Università Politecnica delle Marche

Facoltà di Ingegneria

Dipartimento di Ingegneria dell'Informazione

Corso di Laurea Magistrale in Ingegneria Informatica e dell'Automazione

---

Tesi di Laurea

Studio del ciclo di vita delle challenge di Tik Tok ed estrazione di time pattern per caratterizzarle

Investigating the lifecycle of Tik Tok challenges and extraction of time patterns to characterize them

Relatore	Candidata
Prof. Domenico Ursino	Silvia Cecchini

Anno Accademico 2020-2021

Indice

Introduzione

Con l'avvento di Internet e del Web 2.0, lo studio delle reti sociali (o Social Network) ha acquistato sempre più importanza sociale, economica e politica. Grazie a siti come Facebook, Instagram, Twitter e TikTok, l'espressione "social network" è entrata nel vocabolario comune e sta ad indicare quei siti web nei quali si possono contattare amici e conoscenti attraverso computer o dispositivi mobili (quali smartphone e tabletPC). Il significato originale del termine si riferisce invece ad "un qualsiasi gruppo di persone connesse tra loro da diversi legami sociali, che vanno dalla conoscenza casuale, ai rapporti di lavoro, fino ad arrivare ai vincoli familiari". Molteplici sono, infatti, le pubblicazioni riguardanti l'argomento in questione sin dagli anni '50, riguardanti vari aspetti matematico-scientifici, ingegneristici, psicologici ed economico-sociologici.

Grazie allo studio delle reti sociali che si creano all'interno dei social network, è possibile attuare un gran numero di analisi, a partire dall'evoluzione di un'azione svolta dagli utenti, per arrivare all'individuazione di cluster di utenti che interagiscono tra loro, permettendo la creazione di vere e proprie community.

Lo scopo di questa tesi è quello di analizzare l'evoluzione del ciclo di vita delle challenge di TikTok, mettendo in risalto le differenze che intercorrono tra una serie di challenge positive e una serie di challenge negative selezionate. Si è partiti estrapolando dati dalla piattaforma di TikTok mediante un'API non ufficiale, seguita da un'attività di ETL, manipolazione dati e creazione di reti mediante librerie di Python, come Pandas, Matplotlib e NetworkX. Infine, si è posta l'attenzione sull'evoluzione del ciclo di vita di ciascuna challenge e della loro media, in modo da analizzarne gli andamenti ed eventuali pattern ricorrenti, per poi trarre le opportune conclusioni.

La tesi è strutturata come di seguito specificato. Il primo capitolo tratta la Big Data Analytics, analizzando diversi punti di vista ed approfondendo le parti fondamentali di questo settore scientifico così innovativo. Il secondo capitolo illustrerà la Social Network Analysis, entrando nello specifico su quello che riguarda la teoria dei grafi, le reti sociali e le community detection. All'interno del terzo capitolo si entrerà più nello specifico per ciò che riguarda il social network del momento, ovvero Tik Tok.

Dal quarto capitolo in poi ci si addentrerà maggiormente sul lavoro svolto, considerando il linguaggio utilizzato, la raccolta dati e l'attività di ETL. Nel quinto capitolo verranno elaborati i dati

ottenuti, connettendoli in grafi. All'interno del sesto capitolo, si analizzeranno gli andamenti delle challenge e il loro ciclo di vita, mettendo in evidenza le differenze tra gli andamenti delle challenge positive e quelli delle challenge negative. Nel settimo capitolo ci si concentrerà sull'estrazione di time pattern, mentre nell'ottavo capitolo si effettuerà un confronto con gli approcci correlati. Infine, nel nono capitolo verranno tratte delle conclusioni relative ai risultati ottenuti, con uno sguardo ad eventuali sviluppi futuri.

1. Big Data Analytics

1.1 Big Data

Il termine inglese "Big Data" sta ad indicare delle grandi masse di dati. Si tratta di una raccolta di dati talmente estesa da richiedere delle tecnologie in grado di estrarne il valore, analizzarli e gestirli. La definizione "big data" nasce dalla possibilità di acquisire una grandissima quantità di dati, poiché questi ultimi crescono continuamente, e questa cosa è dovuta alla crescita di dispositivi IoT (Internet of Things), così come le *smart car* in circolazione, ma anche l'utilizzo dei social network. La maggior parte delle azioni che effettuiamo ogni giorno produce dei dati, come ad esempio: effettuare una ricerca su Google, scattare una foto, inviare un messaggio vocale, pubblicare un tweet, e molto altro. I dati prodotti da queste azioni si possono raccogliere grazie all'esistenza di Internet, o semplicemente dell'IoT (Internet of Things). Ogni giorno vengono costantemente prodotti dati di ogni tipo; basti pensare ai dati degli utenti di un sito web, dei post sui social, dell'uso di un'applicazione, degli apparecchi connessi alla rete, e così via. Per capire di che mole di dati si sta parlando, basti pensare che i big data sono misurati in zettabyte, ovvero in miliardi di terabyte, e che vengono raccolti da fonti eterogenee. Lo scopo è, quindi, quello di mettere in relazione un'enorme mole di dati eterogenei, strutturati e non strutturati, per scoprire i legami tra fenomeni diversi per poi prevedere quelli futuri.

1.1.1 Le 5 V dei Big Data

I Big Data sono, solitamente, disponibili con dei volumi enormi, prodotti ad una velocità estrema e si presentano destrutturati e con caratteristiche eterogenee. Spesso si sente parlare delle 3 V dei Big Data e cioè i tre principali fattori che li caratterizzano: Volume, Varietà e Velocità. Oltre alle 3 V, che hanno sempre caratterizzato questi dati dai primi anni 2000, ne sono state aggiunte altre 2 per definire come andrebbero utilizzati: Veridicità e Variabilità. Proprio per questo motivo si definiscono le seguenti 5 V dei Big Data:

- *Varietà*: con varietà si fa riferimento ai diversi tipi di dati disponibili, solitamente provenienti da fonti diverse: sistemi transazionali e gestionali aziendali, social network, siti web, open data e sensori. Si tratta di dati strutturati e non, che includono, anche, documenti di vario tipo (PDF, Word, Excel, txt, csv, e altri), commenti sulle piattaforme social e blog post.

- *Velocità*: una caratteristica peculiare di questi dati è la velocità con cui si rendono disponibili, e proprio per questo motivo sono necessari degli strumenti in grado di immagazzinarli in maniera corretta. La sfida delle aziende è, anche, quella di riuscire a raccogliere questi dati e analizzarli in tempo reale, per prendere decisioni di business in maniera tempestiva.
- *Veridicità*: i dati devono essere affidabili. L'attendibilità delle fonti è fondamentale, e quando si tratta di dati non è sempre detto che siano veritieri. Magari possono essere stati effettuati dei click per errore, oppure gli stessi dispositivi possono essere stati utilizzati da persone diverse. Bisogna sempre valutare con molta attenzione i Big Data; è necessario analizzarli accuratamente e verificarne la veridicità.
- *Variabilità*: è necessario saper contestualizzare i dati, poiché l'interpretazione di un dato può variare a seconda del contesto in cui viene raccolto e analizzato. La variabilità può esserci anche in base al momento in cui viene effettuata l'analisi, e per questo motivo è fondamentale l'analisi in tempo reale (velocità).
- *Volume*: questo parametro indica la quantità di Big Data che ogni giorno viene generata. Si tratta di una massa gigantesca di dati che non si può raccogliere con le tecnologie tradizionali. Il volume dei dati è in costante crescita a causa delle attività effettuate nella vita quotidiana attraverso apparecchi collegati ad Internet. Possiamo prendere come esempio i Big Data delle transazioni bancarie e i movimenti sui mercati finanziari, che raggiungono un volume talmente alto che i Database Management System tradizionali non riescono a gestire. Si può parlare di Big Data quando ci sono volumi che crescono più del 50% all'anno o quando si supera la soglia dei 50 terabyte.

1.2 Dai dati alla saggezza

Con il passare del tempo è naturale che nelle aziende si creino nuove procedure, si risolvano problemi, si trovino soluzioni, si prendano decisioni per i processi organizzativi, ma non è raro che in alcune il tracciamento delle novità e delle modifiche non sia previsto o sia disorganizzato. In particolare, le decisioni *on the go* sono le più rischiose per la salute organizzativa dell'azienda, dato che, talvolta, nascono senza aver analizzato il pregresso della stessa. È invece fortemente necessario raccogliere e organizzare le informazioni, affinché siano utili e gestibili. In base alla quantità di dati da elaborare, il processo complessivo di gestione delle informazioni può essere relativamente lento. Possiamo suddividerlo in più step o, se vogliamo, livelli di conoscenza:

- *Dati*: fatti e osservazioni. All'inizio i dati sono frammentari o disorganizzati; c'è da valutare chi ne detiene la conoscenza e quali sono gli argomenti. Si valuta, quindi, se sono già esistenti delle procedure interne, dei manuali (anche in bozza o obsoleti), delle informazioni note da parte del personale, degli appunti sparsi, della documentazione riservata e/o pubblica (come la conoscenza di quali infrastrutture sono in uso in azienda, se esiste un elenco di soluzioni in uso da parte dei tecnici, dei nominativi, riferimenti, contatti per le nostre attività, e molto altro ancora). Si deve raccogliere tutto ciò che non è inquadrato in un contesto e tutto ciò che potrebbe diventare tracciabile in seguito.
- *Informazioni*: si organizzano i dati. Sono stati raccolti i nostri dati frammentari e ora si devono organizzare in maniera ragionata. Si archiviano, si ordinano, si indicizzano, si registrano. Diversi tipi di software possono essere utili per ogni tipo di dato da tracciare. Alcuni esempi possono essere i software per Help Desk con sistemi di knowledge base, gli ERP (Enterprise Resource Planning) per gestire produzione e approvvigionamento, tenendo insieme le informazioni su clienti e fornitori; i CMS (Content Management System) per la gestione documentale condivisa, i CRM (Customer Relationship Management) per la gestione estesa dei contatti e i CASE (Computer-Aided Software Engineering) per la gestione del lifecycle del software.

- *Conoscenza*: le informazioni vengono applicate alla pratica. Dopo aver raccolto e organizzato le informazioni, queste sono state suddivise e registrate. Da questo momento in poi è necessario che il lavoro fatto non venga "riposto in un cassetto" e dimenticato. La conoscenza è data dalla messa in pratica di quanto è in nostro possesso a livello di informazione. È necessario elaborare le informazioni e applicarle nella pratica, tenendo a mente che tutte le nuove conoscenze sono, comunque, da tracciare, per registrare i cambiamenti e i nuovi dati acquisiti.
- *Saggezza*: esperienza e intuizione ne sono i cardini. La saggezza è immateriale. A differenza dei nostri libri e hard disk, essa è intangibile, ma la saggezza è il giudizio, la capacità di aumentare l'efficienza affinché le cose diventino efficaci. Essa aggiunge valore ed è unica e personale. La saggezza è l'esperienza, accompagna la conoscenza e permette di fare le scelte migliori.

I sopracitati dati (Data), informazioni (Information), conoscenza (Knowledge) e saggezza (Wisdom), compongono il sistema piramidale DIKW (Figura 1.1).

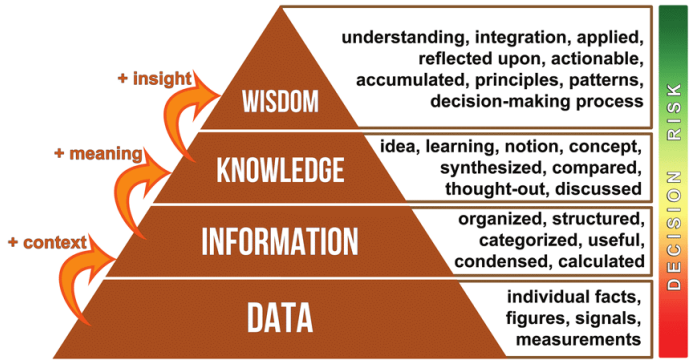


Figura 1.1: Piramide DIKW *Fonte: SEO Leader*

Uno dei casi più lampanti relativo alla necessità dell'uso intelligente delle informazioni è dato dai service desk (meglio noti come help desk), dato che sono tra i servizi esterni più colpiti dalla mancanza o disorganizzazione delle stesse. Le attività di Request Fulfillment, di Incident Management e di Problem Management devono, infatti, essere strettamente legate a un sistema di gestione della Knowledge Base (KB). La Knowledge Base è un contenitore di dati, un database, in cui è raccolta tutta una serie di informazioni relative a un determinato argomento. Le KB nascono con una base di informazioni già "note" e vengono costantemente mantenute aggiornate con le nuove versioni delle informazioni già esistenti. In pratica, nel tempo, i prodotti/servizi in uso forniti, vengono modificati, aggiornati e dismessi; assistiamo a cambiamenti costanti sia dell'oggetto del servizio, sia di chi può usufruirne; per questo il personale che interagisce con esso deve essere messo a conoscenza di ogni parte che lo può riguardare, deve sapere come reperire in breve tempo tutte le informazioni necessarie in merito all'erogazione del servizio: dai livelli di fornitura (SLA) ai dati dei clienti, dai riferimenti per le escalation ai contatti con i fornitori, etc. Ciò non toglie che, oltre al Service Operation, anche gli altri processi del service management devono essere supportati da un'adeguata documentazione.

Il giornalista Gilberto Corbellini, recensendo il libro "The book of why, the new science of causes and effect", che parla dell'approccio probabilistico all'Intelligenza Artificiale attuato dal matematico e filosofo Judea Pearl, afferma che i dati sono stupidi; il fattore più importante per incrementare la propria conoscenza è la capacità di contestualizzare e far transitare verso uno stato conoscitivo superiore i dati in nostro possesso.

1.3 Big Data Analytics

La definizione di big data analytics fa riferimento al processo che include la raccolta e l'analisi dei big data per ottenerne informazioni utili al business. Le tecniche di big data analytics consentono, infatti, di fornire alle aziende intuizioni originali, per esempio sulla situazione del mercato. D'altra parte offrono idee sul comportamento dei clienti, su come raffinare le strategie di customer experience, e così via. Per compiere le attività tese a fornire queste e tante altre informazioni preziose per migliorare l'attività dell'impresa sono necessari:

- Software: dai database e strumenti utili per acquisire ed elaborare informazioni agli applicativi dedicati per specifici processi aziendali.
- Servizi: per esempio, per personalizzare le tecnologie e integrarle con successo nei sistemi pre-esistenti.
- Risorse infrastrutturali: capacità di calcolo, storage, etc.

Impostare un progetto di analisi e gestione dei big data nella propria azienda significa affrontare molteplici aspetti. Non ci si può, naturalmente, limitare a quelli tecnologici; è necessario valutare le esigenze di business a cui si vuole rispondere. E porsi obiettivi precisi, coinvolgendo numerose competenze. Definire e implementare una strategia di big data analytics significa, infatti, avere la possibilità di trarre preziose informazioni per fare innovazione.

1.3.1 Metodologie di analisi

L'acquisizione e l'elaborazione dei big data ha richiesto la progettazione di nuove metodologie e tecnologie di raccolta e di analisi. In particolare, possiamo distinguere quattro tipi o metodologie di analisi dei big data:

- *Analisi Descrittiva*, ovvero le metodologie e le tecnologie utilizzate per descrivere la situazione attuale e quella passata dei processi aziendali o di progetti di business, rappresentando in modo sintetico e grafico gli indicatori di prestazione dell'attività.
- *Analisi Predittiva*, ovvero gli strumenti di analisi dei dati che aiutano a capire cosa potrebbe accadere nel futuro utilizzando tecniche matematiche, come la regressione e i modelli predittivi.
- *Analisi Prescrittiva*, impiegata per individuare soluzioni strategiche e operative efficaci.
- *Analisi Automatizzata*, che comprende gli strumenti che consentono di implementare autonomamente e in maniera automatizzata le azioni desiderate e secondo il risultato delle analisi che sono state condotte.

1.3.2 A cosa serve

I big data sono fondamentali per sviluppare modelli di business competitivi, innovativi e in grado di crescere e migliorare costantemente. Consentono alle aziende di raccogliere una gran quantità di preziosissime informazioni sull'andamento aziendale, sui risultati delle strategie adottate e delle operazioni attuate e sulle possibili previsioni future. Inoltre, permettono di ottenere informazioni sul comportamento di acquisto della clientela, sulle sue abitudini e sui suoi gusti, sulle reazioni a determinati avvenimenti, sulle caratteristiche e sulle consuetudini. Analizzare questi dati consente di assumere decisioni aziendali mirate con l'obiettivo di rintracciare nuovi potenziali clienti, di aumentare il numero dei clienti e delle vendite, di rendere più efficaci le strategie di fidelizzazione del cliente e di attuare efficaci strategie di *cross-selling*. E, ancora, le analisi dei big data consentono di pianificare le azioni di marketing in modo mirato e più efficace su ogni media; ne consentono, altresì, di studiare in maniera approfondita il target di riferimento, di segmentarlo e di capire come e quando raggiungerlo. Permettono di individuare i fattori psicologici e motivazionali che convincono le persone ad acquistare un prodotto o un servizio e di realizzare simulazioni predittive per capire come evolverà il mercato e il settore in cui l'azienda opera.

Negli ultimi anni sempre più aziende investono nell'analisi dei big data per sviluppare modelli di business efficaci e innovativi e per rendere più competitive le proprie strategie di vendita, di marketing e di comunicazione. Investire in big data, quindi, significa investire nel presente e nel futuro del proprio business e cogliere quest'opportunità è la chiave della riuscita e del successo nei mercati contemporanei.

1.3.3 Migliorare la strategia aziendale

Saper gestire ed effettuare un processo di Big Data Analytics può fare la differenza per il successo di un'azienda o startup. Coloro che hanno già investito in questo settore hanno potuto constatare degli impatti non indifferenti in termini di trasversalità e quantificabilità dell'utilizzo di questa tecnologia. Si parla in termini di trasversalità perché la Big Data Analytics ha un impatto su tutta la supply chain aziendale, dalle fasi iniziali a quelle finali. Per quanto riguarda la quantificabilità, è importante perché permette di tenere traccia dell'impatto generato da alcuni eventi. Questi possono essere:

- quantificabili economicamente, cioè traducibili in indicatori finanziari/economici che misurino l'efficacia della strategia;
- non quantificabili economicamente, ma che è possibile utilizzare come indicatori di prestazione;
- non quantificabili, ma che bisogna comunque tracciare anche se non sarà possibile averne una misurazione o averli come indicatori quantitativi.

1.3.4 Big Data Management: gestione di dati strutturati e non strutturati

La sfida nel Big Data Management è comprendere i dati che circolano in Internet e renderli comprensibili per trasformarli in azioni strategiche per il business. Ogni giorno vengono prodotte grandi quantità di dati da fonti diverse, quindi bisogna saper sempre identificare le nuovi fonti e incorporarle nelle piattaforme di Data Management. I dati non vanno mai persi; è necessario saperli gestire e archiviare correttamente, perché alcuni potrebbero non essere rilevanti all'inizio, ma lo potrebbero diventare in un secondo momento. Se volessimo racchiudere in dei semplici *step* un processo di Data Management, essi sarebbero i seguenti:

- usare le fonti giuste per raccogliere i dati, necessarie per ottimizzare le interazioni tra macchine, stakeholder e software;
- decifrare i dati correttamente per usarli in azienda;
- utilizzare uno storage adeguato per i dati;
- unire insieme Big Data Management, Business Intelligence e Data Science.

