente utilizzato per monitorare il traffico in modo non intrusivo (ad esempio in uffici o scuole).
- **Proxy Inverso**: Gestisce il traffico in ingresso verso il server web (backend) e lo inoltra ai server interni appropriati.

[2] L'ordine dei livelli del modello OSI, dal livello 1 al livello 7, è il seguente: 1) Fisico, 2) Data Link, 3) Rete, 4) Trasporto, 5) Sessione, 6) Presentazione, 7) Applicazione.

Proxies possono anche essere utilizzati per garantire anonimato e privacy, nascondendo l'indirizzo IP reale del client. Tuttavia, non tutti i proxy offrono questo servizio. Ad esempio:

- **Proxy Trasparenti**: Inoltrano l'indirizzo IP dell'utente e non offrono anonimato.
- **Proxy Anonimi**: Nascondono l'IP del client omettendo determinati tipi di dati, come gli identificatori del client.
- **Proxy ad Alta Anonimato (Proxy Elite)**: Garantiscano la massima privacy, camuffando l'indirizzo IP dell'utente e l'uso del proxy.

Sebbene i proxy offrano vantaggi in termini di privacy, presentano anche limitazioni e rischi. Ad esempio, i proxy pubblici o non protetti potrebbero intercettare e registrare dati sensibili, esponendo a potenziali violazioni della privacy o rischi di furto di dati. Pertanto, le informazioni private o le credenziali dovrebbero essere nascoste. Inoltre, le prestazioni della connessione potrebbero subire rallentamenti, specialmente se il proxy è geograficamente lontano dal client o dal server di destinazione, causando latenza e tempi di risposta più elevati.

Scegliere il tipo di proxy più adatto e configurarlo correttamente, facendo alcuni compromessi, può bilanciare la sicurezza dei dati con le prestazioni della connessione.

Nel nostro progetto, abbiamo utilizzato un **Proxy in avanti** gratuito per implementare la **rotazione dei proxy**. Questa è una tecnica comune utilizzata nel web scraping per evitare che il programma venga interrotto a causa del superamento dei limiti di richiesta. Quando viene raggiunto il numero massimo di richieste, attivando l'errore 429, il client che effettua le richieste viene aggiornato con un nuovo proxy. Per implementare la rotazione dei proxy, aggiorniamo semplicemente i parametri della richiesta GET specificando il server proxy da utilizzare.

3.6 About the Efficiency

Inizialmente abbiamo optato per un approccio lineare.

- **Approccio sequenziale**:

  La pratica comune di fare più richieste HTTP può essere eseguita in modo sequenziale.

  Ci sono due modi per fare queste richieste: con o senza mantenere una sessione persistente.

  Quest'ultima è comunemente utilizzata da chi è alle prime armi (come nel mio caso quando ho iniziato questo progetto). Quando si utilizza `requests.get()`, si effettua una richiesta HTTP senza mantenere una sessione persistente. Questo significa che la libreria `requests` apre una nuova connessione HTTP al server, invia la richiesta, riceve la risposta e quindi chiude la connessione. Questo metodo è dispendioso in termini di tempo, poiché richiede di negoziare una connessione TCP e, se si tratta di una richiesta HTTPS, il "handshake"

SSL ogni volta. Ciò può rallentare il programma, specialmente se vengono fatte molte richieste allo stesso server. Inoltre, poiché la connessione non viene mantenuta, le risorse delle connessioni esistenti (come i cookie o gli header persistenti) non possono essere riutilizzate per richieste successive, limitando la continuità della sessione.

D'altra parte, quando si utilizza `requests.Session()`, viene creata una sessione HTTP che mantiene la connessione TCP aperta per una serie di richieste consecutive al server. Questa sessione consente alle richieste di riutilizzare la stessa connessione per più richieste, creando un'operazione più efficiente e coerente. Riduce la latenza evitando l'overhead di riaprire la connessione e conserva i cookie, gli header personalizzati e altre impostazioni tra le richieste. Ad esempio, se un sito utilizza l'autenticazione basata su cookie, una volta che l'autenticazione è avvenuta con successo, i cookie di sessione verranno mantenuti nella sessione, consentendo un accesso continuo alle richieste successive.

3.7 Approccio Asincrono

Successicamente abbiamo optato per un approccio asincrono per ottimizzare il flusso di lavoro.

- **Approccio asincrono**:

  L'asincronicità è un concetto di programmazione che consente di gestire più operazioni contemporaneamente in modo efficiente, sfruttando un singolo thread che richiede una quantità minima di memoria e potenza di calcolo dalla CPU, a differenza del multiprocessing. È particolarmente utile per operazioni che richiedono di attendere una risposta (come richieste di rete, lettura/scrittura su file o interazioni con database).

  L'asincronicità è ideale per operazioni di rete latenti, come il recupero dei dati tramite l'utilizzo di API, e l'interazione con sistemi o dispositivi esterni dove il programma deve attendere una risposta. Tuttavia, non è adatta per calcoli complessi che richiedono molta potenza di elaborazione.

  In un programma tradizionale sincrono, le istruzioni vengono eseguite una alla volta, con ogni funzione che termina prima che inizi la successiva. L'asincronicità, invece, opera utilizzando un **loop degli eventi** che decide quale operazione eseguire in un dato momento. Questo gestisce diverse operazioni chiamate **coroutines** [3] e, ogni volta che un'operazione sta aspettando una risposta dal server, passa al prossimo compito senza bloccare l'intero programma. Questa funzionalità è resa possibile dalla funzione `await`.

  Il loop degli eventi è quindi come un direttore d'orchestra che chiama le coroutines (o i suoi membri dell'orchestra) per eseguirle, ma se una non è pronta, passa a quella successiva.

[3] Le coroutines sono definite con la parola chiave `async def`. Quando si chiama una coroutine, invece di eseguire il codice immediatamente, essa restituisce un oggetto che può essere eseguito quando è pronto, ma non viene eseguito immediatamente.

Un approccio sincrono sarebbe come avere un solo cameriere che prende un ordine, aspetta che il cibo sia pronto, lo serve e poi si sposta al tavolo successivo, mentre gli altri tavoli devono aspettare. Un approccio asincrono è come avere un cameriere che prende l'ordine e, mentre aspetta che il cibo sia pronto, prende altri ordini.

- **Httpx**:

  Httpx è una libreria Python per effettuare richieste HTTP e HTTP/2 che combina la semplicità di `requests` con il supporto nativo per operazioni sia sincrone che asincrone. In modalità asincrona, `httpx` utilizza *asyncio*, supportando efficientemente molte richieste concorrenti senza bloccare il thread principale.

  Progettata per essere facile da usare, `httpx` mantiene una sintassi simile a quella di `requests` ma con un potente supporto per operazioni sincrone e asincrone, timeouts, autenticazione avanzata e gestione delle connessioni persistenti, migliorando le prestazioni quando si effettuano molte richieste allo stesso server. Come `aiohttp`, `httpx` non è adatto per attività ad alta intensità di CPU.

  `Httpx` utilizza un modello di client persistente grazie a `Client` e `AsyncClient`, che impiegano una gestione avanzata delle connessioni chiamata **connection pooling**. Questo mantiene la connessione aperta per le richieste successive, evitando la latenza aggiuntiva dovuta all'apertura e chiusura continua delle connessioni TCP.

  Dal punto di vista del progetto, è importante notare che `httpx` offre il supporto integrato per lavorare con proxy HTTP e HTTPS, migliorando la sicurezza, la privacy o le esigenze di rete.