1.3.5 Il valore del mercato Analytics in Italia

1,7 miliardi di euro era il valore del mercato dell'Analytics nel 2019 in Italia, secondo l'Osservatorio Big Data Analytics e Business Intelligence, dato che era in aumento del 23% rispetto all'anno precedente e che segnava un incremento di oltre il doppio rispetto a soli 5 anni fa (nel 2015 i ricavi erano pari a 790 milioni), periodo in cui il tasso medio annuo di crescita è stato del 21,3%. Nel fatturato totale la principale voce di spesa era relativa ai software (47%). In tale ambito i tool per la visualizzazione e analisi dei dati pesavano per il 53%, mentre il 47% era costituito da strumenti di *ingestion* dei dati, integrazione, preparazione e *governance*. Il 20% della spesa era dedicato a risorse infrastrutturali, soluzioni per abilitare gli strumenti di Analytics e fornire capacità di calcolo e storage ai sistemi, primo fra tutti il cloud. Il 33% degli investimenti era destinato a servizi per la personalizzazione del software, l'integrazione e la consulenza per la riprogettazione dei processi. Tra i settori che si sono distinti nell'impegno su questo fronte, si collocavano al primo posto, per quote di mercato, le banche, con il 28% della spesa, seguite dal settore manifatturiero (24%), telecomunicazioni e media (14%), servizi, GDO e retail (8%), assicurazioni (6%), utility

(6%) e PA e sanità (5%). Resta il divario fra le imprese di grandi dimensioni e le PMI in termini di investimenti e competenze di Data Science. Il 93% delle grandi imprese, infatti, investe in progetti di Analytics, contro il 62% delle PMI.

1.3.6 Tecnologie chiave e trend evolutivi dei Big Data

Non esiste una vera e propria tecnologia di big data analytics. Esistono invece advanced analytics che possono essere applicati ai big data, dove diversi tipi di tecnologie lavorano insieme per aiutare a ottenere il massimo valore dai dati aziendali. Si fa riferimento a:

- *Machine Learning:* il Machine Learning è una sottocategoria specifica dell'Intelligenza Artificiale, che addestra le macchine ad apprendere e permette di produrre in modo rapido e automatico modelli in grado di analizzare i dati più grandi e complessi e di fornire risultati accurati, anche su vasta scala. Con la costruzione di modelli precisi, un'azienda ha maggiori possibilità di individuare opportunità redditizie o di evitare rischi sconosciuti.
- *Data management:* i dati devono essere di alta qualità e gestiti al meglio prima di poter essere analizzati in modo affidabile. Con il costante flusso di dati aziendali in ingresso e in uscita, è importante stabilire processi ripetibili per costruire e mantenere standard elevati di *data quality*. Dopo essersi assicurati che i dati siano affidabili, le aziende devono definire un programma di master data management che ponga l'intera organizzazione sulla stessa lunghezza d'onda.
- *Data mining:* la tecnologia di data mining aiuta le aziende ad esaminare grandi quantità di dati per scoprirne i modelli. Le informazioni possono essere utilizzate per ulteriori analisi e per rispondere a complesse domande di business. Con il software di *data mining*, è possibile filtrare i valori caotici e ripetitivi presenti nei dati, individuare ciò che è rilevante, utilizzare tali informazioni per valutare i probabili risultati e, quindi, accelerare il processo decisionale.
- *Hadoop:* questo framework software open source è in grado di memorizzare grandi quantità di dati ed eseguire applicazioni su cluster di hardware commodity. È diventata una tecnologia chiave per il business grazie al costante aumento nei volumi e nella varietà di dati, e il suo modello di calcolo distribuito elabora i big data velocemente.
- *Analytics in-memory:* analizzando i dati dalla memoria di sistema (invece che dal disco rigido), è possibile ricavare insight di business immediati dai dati e agire su di essi rapidamente. Questa tecnologia è in grado di rimuovere le latenze nella preparazione dei dati e nell'elaborazione analitica per testare nuovi scenari e creare modelli; non solo è un modo semplice per rimanere agili e prendere decisioni di business migliori per le organizzazioni, ma permette, anche, di eseguire scenari di analisi iterativi e interattivi.
- *Analisi predittiva:* la tecnologia di analisi predittiva utilizza dati, algoritmi statistici e tecniche di machine learning per identificare la probabilità di risultati futuri sulla base di dati storici. Si tratta di fornire una valutazione ottimale di ciò che accadrà in futuro, perché le aziende possano sentirsi più sicure di prendere la migliore decisione di business possibile. Alcune delle applicazioni più comuni dell'analisi predittiva includono l'individuazione delle frodi, il rischio, le operations e il marketing.
- *Text mining:* con la tecnologia di text mining è possibile analizzare i dati testuali provenienti dal web, campi di commento, libri e altre fonti testuali per scoprire intuizioni di business che non erano visibili a prima vista. Il text mining utilizza la tecnologia di machine learning o di elaborazione del linguaggio naturale per ispezionare i documenti (email, blog, feed di Twitter, sondaggi, post e altro), analizzare grandi quantità di informazioni per scoprire nuovi argomenti e relazioni tra i termini.
- *SIEM:* nel campo della sicurezza informatica con l'acronimo SIEM (Security Information and Event Management) ci si riferisce ad una serie di prodotti software e servizi che combinano/integrano le funzionalità offerte dai SIM (Security Information Management)

a quelle dei SEM (Security Event Management). Il SEM provvede al monitoraggio e alla gestione del sistema real-time degli eventi che accadono all'interno della rete e sui vari sistemi di sicurezza, fornendo una correlazione e aggregazione tra essi. Presenta una console centralizzata adibita al monitoraggio, alla segnalazione e alla risposta automatica a determinati eventi. Il SIM è un software utilizzato per automatizzare il processo di raccolta e gestione dei log non in tempo reale. I dati vengono raccolti e spediti ad un server centralizzato tramite l'utilizzo di software agent installati sui vari dispositivi del sistema monitorato. La possibilità di usufruire di spazi di archiviazione a lungo termine, unita all'analisi dei dati, consente la generazione di report personalizzati. I SIEM integrano le funzionalità offerte dai SEM e SIM al fine di poter combinare all'analisi svolta in tempo reale degli eventi la possibilità di fornire report inerenti ai dati raccolti, rispondendo alle esigenze di *incident response*, *compliance* e di analisi forense. I principali casi d'uso sono il tracciamento delle autenticazioni attraverso i sistemi, individuando, se presenti, gli accessi non autorizzati, il rilevamento di malware e parti infette del sistema, il monitoraggio delle connessioni sospese e trasferimento dei dati, il rilevamento di connessioni sospette verso l'esterno, la segnalazione nel caso di Intrusion Prevention System (IPS) o Intrusion Detection System (IDS) basandosi sui dati inerenti alle vulnerabilità e altri dati contestuali raccolti dal SIEM, la violazione delle politiche interne del sistema e l'individuazione di tentativi di attacco e compromissione al corretto funzionamento delle applicazioni web.

Alcune tra le tendenze in atto a livello italiano e internazionale nel mondo Big Data Analytics vengono riassunte in Figura 1.2:

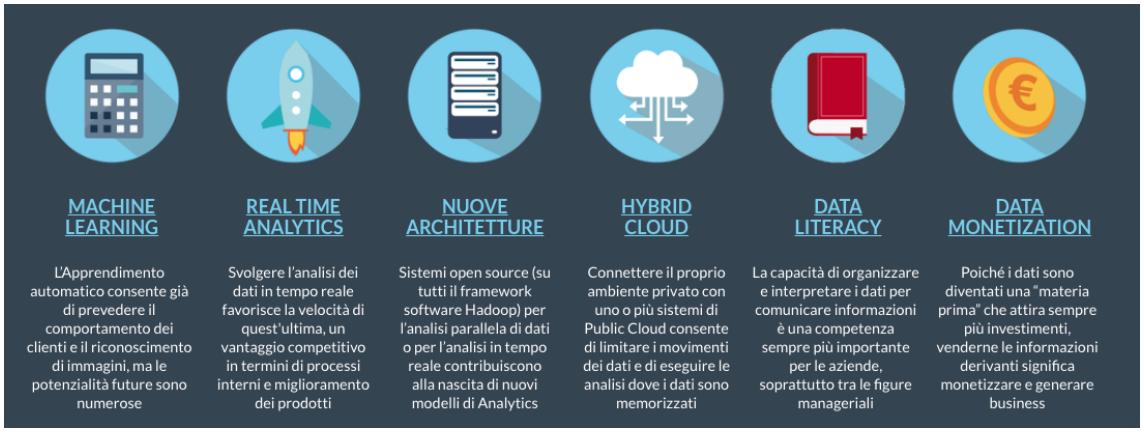


Figura 1.2: Trend evolutivi di Big Data Analytics, *Fonte: blog.osservatori.net*

1.4 Specialisti della Data Science

I professionisti della Data Science provengono dai percorsi di formazione più disparati: dall'informatica all'economia, passando per statistica, matematica o fisica. Una delle figure di spicco è il *Data Scientist*, ovvero la figura professionale che comunemente si associa alla capacità di gestire i Big Data e trarre delle informazioni rilevanti. Oltre a questa figura, troviamo il *Data Analyst*, ovvero colui che esplora, analizza e interpreta i dati, con l'obiettivo di estrapolare informazioni utili al processo decisionale. In altre parole, l'obiettivo del suo lavoro è ricercare evidenze quantitative all'interno di grandi moli di dati, supportando in tal mondo le decisioni di business. Oltre a questi professionisti esiste una terza figura: il *Data Engineer*. Il suo compito è quello di rendere i dati disponibili per le analisi. È fondamentale quando si tratta di integrare fonti eterogenee o di gestire flussi di dati in streaming. In contesti complessi, la presenza di un *Data Engineer* è di vitale importanza per rendere le analisi tempestive e per far funzionare l'intera

infrastruttura. Il *Data Engineer* è, quindi, un ruolo estremamente tecnico, focalizzato sulla capacità di gestire database relazionali e non, sulla conoscenza di tecnologie di Big Data (ad esempio Hadoop o Spark) o di *stream processing* (ad esempio Kafka) e, più in generale, sulla costruzione e manutenzione dell'architettura che rende i dati disponibili per le analisi.

Più a livello generale, e ragionando per macro-categorie, è possibile riassumere le competenze afferenti alla Data Science in cinque ambiti:

- *Tecnologia:* capacità di gestire dati strutturati e non, di estrarre dati da fonti esterne tramite metodologie e tool specialistici, capacità di manipolare e distribuire grandi quantità di dati.
- *Analytics/Machine Learning:* conoscenza di modelli e tecniche matematico-statistiche, capacità di sviluppare e implementare algoritmi di machine learning e conoscenza di tool per effettuare analisi.
- *Programmazione:* capacità di programmazione nei principali linguaggi per raggiungere uno scopo desiderato.
- *Knowledge Deployment:* capacità di creare una rappresentazione dei dati, che sia interessante e intuitiva, per aiutare l'interpretazione degli stessi. Non si fa riferimento soltanto a *soft skill* di comunicazione, ma anche a competenze tecniche, quali capacità di utilizzo di software di *data visualization* e design di dashboard.
- *Comprensione del business:* conoscenza di aspetti di business (effetti di micro e macro-economia, processi funzionali, linguaggi e obiettivi di management) e di governo di variabili, legate al settore in cui l'azienda opera.

1.5 Campi di utilizzo dei Big Data

Le applicazioni dei Big Data sono pressoché infinite e si possono estendere a una grande quantità di settori. L'analisi dei big bata può essere utilizzata in ambito politico e governativo e può essere sfruttata in diversi modi per cause di interesse pubblico. Si pensi, per esempio, alle applicazioni da parte di un ente governativo per la sicurezza stradale, ottenendo, così, dati relativi agli incidenti stradali o a zone e orari più trafficati, per guidare la pianificazione urbana e rendere più sicure le strade. Grandi volumi di dati vengono anche usati in periodi di elezioni, da un lato per conoscere orientamenti, abitudini e preferenze dei cittadini, dall'altro per fare previsioni sui risultati elettorali. Nel campo dell'agricoltura le aziende di biotecnologia riescono, oggi, grazie ai dati provenienti da sensori ad hoc, ad ottimizzare l'efficienza delle coltivazioni. Attraverso simulazioni o test di coltivazioni vengono monitorate le risposte delle piante a differenti condizioni climatiche o, comunque, a delle variazioni nell'ambiente. Sulla base dei dati raccolti è possibile adeguare, man mano, la temperatura, l'acqua, la composizione della terra, tra i vari fattori, per riuscire a identificare qual è l'ambiente ottimale per la crescita delle differenti tipologie di piante. In ambito medico, invece, le potenzialità della *big data analytics* sono grandi: dei sensori altamente sofisticati e particolarmente precisi vengono inseriti sia negli strumenti medici che nel corpo dei pazienti, ma anche nei dispositivi indossabili, come orologi o occhiali. Questi ultimi, per esempio, consentono di monitorare in tempo reale lo stato di salute di pazienti con problemi specifici, permettendo ai medici di ottenere delle informazioni molto precise sulla situazione dell'individuo e di poter agire in maniera tempestiva, se necessario. Ovviamente una grande raccolta di dati in questo ambito risulta di particolare rilevanza per la ricerca di nuovi farmaci e terapie più efficaci.

Come visto precedentemente, sempre più imprese ricorrono alla figura del *data scientist* perché i risultati positivi, in termini di profitto, di un'analisi efficiente dei dati sono evidenti. Più sono grandi il volume e la varietà dei dati, più funzionale risulterà la loro analisi ai fini del business. Dati provenienti per esempio dai social network possono aiutare a definire bene il target (o i differenti target) di un'azienda e a migliorare i prodotti o i servizi sulla base delle preferenze espresse dagli utenti. Si pensi a casi come quello di Netflix: esso sfrutta i dati di visualizzazione e le preferenze

degli utenti che usano la piattaforma di streaming per creare dei prodotti audiovisivi mirati e per ottimizzare anche la piattaforma stessa.

Grazie all'IOT le aziende possono anche ottenere dei dati provenienti dai macchinari industriali e, così, identificare velocemente eventuali problemi tecnici e risolverli in maniera più efficiente. Ugualmente importante è l'utilizzo di questi dati per evitare casi di frode o almeno per ridurne i danni, cercando di prevenire situazioni future. Infatti, è possibile migliorare la qualità dei prodotti e processi produttivi, ottimizzando la loro manutenzione presso i clienti e abbassando, anche, i costi di produzione grazie alla riduzione di guasti inaspettati.

La big data analytics viene applicata anche in ambito pubblicitario, mediante i Recommender System. Un sistema di raccomandazione, o motore di raccomandazione, è un software di filtraggio dei contenuti che crea delle raccomandazioni personalizzate specifiche per l'utente, così da aiutarlo nelle sue scelte. Viene utilizzato per diversi prodotti, come libri, musica, film, video, notizie e social media. Esistono due tecniche di recommender system, ovvero l'approccio "content based" e il "collaborate filtering", a cui si aggiunge una tecnica ibrida che le considera entrambe.

L'approccio basato sui contenuti incrocia il contenuto di un elemento e il profilo di un utente. Il contenuto di un elemento è costituito dalla sua descrizione, dai suoi attributi, dalle sue parole chiave e dalle sue etichette. Questi dati vengono messi a confronto con il profilo utente, il quale racchiude le preferenze dell'utente stesso, costruite analizzando gli elementi che ha visualizzato durante la navigazione. Anche il profilo delle preferenze è espresso tramite attributi, parole chiave ed etichette. Mettendo a confronto il contenuto degli elementi e il profilo delle preferenze, il motore di raccomandazione suggerisce all'utente articoli che possono essere di suo interesse. L'approccio basato sul contenuto utilizza sistemi per creare delle stime probabili e affidabili. Viene spesso usato il classificatore Bayesiano. Un esempio di sito che applica l'approccio basato sul contenuto è Pandora Radio. Un utente che usa Pandora Radio deve creare una lista di riproduzione aggiungendo canzoni e artisti di suo interesse. In questo modo, il sistema consiglia all'utente altre canzoni e autori che potrebbero essere di suo interesse. In questo caso, il sistema fornisce all'utente raccomandazioni anche basandosi sui like che l'utente ha attribuito alla canzone o all'artista.

L'approccio collaborativo, invece, crea dei suggerimenti utilizzando la similarità tra gli utenti. Quest'ultima viene definita con il termine "omofilia" e si basa sul presupposto che utenti simili abbiano probabilmente gusti simili. L'approccio collaborativo si basa sul tipo di elementi visualizzati dall'utente in precedenza e su eventuali voti che l'utente ha attribuito all'elemento. Partendo dal principio che utenti simili potrebbero attribuire allo stesso elemento una valutazione simile, l'algoritmo fornisce all'utente gli elementi votati da altri utenti con le valutazioni più alte. Utilizzando l'approccio collaborativo, non è necessaria la creazione esplicita del profilo dell'utente perché il sistema ne crea uno attraverso i dati, come il tipo di acquisto.

Il problema più diffuso che si verifica utilizzando l'approccio collaborativo è il problema della *cold start*. Questa situazione si presenta quando un nuovo utente si unisce al sito. Il sistema non ha a disposizione sufficienti informazioni di acquisto dell'utente e, di conseguenza, la raccomandazione è meno accurata. La stessa situazione avviene quando un nuovo prodotto è aggiunto al catalogo. La radio online "Last.fm" è un esempio di sito che utilizza l'approccio collaborativo. L'elenco di riproduzione di un utente viene comparato a quello di utenti simili, in modo da creare una lista di canzoni che l'utente non ha ancora ascoltato ma che altri utenti simili hanno ascoltato nel passato.

L'approccio ibrido è ottenuto dalla combinazione tra l'approccio collaborativo e quello basato sul contenuto. In questo caso, non si può constatare come il sistema crea delle raccomandazioni perché dipende dalla combinazione degli altri due approcci. I metodi utilizzabili sono i seguenti:

- applicazione dell'approccio collaborativo e dell'approccio basato sul contenuto in maniera separata, per poi incrociare i risultati;
- inserimento nell'approccio collaborativo delle caratteristiche dell'approccio basato sul contenuto;

- inserimento nell'approccio basato sul contenuto delle caratteristiche dell'approccio collaborativo;
- costruzione di un modello che unisce l'approccio collaborativo e l'approccio basato sul contenuto.

Netflix utilizza l'approccio ibrido per raccomandare prodotti all'utente. Per creare delle raccomandazioni, vengono utilizzate le valutazioni degli utenti simili, sfruttando l'approccio collaborativo, e il contenuto dell'elemento incrociato al profilo dell'utente, sfruttando l'approccio basato sul contenuto.

1.5.1 Applicazione dei Big Data ai Social Network

Le tecniche di social media analytics, e in particolare la sentiment analysis, si utilizzano proprio per classificare i testi scritti sul web dagli utenti. Grazie a questo strumento l'azienda o il data scientist può:

- capire quali sono i trend generali tra le conversazioni dei consumatori;
- capire come il proprio marchio viene percepito dai consumatori;
- identificare immediatamente eventuali commenti negativi (e reagire con tempestività);
- identificare le parole associate al proprio prodotto o servizio;
- monitorare più in generale le conversazioni online sul brand e sui prodotti per programmare strategie di marketing più mirate;
- studiare possibili reti ed individuare persone con maggiore influenza (ovvero, gli *influencers*).

Analizzare i dati della sentiment analysis e integrarli con i dati aziendali sul mercato di riferimento permette di migliorare la propria offerta di prodotti/servizi.

1.6 Big Data e Privacy

Affrontare il fenomeno dei Big Data dal punto di vista giuridico significa approfondire le problematiche relative al trattamento dei dati personali, sempre più preziosi e sempre più a rischio. Il rapporto tra Big Data e privacy non è semplice e il GDPR (General Data Protection Regulation), in vigore dal 25 maggio 2018, ha modificato e reso ancor più complesso lo scenario a livello europeo.

Nel settembre 2014, il Gruppo di Lavoro "Article 29 Data Protection Working Party", istituito dal Garante Europeo della Protezione dei Dati, definisce i Big Data come un termine generico che comprende un gran numero di operazioni di trattamento dei dati. In concreto, la crescita di questo enorme patrimonio informativo può comportare dei rischi per la tutela e la riservatezza dei dati trattati. Dal punto di vista delle aziende, ciò implica che i benefici derivanti dall'utilizzo dei Big Data possono essere sfruttati solo a condizione che siano adeguatamente soddisfatte le aspettative degli utenti e che sia garantita loro la tutela della privacy.

1.6.1 Dati Personali

Tracciare dei confini tra dati personali e non, è meno banale di quanto si pensi. Per "dato personale" si intende qualsiasi informazione riguardante una persona fisica identificata o identificabile. La *Foundation for Accountability Information* distingue quattro tipologie di dati personali:

- *Provided Data*, forniti consapevolmente e volontariamente dagli individui (ad esempio, la compilazione di un modulo online);
- *Observed Data*, raccolti automaticamente (ad esempio, dati raccolti tramite cookie o sistemi di videosorveglianza collegati al riconoscimento facciale);
- *Derived Data*, prodotti da altri dati in modo relativamente semplice e diretto (ad esempio, calcolando la redditività del cliente dal numero di visite a un negozio e agli oggetti acquistati);

- *Inferred Data*, prodotti utilizzando un metodo analitico complesso per trovare le correlazioni tra i set di dati e utilizzarli per categorizzare o profilare le persone (ad esempio, calcolare i punteggi di credito o predire lo stato di salute futuro di un soggetto). Si basano sulle probabilità e possono, dunque, essere meno "certi" dei dati derivati.

1.6.2 Principali problemi normativi

Uno dei problemi principali è la scarsa trasparenza in materia di trattamento dei dati. Una volta inseriti nei sistemi, i dati contenuti negli strumenti di storage dei Big Data vengono “persi di vista”. Questo comporta un elevato rischio per l’interessato, in quanto il titolare del trattamento potrebbe utilizzarli per finalità non perseguibili rispetto alle informative e ai consensi raccolti. Il tema delle informative e dei relativi moduli di raccolta di consenso è cruciale; questi devono essere adeguati ai principi normativi: una descrizione troppo vaga e generica delle finalità del trattamento determina la nullità del consenso. Inoltre, la numerosità delle fonti informative fa sì che le persone interessate abbiano grandi difficoltà nel comprendere come i dati vengono integrati tra loro e trattati.

Inoltre, anonimizzare non è sufficiente. Anche informazioni apparentemente anonimizzate possono presentare delle problematiche. Attraverso la fusione di diverse banche dati, infatti, si può riuscire a "re-identificare" un interessato anche attraverso informazioni apparentemente anonime. In molti casi, dunque, l’anonimizzazione di singoli identificatori univoci non è sufficiente per escludere le re-identificazioni (un dato considerato «anonimo» può essere successivamente attribuito a una determinata persona). Inoltre, gli algoritmi che vengono applicati nell’analisi dei Big Data permettono di analizzare in modo autonomo e automatizzato banche dati di grandi dimensioni, anche nelle loro connessioni reciproche. Queste procedure di analisi generano nuove informazioni e spesso nuovi dati personali. Per questi motivi è essenziale che chi intende avvalersi di Big Data consideri il tema della protezione dei dati sin dalla fase iniziale di un progetto attraverso la preventiva interazione di criteri in materia di protezione dei dati («Privacy by Design»).

1.6.3 Possibili soluzioni

Le Autorità mondiali di protezione dei dati, al fine di tutelare i diritti degli interessati al trattamento, hanno ritenuto opportuno raccomandare alle aziende delle linee guida, riassunte nel seguente elenco:

- trasparenza delle attività di raccolta dei dati, l’elaborazione, l’uso e la loro condivisione;
- manifestazione espressa, prima del trattamento, da parte degli interessati, del consenso all’utilizzo dei propri dati per fini di analisi o di profilazione;
- adozione di misure idonee a tutelare i dati e a garantirne il controllo;
- utilizzo, quando possibile, di dati anonimi;
- limitazione delle finalità (utilizzo dei dati per scopi compatibili con le finalità note all’utente);
- accesso ai dati raccolti dai legittimi titolari, nonché accesso, da parte dei medesimi, alle informazioni e decisioni che li riguardano;
- tutela del diritto degli interessati di correggere/modificare i propri dati;
- configurazione delle tecniche e delle procedure relative ai Big Data affinché i relativi criteri vengano considerati sin dalla fase iniziale di un progetto e affinché la sicurezza dei dati sia garantita.

2. Social Network Analysis

2.1 Rete sociale

La Social Network Analysis ha a che fare con il significato che attribuiamo comunemente al termine "social network" tanto quanto ha a che fare con qualsiasi gruppo di individui connessi tra loro da un qualunque legame, anche offline. Si pensi ai rapporti di lavoro, ai vincoli familiari, al fatto di fare parte di un club, o di avere la tessera di un supermercato, così come essere amici su Facebook. Per comprendere meglio di cosa si sta parlando si può tradurre *Social Network Analysis* in italiano: rappresenta l'analisi delle reti sociali, anche detta "teoria della rete sociale", ovvero una metodologia di studio delle relazioni che trova applicazione in diverse scienze: la sociologia, la psicologia, l'economia, l'antropologia, ma anche la genetica e la computer science, ambiti dove il termine "social" è inteso in senso lato, non più con diretto riferimento alle relazioni tra esseri umani, bensì, più genericamente, alle relazioni tra individui, enti, attori di una relazione.

Una rete sociale è un insieme fatto di punti e di linee che collegano i punti, come mostrato in Figura 2.1.

Figura 2.1: Tipica composizione di una rete sociale *Fonte: www.thismarketerslife.it*

Nell'analisi delle reti sociali, di solito, i punti vengono chiamati "nodi" e le linee "ponti" o "archi". L'analisi delle reti sociali studia le relazioni che sussistono all'interno di un gruppo e

permette di comprendere i collegamenti in un insieme di persone, i loro legami e l'intensità degli stessi, di identificare le persone più influenti in un gruppo (ad esempio gli *influencer*), di scoprire quali persone rappresentano l'aggancio necessario per mettere in contatto due individui, o due gruppi di individui, che altrimenti non comunicherebbero tra loro.

Ad un livello più alto, l'analisi delle rete sociali mostra com'è fatto un network: ci si può chiedere se le relazioni sono biunivoche e perlopiù omogenee, oppure se il gruppo osservato è formato da sottogruppi, nicchie: community. Si può comprendere quanto comunicano queste community tra loro e quali sono i membri delle community che costituiscono un ponte, ovvero un passaggio utile e strategico che permette di raggiungere un'altra community.

L'ovvio legame tra una quantità enorme di dati e le piattaforme online, dove i network prendono forma, sta spingendo sempre più a considerare l'analisi delle reti sociali come una nuova, importante e tuttora poco esplorata branca di indagine, utilissima in una strategia di marketing dall'approccio più scientifico. I social network ci danno un accesso, più o meno semplice, ai loro magazzini di informazioni, che possiamo usare per ricostruire come il network è fatto, quali sono i legami che intercorrono tra i suoi membri, per poi implementare tali informazioni nelle nostre strategie di marketing.

2.2 Community Detection

Una community è un gruppo di nodi densamente connessi tra loro. Non è auspicabile analizzare un gruppo dato da "tutti gli utenti di Twitter": si tratta di un insieme troppo vasto e variegato per pensare di ricavarci un'indagine significativa. Più interessante potrebbe essere analizzare il sottogruppo di tutti gli utenti di Twitter legati dal twittare frequentemente in relazione ad un determinato hashtag. Questo concetto vale per tutti i social network, considerando per ciascuno le caratteristiche relative a quella piattaforma specifica. La *community detection* è, allora, l'indagine che permette di individuare le community dentro ad un network e di analizzarle per scoprire informazioni del tipo: quanto stretti sono i legami tra i suoi membri, chi sono gli *influencer*, come circolano le informazioni all'interno della rete e attraverso quali nodi far passare i messaggi dall'esterno.

2.3 Social Network Analysis

Secondo Scott (1991), l'origine della Social Network Analysis è da attribuire agli studi effettuati da Jacob Moreno (1889-1974) sulla "sociometria", la scienza che studia i metodi di rilevazione e misurazione delle relazioni che intercorrono all'interno di un gruppo o comunità, e da Fritz Heider (1896- 1988) sulla "Triad Equilibrium Analysis". Queste prime idee vennero successivamente riprese da Frank Harary (1921-2005) e Dorwin Cartwright (-1988), i quali individuarono nei metodi propri della teoria dei grafi un potente metodo formale per lo studio delle strutture sociali. Intorno agli anni '30 e '40 alcuni ricercatori di Harvard, tra cui Lloyd Warner (1898-1970) e Eltan Mayo (1880-1949), in uno studio sulle comunità del New England e in uno successivo sulle condizioni di lavoro degli operai della "Chicago Central Plant", introdussero l'uso dei sociogrammi, una metodologia di indagine, usata ancora oggi nelle scienze dell'educazione, che, tramite l'uso di questionari, si pone l'obiettivo di ricostruire le posizioni degli individui all'interno di un gruppo, per la valutazione delle proprietà informali dei grafi, definendo, inoltre, il concetto di "clique". Una profonda revisione critica di questi concetti venne operata da George Homans (1910-1989). Un fondamentale sviluppo alla Social Network Analysis venne dato da due ricercatori del dipartimento di antropologia sociale della Manchester University, ovvero John Barnes e (1918-) e Siegfried Nadel (1903-1956), i quali puntarono l'attenzione sulle relazione date dal potere e dai conflitti tra gli individui, piuttosto che su quelle relazioni date dalle configurazioni sociali pre-imposte, come potevano essere i rapporti gerarchici, mostrando come le relazioni visibili in una rete sociale sono

molto diverse da quelle reali. Essi introdussero, inoltre, l'utilizzo di strumenti algebrici per l'analisi dei ruoli. Gli studi furono successivamente portati avanti da Clyde Mitchell (1918-1995), il quale introdusse la differenza tra rete "completa" e rete ego-centrica, oltre all'utilizzo degli indici per l'analisi. Successivi studi di Harrison C. White (1930-) sostennero che la ricerca di proprietà del network non andava fatta basandosi su categorie prestabilite, ma sulle reali proprietà della rete e, in questo contesto, Mark Granovetter dimostrò l'importanza dei legami deboli all'interno dei network.

Tra i vari tipi di analisi, una teoria originale, conosciuta come "teoria del mondo piccolo" e proposta da Stanley Milgram (1933-1984), si è rivelata piena di interessanti spunti di ricerca; essa afferma che, in una rete sociale, anche grande, il percorso di relazioni più breve necessario per collegare tra di loro due individui è solitamente molto piccolo rispetto alla dimensione della rete. Questa teoria, nonostante le molte critiche relative al paradigma di sperimentazione originale, si è rivelata un importante ambito di studio; le proprietà di questi "mondi piccoli" e la definizione di sistemi di questo tipo risultano ancora di notevole interesse scientifico. Approcci empirici al problema si sono rivolti anche al mondo web e ai network virtuali, dando vita a sistemi quali *Friendster* e ai software di *Instant Messaging*. Ai giorni nostri, come detto in precedenza, l'analisi dei Social Network interessa i più disparati campi di ricerca sociali, psicologici, antropologici, matematici, fisici, economici e matematici.

2.3.1 Sviluppi Recenti

Tra i recenti sviluppi della ricerca sulla Social Network Analysis, una parte di studi ha cercato di trovare delle applicazioni concrete in termini di strumenti di diagnosi dei gruppi sociali, di informazione e monitoraggio della rete in tempo reale. Verranno, di seguito, presentati alcuni esempi di approcci che impiegano la Social Network Analysis sullo studio dei metodi di feedback per il monitoraggio delle relazioni sociali e per l'intervento su di esse, rivolto al miglioramento della qualità delle connessioni tra i componenti. Moltissimi altri esempi sono reperibili in letteratura; il primo caso si occupa del monitoraggio della salute sociale di un gruppo di persone anziane tramite l'ausilio di sensori e di sistemi di visualizzazione (Margaret E. Morris, 2005). Il network considerato è quello che ha come soggetti un anziano e le persone che conosce e vede regolarmente. Questo tipo di network, durante la vita di una persona, è soggetto a grandi variazioni, e tende a diventare molto ristretto con l'aumento dell'età, rendendo gli anziani rassegnati alla solitudine. Inoltre, i contatti che un anziano ha con altre persone sono, molto spesso, mediati e favoriti da pochissime persone, come, ad esempio, figli ormai adulti, che diventano, perciò, nodi focali della rete, che spesso sentono il peso di questa responsabilità.

Per il monitoraggio delle relazioni sono stati utilizzati dei sensori che hanno permesso di monitorare parametri quali le chiamate telefoniche, la presenza di visite in casa e la loro durata; ad essi, per completezza, sono stati aggiunti dei report fatti dai vari soggetti, tramite intervista. I dati così raccolti sono stati resi disponibili ai vari attori sociali tramite alcune schermate di facile interpretazione, a cui poteva avere accesso sia la persona anziana, sia un'altra persona che si prendeva cura di essi (ad esempio, il figlio). Lo studio ha messo in evidenza come una conoscenza dello stato delle relazioni da parte dell'anziano ne migliori notevolmente la vita sociale e, al contempo, alleggerisca il carico di responsabilità dei nodi focali della rete, che prima erano completamente responsabili; con l'introduzione di questo feedback, invece, essi sentono venir meno il peso delle responsabilità.

Il secondo lavoro riguarda lo studio di un social network per un gioco on-line (Baù, Gamberini, Martino, Spagnolli, 2009). Lo scopo della suddetta ricerca era lo studio di come un particolare feedback sulla partecipazione al network dei soggetti influenzi l'attività del soggetto stesso. Lo studio era volto a comprendere quali tipi di feedback influenzavano maggiormente le attività comunicative dentro una rete per un compito specifico. L'esperimento effettuato consisteva nell'impiego di un gioco on-line chiamato "Crossfire", abbinato ad un client per la messaggistica

istantanea, utilizzato per lo scambio dei messaggi tra i partecipanti. Il task dato ai soggetti era quello di ricercare il maggior numero di calici nascosti all'interno di una mappa fornita dall'ambiente di gioco, dentro la quale l'avatar del partecipante poteva muoversi liberamente. Le informazioni per il ritrovamento dei calici erano fornite tramite dei cartelli situati in differenti città costitutive dell'ambiente. Gli utenti, divisi in gruppi, dovevano trovare il maggior numero possibile di oggetti per far vincere il match alla propria squadra. L'utente, in ogni istante di tempo, poteva vedere solo una determinata porzione di mappa: la ricerca poteva essere effettuata focalizzandosi sull'ambiente oppure utilizzando il gruppo per ricevere informazioni utili alla localizzazione dei calici. Sulla base dei dati raccolti manualmente sull'utilizzo della chat testuale e dei flussi di messaggi scambiati per comunicare con il resto del team vennero calcolati, in tempo reale, gli indici "Centrality Degree" e "Reciprocity" dei singoli soggetti, dividendo i soggetti in tre gruppi (indipendenti dalla suddivisione in team), fornendo al primo come valore del feedback il valore di "Centrality Degree", al secondo il valore di "Reciprocity" e al terzo niente. Per una facile interpretazione dei valori essi sono stati presentati all'utente tramite dei *gauge*. Infine ogni team doveva partecipare a tre match differenti per poterne studiare il trend. Lo studio ha dimostrato che l'utilizzo di feedback si dimostra efficace per il monitoraggio della rete e per l'incremento della comunicazione, benché questa maggior coesione sociale non sia rilevata dai soggetti esposti ai feedback stessi.

2.4 La teoria dei grafi

La paternità della teoria dei grafi è attribuita ad Eulero, uno dei più importanti matematici del '700, che la usò per risolvere un problema di geometria, ovvero quello dei sette ponti di Königsberg. Si trattava di un problema in cui si considerava una città all'interno della quale vi erano sette ponti. L'obiettivo era quello di scoprire se fosse possibile, passeggiando, attraversarli tutti, ma una volta sola. Nella teoria dei grafi il problema può essere rappresentato con punti (le zone della città) collegati da linee (i ponti). La teoria dei grafi fornisce i concetti e gli strumenti per analizzare e studiare queste mappe.

La Network Analysis e la Social Network Analysis (SNA) usano la teoria dei grafi per creare dei modelli semplificati della realtà da studiare o del problema da risolvere. Come la matematica sta alla fisica o ad altre scienze, così la teoria dei grafi sta alla SNA. Dopo i sociogrammi, usati da Jacob Moreno per rappresentare le relazioni tra gruppi di persone, già nel 1950 si parla di social network quando J.A. Barnes usa i grafi per rappresentare i legami sociali all'interno di un villaggio di pescatori scandinavo. Nel 1967 un esperimento di Stanley Milgram lancia l'espressione "sei gradi di separazione", un concetto oramai noto grazie ai moderni social network e ad un film: si presume che ogni persona sia collegata ad una qualsiasi altra persona nella terra da una catena di conoscenze fatta da non più di 6 persone. L'esperimento originale prevedeva il recapito di lettere da diverse zone degli Stati Uniti ad una specifica persona di Boston. Molti anni dopo l'esperimento fu ripetuto, con la differenza che vennero utilizzate le e-mail. Recentemente il gruppo di analisti di Facebook lo ha ripetuto all'interno della rete del social network. Negli anni la catena sembra essersi accorciata: dai sei gradi di separazioni rilevati da Milgram si passa ai 4 scarsi rilevati dagli studiosi Facebook.