Chapter 4

Mastoanalyzer

4.1 Mastoanalyzer: uno strumento per l'analisi dei dati di Mastodon e il rilevamento dei bot

Mastoanalyzer mira a creare uno strumento di analisi che faciliti la raccolta dei dati su Mastodon e supporti il rilevamento dei bot. Questo strumento è progettato per consentire ai ricercatori e agli analisti di estrarre, organizzare e analizzare informazioni rilevanti su utenti e post di Mastodon per ottenere intuizioni significative sul comportamento degli utenti.

Lo strumento utilizza le tecnologie menzionate nel capitolo precedente, inclusi il web scraping, che sfrutta le API REST fornite da Mastodon per accedere in modo sicuro e affidabile alle informazioni pubbliche. Utilizza anche dataset annotati memorizzati in MySQL, scelto per la sua robustezza nella gestione di query strutturate e veloci e per la facile integrazione con strumenti di analisi. Con l'archiviazione dei dati in MySQL, lo strumento può supportare analisi a lungo termine, gestire set di dati più ampi e facilitare l'accesso rapido ai dati senza problemi di prestazioni. Inoltre, il formato JSON consente l'integrazione con altri software di analisi.

L'architettura dello strumento è progettata per promuovere una crescita organica, consentendo di aggiungere funzionalità aggiuntive senza dover riscrivere il sistema da zero. Ad esempio, la rapida modifica degli endpoint (istanze e timeline di interesse) lo rende adattabile a diverse esigenze di ricerca senza richiedere competenze avanzate di programmazione da parte dell'utente. È anche modulare, con spazi dedicati per configurazioni e parametri che promuovono l'evoluzione futura dello strumento senza compromettere la sua stabilità. Inoltre, è abbastanza versatile da supportare l'implementazione di algoritmi di machine learning, consentendo la creazione di modelli di classificazione dei bot in tempo reale utilizzando i dati raccolti.

Questo progetto ha cercato di combinare praticità e considerazioni etiche, offrendo una soluzione accessibile per analizzare il contesto digitale sempre più complesso delle reti sociali decentralizzate. La flessibilità e l'interoperabilità dello strumento mirano a soddisfare le esigenze immediate di raccolta dei dati, promuovendo al contempo una maggiore consapevolezza dell'impatto dell'automazione e della qualità delle interazioni online. Contribuendo a nuove possibilità di analisi trasparente, questo strumento si propone come una risorsa preziosa per comprendere meglio le

dinamiche delle comunità digitali e per supportare una gestione responsabile dei dati, rispettando l'autonomia di piattaforme come Mastodon.

4.2 Architettura del programma

Il cuore del programma ruota attorno a due funzioni principali: la prima consiste nel raccogliere tutti gli utenti che interagiscono in una timeline specifica, e la seconda nel registrare gli stati che questi utenti hanno pubblicato sul loro profilo.

1. **Recuperare gli utenti (def get_timeline_posts())**:

   Durante questa fase, viene interrogata la timeline di mastodon.social. Sfruttando le API REST di Mastodon, possiamo recuperare tutti gli stati pubblicati qui. Ogni richiesta può contenere un massimo di 40 stati[1], quindi sono state effettuate 300 richieste[2], raccogliendo un totale di 12.000 stati dai post più recenti. Per restringere il campo di analisi, abbiamo aggiunto tag all'*url* da cui vengono raccolti i post. Il tag principale utilizzato è stato *politics*, con tag secondari come *technology*, *science* e *internet*.

   Poiché l'attenzione si è spostata sugli utenti, la risposta JSON di ogni richiesta contiene non solo informazioni sullo stato, ma anche sull'account che lo ha pubblicato. In questo modo, abbiamo raccolto informazioni sugli utenti.

   Una volta che il programma è stato eseguito correttamente e per ogni post abbiamo le informazioni sull'utente, possiamo caricare ogni tupla[3] nella tabella *users*, che contiene i seguenti campi:

   Table 4.1. users

Field	Type	Null	Key
user_id	varchar(255)	NO	PRI
username	varchar(255)	YES	UNI
bot	tinyint(1)	YES	
url	varchar(255)	YES	
followers	int	YES	
following	int	YES	
statuses	int	YES	
description	text	YES	

   Il seguente grafico mostra le differenze tra l'apertura di una sessione per ogni richiesta e l'utilizzo di una singola sessione per tutte le richieste.

   Come illustrato nella figura, il tempo di risposta aumenta gradualmente con il numero di richieste, passando da poche centinaia di millisecondi a quasi

[1] L'API REST di Mastodon imposta come numero predefinito di stati che è possibile recuperare per richiesta fino a 20, ma è adattabile da 1 a 40.

[2] Per account e per IP, tutti gli endpoint e i metodi possono essere chiamati 300 volte ogni 5 minuti.

[3] Mastodon consente di auto-dichiararsi bot, nella risposta JSON viene riportato e la chiave ha valore 1.

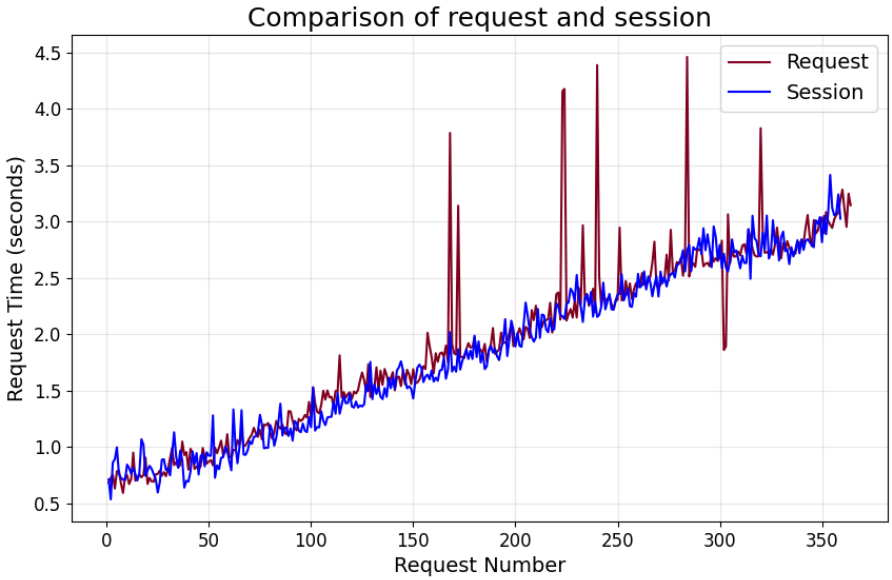


Figure 1. Comparison beetween *request.get()* and *session.get()*

tre secondi. I grafici sono stati generati da un totale di 350 richieste. Nel caso di *request*, il tempo totale di elaborazione è stato di 697 secondi, mentre nel caso di *session* è stato di 660 secondi. Il confronto tra i due scenari mostra che l'approccio basato sulla sessione è più vantaggioso, poiché riduce il tempo di elaborazione complessivo e garantisce anche una risposta più stabile e prevedibile, evitando picchi anomali nel tempo di risposta.

2. **Statuses per utente (def fetch_posts()):**

   Una volta che la tabella degli utenti è stata riempita, possiamo iniziare a recuperare i post di questi utenti. Partendo dagli utenti più attivi (quelli con più status), recuperiamo un numero dato di status per utente, nel nostro caso, 200 status. Questo ci consente di interrogare fino a 60 utenti a causa dei limiti di frequenza.

   Tuttavia, una volta raggiunto il limite di frequenza, il programma solleva l'errore 429.[4] Per risolvere questo problema, viene avviato un *time.sleep* con una durata variabile a seconda dei secondi rimasti fino al tempo di reset. Il valore predefinito per *time.sleep* è di 300 secondi, poiché il reset avviene ogni 5 minuti. Questo automatizza il programma e garantisce che tutti gli utenti vengano interrogati.