La Network Analysis può essere usata per studiare sistemi complessi di qualsiasi genere, purché rappresentabili con un grafo. I campi di applicazione sono moltissimi e le scoperte spesso sorprendenti. In biologia la si è usata per studiare le interazioni tra i geni di un organismo, in logistica per studiare le reti di trasporti, in informatica per ottimizzare l'instradamento dei pacchetti di informazione in Internet. Capire i collegamenti tra le persone e come le informazioni si propagano nella rete di relazioni che le legano può aiutare a migliorare la comprensione del proprio pubblico e ottimizzare sia il messaggio che la spesa per sponsorizzarlo. Secondo alcuni studiosi, tra cui Duncan Watts, esperto teorico delle reti e autore di libri come "Six degrees: the science of a connected age" e "Everything Is Obvious: Once You Know the Answer", non bisognerebbe preoccuparsi

troppo di identificare e coinvolgere gli *influencer* per lanciare un prodotto. Sarebbe preferibile, e forse meno costoso (secondo Watt), identificare nella rete le persone sensibili al messaggio da propagare e raggiungerne il maggior numero possibile, incoraggiandole a condividerlo. La struttura della rete farà il resto. Secondo gli studiosi, infatti, non è il numero di persone quello che conta, ma come sia strutturata la rete che li lega. Incoraggiare le connessioni tra le persone aumenta il numero di possibili vie di propagazione delle informazioni. Maggiore è il numero di vie che le informazioni da propagare potrebbero percorrere, più alta sarà la possibilità che viaggino più velocemente (tecnicamente la lunghezza media del percorso tra due punti della rete si accorcia).

Una curiosa ricerca della Nothwestern University relativa ai musical di Broadway fece una scoperta interessante: il successo di un opera teatrale non era tanto legata al budget per il marketing o alla notorietà del regista. Per il successo e per un maggior numero di critiche positive risultò, invece, determinante quanto fossero strette le relazioni tra il cast e gli altri lavoratori. Infine, una adeguata conoscenza della rete può aiutare a fare previsioni accurate sulle reazioni possibili delle persone, anticipando possibili crisi e permettendone una gestione migliore. Accade esattamente la stessa cosa per le epidemie: è possibile identificare gruppi di persone che possano favorire il contagio. Uno studio relativo all'epidemia di influenza aviaria del 2009 permise di scoprire che la si sarebbe potuta fermare con due settimane di anticipo e vaccinando un minor numero di persone, identificando il gruppo, nella rete, più connesso. Analizzare una rete, specie se complessa, come quelle presenti nei social network, non è semplice, sia per le competenze necessarie, sia per la disponibilità dei dati. Non sempre i social network mettono completamente a disposizione i dati e, nel caso in cui ciò avvenga, lo fanno sotto determinate condizioni.

2.4.1 Dati Relazionali

È importante definire che tipo di dati, all'interno di una rete sociale, sono i più adatti per essere studiati tramite i metodi dell'analisi di rete. Quando c'è la necessità di studiare un "social network", spesso i dati da raccogliere, utili per l'indagine da svolgere, non sono misurabili con l'ausilio di strumenti di misura scientifici, ma vanno raccolti con tecniche socio/psicologiche e sono soggetti, in parte, a interpretazione. Sulla base di simili processi di interpretazione si possono distinguere tipi distinti di dati, cui corrispondono diversi metodi di analisi.

I principali tipi di dati sono gli "attributi" e i dati sono "relazionali". Gli attributi si riferiscono agli atteggiamenti, alle opinioni e ai comportamenti di individui o gruppi, considerati come proprietà, qualità o caratteristiche appartenenti ai soggetti agenti. Le informazioni, raccolte solitamente attraverso sondaggi o interviste, vengono spesso considerate come attributi di singoli individui, suscettibili di essere quantificati e analizzati attraverso le tecniche statistiche disponibili. I metodi appropriati agli attributi sono quelli dell'analisi delle variabili, in cui gli attributi sono misurati come valori di particolari variabili. I dati relazionali, invece, si riferiscono ai contatti, ai vincoli e ai collegamenti, alle appartenenze e agli incontri di gruppo, che mettono in relazione un attore con l'altro e non possono, quindi, essere ridotti a proprietà degli stessi individui agenti. Le relazioni non sono proprietà degli attori, ma di sistemi di attori; esse collegano coppie di attori in sistemi relazionali.

I metodi appropriati ai dati relazionali sono proprio quelli dell'analisi delle reti, in cui le relazioni sono esaminate, in quanto esprimono i legami che intercorrono fra gli attori. Anche se è possibile, ovviamente, eseguire conteggi quantitativi e statistici delle relazioni, l'analisi delle reti comporta un corpo di misure qualitative della struttura del network. I dati relazionali possono essere rappresentati in vari modi, ma non con le usuali matrici "casi-per-variabili". Infatti, questo tipo di rappresentazione descrive dettagliatamente gli attributi dei singoli soggetti, ma non dà alcuna informazione sulle relazioni presenti tra i soggetti. La rappresentazione dei dati relazionali richiede un formato che metta in evidenza le relazioni tra i soggetti e non le loro proprietà. Una rappresentazione tabellare utile per la rappresentazione dei dati relazionali è detta "matrice di

incidenza"; in essa si rappresentano nelle righe le persone coinvolte nell'indagine, mentre nelle colonne si inseriscono gli eventi considerati. Quando uno specifico individuo prende parte ad un evento, nella corrispondente cella della matrice è posto un "1"; la non-partecipazione è, invece, indicata con uno "0". Questo tipo di rappresentazione mostra quali soggetti hanno partecipato a quali eventi e ci indica, perciò, la relazione che intercorre tra di essi: se due o più soggetti hanno partecipato ad un stesso evento allora saranno in relazione tra di loro. Da questo tipo di rappresentazione ne sono state ideate altre, molto più simili alle matrici che verranno utilizzate, poi, nell'analisi delle reti tramite la teoria dei grafi; si tratta delle tabelle "caso-per-caso". Questo tipo di rappresentazione, che pone su righe e colonne i soggetti, mostra direttamente le relazioni che intercorrono tra essi, ripartendo nel campo corrispondente un "1" se i soggetti sono in relazione, e uno "0" se, invece, non lo sono.

Un altro tipo di rappresentazione possibile è detta "affiliazione-per-affiliazione"; in essa, in una tabella "eventi x eventi", vengono indicati gli eventi che hanno almeno un soggetto partecipante ad entrambi. Questi ultimi due tipi di rappresentazione sono dette matrici di adiacenza. I dati contenuti nella matrice possono essere suddivisi secondo due importanti criteri: direzionalità e numerazione. Per direzionalità si intende il grado di simmetria di una relazione. Per numerazione si intende il tipo di dati contenuto nei campi; tale tipo può essere binario se il campo può contenere solo i valori "0" oppure "1", numerico se il campo può contenere qualsiasi numero positivo. In questo caso il valore viene detto "peso".

2.4.2 Grafi

Una base comune per i programmi dell'analisi delle reti sociali è l'approccio matematico della teoria dei grafi, che fornisce un linguaggio formale per la descrizione delle reti e dei loro caratteri.

Definizione: "Si definisce grafo non orientato una coppia G=(V,E) in cui V è un insieme finito di elementi, detti nodi o vertici, ed E è una famiglia di coppie non ordinate di elementi appartenenti a V. Agli elementi di E, detti archi o lati, è possibile associare una funzione f:E→R; in questo caso gli archi vengono detti archi pesati. Se $(i, j) \in E$ compare 2 o più volte all'interno dell'insieme si parla di archi multipli.

Si definisce grafo orientato una coppia G=(V,A), in cui V è un insieme finito di elementi detti nodi o vertici e A è un insieme di coppie ordinate di vertici.

È possibile rappresentare i grafi tramite due soluzioni di facile interpretazione: la matrice di adiacenza (presentata nella sezione precedente), e il diagramma archi-nodi, nel quale sono rappresentati i nodi, numerati, collegati da archi (Figura 2.2).

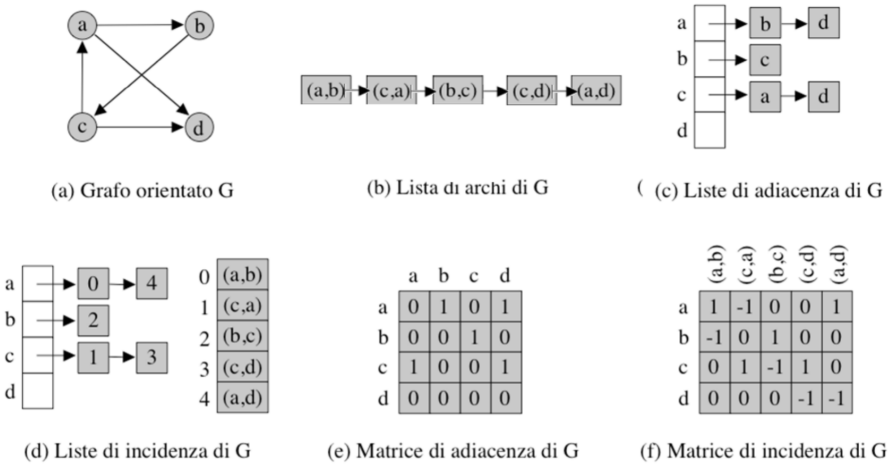


Figura 2.2: Rappresentazione dei dati e teoria dei grafi *Fonte: mat.uniroma2.it*

Nel caso sia importante considerare il peso dell'arco è possibile aggiungere un'etichetta ad ogni arco riportando il valore della funzione peso. Per l'utilizzo dei grafi nella rappresentazione delle reti sociali si possono considerare i nodi in luogo dei soggetti presi in considerazione nello studio, e gli archi in luogo delle relazioni che si instaurano tra i soggetti. Se la relazione è asimmetrica il grafo risulterà orientato, altrimenti il grafo sarà non orientato. Se in una relazione sarà importante considerare la forza del legame che una relazione esprime tra due soggetti, il grafo generato sarà pesato.

Un percorso W (*walk*) è una sequenza di nodi e linee (non necessariamente tutti distinti), che descrive un tragitto all'interno del grafo. Un percorso con nodi e linee tutti distinti prende il nome di sentiero P (*path*). Un percorso chiuso in cui ogni linea e ogni nodo sono inseriti in sequenza una ed una sola volta, tranne il nodo di origine, si chiama ciclo C (*cycle*). Un grafo si dice, inoltre, connesso se esiste un percorso tra ogni coppia di nodi nel grafo (cioè non contiene nodi isolati). Se un grafo, invece, contiene anche solo un nodo isolato, si dice sconnesso. Inoltre, vengono definiti ponte (*bridge*) e punto di separazione (*cutpoint*), rispettivamente, una linea ed un nodo che, se soppressi, sconnettono il grafo.

Un nodo viene definito raggiungibile se esiste un percorso che lo collega agli altri nodi, indipendentemente dalla sua lunghezza (e quindi dagli intermediari che dovranno essere attraversati dal percorso). Un nodo isolato, al contrario, è definito come non raggiungibile e la sua distanza dagli altri è infinita.

Possono esistere due o più percorsi tra due nodi con lunghezze differenti. Quelli generalmente usati nei calcoli sono i percorsi più brevi (*shortest path*). La distanza geodesica rappresenta il sentiero più breve tra due nodi. Il diametro, invece, è la lunghezza del percorso più lungo che collega coppie di nodi (*largest path*).

2.4.3 Indici della rete

La densità di una rete descrive il livello generale dei legami fra i punti di un grafo. Più sono numerosi i nodi direttamente collegati fra loro, più un grafo è denso. La densità di un grafo si calcola come rapporto tra il numero delle linee di un grafo e il numero possibile di linee tra i nodi.

L'inclusività, inoltre, misura la percentuale di soggetti coinvolti nei legami, o negli scambi del gruppo, ed è calcolata come numero totale di punti in un grafo meno il numero di punti isolati.

La centralità è di due tipi: si parla di centralità locale se un punto ha un gran numero di connessioni con altri punti del suo ambiente circostante; si parla di centralità globale se un punto ha una posizione di importanza strategica nella struttura complessiva della rete. L'indice di centralizzazione serve a misurare la centralità del grafo nel suo complesso, osservando le distanze fra i punteggi di centralità del punto più centrale e quelli di tutti gli altri punti. L'indice di centralità, invece, localizza la posizione dell'attore in relazione a quella degli altri nella rete. La misura più semplice della centralità si ottiene dal calcolo dei gradi, ovvero un nodo è centrale se ha un grado elevato, cioè se è adiacente a molti degli altri nodi.

Esiste, inoltre, un indice basato sulla vicinanza. Esso focalizza la propria attenzione su quanto un attore è vicino agli altri. Un attore, quindi, è tanto più centrale nella rete, quanto più è nella posizione di interagire velocemente con gli altri attori. La centralità come vicinanza è, pertanto, inversamente proporzionale alla distanza geodesica: meno si è distanti dagli altri e più si è centrali (e viceversa). Questo indice di centralità si calcola per grafi connessi.

Si può identificare, inoltre, un indice basato sul ruolo di mediatore; i nodi che si collocano in una posizione di intermediari (cioè localizzati sui percorsi che collegano coppie di nodi non adiacenti), possono esercitare un potere di controllo sul flusso delle informazioni. Il concetto di *betweenness* prende in considerazione le geodesiche presenti nel grafo, contando quante volte ogni nodo si trova coinvolto in quelle degli altri attori.

3. Il Social Network Tik Tok

3.1 Social Network

Un social network è una rete sociale che consiste in un qualsiasi gruppo di individui connessi tra loro da diversi legami sociali. Per gli esseri umani i legami vanno dalla conoscenza casuale, ai rapporti di lavoro, ai vincoli familiari. Un social network è un servizio Internet, tipicamente fruibile mediante browser o applicazioni mobili, che si appoggia sulla relativa piattaforma per la gestione dei rapporti sociali. Esso consente la comunicazione e condivisione per mezzi testuali e multimediali tra gli utenti iscritti.

I servizi di questo tipo, nati come comunità alla fine degli anni 1990 e divenuti enormemente popolari nel decennio successivo, permettono agli utenti di creare un proprio profilo, organizzare una lista di contatti, pubblicare un proprio flusso di aggiornamenti e accedere a profili altrui.

I servizi sono distinguibili in base alla tipologia di relazioni cui sono orientati, per esempio quelle amicali, lavorative o pubbliche, o anche a seconda del formato delle comunicazioni che prevedono testi brevi, immagini o musica. Il loro utilizzo è spesso offerto gratuitamente, poiché i fornitori sono remunerati dagli inserzionisti pubblicitari online.

Si possono definire siti di reti sociali tutti i servizi web che permettono:

- la creazione di un profilo pubblico, o semi-pubblico, all'interno di un sistema vincolato;
- l'articolazione di una lista di contatti;
- la possibilità di scorrere la lista di amici dei propri contatti.

Per entrare a far parte di una rete sociale online occorre registrarsi, indicando un indirizzo di posta elettronica (e-mail) ed una password (parola chiave). Dopodiché si passa alla creazione e alla gestione di un profilo personale dell'utente attraverso una serie di domande che vengono poste al nuovo iscritto (sulla città di nascita, sul luogo in cui abita, sulla scuola che frequenta o è stata frequentata, sul datore di lavoro, sugli interessi personali, hobby e altro); in questo modo si delinea la pagina web del profilo personale dell'utente. Questa pagina contiene le informazioni generali della persona, che possono essere supportate da immagini o foto, da video e da una breve auto-descrizione, che commenta più dettagliatamente il carattere personale o l'attività che si svolge. L'utente, in questo modo, ha la possibilità di scegliere quali informazioni personali rendere pubbliche, per farsi conoscere dagli altri utenti (che non lo conoscono) e quali rendere private,

per condividere i contenuti solo con gli amici. Costruire il proprio profilo personale, partendo da informazioni come il proprio indirizzo e-mail fino ad arrivare agli interessi e alle passioni (utili per le aree "amicizia"), alle esperienze di lavoro passate e alle relative referenze (informazioni necessarie per il profilo "lavoro"), è di fondamentale importanza per delineare le caratteristiche degli utenti che usufruiscono del servizio.

Successivamente si passa alla creazione di una lista di amici che si costruisce soggettivamente, indicando direttamente il nome delle persone che si vogliono includere nell'elenco delle nostre amicizie. La lista degli amici viene ampliata con l'aiuto di programmi informatici del social network che, in riferimento alle risposte che abbiamo dato nella fase di compilazione del profilo utente, ci suggeriscono amici conosciuti nella vita reale e "potenziali nuove amicizie", selezionando, tra gli utenti iscritti, le persone che hanno caratteristiche corrispondenti alle nostre indicazioni. A questo punto è possibile invitare i propri amici a far parte della propria rete; questi, a loro volta, possono fare lo stesso, cosicché ci si trova ad allargare la cerchia di contatti con gli amici degli amici, e così via, idealmente fino a comprendere tutta la popolazione del mondo, come prospettato nella teoria dei *sei gradi di separazione* dello psicologo *Stanley Milgram* (1967), la cui validità, anche su Internet, è stata recentemente avvalorata dai ricercatori della Columbia University, come spiegato in precedenza.

Una caratteristica propria dei social network è la possibilità di esplorare i profili degli amici che fanno parte del nostro elenco di amicizie e quelli che fanno parte delle amicizie dei nostri amici (anche se non hanno stretto amicizia direttamente con noi); si possono visitare le pagine personali degli utenti (amici), osservare le attività preferite, i gusti musicali e, naturalmente, interagire direttamente anche con persone che non conosciamo direttamente, ma che possono essere "amici dei nostri amici" o persone che non conosciamo affatto.

Diventa, quindi, possibile costituire delle comunità tematiche in base alle proprie passioni o aree di affari, aggregando ad esse altri utenti e stringendo contatti di amicizia o di affari. I servizi di rete sociale consentono ai detentori di siti di trarre guadagno principalmente dalla fornitura a terzi delle informazioni degli utenti, che alimentano gratuitamente la base di conoscenza, in secondo luogo dalla pubblicità mirata, che le aziende indirizzano agli utenti in base ai siti visitati, ai link aperti, alla permanenza media, alle informazioni da loro stessi inserite. Come piattaforma tecnologica con le sue funzionalità, applicazioni e risorse, il social network facilita le interazioni e lo sviluppo di connessioni e relazioni attraverso l'utilizzo di contenuti diversi (testuali, video, audio, fotografici), la condivisione sociale su piattaforme e dispositivi eterogenei (PC, smartphone e tablet), il coinvolgimento attivo dei membri della rete e la rapidità con cui si può conversare e disseminare informazioni.

L'aspetto da considerare, però, è come l'avvento di queste comunità online abbia creato effetti concreti sul modo di sentire e di pensare degli utenti, che finiscono per modificare le loro pratiche di interazione sociale usuali. Alla corporeità dell'incontro faccia a faccia si sostituisce la virtualità del profilo da cui si elimina il corpo, ma soprattutto i suoi significati. L'utilizzo dei social network e delle nuove tecnologie inducono, infatti, molti cambiamenti: cambia il rapporto con se stessi e, soprattutto, con gli altri, che diventa più diretto, ma molto più mediato. Le nuove tecnologie ci promettono di incontrare molte persone, ma tendono a togliere il sapore, la genuinità, l'originalità e la freschezza alla relazione interpersonale vera e propria. Cambia, inoltre, il modo di concepire la quotidianità. È difficile pensare alle giornate senza aprire il computer o usare il cellulare; la nostra esperienza quotidiana subisce dei pesanti condizionamenti poiché può cambiare il modo di partecipare alla vita di società. Oggi circa tre miliardi di persone si connettono ogni giorno ad Internet, e la maggior parte di queste persone lo fa per utilizzare i "Social Network", per lavoro, per tenersi in contatto con gli amici o per divertimento.