   Un altro problema si verifica perché tutte le richieste vengono fatte dallo stesso indirizzo IP. Mastodon rallenta il tempo di risposta fino a chiudere la connessione e il programma solleva l'errore 503.[5]

   La soluzione agli errori 429 e 503 è l'uso della rotazione dei proxy. Quando il programma incontra un errore di superamento del limite di frequenza, invece di entrare in *time.sleep*, cambiamo il client che effettua la richiesta con un nuovo proxy. Questo garantisce una continuità quasi infinita del programma, impedendo al server di rallentare i tempi di risposta e dipendendo solo dall'efficienza del proxy. Le richieste vengono distribuite tra diversi client che si alternano, riducendo il rischio che Mastodon chiuda la connessione e blocchi l'esecuzione del programma. Riutilizzando ogni client nel corso di alcuni minuti, possiamo mantenere attive le connessioni con interruzioni minime.

   Con il programma in esecuzione e completato con successo, la tabella *posts* contiene 200 status per quasi ogni utente nella tabella *users* (con il numero di status regolabile liberamente in qualsiasi momento). Alcuni utenti potrebbero non avere tutti i loro messaggi recuperati perché appartengono a istanze con strutture e tecnologie diverse, in tal caso vengono eliminati dalla tabella *users*.

   La tabella è strutturata come segue:

[4] Errore 429: Hai inviato troppe richieste in un determinato periodo di tempo. Riprova più tardi.

[5] Errore 503: Il server è temporaneamente incapace di elaborare la tua richiesta a causa di lavori di manutenzione o problemi di capacità. Riprova più tardi.

Table 4.2. posts

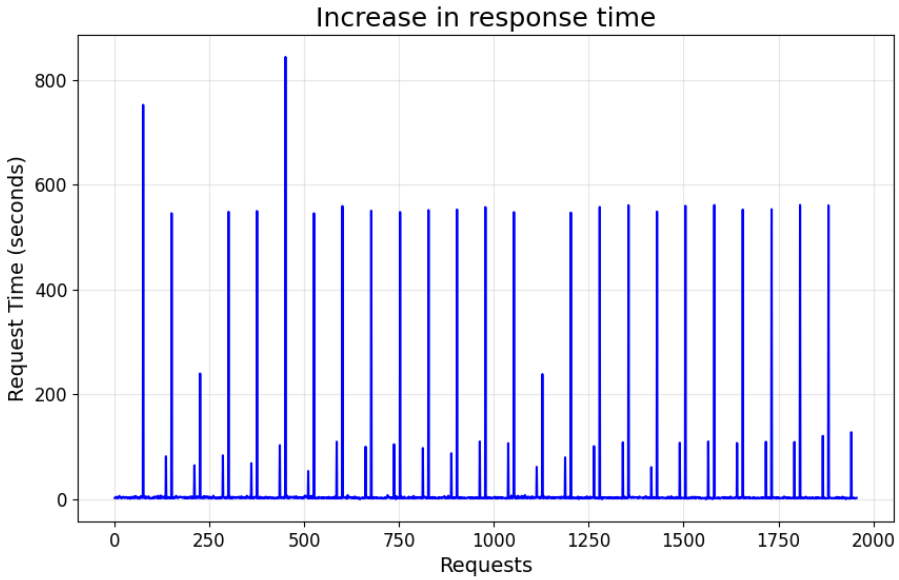
Field	Type	Null	Key
post_id	varchar(255)	NO	PRI
created_at	datetime	NO	
in_reply_to_id	varchar(255)	YES	
in_reply_to_account_id	varchar(255)	YES	
sensitive	tinyint(1)	YES	
spoiler_text	text	YES	
visibility	varchar(255)	YES	
language	varchar(10)	YES	
uri	text	YES	
url	text	YES	
replies_count	int	YES	
reblogs_count	int	YES	
favourites_count	int	YES	
favourited	tinyint(1)	YES	
reblogged	tinyint(1)	YES	
muted	tinyint(1)	YES	
bookmarked	tinyint(1)	YES	
pinned	tinyint(1)	YES	
content	text	YES	
media_attachments	json	YES	
account_id	varchar(255)	NO	PRI
account_username	varchar(255)	NO	
account_display_name	varchar(255)	YES	
account_url	varchar(255)	YES	
reblog_id	varchar(255)	YES	
reblog_content	text	YES	
reblogged_from_account	varchar(255)	YES	

La selezione di questi campi si basa su un'analisi del lavoro precedente e della letteratura esistente, che ha identificato questi temi come cruciali per la nostra analisi, in particolare per la rilevazione dei bot sui social media (rilevamento dei bot sociali). Esaminando la struttura del nome utente su Mastodon e altre caratteristiche rilevanti, abbiamo confermato che questi dati sono fondamentali per i nostri obiettivi di ricerca.

La tabella degli utenti include sia i bot dichiarati che gli utenti reali e, di conseguenza, la tabella dei post contiene status appartenenti a entrambe le categorie. Questa distinzione è fondamentale per l'analisi, poiché ci consente di studiare le differenze di comportamento tra queste due categorie.

Esaminiamo l'efficienza del programma quando vengono impiegate diverse tecniche.

1. **Effetto di *Time.Sleep()* sui Tempi di Richiesta**
   Il grafico illustra come l'efficienza temporale del programma sia influenzata dall'implementazione di *time.sleep()* per mettere in pausa l'esecuzione quando si incontra l'errore 429. Sfortunatamente, quando un singolo client effettua un alto volume di richieste, il server potrebbe eventualmente terminare la connessione dopo che sono stati interrogati centinaia o migliaia di utenti. In una sessione singola, interrogare 300 utenti ha richiesto al programma 3428 secondi, ovvero circa 57 minuti, prima che il server interrompesse la connessione. Ogni punto sul grafico rappresenta il tempo necessario per eseguire 5 richieste per interrogare un singolo utente e recuperare i suoi ultimi 200 post. Ogni richiesta richiede pochi centesimi di secondo fino a pochi secondi, tuttavia ci sono alcuni picchi anomali: quelli più bassi sono causati dal *time.sleep* per raggiungere il tempo di reset, mentre quelli più alti sono causati dall'**abbandono della connessione**, in questo caso viene gestito ripetendo *time.sleep* fino a quando la connessione non riprende a funzionare.



2. **Effetto della *Rotazione dei Proxy* sui Tempi di Richiesta**
   Il grafico illustra come l'efficienza temporale del programma sia influenzata dalla rotazione di otto proxy e del nostro indirizzo IP. In una singola sessione, interrogare 300 utenti ha richiesto al programma 2646 secondi, ovvero circa 44 minuti. Teoricamente, ogni client può interrogare fino a 60 utenti ed eseguire un totale di 1.500 richieste per recuperare 60.000 post, tuttavia, come mostrato nei grafici, ogni picco negativo rappresenta una sostituzione di proxy, che sono uno strumento molto insicuro e poco affidabile, tanto che non funzionano correttamente anche se precedentemente testati (proxy numero 1, 2, 3, 5, 8) oppure sono molto lenti (proxy numero 4, 6, 7). Tuttavia, con questa tecnica, il programma riesce a mantenere un accesso consistente, aumentare l'anonimato

e prevenire le restrizioni lato server che potrebbero derivare da troppe richieste provenienti da un singolo indirizzo IP.

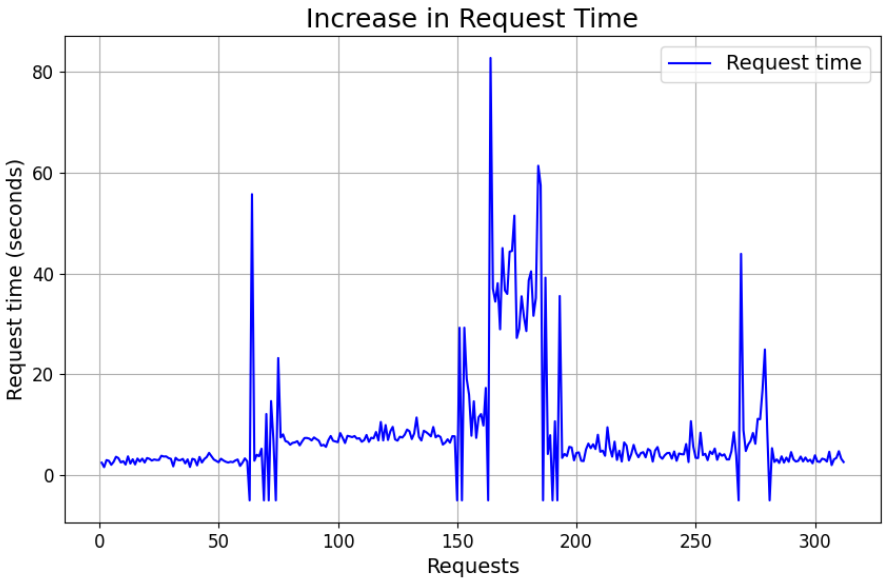


Figure 2

3. **Effetto del *Compito Asincrono* sui Tempi di Richiesta**

   Per massimizzare le prestazioni del client, possiamo eseguire richieste asincrone che interrogano diverse porzioni di utenti nella tabella *users*. Generiamo tre compiti che eseguono simultaneamente richieste al server Mastodon, ciascuno interrogando 100 utenti, per un totale di 300 utenti e 1.500 richieste. Sfortunatamente, a causa del limite di richiesta imposto, il client incontra l'errore **429** e, per risolvere questo problema, possiamo implementare *time.sleep* come descritto sopra. La mediana del tempo di richiesta è di 2,7 secondi e i picchi mostrati nel grafico sono causati da *time.sleep*, con un tempo di esecuzione complessivo di 2783 secondi, ovvero 46 minuti. Il seguente grafico mostra le prestazioni in secondi di ciascun compito.

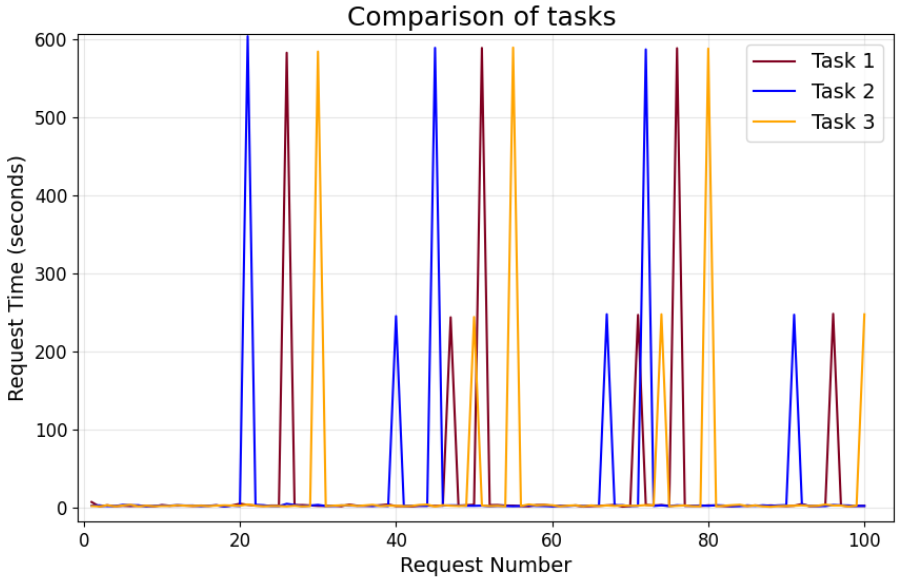


Figure 3

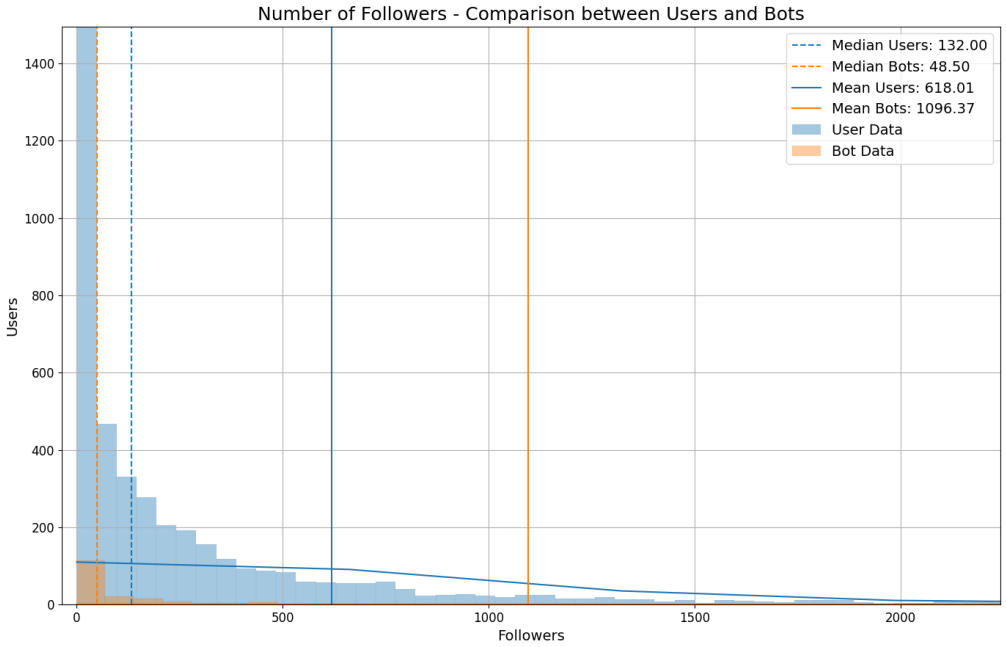
4. **Effetto della *Rotazione Asincrona dei Proxy* sui Tempi di Richiesta**

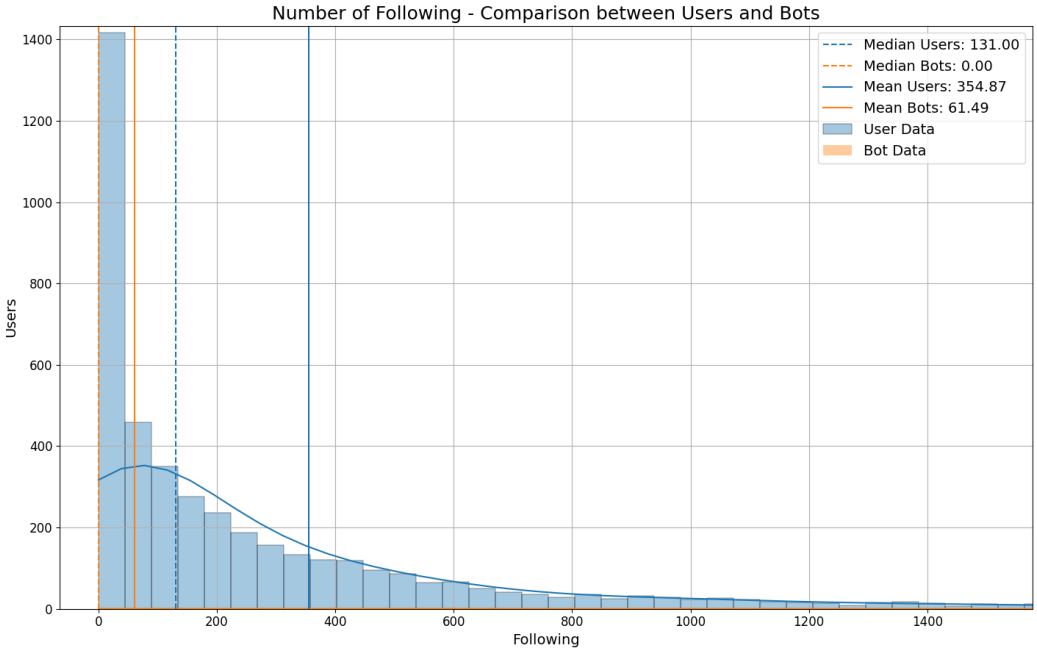
   Purtroppo, i proxy gratuiti tendono a essere instabili, lenti e sovraccarichi, causando numerosi problemi di rete o timeout, che portano a frequenti errori di connessione come **httpx.ProxyError** o **httpx.NetworkError**. Per questo motivo, non siamo riusciti a trovare abbastanza proxy per concludere l'esecuzione, anche se lo script funziona.
   In una singola sessione, interrogare 300 utenti dovrebbe richiedere al programma un terzo del tempo rispetto all'approccio precedente, grazie all'utilizzo simultaneo di tre compiti. Ogni compito interroga 100 utenti, avendo a disposizione una lista di diversi proxy gratuiti che vengono scambiati quando si supera il limite di richieste.