Negli ultimi anni, la tendenza degli utenti ad utilizzare queste "piattaforme di comunicazione", e la diffusione sul mercato degli innumerevoli "dispositivi mobile", hanno fatto diventare i social

network gli indiscussi padroni della comunicazione mediatica sul web, diventando un fenomeno globale che ha visto un aumento esponenziale negli ultimi 10 anni: Facebook, Instagram, Twitter, Linkedin, Twitch e TikTok sono solo alcune delle piattaforme social che, dal loro avvento ad oggi, hanno conquistato il Web. Tutte le applicazioni di social network si basano sulla costruzione, manutenzione, gestione e visibilità di profili e di pagine Web personali. Sono tutti social media che differiscono per contenuti e per audience di utenti, con l'obiettivo comune di ottenere un tempo medio di connessione sempre più elevato da parte degli utenti. La diversità tra le piattaforme nasce dall'eterogeneità delle persone, fortemente influenzate dall'ambiente e dal periodo storico in cui vivono. Di conseguenza, ogni generazione si differenzia da quella precedente e successiva, portando gli utenti a scegliere il social network più adatto alle proprie esigenze e consuetudini.

La loro espansione è, sicuramente, stata incrementata dal fatto che molte persone hanno deciso di investire sui social media, perché sono diventati il primo luogo virtuale in cui si cercano informazioni su brand o attività, battendo il primato dei motori di ricerca. Alle imprese che vogliono farsi conoscere sui social, quindi, conviene scegliere almeno uno tra i social media principali e creare un profilo aziendale dove condividere contenuti legati all'attività. Tra i social media di vecchia data che hanno avuto maggiore espansione troviamo Facebook e YouTube. Attraverso Facebook è possibile raggiungere un enorme numero di potenziali clienti, grazie ad un pubblico eterogeneo e ben rappresentato per ogni fascia d'età. Facebook è, anche, uno dei social media più versatili, prestandosi tanto per scopi professionali quanto per il personal branding o per un uso personale. Per quello che riguarda YouTube, si può notare che è stata registrata una crescita nell'ultimo periodo, e questo dipende dal fatto che i trend social, negli ultimi anni, stanno subendo un cambiamento radicale, privilegiando i contenuti video rispetto alle foto e ai contenuti testuali. Per questa motivazione, Instagram ha apportato recenti cambiamenti, introducendo novità nel meccanismo delle *Live* e i *Reel*, ovvero video molto brevi da poter pubblicare sul proprio profilo. Essendo un mezzo social legato ai video e alle immagini, forse Instagram non è adatto a tutti come Facebook, ma nel 2021 ha senso puntare su Instagram come piattaforma social proprio per la possibilità di usufruire delle novità introdotte in fatto di video e per il target vasto di età a cui si rivolge.

Durante il 2021, inoltre, è esploso Clubhouse, nuova stella tra i Social Media trend, lanciato nell'aprile dello scorso anno (ma giunto in Italia solo a gennaio); se ne sente parlare ormai tantissimo negli ultimi mesi, soprattutto tra coloro che hanno a disposizione un dispositivo iOS (in quanto l'app non è ancora disponibile per Android). Clubhouse si basa sulla creazione di vere e proprie stanze virtuali, all'interno delle quali le persone possono riunirsi e discutere su un determinato argomento. Nonostante non siano ancora stati implementati strumenti per uso professionale, Clubhouse offre numerose potenzialità da sfruttare in ambito marketing: è infatti possibile usarlo per fare personal branding e/o creare un proprio format legato all'attività o azienda. Questa potrebbe essere un'arma a doppio taglio, perché Clubhouse prevede che i contenuti delle stanze non possano essere modificati. Dunque, prima di prendere parola in una stanza, conviene sapere esattamente cosa si vuole comunicare a chi sta ascoltando: non si avrà possibilità di modificare il proprio intervento dopo essere andato live, dato che le conversazioni si svolgono in tempo reale. In ogni caso, se utilizzato con una consapevole strategia di personal branding, Clubhouse rappresenta un terreno ancora fertile, essendo un sito social di nicchia al quale si accede solo tramite invito. Si tratta, insomma, di una piattaforma ancora acerba, ma sono in programma implementazioni per integrare la pubblicità ed i contenuti per i brand. Una delle applicazioni più amate dalla nuova generazione, però, è TikTok, il vero social network del momento.

3.2 Tik Tok

Tik Tok, noto come *Douyin* in Cina, è un social network che permette ai suoi utenti di realizzare video divertenti e creativi di breve durata (tipicamente dai 15 ai 60 secondi), che spaziano da un

tema all'altro. Gli utenti possono creare nuovi contenuti o reinterpretare delle scene di un film o di una serie TV. Questa applicazione deriva da una precedente versione chiamata *Musical.ly*, un'app di videosharing dove padroneggiava il *lip sync*, aggiornata all'applicazione attuale il 2 Agosto 2018. L'app Musical.ly si è fusa con la già esistente App video TikTok, diventando una realtà unica, consacrandosi l'applicazione più scaricata al mondo.

Musical.ly Inc. è stata fondata in Cina nel 2015 da Alex Zhu e Luyu Yang. Prima di lanciare *Musical.ly*, essi avevano creato un social network con scopi educativi, dove gli utenti potevano imparare ed insegnare diverse materie mediante brevi video (3–5 minuti). Tuttavia, pur avendo trovato degli investitori disposti a supportare il progetto, la piattaforma non ebbe successo, così Alex Zhu e Luyu Yang decisero di cambiare target e di puntare sugli adolescenti. Il 24 luglio 2016 *Musical.ly* ha lanciato *live.ly*, una piattaforma per lo streaming video in diretta. Nel novembre 2017 l'azienda cinese ByteDance, sviluppatrice dell'aggregatore di notizie Toutiao, ha acquistato *musical.ly* per una cifra intorno ai 750 milioni di euro. Il 2 agosto 2018 ByteDance ha unito, attraverso un aggiornamento, le piattaforme TikTok e Musical.ly al fine di allargare la base degli utenti, mantenendo TikTok come nome. Gli utenti che utilizzano la piattaforma di TikTok sono noti come *tik tokers*.

Tik Tok è rapidamente diventato la principale app di social media al mondo relativamente a diverse categorie, in molto meno tempo rispetto ad altre app leader. Entro poco meno di un anno dal lancio della versione internazionale di Douyin (Tik Tok), l'app e diventata l'app iOS piu scaricata al mondo durante la prima meta del 2018, anno in cui e stata scaricata oltre 104 milioni di volte sull'App Store di Apple. Ad agosto 2020, TikTok ha superato la soglia di un miliardo di utenti in tutto il mondo considerando i download di App Store e Google Play (esclusi gli utenti Android cinesi), incassando 100 milioni di dollari dagli acquisti in-app, con un tasso di crescita quasi esponenziale (Figura 3.1).

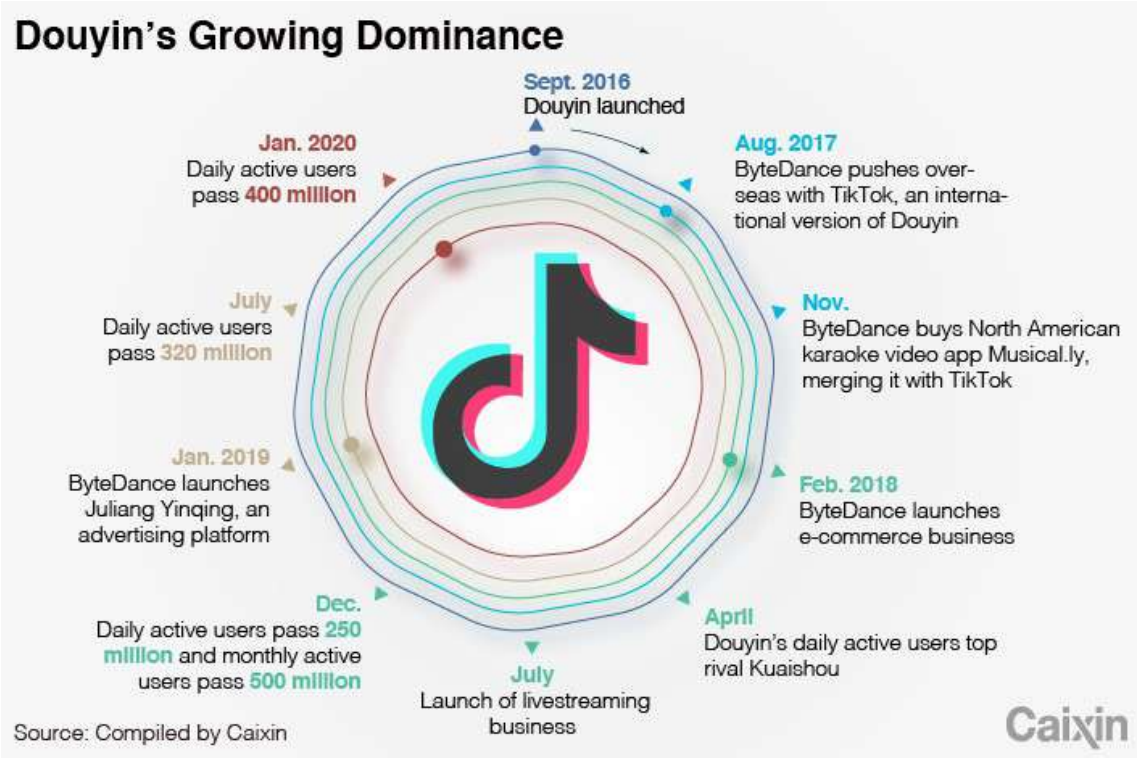


Figura 3.1: Evoluzione della piattaforma *Fonte: Caixin*

3.2.1 Descrizione e funzionamento

Gli utenti possono caricare video della durata massima di 60 secondi e aggiungere canzoni, suoni o voci da doppiare. L'applicazione mobile TikTok permette agli utenti di accelerare, rallentare, o modificare mediante un filtro, il suono o la musica di sottofondo, selezionabile all'interno di una vasta gamma di generi musicali. Tramite un'opzione dedicata, è possibile registrare audio e video della propria reazione, mentre la clip viene visualizzata sullo smartphone. Alternativamente l'opzione "duo" permette di condividere le videoripresa di due terminali mobili in un unico video. Tale funzionalità è utilizzata, ad esempio, fra due persone vicine per filmarsi l'una col cellulare dell'altra.

L'applicazione consente agli utenti di configurare i propri account come "privati". Il contenuto di questi account rimane a disposizione di TikTok, ma è visualizzabile soltanto dagli utenti autorizzati dal titolare, che, similmente ad altri social, possono scegliere se rendere il profilo pubblico o privato, ovvero se interagire soltanto con gli amici, tramite commenti, messaggi o video di "reazione" e "duo". TikTok utilizza l'intelligenza artificiale per analizzare gli interessi e le preferenze manifestate dagli utenti dell'applicazione, in modo tale da poter personalizzare i contenuti loro proposti (Figura 3.2).

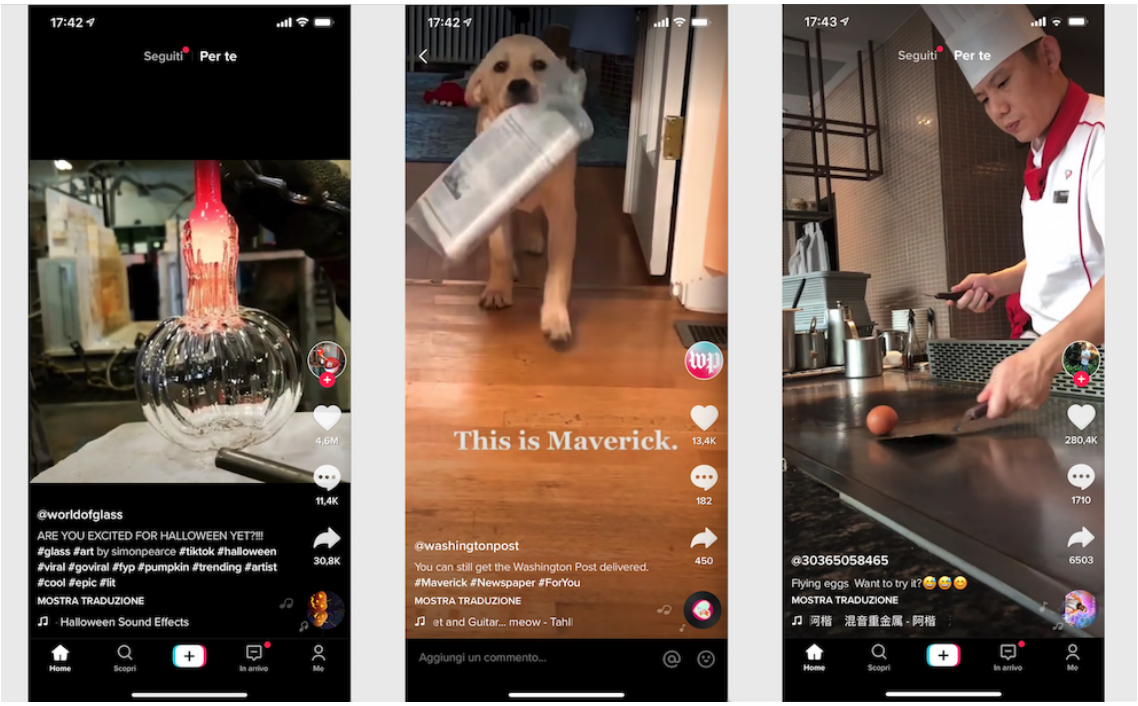


Figura 3.2: Immagini dalla Home Page della piattaforma, *Fonte: autoprodotta*

Appena autenticati all'interno della piattaforma, si potranno vedere diversi video in full screen, i quali rientrano nella sezione dei "per te", ovvero la sezione dei video consigliati. Solitamente si tratta di video con un elevato numero di visualizzazioni e di interazioni in un breve lasso di tempo. È possibile mettere *like* al video visualizzato, commentarlo, oppure inviarlo a terzi. Per scorrere è sufficiente effettuare il movimento di *swipe-up* col dito, ovvero di trascinamento dal basso verso l'alto. I video vengono prodotti automaticamente dalla piattaforma, per questo si dice che l'esperienza di visualizzazione è passiva.

Il principio di base su cui si fonda Tik Tok è la pubblicazione di video di durata massima pari ad un minuto, anche se solitamente si predilige la pubblicazione di video pari a 15 secondi. Il

formato dei video è di 1080×1920, si tratta quindi di video allungati, non di formato quadrato (come quello che viene utilizzato, ad esempio, su Instagram).

3.2.2 Caratteristiche

Ci sono diverse caratteristiche che identificano in maniera univoca questa piattaforma, tra queste citiamo:

- *Risoluzione HD e visualizzazione Full Screen:* la visualizzazione a schermo intero è un elemento fondamentale di Tik Tok, perché permette di catturare l'attenzione dell'utente in maniera più efficace, creando una "dipendenza" maggiore e inducendo l'utente stesso a trascorrerci più tempo.
- *Funzionalità avanzate di video editing:* considerando che la missione principale di TikTok è quella di aiutare le persone a condividere la loro creatività, vengono offerte funzionalità accattivanti, come filtri, effetti di bellezza, sticker e tool di editing.
- *Possibili background musicali:* è possibile aggiungere un clip musicale al video pubblicato.
- *Challenge:* gli hashtag ricoprono un ruolo fondamentale all'interno di questa piattaforma. Alcuni hashtag vengono considerati dei trend a cui gli utenti possono ispirarsi per creare nuovi video.

CHALLENGE

Una challenge è una "sfida" virale a cui sono chiamati gli utenti di TikTok, per cui molti ragazzi decidono di copiare il comportamento contenuto all'interno di un video per replicarlo a proprio modo. Questo rappresenta il vero collante che rende la semplice piattaforma un vero e proprio social. Ogni challenge è identificata da un hashtag che gli utenti utilizzano nella *caption*, ovvero nella didascalia, del video pubblicato. Tutti i video corrispondenti all'hashtag della challenge vengono raggruppati, in modo da offrire all'utente una maggiore visibilità.
Le challenge, nell'ultimo periodo, sono prevalentemente innocue, ovvero sono sfide relative ad attività che non hanno risvolti pericolosi, poiché TikTok ha incrementato i controlli di sicurezza. Non è da escludere, però, la possibilità che ci siano challenge pericolose, o negative, a cui un utente può prendere parte; per questo occorre prestare molta attenzione.

3.2.3 A chi è rivolta la piattaforma

Secondo la ricerca svolta da Johannes Ahlse, Felix Nilsson, Nina Sandström, relativamente alla pubblicazione *"It's time to TikTok"*, ci sono diverse motivazioni che spingono un ragazzo a partecipare ad una challenge. Tutti questi aspetti vengono posti all'interno di una piramide con un approccio top-down. La figura viene letta dall'alto, a partire dall'aspetto maggiormente motivante, fino alla base della piramide, in cui viene riportato il motivo meno importante. Sono state individuate sei categorie (Figura 3.3):

1. Entertainment
2. Personal Identity
3. Socializing
4. Status
5. Convenience
6. Information Seeking

Figura 3.3: Piramide che mostra i motivi per cui gli utenti prendono parte alle challenges, *Fonte: Jonkoping University*

Alla base vediamo l'apprendimento ("information seeking"), che è stato visto come un aspetto positivo dell'impegno nelle sfide, ma non come una motivazione principale. L'aspetto rilevante è proprio quello relativo al divertimento, che risulta essere il motivo chiave per cui i teenager decidono di prendere parte alle challenges.

TikTok, infatti, si focalizza proprio sulla generazione Z, ovvero la generazione formata da coloro che sono nati dopo il 1995. Questo avviene perché i ragazzi della generazione Z sono stati abituati a crescere con lo smartphone, piuttosto che tablet, computer e dispositivi digitali tra le mani. Sono quasi ossessionati da contenuti visivi e questo social network è perfetto per condividere col resto del mondo la propria vita. Nella Figura 3.4 si riporta un'analisi degli utenti sulla base di genere ed età:

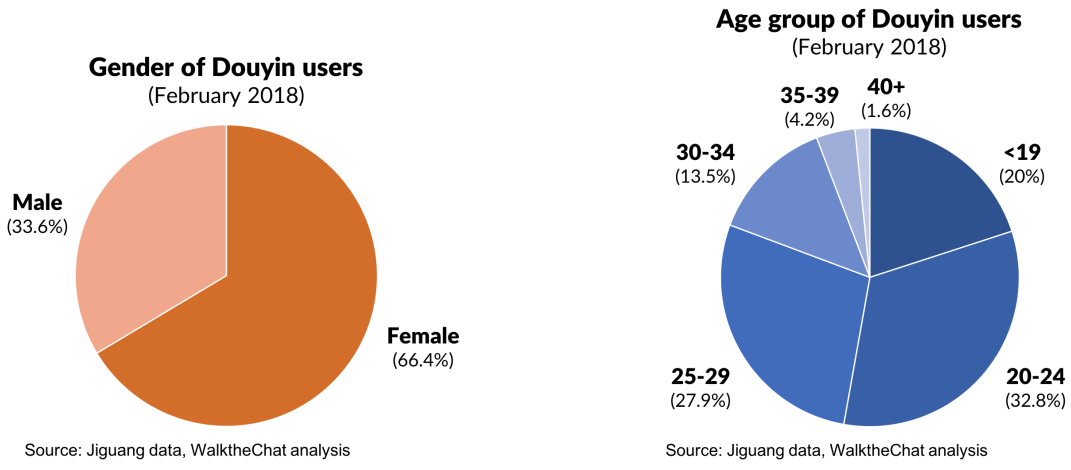


Figura 3.4: Genere ed età degli utenti di Tik Tok fino al 2018 *Fonte: WalktheChat analysis*

Come vediamo dalla Figura 3.4, gli utenti sono prevalentemente donne e, per più della metà, gli iscritti hanno un'età inferiore ai 24 anni. Con il passare del tempo, però, l'età media si sta alzando, fenomeno che ha caratterizzato l'evoluzione della quasi totalità di social presenti sul mercato. Come possiamo osservare dalla Figura 3.5, durante l'anno 2020 si sono iscritti prevalentemente utenti

con fascia d'età compresa tra 31 e 35 anni, sinonimo del fatto che TikTok sta iniziando a ricoprire anche una diversa fascia di utenti rispetto al passato.

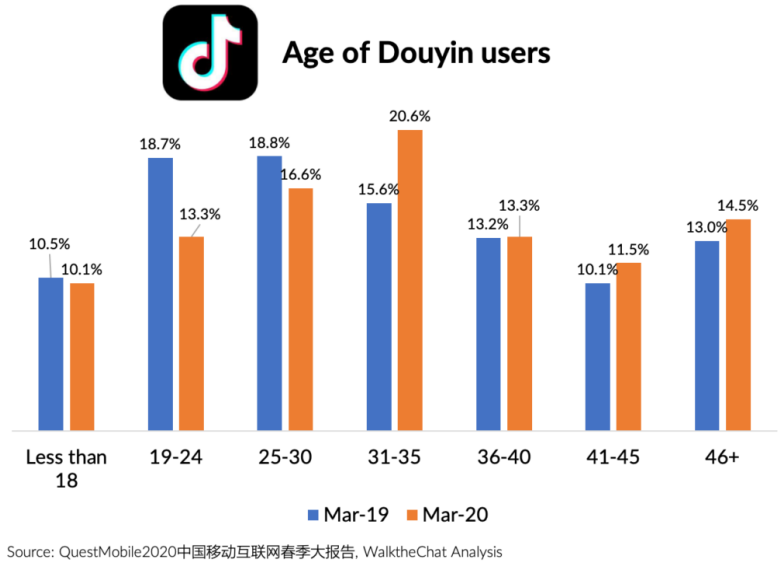


Figura 3.5: Età degli iscritti negli anni 2019 e 2020 *Fonte: WalktheChat analysis*

3.2.4 Principali problematiche

Quando si parla di un social network così diffuso, si fa riferimento inevitabilmente non solo ai pregi di una piattaforma di questo tipo, ma anche ai difetti, che, in alcuni casi, possono diventare veri e propri pericoli. Alcuni svantaggi o possibili problematiche sono le seguenti:

- *Accounts fake:* c'è la possibilità che qualche malintenzionato crei account fasulli, che non corrispondano alla reale identità della persona stessa. Questo può mettere in pericolo gli utenti che entrano in contatto con questo tipo di account, in particolar modo se si tratta di adolescenti e, soprattutto, bambini.
- *Abuso di utilizzo e dipendenza:* questo social, in particolare, è stato pensato per catturare l'attenzione completa dell'utente, grazie ai video full screen e la navigazione passiva; per questo è importante avere il pieno controllo sull'utilizzo della piattaforma. Nel momento in cui la maggior parte di utenti sono componenti della generazione Z, il rischio che si incorra in una forma di dipendenza è molto alto. L'applicazione usa l'intelligenza artificiale per capire quali sono gli interessi principali dell'utente, così da proporre contenuti mirati e prolungare la permanenza all'interno dell'app. Questo permette ai creator di TikTok di avere una potenza maggiore dal punto di vista del marketing e delle sponsorizzazioni.
- *Contenuti offensivi e calo dell'autostima:* uno dei problemi principali che si è riscontrato nei giovani è la crescente insicurezza e l'innalzamento dei numeri relativamente alla depressione e ai suicidi. Questo dato è allarmante e parte del problema potrebbe essere imputato all'abuso dei social. Molto spesso si cercano di inseguire canoni estetici irraggiungibili o non realistici che vengono proposti sui social, e questo contribuisce al calo dell'autostima personale degli adolescenti. In particolare, in questa piattaforma specifica, c'è un inasprimento di odio e contenuti offensivi, che vanno ad acuire le problematiche sociali già presenti.

3.2.5 Dati generali

L'app ha 800 milioni di utenti attivi in tutto il mondo (guadagnati in tre anni solamente). Il Paese che ha scaricato più volte l'app è l'India, con 611 milioni di download. A seguire troviamo la Cina con 197 milioni di download, gli USA con 165 milioni e l'Italia con 8 milioni.

Gli utenti di TikTok trascorrono una media di 52 minuti al giorno sulla piattaforma, ovvero circa un'ora, attirati dalla navigazione passiva e dalla challenge riproposte da diversi utenti. Il 90% degli utenti accede all'app quotidianamente, anche più di una volta al giorno. Questo dato riconferma la potenza psicologica di TikTok. Il numero di utenti è continuamente in crescita; basti pensare che in Italia, in soli 3 mesi, gli utenti sono triplicati.

Come è possibile notare dalla Figura 3.6, TikTok ha superato ogni social presente sul mercato, mentre al secondo posto troviamo YouTube, seguito da Instagram.

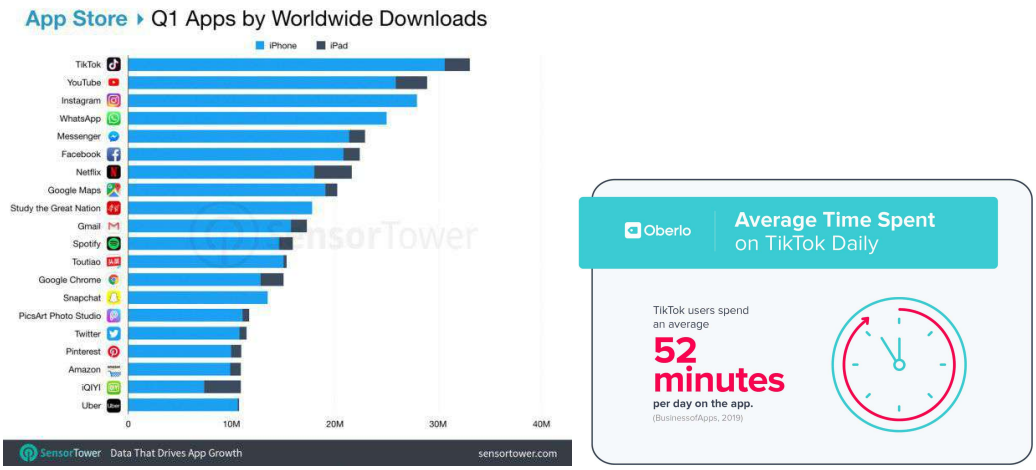


Figura 3.6: App di maggior successo e tempo medio di utilizzo *Fonte: sensortower.com*

4. Costruzione del dataset di riferimento

4.1 Python

Python è un linguaggio di programmazione di più "alto livello" rispetto alla maggior parte degli altri linguaggi, orientato agli oggetti, adatto, tra gli altri usi, a sviluppare applicazioni distribuite, scripting, computazione numerica e system testing. Ideato da Guido van Rossum all'inizio degli anni novanta, il nome fu scelto per la passione dello stesso inventore verso i Monty Python e per la loro serie televisiva Monty Python's Flying Circus. Spesso viene, anche, studiato tra i primi linguaggi per la sua somiglianza ad uno pseudo-codice e, di frequente, viene usato per simulare la creazione di software grazie alla flessibilità di sperimentazione consentita, che permette al programmatore di organizzare le idee durante lo sviluppo. È un linguaggio multi-paradigma che ha, tra i principali obiettivi, la dinamicità, la semplicità e la flessibilità. Python supporta il paradigma object oriented, la programmazione strutturata e molte caratteristiche di programmazione funzionale e riflessione.

I principali punti di forza di Python sono:

- *Free:* Python è completamente gratuito ed è possibile usarlo e distribuirlo senza restrizioni di copyright. Nonostante sia free, da oltre 25 anni Python ha una comunità molto attiva, e riceve costantemente miglioramenti che lo mantengono aggiornato e al passo coi tempi.
- *Multi-paradigma:* Python è un linguaggio multi-paradigma, che supporta sia la programmazione procedurale (che fa uso delle funzioni), sia la programmazione ad oggetti (includendo funzionalità come l'ereditarietà singola e multipla, l'overloading degli operatori, e il duck typing). Inoltre supporta anche diversi elementi della programmazione funzionale (come iteratori e generatori).
- *Portabilità:* Python è un linguaggio portabile sviluppato in ANSI C. È possibile usarlo su diverse piattaforme, come Unix, Linux, Windows, DOS, Macintosh, Sistemi Real Time, OS/2, cellulari Android e iOS. Ciò è possibile perché si tratta di un linguaggio interpretato; quindi lo stesso codice può essere eseguito su qualsiasi piattaforma, purché abbia l'interprete Python installato.
- *Facilità di utilizzo:* Python è un linguaggio di alto livello che è, al tempo stesso, semplice e potente. La sintassi, le funzioni e i diversi moduli, che sono già inclusi nel linguaggio, sono

consistenti, intuitivi e facili da imparare; il design del linguaggio si basa sul principio del least astonishment (cioè della "minor sorpresa": il comportamento del programma coincide con quanto ci si aspetta).

- *Ricco di librerie:* ogni installazione di Python include la *standard library*, cioè una collezione di oltre 200 moduli per svolgere i compiti più disparati, come ad esempio l'interazione con il sistema operativo e il file system, nonché la gestione di diversi protocolli. Inoltre, il Python Package Index consente di scaricare ed installare migliaia di moduli aggiuntivi creati e mantenuti dalla comunità.
- *Performante:* anche se Python è considerato un linguaggio interpretato, i programmi vengono automaticamente compilati in un formato chiamato bytecode prima di essere eseguiti. Questo formato è più compatto ed efficiente, e garantisce, quindi, prestazioni elevate. Inoltre, diverse strutture dati, funzioni e moduli di Python sono implementati interamente in C per essere ancora più performanti.
- *Gestione automatica della memoria:* Python è un linguaggio di alto livello che adotta un meccanismo di *garbage collection* che si occupa automaticamente dell'allocazione e del rilascio della memoria. Questo consente al progammatore di usare le variabili liberamente, senza doversi preoccupare di dichiararle e di allocare e rilasciare spazi di memoria manualmente (cosa che è, invece, necessaria in linguaggi di più basso livello, come il C o il C++).
- *Integrabile con altri linguaggi:* oltre all'interprete classico scritto in C (e chiamato CPython), esistono, anche, altri interpreti che consentono l'integrazione con diversi altri linguaggi. IronPython consente di utilizzare Python all'interno del framework .NET, di usarne le sue funzioni, e di interagire con altri linguaggi .NET. Per poter, invece, integrare Python e Java è possibile utilizzare Jython. Esistono, poi, altri interpreti, come PyPy; un'implementazione altamente performante scritta in Python.

4.1.1 Possibilità di utilizzo

La dotazione standard e le librerie di terze parti completano Python con funzionalità che lo rendono uno strumento duttile in svariati ambiti. Tra questi citiamo:

- programmazione GUI;
- sviluppo web;
- accesso ai database;
- realizzazioni di giochi;
- realizzazione di applicazioni scientifiche.

4.1.2 Librerie utilizzate

Pyhton offre una grande moltitudine di librerie; in particolare sono risultate molto utili, al fine di completare questo lavoro, librerie come Pandas, Matplotlib, NetworkX e Scikit-learn, ognuna delle quali è stata utilizzata per un diverso scopo e, molto spesso, sono state combinate tra loro.

Pandas è una libreria software scritta per la manipolazione e l'analisi dei dati. In particolare, offre strutture dati e operazioni per manipolare tabelle numeriche e serie temporali. Il nome deriva dal termine "panel data", termine econometrico per set di dati che include osservazioni su più periodi di tempo per gli stessi individui. È stato utilizzato Pandas per la lettura di csv esterni, l'importazione e la manipolazione degli stessi.

Matplotlib è una libreria adibita alla creazione di grafici. È possibile combinarla alla libreria matematica NumPy, per calcolare determinate metriche da graficare. La stessa fornisce anche API orientate agli oggetti che permettono di inserire grafici all'interno di applicativi, usando toolkit GUI generici. È stata particolarmente utile per la creazione di grafici durante l'intero lavoro svolto.

È stata utilizzata anche Seaborn, una libreria di visualizzazione dati Python basata su Matplotlib. Essa fornisce un'interfaccia di alto livello per disegnare grafici statistici interessanti e informativi.

Tramite NetworkX sono stati creati non solo grafici, ma anche misurazioni critiche relative ai grafi ottenuti. Come visto in precedenza, le reti sono fondamentali per comprendere le connessioni tra nodi ed archi, e questa analisi è stata resa possibile grazie all'utilizzo della suddetta libreria.

Scikit-learn è un'altra libreria utilizzata all'interno del progetto; è open source e permette di effettuare l'apprendimento automatico per il linguaggio di programmazione Python. Contiene algoritmi di classificazione, regressione e clustering (raggruppamento), support vector machine, regressione logistica, classificatore bayesiano, k-means e DBSCAN, ed è progettato per operare con le librerie NumPy e SciPy. SciPy è una libreria open source di algoritmi e strumenti matematici che uscì dalla collezione originale di moduli d'estensione Multipack per Python di Travis Oliphant del 1999. Contiene moduli per l'ottimizzazione, l'algebra lineare, l'integrazione, funzioni speciali, FFT, elaborazione di segnali ed immagini, solver ODE e altri strumenti comuni nelle scienze e nell'ingegneria.

4.2 API: Application Programming Interface

Le API sono un insieme di regole e di codici, ovvero vere e proprie librerie di funzioni che permettono di far interagire un programma con un altro. Semplificando, le API sono una sorta di medium: qualcosa che sta in mezzo tra un programma che contiene dei dati che ci interessano (Tik Tok, ad esempio) e chi prova ad accedervi (con un altro programma, generalmente). Le API sono una porta di accesso a quel programma; esse consentono di interrogarlo per ottenere informazioni non immediatamente visibili altrimenti. Per richiedere al programma di effettuare determinate operazioni, è necessario seguire delle regole, dettate, appunto, dalle API. Lo svantaggio delle API è che ogni programma ha le sue: le regole con cui interrogherò Tik Tok, quindi, non sono le stesse con cui posso interrogare Twitter.

Le API non servono solo ad accedere ai dati di un programma, ma anche per costruire nuovi tool, i quali sfruttano quel programma stesso. Un esempio lo possiamo identificare in un'azienda che distribuisce libri. Il distributore potrebbe offrire ai propri clienti un'app che consente ai commessi delle librerie di controllare la disponibilità dei libri. L'app potrebbe essere, però, costosa da sviluppare, vincolata a una piattaforma, esigente in termini di tempi di sviluppo e soggetta a manutenzione costante. In alternativa, il distributore può fornire un'API per controllare la disponibilità in magazzino.

Questo approccio offre diversi vantaggi:

- consentire ai clienti di accedere ai dati tramite un'API aiuta a raccogliere le informazioni in un inventario unificato;
- il distributore può apportare modifiche ai sistemi di distribuzione interni senza impatto sui clienti, a condizione che il comportamento dell'API non subisca modifiche;
- con un'API pubblicamente disponibile, gli sviluppatori che lavorano per il distributore, i rivenditori e i provider di altri servizi possono sviluppare un'app che aiuti i consumatori a trovare i libri che stanno cercando, incrementando le vendite o altre opportunità di business.

È possibile connettersi alle API e creare applicazioni che usano i dati o le funzionalità esposte da esse tramite una piattaforma di integrazione distribuita, in grado di connettere ogni elemento, inclusi i sistemi esistenti e l'Internet of Things (IoT). Esistono tre diversi approcci ai criteri di rilascio delle API, come espresso in Figura 4.1.

Figura 4.1: Tipi di rilascio di API

L'esposizione delle API ai partner o al pubblico consente di:

- creare nuovi canali di reddito o espandere quelli esistenti;
- espandere la copertura del brand;
- facilitare l'apertura all'innovazione o l'incremento dell'efficienza grazie allo sviluppo e alla collaborazione con l'esterno.

Tornando all'esempio di distributore di libri, uno dei partner dell'azienda sviluppa un'app che aiuta i clienti a trovare i libri sugli scaffali della libreria. Questa esperienza migliorata aumenta il numero di acquirenti nel negozio (il cliente del distributore) ed espande un canale di reddito già esistente. Una terza parte utilizza forse un'API pubblica per sviluppare un'app che consente di acquistare i libri direttamente dal distributore anziché dalla libreria. Ciò apre un nuovo canale di reddito per il distributore. La condivisione delle API, sia con partner selezionati che con chiunque altro, può avere effetti positivi. Ogni partnership estende il riconoscimento del marchio ben oltre l'impegno del team che si occupa delle operazioni di marketing. Esporre la tecnologia a chiunque, come accade con un'API pubblica, incoraggia gli sviluppatori a creare un sistema di app incentrato sull'API. Maggiore è il numero di persone che utilizzano la tecnologia condivisa, maggiori le opportunità di business.

4.2.1 Evoluzione delle API

L'avvento delle API risale agli albori dell'era dell'informazione, ben prima dei personal computer. A quell'epoca, un'API era in genere utilizzata come libreria per un sistema operativo. Era a livello locale rispetto al sistema in cui operava, sebbene a volte passasse messaggi tra mainframe. Dopo circa 30 anni, le API sono emerse dai loro ambienti locali. Nei primi anni 2000 si trasformarono in un'importante tecnologia per l'integrazione remota dei dati. Le *API Remote* sono concepite per interagire tramite una rete di comunicazione. Si chiamano API remote perché le risorse gestite dall'API sono esterne al computer che invia la richiesta. Poiché il canale di comunicazione più diffusamente utilizzato è Internet, la maggior parte delle API è progettata in base a standard web. Non tutte le API remote sono API web, ma possiamo affermare che le API web sono remote. Queste ultime utilizzano in genere il protocollo HTTP per richiedere messaggi e fornire una definizione della struttura dei messaggi di risposta. Questi ultimi assumono in genere la forma di file XML o JSON, due formati diffusi perché presentano i dati in modo da consentire alle altre app di gestirli

facilmente. Le API si sono evolute in API web, ormai molto diffuse. Nel tempo sono stati fatti molti tentativi per semplificarne il design e renderne l'implementazione più utile.

4.2.2 TikTokApi Wrapper: l'API utilizzata

Per questa analisi è stata utilizzata un'API non ufficiale di Tik Tok, creata da David Teather. Si tratta di un wrapper in grado di effettuare il download di dati provenienti da Tik Tok, selezionandoli per hashtag, suono o per utente. TikTokApi utilizza Playwright, che è una libreria Python per automatizzare i browser Chromium, Firefox e WebKit con una singola API.