4.3 Euristic Bot Detection

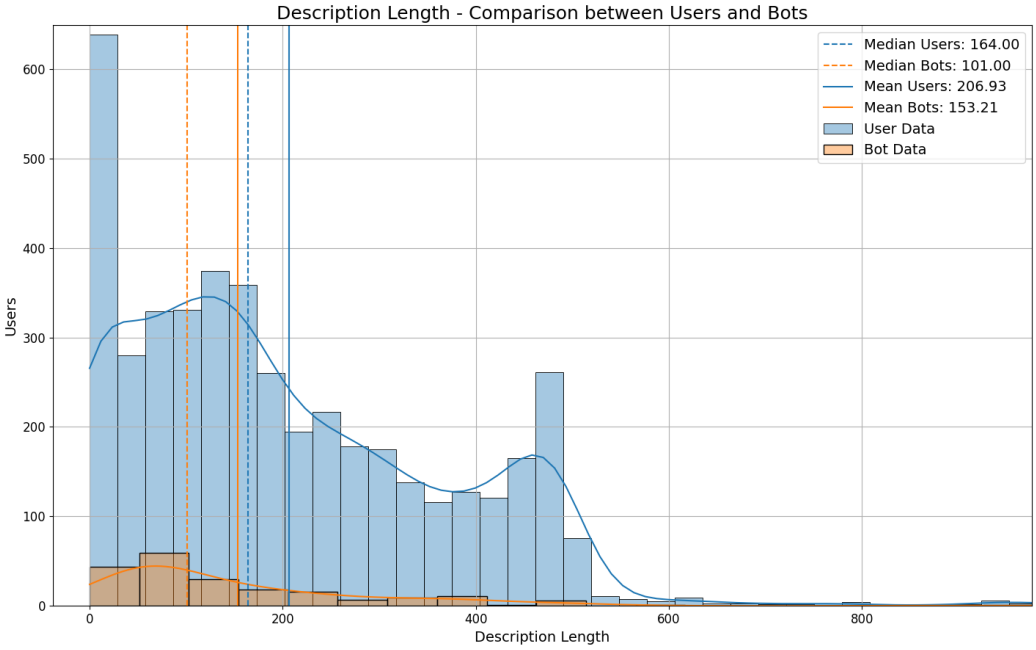
Le sezioni seguenti presentano un confronto dettagliato tra utenti reali e bot appartenenti a un dataset che conta circa 4600 utenti. Basato sull'analisi delle principali differenze, come *follower*, *following*, *lunghezza della descrizione*, *caratteristiche del nome utente* e *numero di stati*, le principali conclusioni sono le seguenti:

1. **Follower e Following:**
   Gli utenti il cui conteggio di follower e following è simile alle mediane osservate per i bot verranno classificati come bot. Questa combinazione di valori mediani funge da indicatore chiave per la classificazione. Il primo grafico mostra come i bot tendano ad essere seguiti da molte meno persone rispetto agli utenti reali, circa alcune decine, a differenza degli utenti reali che hanno un seguito maggiore. Mentre il secondo grafico mostra che il numero di utenti seguiti dai bot è minimo e che include altri bot.