4.3 Challenge: quando diventano pericolose

Come precedentemente detto, su Tik Tok, assumono vitale importanza i trend da seguire, attraverso la simulazione di challenge. Le challenge sono le sfide che gli utenti della rete fanno tra loro, per divertirsi ed intrattenere il pubblico. Alcune sono così divertenti da diventare virali: tutti si cimentano nelle imprese più esilaranti, facendo a gara a chi fa la sfida più divertente, per goliardia ma anche per guadagnare follower, like e consensi. Il problema principale è che ne esistono alcune che possono considerarsi "sfide al rischio". Purtroppo ci sono molti adolescenti, o addirittura bambini, che cercano di simulare alcune sfide che vengono loro proposte, finendo per mettersi in serio pericolo.

Attorno al mese di settembre 2020 presero piede diverse challenge allarmanti, ad esempio:

- *Planking challenge*: si tratta di sdraiarsi in posti sempre più inconsueti. Il centro di un incrocio stradale, la ringhiera di una scala, il cornicione di un palazzo alto, qualsiasi posto non adatto a sdraiarsi senza rischiare di cadere e farsi male. La pericolosità di questa challenge risiede nel fatto che spesso viene eseguita da ragazzi molto giovani, come è la quasi totalità degli utenti di Tik Tok, ovvero dai 12 ai 14 anni. In alcune città italiane sono stati colti ragazzini di queste età mentre si sfidavano a chi si sdraiava più vicino alla carreggiata e alle automobili in arrivo. Queste challenge pericolose poi venivano pubblicate su WhatsApp o su Tik Tok, appunto.
- *Coronavirus challenge*: questa sfida è diventata virale durante la quarantena, creando un grande allarme. La gran parte dei giovanissimi che vi ha aderito evidentemente non ha preso sul serio il virus. Essa consiste nel leccare i servizi igienici, in modo particolare le tavolette dei wc, incuranti dei rischi igienici conseguenti. È stata ideata da Ava Louise, una influencer di 21 anni che su Instagram ha 175k follower.
- *Skullbreaker challenge*: in questa sorta di "gioco" si cerca di far cadere una persona all'indietro, facendole sbattere la testa: tre persone si mettono d'accordo per saltare tutte insieme nello stesso momento ma, sfortunatamente per la persona che sta nel mezzo, quelle di lato si sono messe d'accordo tra loro per darle un calcio mentre sta compiendo il salto, facendole perdere l'equilibrio e cadere.
- *La sfida della cannella*: la sfida della cannella consiste nel filmarsi mentre si ingoia un cucchiaio pieno di cannella, senza l'aiuto dell'acqua. La pericolosità di questa sfida sta nel fatto che la cannella secca la bocca e la gola, provocando tosse, vomito, irritazione e difficoltà respiratorie che possono causare il collasso dei polmoni.

Oltre a queste challenge, ce ne sono tantissime altre, che rendono il mondo del web particolarmente pericoloso, soprattutto per chi non riesce ad individuare il rischio del trend stesso. Un chiaro esempio è da ricercare nella triste storia di una bambina di 10 anni, palermitana, che non ha saputo cogliere il pericolo di un trend virale fino a poco tempo fa. La challenge in questione prendeva il nome di "blackout challenge", una prova di soffocamento estrema. Il gioco del soffocamento, detto anche "pass out challenge", consisteva nel filmarsi mentre ci si provocava un'asfissia temporanea. Lo scopo di questa pericolosa challenge era quello di provocarsi piacere e

una sensazione di euforia che, si dice, sia legata ad esperienze del genere. La prova prevedeva che si stringesse attorno al collo una cintura. La bambina prese quella di un accappatoio, provocandosi così la morte cerebrale.

Recentemente, secondo lo studio svolto, Tik Tok ha iniziato a prestare molta attenzione ai trend considerati pericolosi per la salute degli utenti, togliendo tutto ciò che concerne gli hashtag relativi a challenge allarmanti. Con lo scopo di individuare la propagazione di challenge positive e negative, sono state considerate quelle disponibili al momento del lavoro, considerando che sulla piattaforma non c'è più la minima traccia delle sfide precedentemente citate.

4.4 Individuazione delle challenge

Sono state individuate una serie di challenge di partenza, considerando quelle con un numero elevato di video presenti e scegliendo degli hashtag il più possibile coincidenti con il nome del suono utilizzato per il video.

Le challenge scelte si suddividono in positive e negative, in base all'impatto che possono avere sugli utenti. È chiaro che un hashtag considerato negativo, diventa tale nella misura in cui un utente decide di estremizzare la sfida che è stata posta.

Sono state individuate le seguenti challenge positive:

- *Bussitchallenge:* principalmente ricreata da ragazze che si presentano senza trucco e in tenuta casalinga, per poi passare improvvisamente ad essere vestite al meglio, sistemate in trucco e parrucco, il tutto enfatizzato dall'utilizzo delle transition.
- *Copinesdancechallenge:* si esegue un balletto, diventato un trend associato alla musica utilizzata.
- *Emojichallenge:* sfida di grande successo, in cui l'obiettivo è quello di simulare le emoji che si utilizzano nelle chat attraverso le espressioni facciali.
- *ITookANap:* si riprende una persona o un animale domestico che sta facendo un pisolino.
- *Colpiditesta:* si tratta di un gioco in cui la persona che viene ripresa deve cercare di colpire un pallone virtuale con la propria testa, simulando dei veri e propri palleggi.
- *Boredinthehouse:* challenge che ha acquisito grande successo durante il periodo di quarantena, in cui le persone si mostravano annoiate in casa, intente ad effettuare azioni quotidiane.
- *Plankchallenge:* sfida di resistenza fisica, in cui si devono effettuare molti plank, in modo da mettere in risalto la propria forma fisica e la sportività della persona.

Le challenge considerate negative sono:

- *Silhouettechallenge:* da un momento all'altro cambiano le luci e ci si mostra senza troppi vestiti, rimanendo in ombra. Il brano scelto come sottofondo è "Put your head on my shoulder". Il pericolo consiste nel fatto che è stato scoperto il modo per eliminare il filtro. Questo potrebbe mettere a rischio la privacy di centinaia di migliaia di utenti. Sebbene il trend sia nato per combattere ogni forma di discriminazione legata al peso e alla forma del corpo, non è auspicabile che le proprie foto nude possano circolare sul web.
- *Bugsbunnychallenge:* le ragazze simulano di avere le orecchie da coniglio, utilizzando i piedi e una prospettiva di ripresa che permette di ottenere la sagoma corretta. La negatività di questa challenge sta nel fatto che diversi utenti la effettuino con pochi indumenti, permettendo la visione di parti intime.
- *Strippatiktok:* in questa challenge diverse stripper professioniste mostrano la loro vita e i loro guadagni, inducendo possibili utenti adolescenti all'emulazione.
- *Firewroks:* consiste nel far esplodere fuochi d'artificio in prossimità di soggetti ed oggetti a rischio. Questo può essere potenzialmente dannoso per chi si trova vicino ad un'esplosione indotta da persone non professioniste, poco consce del pericolo della loro azione.
- *Fightchallenge:* è la challenge in cui adolescenti, ragazzi o adulti ne riprendono altri mentre si picchiano.

- *Sugarbaby:* alcune persone mostrano la vita e gli svaghi che si concedono grazie a particolari lavori. Viene considerata negativa per l'influenza che potrebbe avere sugli adolescenti e le nuove generazioni.
- *Updownchallenge:* è un gioco di coppia in cui ci si afferrano mani e piedi e si cerca di ribaltare la posizione attuale da terra. Il rischio è quello di sbattere la testa nel momento in cui ci si capovolge, con il rischio di farsi davvero male.

In tutto sono state individuate sette challenge positive e sette negative. Successivamente sono stati scaricati i dati relativi all'hashtag che rappresenta ciascuna di queste challenge.

4.5 Raccolta Dati

Dopo aver individuato una serie di challenge, alcune positive e altre negative, è stato necessario procedere alla raccolta dei dati. Come precedentemente spiegato, è stata utilizzata l'API non ufficiale di Tik Tok, realizzata da David Teather, per poi essere manipolata secondo le esigenze del caso. Come si evince dal codice in appendice, è stato necessario creare un meccanismo di raccolta dati.

Per avviare la procedura, è sufficiente aprire il prompt dei comandi, identificare la directory contenente il file "main.py" e avviare il processo relativamente ad una determinata challenge. Si dovranno, quindi, digitare i seguenti comandi:

```
cd <filepath>
python main.py -c <nomechallenge>
```

In questo modo viene attivato il file main.py all'interno della directory specificata, passando ad esso come challenge quella definita all'attivazione attraverso il comando. Questa procedura è mostrata all'interno della Figura 4.2:

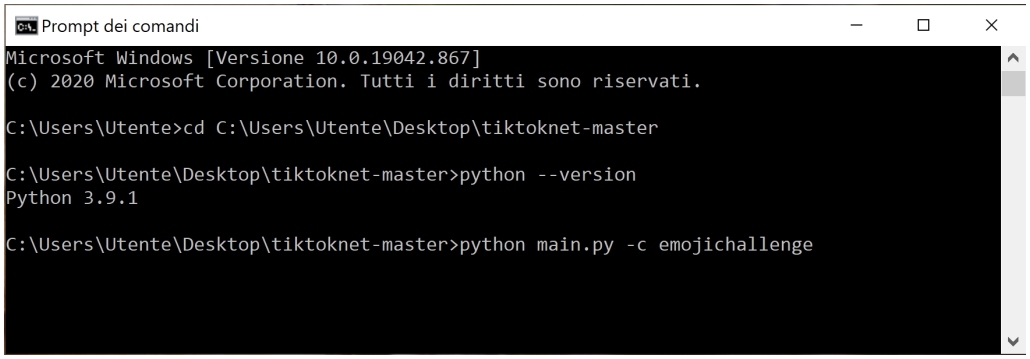


Figura 4.2: Comandi di avvio per la raccolta dati dal prompt dei comandi *Fonte: autoprodotta*

4.6 Algoritmo sviluppato

È stato sviluppato un algoritmo per la raccolta dei dati, la loro elaborazione ed il conseguente lavoro di ETL. In seguito è stato utilizzato Network per la creazione delle reti, oltre all'esplicitazione di dati e parametri utili alla comprensione e allo studio delle reti stesse.

4.6.1 Struttura dell'algoritmo sviluppato

L'algoritmo sviluppato è composto dai file riportati in Figura 4.3.

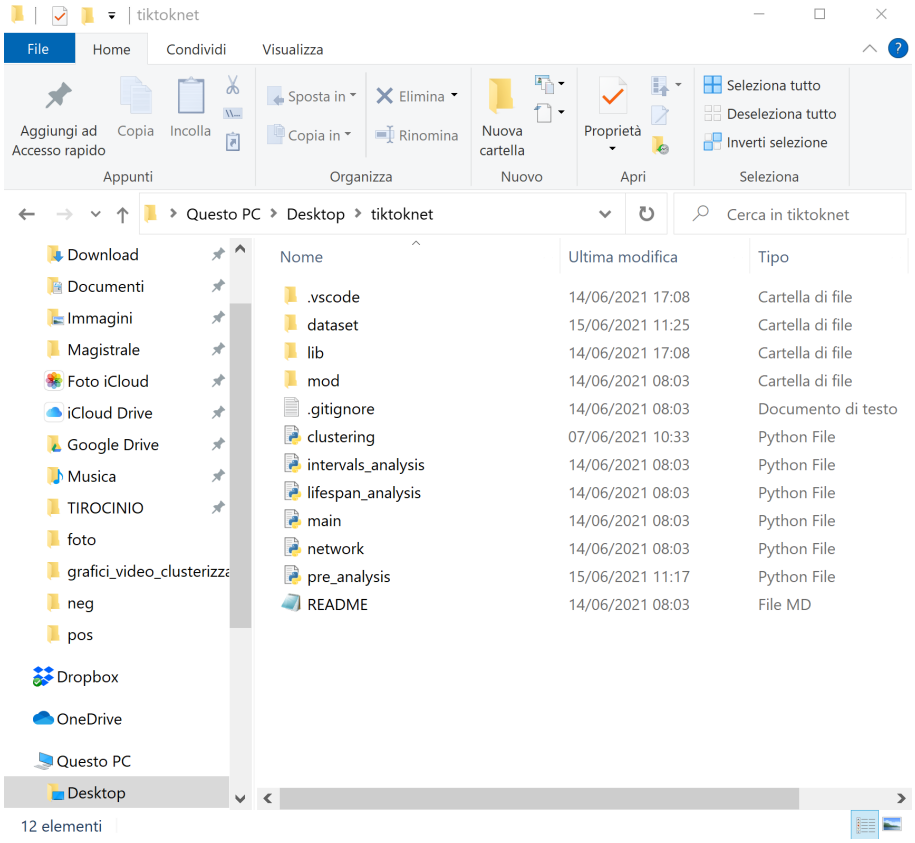


Figura 4.3: Cartella tiktoknet-master *Fonte: autoprodotta*

All'interno della *working directory* troviamo la cartella "dataset", che contiene, a sua volta, un'altra cartella denominata "authors". All'interno di "lib" troviamo i file: "challenges.py", "tiktokdataset.py", "tiktoknetwork.py" e "utils.py". In "mod" è presente un file denominato browser.py, che approfondiremo in seguito. Come file principali sono presenti "main.py", "network.py", "pre_analysis", "lifespan_analysis", "intervals_analysis" e "clustering".

4.6.2 Funzionamento dell'algoritmo

Il file "challenges.py" contiene la lista delle challenge considerate, attraverso un dizionario che ne specifica il nome e l'eventuale URL del video originale (se è stato reperito). All'interno del file main.py, posizionato nella directory principale, viene inizializzata una nuova istanza tramite il comando TikTokApi.get_instance; dopodiché si costruisce il dataset procedendo con l'estrazione dei dati a partire dall'hashtag considerato. Viene, infatti, richiamata la funzione "buildDatasetByHashtag" presente all'interno del file "tiktok_dataset.py", a cui vengono passati diversi parametri: la variabile api, che consiste nell'istanza appena creata, il nome della challenge e l'eventuale URL del video originale, oltre al count, ovvero al numero desiderato di video scaricati relativi alla challenge in questione. Attraverso questa funzione si estraggono una serie di video relativi all'hashtag considerato, scegliendolo tra quelli presenti all'interno del dizionario creato nel file "challenges.py". Si identifica inoltre il video originale attraverso un flag, solo se il video originale sia realmente presente. Tutti i dati scaricati vengono salvati all'interno della cartella "dataset", con l'attribuzione di un nominativo standard, composto da: "dataset_hashtag.csv". Al termine di questa operazione viene stampato sul video un messaggio del tipo: "downloaded <numero_di_video> for <hashtag_considerato>".

Successivamente, dal "main.py", viene richiamata una seconda funzione: "pubAuthList", presente all'interno di "tiktok_dataset.py". Questa ha l'obiettivo di restituire la lista di utenti che hanno la sezione di video piaciuti pubblici. Il problema principale è proprio il fatto che la maggior parte degli utenti di TikTok ha i like impostati come privati. Tutti questi utenti sono impossibili da elaborare e vengono considerati come dati persi. Tramite questa funzione è possibile, comunque, effettuare uno studio relativo a coloro che hanno reso pubblica la lista dei video a cui hanno messo like. Viene creato un csv denominato "pubLiked_hashtag", il quale contiene tre colonne specifiche: "author_id","authorsec_Uid" e "author_uniqueId", ovvero l'id dell'utente con like pubblici, il suo secondo id e il suo *nickname*. Questo file prende il nome di pubLikedhashtag e viene inserito all'interno della cartella authors presente in dataset.

Durante questa fase, viene mostrato sul video l'andamento del parser, che scorrerà tutti i video scaricati nella prima fase. Uno dei problemi più rilevanti si è presentato per ciò che concerne un'impostazione propria dell'API. Dopo svariati minuti, infatti, la connessione al browser si interrompeva improvvisamente, non permettendo di terminare né il download di video, né il completamento del parsing sugli stessi. Per poter ovviare a questo problema, è stato modificato il file "browser.py" presente nella libreria di TikTokApi. È possibile trovarlo all'interno della directory "tiktoknet", nella cartella "mod". È stato, così, riavviato il modulo "playwright", con conseguente richiamo del comando "get_instance", a partire dall'esatto punto in cui si era interrotto. Nel momento in cui avviene il riavvio, viene stampata sul video la dicitura "Recovery mode".

Terminata la fase di parsing per gli utenti con like pubblici, viene avviata la funzione "checkConnections". In questa fase viene letto il csv originario, in cui sono riportati tutti i video scaricati relativi ad una challenge. Vengono, inoltre, aggiunte le seguenti colonne:

- "likedBy_id": rappresenta l'id dell'utente che ha messo like al video considerato (ovvero al video presente in quella riga del csv);
- "likeBy_secUid": rappresenta il secondo identificativo dell'utente che ha messo like al video considerato;
- "likedBy_uniqueId": rappresenta il nickname dell'utente che ha messo like al video considerato.

Se un video non ha ancora ottenuto nessuna corrispondenza con i like, il valore all'interno della cella "likedBy_id" sarà pari ad un trattino "-". Una volta che è stato identificato l'id del video (presente nel csv originale) all'interno della lista dei video piaciuti (considerando coloro che hanno i like pubblici), allora si andrà a verificare il valore della cella: se la cella avrà il valore "-" significa che non è stato trovato ancora nessun utente che abbia messo mi piace a quel determinato video; quindi verranno aggiunti i dati dell'utente corrispondente trovato. In caso contrario, verrà creata una nuova riga identica a quella attuale e verrà aggiunto l'utente che ha messo like. Questa parte è, senza dubbio, la più onerosa in termini di tempo; per questo motivo è stata aggiunta una *progress bar*, cosicché venga indicato il progresso del processo in tempo reale. Al termine del processo, il nuovo csv ottenuto viene salvato e si tratterà di un file contenente tutte le nuove connessioni, denominato "dataset_<hashtag>_connections". Inoltre, se viene trovato un nuovo video relativo alla challenge considerata, non presente nella lista originaria, esso viene aggiunto.

Infine, si avvia la funzione ETL, attraverso la quale viene creato un ulteriore file csv, a cui viene applicata la tecnica *Extract, Transform and Load*. Vengono, quindi, considerate soltanto le colonne necessarie alla successiva analisi, eliminando i dati in eccesso e non utili.

4.7 ETL: cos'è e come si mette in pratica

In informatica Extract, Transform, Load (ETL) è un'espressione in lingua inglese che si riferisce al processo di estrazione, trasformazione e caricamento dei dati in un sistema di sintesi (data warehouse, data mart, big data, etc). I dati vengono estratti da sistemi sorgenti quali database

transazionali (OLTP), comuni file di testo o altri sistemi informatici (ad esempio, sistemi ERP o CRM). Essi subiscono, quindi, un processo di trasformazione, che consiste, ad esempio, nel:

- selezionare solo quelli che sono di interesse per il sistema;
- normalizzare i dati (per esempio, eliminando i duplicati);
- tradurre dati codificati;
- derivare nuovi dati calcolati;
- eseguire accoppiamenti (join) tra dati recuperati da differenti tabelle;
- raggruppare i dati.