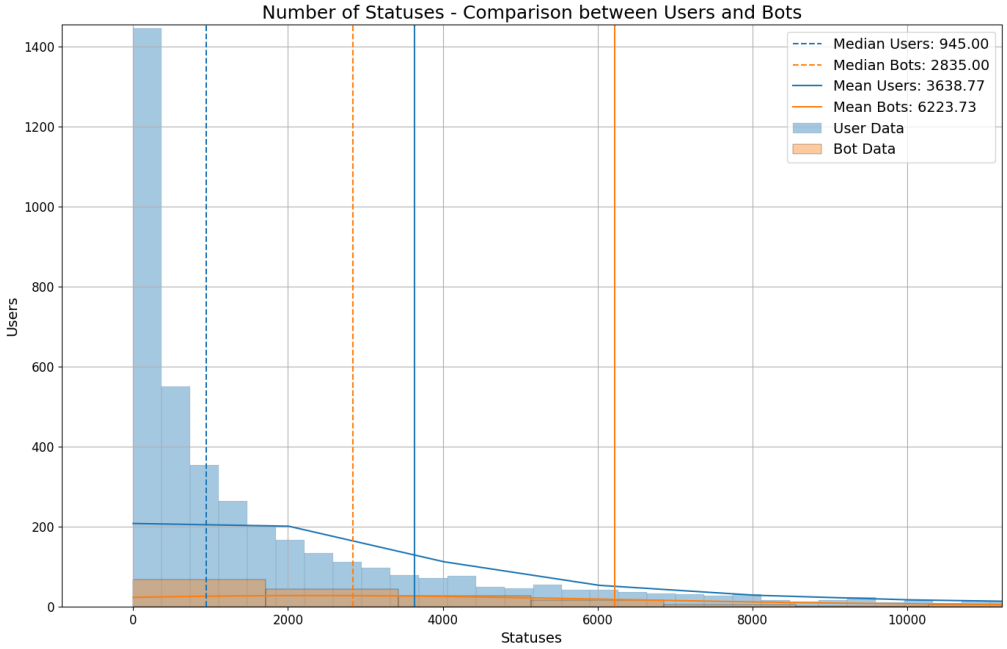




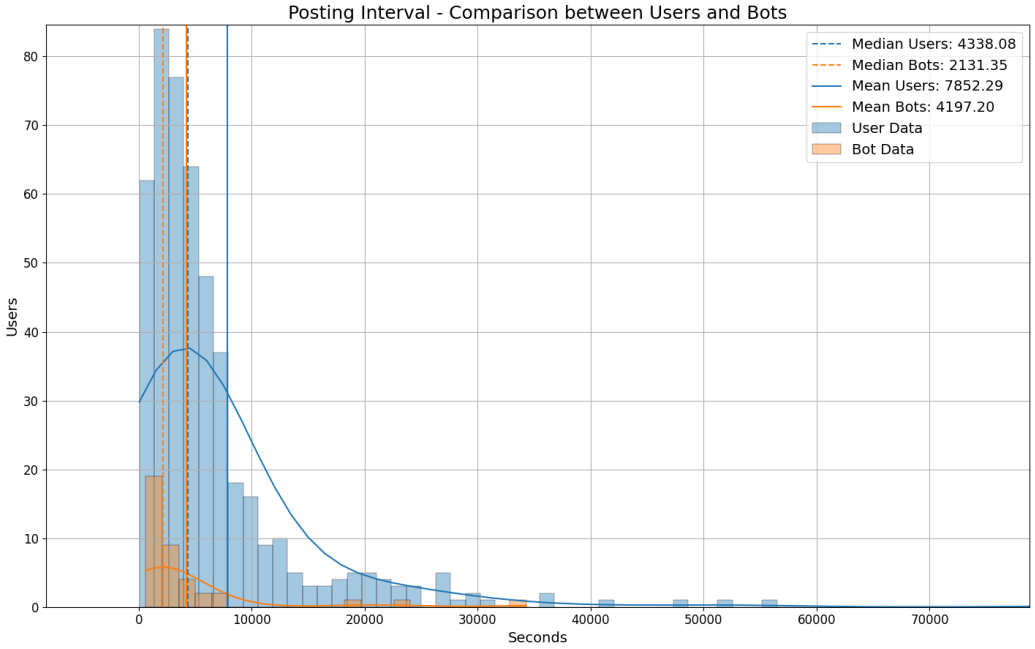
2. **Lunghezza della Descrizione e Link:**
   Gli utenti la cui lunghezza della descrizione è vicina alla lunghezza media delle descrizioni dei bot, specialmente se la descrizione contiene un link, verranno classificati come bot. Questo evidenzia la tendenza dei bot ad includere link nei loro profili, a differenza degli utenti reali che scrivono sulla loro vita e sui loro hobby.



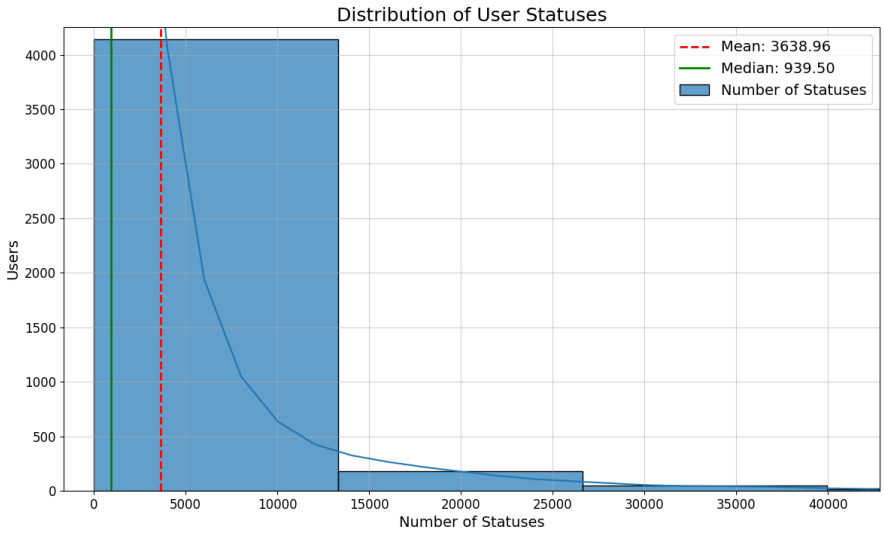
3. **Numero di Stati:**
   Gli utenti con un numero di post approssimativamente uguale alla mediana del numero di stati dei bot saranno anch'essi classificati come bot. Questo è in linea con il comportamento tipico di pubblicazione osservato negli account automatizzati. La mediana del numero di stati degli utenti dimostra che gli utenti pubblicano meno frequentemente rispetto ai bot, poiché adottano un comportamento schematico e algoritmico.



4. **Pubblicazione ad Alta Frequenza:**
   Gli utenti che pubblicano con una frequenza molto alta, approssimativamente ogni 1500 secondi, saranno anch'essi classificati come bot. Questo comportamento suggerisce una programmazione automatizzata piuttosto che un'interazione umana. La pubblicazione ad alta frequenza è un comportamento comune tra i software automatizzati a causa di spam, promozioni e altri motivi.



5. **Attività Anomala:**
   Il grafico mostra che 41000/4600 utenti hanno meno di 15.000 post e altri ancora meno di 28.000. Gli utenti con un numero eccezionalmente alto di post saranno contrassegnati come bot. Precisamente, coloro che hanno un numero di post molto superiore ai 30.000 stati. Questi schemi di attività estremi spesso distinguono gli account automatizzati dal comportamento umano.



Applicando questi criteri, il programma sfrutta le mediane statistiche su più attributi per identificare gli utenti sospetti con una probabilità elevata di essere bot.

L'obiettivo finale del programma è automatizzare il rilevamento degli account sospetti su Mastodon, utilizzando modelli statistici per confrontare il comportamento degli utenti con quello tipico degli account bot. Al termine del processo, il programma genera un elenco di utenti sospetti, identificando quelli il cui comportamento, basato su determinati parametri, è simile a quello di un bot. Con il dataset precedentemente generato, abbiamo individuato un gruppo di utenti sospetti. Su 4619 utenti, 204 sono bot e 4415 sono utenti reali. Tra questi, 118 utenti sono stati contrassegnati come sospetti.

Conclusioni

Il lavoro di sviluppo di MastoAnalyzer è stato un viaggio affascinante e impegnativo nel mondo del Fediverse e nelle sfide poste dal rilevamento dei social bot. Questo studio ha avuto come obiettivo la creazione di uno strumento per monitorare e comprendere le interazioni all'interno del Fediverse, un'area della rete ancora largamente inesplorata, affrontando nuove complessità e scoprendo straordinarie opportunità.

Un altro aspetto cruciale è stato affrontare il fenomeno dei social bot, che possono influenzare le discussioni pubbliche, diffondere disinformazione o addirittura manipolare le emozioni delle persone. MastoAnalyzer dimostra che combinando diverse tecniche analitiche, è possibile rilevare i bot con buona precisione, anche in ambienti decentralizzati come il Fediverse.

Purtroppo, i bot continuano a evolversi, diventando sempre più sofisticati e richiedendo un monitoraggio continuo di volumi di dati sempre più grandi e complessi. Tuttavia, il lavoro svolto ha gettato solide basi per affrontare queste difficoltà. MastoAnalyzer è progettato per essere uno strumento versatile e utile non solo per i ricercatori, ma anche per le comunità che popolano il Fediverse. I risultati ottenuti dimostrano che questo approccio funziona e che possiamo migliorare il modo in cui analizziamo le piattaforme decentralizzate.

La struttura di MastoAnalyzer è progettata per analizzare i dati in modo efficiente e mirato, rispettando la privacy degli utenti raccogliendo solo dati pubblicamente disponibili tramite API. Questi dati vengono poi elaborati utilizzando algoritmi di rilevamento comportamentale, che possono essere visualizzati graficamente tramite la libreria matplotlib.

Per comprendere meglio le dinamiche tra bot e utenti umani nel contesto del Fediverse, è stato essenziale analizzare i dati rappresentati nei grafici prodotti da MastoAnalyzer. Possiamo dimostrare che una delle differenze più evidenti riguarda i modelli comportamentali: i bot interagiscono con maggiore regolarità e ripetitività, mentre gli utenti umani sono caratterizzati da maggiore variabilità sia temporale che nei contenuti. Infatti, i grafici relativi alla distribuzione temporale dei post hanno messo in evidenza che i bot pubblicano a intervalli regolari, indipendentemente dal contenuto o dal contesto, a differenza degli utenti umani che seguono modelli meno prevedibili basati sul tempo, eventi o emozioni personali.

Un altro aspetto che caratterizza i bot è la loro tendenza a formare cluster isolati o ad interagire prevalentemente con altri bot, generando una rete meno densa e più frammentata. Gli utenti umani, invece, partecipano a conversazioni più diverse e contribuiscono a reti più complesse e interconnesse. Questi dati non solo aiutano

a identificare i bot con maggiore precisione, ma evidenziano anche l'importanza di promuovere reti autentiche e dinamiche all'interno del Fediverse.

MastoAnalyzer è uno strumento modulare, che consente di aggiungere nuove funzionalità o sostituire componenti senza influire sull'intero sistema. È anche adattabile ad altre reti sociali all'interno del Fediverse o anche a quelle centralizzate.

In futuro, potremmo potenziare le capacità di MastoAnalyzer integrando tecnologie più avanzate, come ad esempio:

- **Modelli di intelligenza artificiale** per migliorare ulteriormente il rilevamento dei bot ed espandere l'analisi ad altri aspetti del comportamento online, come l'impatto della disinformazione o il rispetto della privacy.
- **Analisi cross-platform** per raccogliere e analizzare dati provenienti da diverse piattaforme del Fediverse (es. PeerTube, Pixelfed).
- **Rilevamento di attacchi coordinati** per identificare attacchi informatici o campagne di disinformazione pianificate tramite reti di bot.
- **Grafici temporali avanzati**: per rappresentare l'evoluzione delle interazioni tra bot e umani nel tempo.

Ringraziamenti

Come ciliegina sulla torta vorrei ringraziare tutti coloro che hanno condiviso parte della loro energia con me.
Ai miei amici dell'università, figure fondamentali che hanno reso questo percorso speciale: con voi custodirò per sempre ricordi preziosi, dai viaggi in treno alle serate su Discord, dallo studio condiviso agli esami affrontati insieme. Senza ognuno di voi, raggiungere questo traguardo non sarebbe stato così semplice. Grazie di cuore!
Ai miei più cari amici, che sono stati al mio fianco ancor prima di intraprendere questo percorso e che hanno seguito ogni mio passo, sempre pronti a tifare per me. Il vostro supporto è stato un dono inestimabile.
Alla mia famiglia che con il suo sostegno mi hanno dato la possibilità di dedicarmi pienamente allo studio, rasserenandomi nei momenti di tensione e spronandomi quando ne avevo più bisogno.
Infine a me stesso, per non aver mai mollato e per la promessa che mai lo farò.

Bibliography

[1] G. Gow. "Turning to Alternative Social Media". In: *The SAGE Handbook of Social Media Research Methods*. SAGE Publications Ltd., 2022, pp. 568–580. DOI: 10.4135/9781529782943. URL: https://doi.org/10.4135/9781529782943.

[2] Aymeric Mansoux and Roel Roscam Abbing. "Seven Theses on the Fediverse and the Becoming of FLOSS". In: *The Eternal Network*. Ed. by Kristoffer Gansing and Inga Luchs. Institute of Network Cultures, 2020, pp. 124–140. URL: http://journals.sagepub.com/doi/10.1177/2056305115604338.

[3] C. Cerisara et al. "Multi-task dialog act and sentiment recognition on Mastodon". In: *Proceedings of the COLING Conference*. 2018, pp. 745–754. URL: https://www.aclweb.org/anthology/C18-1063/.

[4] J. Trienes, A. T. Cano, and D. Hiemstra. "Recommending users: whom to follow on federated social networks". In: *CoRR* arXiv: 1811.09292 (2018).

[5] M. Zignani, S. Gaito, and G. P. Rossi. "Follow the "Mastodon": structure and evolution of a decentralized online social network". In: *Proceedings of the International Conference on Web and Social Media (ICWSM)*. 2018, pp. 541–551.

[6] A. Raman et al. "Challenges in the decentralised web: the Mastodon case". In: *Proceedings of the ACM IMC Conference*. 2019, pp. 217–229. DOI: 10.1145/3355369.3355572.

[7] D. Zulli, M. Liu, and R. Gehl. "Rethinking the "social" in "social media": insights into topology, abstraction, and scale on the Mastodon social network". In: *New Media & Society* 22.7 (2020), pp. 1188–1205.

[8] M. Zignani et al. "The footprints of a "Mastodon": how a decentralized architecture influences online social relationships". In: *Proceedings of the IEEE INFOCOM Workshops*. 2019, pp. 472–477. DOI: 10.1109/INFCOMW.2019.8845221. URL: https://doi.org/10.1109/INFCOMW.2019.8845221.

[9] O. Varol et al. "Online human–bot interactions: detection, estimation, and characterization". In: *Proceedings of the International Conference on Web and Social Media (ICWSM)*. 2017, pp. 280–289.

[10] R. W. Gehl and D. Zulli. "The digital covenant: Non-centralized platform governance on the mastodon social network". In: *Information, Communication & Society* 0.0 (2023), pp. 1–17. DOI: 10.1080/1369118X.2022.2147400. URL: https://doi.org/10.1080/1369118X.2022.2147400.

[11] World Wide Web Consortium (W3C). *ActivityPub*. https://www.w3.org/TR/activitypub/. Accessed: 2024-11-09. 2017.

[12] Emilio Ferrara et al. "The rise of social bots". In: *Communications of the ACM* 59.7 (2016). Accessed: 1 November 2024, pp. 96–104. DOI: 10.1145/2818717.

[13] Alan M. Turing. "Computing machinery and intelligence". In: *Mind* 49.236 (1950), pp. 433–460.

[14] *Loebner Prize*. http://www.loebner.net/Prizef/loebner-prize.html. Accessed: 1 November 2024.

[15] Emilio Ferrara. "What types of COVID-19 conspiracies are populated by Twitter bots?" In: *First Monday* 25.6 (2020). Accessed: 1 November 2024. DOI: 10.5210/fm.v25i6.10633.

[16] Bjarke Mønsted et al. "Evidence of complex contagion of information in social media: An experiment using Twitter bots". In: *PLoS ONE* 12.9 (2017). Accessed: 1 November 2024, e0184148. DOI: 10.1371/journal.pone.0184148.

[17] Petter Bae Brandtzaeg and Asbjørn Følstad. "Why people use chatbots". In: *Internet Science, Lecture Notes in Computer Science*. Ed. by Ioannis Kompatsiaris et al. Vol. 10673. Accessed: 1 November 2024. Cham, Switzerland: Springer, 2017, pp. 377–392. DOI: 10.1007/978-3-319-70284-1_30.

[18] Zi Chu et al. "Who is tweeting on Twitter: Human, bot, or cyborg?" In: *Proceedings of the 26th Annual Computer Security Applications Conference*. Accessed: 1 November 2024. 2010, pp. 21–30. DOI: 10.1145/1920261.1920265.

[19] Howard C. H. Chang and Emilio Ferrara. "Comparative analysis of social bots and humans during the COVID-19 pandemic". In: *Journal of Computational Social Science* 5 (2022). Accessed: 1 November 2024, pp. 1409–1425. DOI: 10.1007/s42001-022-00173-9.

[20] Tim Hwang, Ian Pearce, and Max Nanis. "Socialbots: Voices from the fronts". In: *Interactions* 19.2 (2012). Accessed: 1 November 2024, pp. 38–45. DOI: 10.1145/2090150.2090161.

[21] Irene Pozzana and Emilio Ferrara. "Measuring bot and human behavioral dynamics". In: *Frontiers in Physics* 8 (2020). Accessed: 1 November 2024. DOI: 10.3389/fphy.2020.00125.

[22] Jahan Ratkiewicz et al. "Truthy: Mapping the spread of astroturf in microblog streams". In: *Proceedings of the 20th International Conference Companion on World Wide Web*. Accessed: 1 November 2024. 2011, pp. 249–252. DOI: 10.1145/1963192.1963301.

[23] Kyumin Lee, James Caverlee, and Steve Webb. "The social honeypot project: Protecting online communities from spammers". In: *Proceedings of the 19th International Conference on World Wide Web*. Accessed: 1 November 2024. 2010, pp. 1139–1140. DOI: 10.1145/1772690.1772843.

[24] Gianluca Stringhini, Christopher Kruegel, and Giovanni Vigna. "Detecting spammers on social networks". In: *Proceedings of the 26th Annual Computer Security Applications Conference*. Accessed: 1 November 2024. 2010, pp. 1–9. DOI: 10.1145/1920261.1920263.

[25] V.S. Subrahmanian et al. “The DARPA Twitter bot challenge”. In: *Computer* 49.6 (2016). Accessed: 1 November 2024, pp. 38–46. DOI: 10.1109/MC.2016.183.

[26] Lei Wu et al. “Different absorption from the same sharing: Sifted multi-task learning for fake news detection”. In: *arXiv* 1909.01720 (2019). Accessed: 1 November 2024.

[27] Christopher Besel, Juan Echeverria, and Sijing Zhou. “Full cycle analysis of a large-scale botnet attack on Twitter”. In: *2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*. Accessed: 1 November 2024. 2018. DOI: 10.1109/ASONAM.2018.8508708.

[28] Diego Pacheco, Alessandro Flammini, and Filippo Menczer. “Unveiling coordinated groups behind white helmets disinformation”. In: *WWW ’20: Companion Proceedings of the Web Conference 2020*. Accessed: 1 November 2024. 2020, pp. 611–616. DOI: 10.1145/3366424.3385775.

[29] Diego Pacheco et al. “Uncovering coordinated networks on social media: Methods and case studies”. In: *Proceedings of the International AAAI Conference on Web and Social Media*. Vol. 15. Accessed: 1 November 2024. 2021, pp. 455–466. DOI: 10.1609/icwsm.v15i1.18075.

[30] Karishma Sharma et al. “Identifying coordinated accounts on social media through hidden influence and group behaviours”. In: *KDD ’21: Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. Accessed: 1 November 2024. 2021, pp. 1441–1451. DOI: 10.1145/3447548.3467391.

[31] Andreas Rössler et al. “Faceforensics++: Learning to detect manipulated facial images”. In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. Accessed: 1 November 2024. 2019. DOI: 10.1109/ICCV.2019.00009.

[32] Zhilin Yang et al. “XLNet: Generalized autoregressive pretraining for language understanding”. In: *NIPS ’19: Proceedings of the 33rd International Conference on Neural Information Processing Systems*. 2019, pp. 5753–5763.

[33] Kai Shu et al. “FakeNewsNet: A data repository with news content, social context and spatial-temporal information for studying fake news on social media”. In: *arXiv* 1809.01286 (2018). Accessed: 1 November 2024.

[34] Gang Wang et al. “Social Turing tests: Crowdsourcing sybil detection”. In: *NDSS*. The Internet Society, 2013.

[35] Paul Heymann, Georgia Koutrika, and Hector Garcia-Molina. “Fighting spam on social web sites: A survey of approaches and future challenges”. In: *Internet Computing* 11.6 (2007), pp. 36–45.

[36] Stefano Cresci et al. “The paradigm-shift of social spambots: Evidence, theories, and tools for the arms race”. In: *WWW ’17 Companion: Proceedings of the 26th International Conference on World Wide Web Companion*. Accessed: 1 November 2024. 2017, pp. 963–972. DOI: 10.1145/3041021.3055135.

[37] Juan Echeverria and Shi Zhou. "Discovery, retrieval, and analysis of the 'Star Wars' botnet in Twitter". In: *ASONAM '17: Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2017*. Accessed: 1 November 2024. 2017, pp. 1–8. DOI: 10.1145/3110025.3110074.

[38] Kai-Cheng Yang, Emilio Ferrara, and Filippo Menczer. "Botometer 101: Social bot practicum for computational social scientists". In: *Journal of Computational Social Science* 5.2 (2022). Accessed: 1 November 2024, pp. 1511–1528. DOI: 10.1007/s42001-022-00177-5.

[39] Emilio Ferrara. "Manipulation and abuse on social media". In: *ACM SIGWEB Newsletter* 2015 (Spring 2015). Accessed: 1 June 2023, pp. 1–9. DOI: 10.1145/2749279.2749283.

[40] Alessandro Bessi and Emilio Ferrara. "Social bots distort the 2016 U.S. Presidential election online discussion". In: *First Monday* 21.11 (2016). Accessed: 1 June 2023. DOI: 10.5210/fm.v21i11.7090. URL: https://firstmonday.org/ojs/index.php/fm/article/view/7090/5653.

[41] Adam Badawy, Emilio Ferrara, and Kristina Lerman. "Analyzing the digital traces of political manipulation: The 2016 Russian interference Twitter campaign". In: *2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*. Accessed: 1 June 2023. 2018, pp. 258–265. DOI: 10.1109/ASONAM.2018.8508646.

[42] Hsin-Chang Han Chang et al. "Social bots and social media manipulation in 2020: The year in review". In: *Handbook of Computational Social Science. Volume 1: Theory, Case Studies and Ethics*. Ed. by Ulrich Engel et al. Accessed: 1 June 2023. London: Routledge, 2021. DOI: 10.4324/9781003024583.

[43] Robert Gorwa and Douglas Guilbeault. "Unpacking the social media bot: A typology to guide research and policy". In: *Policy & Internet* 12.2 (2020). Accessed: 1 June 2023, pp. 225–248. DOI: 10.1002/poi3.184.

[44] Philip N. Howard and Bence Kollanyi. "Bots, #StrongerIn, and #Brexit: Computational propaganda during the UK-EU referendum". In: *arXiv* 1606.06356 (2016). Accessed: 1 June 2023. DOI: 10.48550/arXiv.1606.06356.

[45] Alaa Addawood et al. "Linguistic cues to deception: Identifying political trolls on social media". In: *Proceedings of the International AAAI Conference on Web and Social Media*. Vol. 13. Accessed: 1 June 2023. 2019, pp. 15–25. URL: https://ojs.aaai.org/index.php/ICWSM/article/view/3205.

[46] Luca Luceri et al. "Red bots do it better: Comparative analysis of social bot partisan behavior". In: *WWW '19: Companion Proceedings of The 2019 World Wide Web Conference*. Accessed: 1 June 2023. 2019, pp. 1007–1012. DOI: 10.1145/3308560.3316735.

[47] Massimo Stella, Emilio Ferrara, and Manlio De Domenico. "Bots increase exposure to negative and inflammatory content in online social systems". In: *Proceedings of the National Academy of Sciences* 115.49 (2018). Accessed: 1 June 2023, pp. 12435–12440. DOI: 10.1073/pnas.1803470115.

[48] Natali Ruchansky, Sungyong Seo, and Yan Liu. "CSI: A hybrid deep model for fake news detection". In: *CIKM '17: Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. Accessed: 1 June 2023. 2017, pp. 797–806. DOI: 10.1145/3132847.3132877.

[49] Soroush Vosoughi, Deb Roy, and Sinan Aral. "The spread of true and false news online". In: *Science* 359.6380 (2018). Accessed: 1 June 2023, pp. 1146–1151. DOI: 10.1126/science.aap95.

[50] Luca Nizzoli et al. "Charting the landscape of online cryptocurrency manipulation". In: *IEEE Access* 8 (2020). Accessed: 1 June 2023, pp. 113230–113245. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=9120022.

[51] Emilio Ferrara. "Twitter spam and false accounts prevalence, detection and characterization: A survey". In: *First Monday* 27.12 (2022). Accessed: 1 June 2023. DOI: 10.5210/fm.v27i12.12872. URL: https://firstmonday.org/ojs/index.php/fm/article/view/12872/10749.