Tale trasformazione ha lo scopo di consolidare i dati (cioè renderli omogenei sebbene provengano da sorgenti diverse), oltre a fare in modo che essi siano più aderenti alla logica di business del sistema di analisi per cui viene sviluppato. I dati vengono, infine, memorizzati nelle tabelle del sistema di sintesi (load). Occorre prestare particolare attenzione alla granularità delle informazioni da memorizzare nella struttura a valle. Queste, infatti, non solo devono essere aggregate in modo da non avere un dettaglio eccessivo (cosa che potrebbe portare ad un decadimento delle prestazioni delle interrogazioni effettuate sul sistema), ma devono anche mantenere una granularità che consenta di effettuare le analisi necessarie sui dati.

In particolare questo processo è composto da tre fasi:

1. *Estrazione:* l'obiettivo di un processo di ETL è ottenere dati puliti e accessibili da utilizzare per attività di analisi o processi di business. I dati grezzi vengono inizialmente estratti da una vasta gamma di sorgenti, quali database esistenti, registri di attività (come quelli relativi al traffico di rete piuttosto che report di errori), report su anomalie e prestazioni di applicazioni, eventi di sicurezza e altri. I dati estratti vengono, quindi, collocati in destinazioni quali Data Lake o Data Warehouse.
2. *Trasformazione:* la fase di trasformazione del processo di ETL è la più critica. In questa fase, infatti, ai dati vengono applicate le regole necessarie a soddisfare i requisiti di segnalazione. Durante la trasformazione, ai dati grezzi vengono applicati i formati di segnalazione corretti. Se i dati non sono puliti, applicare le regole di segnalazione diventa complicato. La trasformazione avviene tramite l'applicazione di una serie di regole definite a livello aziendale. Gli standard che assicurano l'accessibilità e la qualità dei dati durante questa fase dovrebbero includere i seguenti passi:
   - *Standardizzazione:* definizione dei dati che saranno presi in considerazione, della modalità con cui verranno formattati e memorizzati, così come di altri fattori chiave che andranno a definire le fasi successive del processo.
   - *Deduplicazione:* segnalazione di duplicazioni agli steward dei dati; esclusione e/o eliminazione dei dati ridondanti.
   - *Verifica:* esecuzione di verifiche automatiche per mettere a confronto informazioni simili, come orari delle transazioni e record di accesso. Le attività di verifica consentono di sfrondare ulteriormente i dati inutilizzabili e di contrassegnare eventuali anomalie in sistemi, applicazioni e dati.
   - *Ordinamento:* ottimizzazione dell'efficienza all'interno dei data warehouse tramite raggruppamento e ordinamento in categorie di elementi, come dati grezzi, audio, multimediali e altri oggetti. Le regole di trasformazione determinano come ogni singolo dato viene classificato e dove sarà collocato nella fase successiva. Spesso, i processi ETL vengono utilizzati per creare tabelle di aggregazione per report riepilogativi. Questo richiede l'ordinamento e la successiva aggregazione dei dati.

   Attraverso questi passaggi di trasformazione, ciò che prima era un ammasso di materiale inutilizzabile viene plasmato in un prodotto di dati pronto per la fase finale del processo di ETL, ovvero quella del caricamento.
3. *Caricamento:* l'ultima fase del processo ETL prevede, in genere, il caricamento dei dati

estratti e trasformati in una nuova destinazione. I dati possono essere caricati in un data warehouse in due diversi modi: tramite caricamento completo o incrementale.

Per questo lavoro è stato effettuato ETL mediante codice, utilizzando la funzione denominata "ETL" all'interno del file "tiktok_dataset.py".

4.7.1 Studio del csv: i campi dopo l'attività di ETL

Il csv finale, ottenuto dopo la fase di ETL, è composto da 27 colonne:

1. Nome Campo: id, Tipo Campo: int64, Descrizione: numero identificativo del video considerato.
2. Nome Campo: createTime, Tipo Campo: data (dateTime), Descrizione: data di creazione del video.
3. Nome Campo: video_id, Tipo Campo: int64, Descrizione: numero identificativo del video considerato (che coincide con l'id alla colonna 1).
4. Nome Campo: video_duration, Tipo Campo: int64, Descrizione: Durata del video (secondi).
5. Nome Campo: author_id, Tipo Campo: int64, Descrizione: numero identificativo dell'autore.
6. Nome Campo: author_uniqueId, Tipo Campo: object (stringa), Descrizione: nominativo univoco con cui si è iscritto ogni utente.
7. Nome Campo: author_nickname, Tipo Campo: object (stringa), Descrizione: nickname dell'utente.
8. Nome Campo: author_verified, Tipo Campo: bool, Descrizione: rappresenta la possibilità o meno che l'utente sia verificato, ovvero che rappresenti un personaggio di spicco all'interno della società (possibili influencers/VIP).
9. Nome Campo: author_secUid, Tipo Campo: object (stringa), Descrizione: è il secondo identificativo dell'utente.
10. Nome Campo: music_id, Tipo Campo: int64, Descrizione: identificativo della musica utilizzata per il video.
11. Nome Campo: music_title, Tipo Campo: object (stringa), Descrizione: titolo del brano utilizzato come sottofondo al video.
12. Nome Campo: music_authorName, Tipo Campo: object (stringa), Descrizione: cantante del brano utilizzato come sottofondo.
13. Nome Campo: stats_diggCount, Tipo Campo: int64, Descrizione: numero di like al video.
14. Nome Campo: stats_shareCount, Tipo Campo: int64, Descrizione: numero di condivisioni del video.
15. Nome Campo: stats_commentCount, Tipo Campo: int64, Descrizione: numero di commenti al video.
16. Nome Campo: stats_playCount, Tipo Campo: int64, Descrizione: numero di visualizzazioni del video.
17. Nome Campo: duetInfo_duetFromId, Tipo Campo: int64, Descrizione: id dell'utente che ha effettuato un duetto insieme al video in considerazione.
18. Nome Campo: authorStats_diggCount, Tipo Campo: int64, Descrizione: numero di like messi dall'autore del video.
19. Nome Campo: authorStats_followingCount, Tipo Campo: int64, Descrizione: numero di following da parte dell'autore verso altri utenti (ovvero il numero di persone che l'autore del video segue).
20. Nome Campo: authorStats_followerCount, Tipo Campo: int64, Descrizione: numero di follower che ha l'autore del video.
21. Nome Campo: authorStats_heartCount, Tipo Campo: int64, Descrizione: numero di mi piace totali ricevuti dall'utente.

22. Nome Campo: authorStats_videoCount, Tipo Campo: int64, Descrizione: numero di video dell'autore che ha pubblicato il video corrente.
23. Nome Campo: duetEnabled, Tipo Campo: bool, Descrizione: abilitazione ai duetti (esiste la possibilità che un utente, quando pubblica un video, decida di opzionare l'abilitazione ai duetti oppure disabilitarla).
24. Nome Campo: originalVideo, Tipo Campo: float64, Descrizione: identifica il video originale, in questo caso utilizzeremo lo 0 per tutti i video non originali, mentre il valore 1 viene assegnato ai video originali.
25. Nome Campo: likedBy_id, Tipo Campo: object (stringa), Descrizione: id della persona che ha messo like al video.
26. Nome Campo: likedBy_secUid, Tipo Campo: object(stringa), Descrizione: secondo uniqueId dell'utente che ha messo mi piace al video.
27. Nome Campo: likedBy_uniqueId, Tipo Campo: object (stringa), Descrizione: uniqueId dell'utente che ha messo mi piace al video considerato.

Per una migliore comprensione dei campi, essi sono stati riportati in forma tabellare mediante script in Python, come riportato in Figura 4.4.

```
>>> df = pd.read_csv('dataset_badchallenge_connections_etl.csv', sep=";")
>>> df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4139 entries, 0 to 4138
Data columns (total 27 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   id                          4139 non-null   int64
 1   createTime                  4139 non-null   object
 2   video_id                    4139 non-null   int64
 3   video_duration              4139 non-null   int64
 4   author_id                   4139 non-null   int64
 5   author_uniqueId             4139 non-null   object
 6   author_nickname             4138 non-null   object
 7   author_verified             4139 non-null   bool
 8   author_secUid               4139 non-null   object
 9   music_id                    4139 non-null   int64
 10  music_title                 4139 non-null   object
 11  music_authorName            4137 non-null   object
 12  stats_diggCount             4139 non-null   int64
 13  stats_shareCount            4139 non-null   int64
 14  stats_commentCount          4139 non-null   int64
 15  stats_playCount             4139 non-null   int64
 16  duetInfo_duetFromId         4139 non-null   int64
 17  authorStats_diggCount       4139 non-null   int64
 18  authorStats_followingCount  4139 non-null   int64
 19  authorStats_followerCount   4139 non-null   int64
 20  authorStats_heartCount      4139 non-null   int64
 21  authorStats_videoCount      4139 non-null   int64
 22  duetEnabled                 4139 non-null   bool
 23  originalVideo               4139 non-null   float64
 24  likedBy_id                  4139 non-null   object
 25  likedBy_secUid              4139 non-null   object
 26  likedBy_uniqueId            4139 non-null   object
dtypes: bool(2), float64(1), int64(15), object(9)
memory usage: 816.6+ KB
>>>
```

Figura 4.4: Elenco delle colonne con la relativa tipologia *Fonte: autoprodotta*

4.8 Challenge esemplificativa: ITookANap

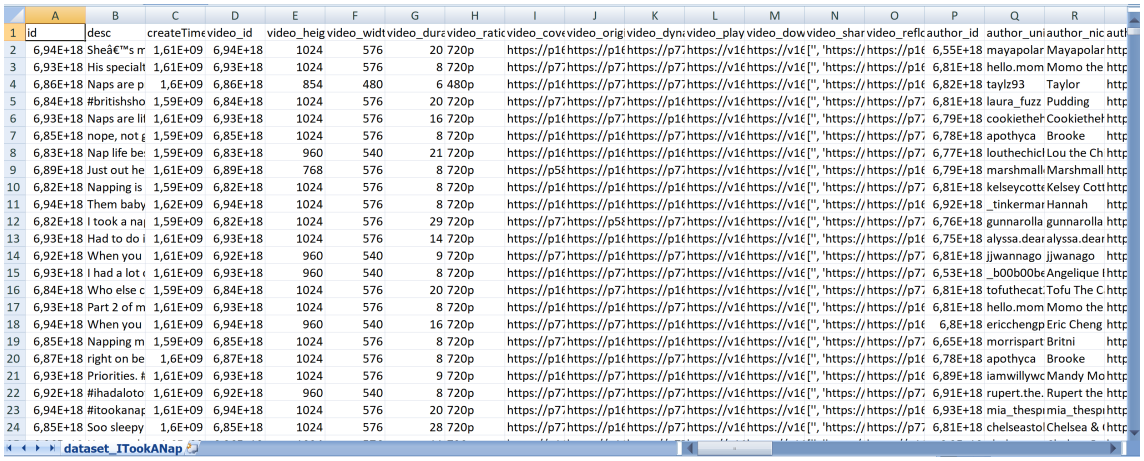
Come challenge di partenza è stata scelta "ITookANap". Si tratta di una challenge positiva, senza alcun risvolto pericoloso. Consiste nel riprendere un soggetto (persona o animale domestico) mentre è intento a schiacciare un pisolino. È stata inserita la challenge in questione all'interno del file "challenges.py", indicando il nome e l'URL del video originale, identificato dalla piattaforma stessa (Figura 4.5).

```
lib > challenges.py > ...
challenges = { "itookanap":
                    {"name": "ITookANap",
                    "url": "https://www.tiktok.com/@gunnarolla/video/6816020939759815942"}, # original video
               "elpepe":
                    {"name":"elpepe",
                    "url":""},
               "ohnanachallenge":
                    {"name":"ohnanachallenge",
                    "url":"https://www.tiktok.com/@saffronbarker/video/6778137328499084549"},
               "makarenachallenge":
                    {"name":"makarenachallenge",
                     "url":""},
               "weirdsoundchallenge":
                    {"name":"weirdsoundchallenge",
                     "url":"https://www.tiktok.com/@angelbrown_19/video/6806305248223759622"},
               "copinesdancechallenge":
```

Figura 4.5: Struttura del dizionario relativo alle challenge *Fonte: autoprodotta*

È possibile specificare anche più di un hashtag per una determinata challenge, inserendo all'interno del campo "name", "hashtag1, hashtag2, hashtag3". Questa eventualità è stata particolarmente utilizzata per le challenge negative, poiché, molto spesso, tendono ad essere identificate con vari nomi per renderne più difficile l'individuazione, e la conseguente interruzione.

A questo punto viene avviato il "main.py" dal prompt dei comandi, eseguendo `python main.py -c itookanap`, come visto in precedenza. Vengono scaricati i dati relativi a 5000 video circa, dopodiché vengono salvati all'interno della cartella "dataset", che si trova nella working directory, sotto forma di un file csv, caratterizzato dal layout riportato in Figura 4.6.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	
1	id	desc	createTime	video_id	video_heig	video_widt	video_dura	video_ratio	video_cove	video_origi	video_dyn	video_play	video_dow	video_shar	video_refl	author_id	author_uni	author_nic	auth
2	6,94E+18	Sheâ€™s n	1,61E+09	6,94E+18	1024	576	20	720p	https://p16	https://p7	https://v1€	https://v1€	https://v1€	", "https://	https://p16	6,55E+18	mayapolar	Mayapolar	http
3	6,93E+18	His specialt	1,61E+09	6,93E+18	1024	576	8	720p	https://p7	https://p7	https://p7	https://v1€	https://v1€	", "https://	https://p16	6,81E+18	hello.mom	Momo the	http
4	6,86E+18	Naps are p	1,6E+09	6,86E+18	854	480	6	480p	https://p16	https://p7	https://p7	https://v1€	https://v1€	", "https://	https://p16	6,82E+18	taylz93	Taylor	http
5	6,84E+18	#britishsho	1,59E+09	6,84E+18	1024	576	20	720p	https://p7	https://p7	https://p16	https://v1€	https://v1€	", "https://	https://p7	6,81E+18	laura_fuzz	Pudding	http
6	6,93E+18	Naps are lil	1,61E+09	6,93E+18	1024	576	16	720p	https://p16	https://p7	https://p16	https://v1€	https://v1€	", "https://	https://p7	6,79E+18	cookiethet	Cookiethet	http
7	6,85E+18	nope, not g	1,59E+09	6,85E+18	1024	576	8	720p	https://p16	https://p7	https://p7	https://v1€	https://v1€	", "https://	https://p7	6,78E+18	apothyca	Brooke	http
8	6,83E+18	Nap life be	1,59E+09	6,83E+18	960	540	21	720p	https://p16	https://p16	https://p16	https://v1€	https://v1€	", "https://	https://p7	6,77E+18	louthechicl	Lou the Ch	http
9	6,89E+18	Just out he	1,61E+09	6,89E+18	768	576	8	720p	https://p5	https://p16	https://p7	https://v1€	https://v1€	", "https://	https://p16	6,79E+18	marshmall	Marshmall	http
10	6,82E+18	Napping is	1,59E+09	6,82E+18	1024	576	8	720p	https://p16	https://p16	https://p16	https://v1€	https://v1€	", "https://	https://p7	6,81E+18	kelseycotte	Kelsey Cott	http
11	6,94E+18	Them baby	1,62E+09	6,94E+18	1024	576	8	720p	https://p16	https://p16	https://p16	https://v1€	https://v1€	", "https://	https://p16	6,92E+18	_tinkermar	Hannah	http
12	6,82E+18	I took a nap	1,59E+09	6,82E+18	1024	576	29	720p	https://p7	https://p5	https://p7	https://v1€	https://v1€	", "https://	https://p7	6,76E+18	gunnarolla	gunnarolla	http
13	6,93E+18	Had to do i	1,61E+09	6,93E+18	1024	576	14	720p	https://p16	https://p16	https://p16	https://v1€	https://v1€	", "https://	https://p16	6,75E+18	alyssa.dear	alyssa.dear	http
14	6,92E+18	When you	1,61E+09	6,92E+18	960	540	9	720p	https://p7	https://p16	https://p7	https://v1€	https://v1€	", "https://	https://p7	6,81E+18	jjwanago	jjwanago	http
15	6,93E+18	I had a lot (	1,61E+09	6,93E+18	960	540	8	720p	https://p16	https://p7	https://p7	https://v1€	https://v1€	", "https://	https://p7	6,53E+18	_b00b00be	Angelique I	http
16	6,84E+18	Who else c	1,59E+09	6,84E+18	1024	576	20	720p	https://p16	https://p7	https://p16	https://v1€	https://v1€	", "https://	https://p7	6,81E+18	tofuthecat	Tofu The C	http
17	6,93E+18	Part 2 of m	1,61E+09	6,93E+18	1024	576	8	720p	https://p16	https://p7	https://p7	https://v1€	https://v1€	", "https://	https://p16	6,81E+18	hello.mom	Momo the	http
18	6,94E+18	When you	1,61E+09	6,94E+18	960	540	16	720p	https://p7	https://p7	https://p16	https://v1€	https://v1€	", "https://	https://p16	6,8E+18	ericchengp	Eric Cheng	http
19	6,85E+18	Napping m	1,59E+09	6,85E+18	1024	576	8	720p	https://p7	https://p16	https://p16	https://v1€	https://v1€	", "https://	https://p7	6,65E+18	morrispart	Britni	http
20	6,87E+18	right on be	1,6E+09	6,87E+18	1024	576	8	720p	https://p16	https://p16	https://p7	https://v1€	https://v1€	", "https://	https://p16	6,78E+18	apothyca	Brooke	http
21	6,93E+18	Priorities. #	1,61E+09	6,93E+18	1024	576	9	720p	https://p16	https://p7	https://p16	https://v1€	https://v1€	", "https://	https://p16	6,89E+18	iamwillywc	Mandy Mo	http
22	6,92E+18	#ihadaloto	1,61E+09	6,92E+18	960	540	8	720p	https://p7	https://p7	https://p7	https://v1€	https://v1€	", "https://	https://p7	6,91E+18	rupert.the.	Rupert the	http
23	6,94E+18	#itookanap	1,61E+09	6,94E+18	1024	576	20	720p	https://p7	https://p16	https://p7	https://v1€	https://v1€	", "https://	https://p16	6,93E+18	mia_thespi	mia_thespi	http
24	6,85E+18	Soo sleepy	1,6E+09	6,85E+18	1024	576	28	720p	https://p7	https://p16	https://p7	https://v1€	https://v1€	", "https://	https://p7	6,81E+18	chelseastol	Chelsea &	http

dataset_ITookANap

Figura 4.6: file csv "dataset_itookanap" *Fonte: autoprodotta*

A questo punto, sul terminale, partirà automaticamente la seconda fase, ovvero il parsing effettuato su ciascun video per ricercare se ci sono utenti con la lista dei like pubblici. È una parte piuttosto onerosa, sebbene il primato resti alla fase successiva. In ogni caso, viene stampata la progressione del parsing su base numerica. La visualizzazione sarà quella riportata in Figura 4.7.

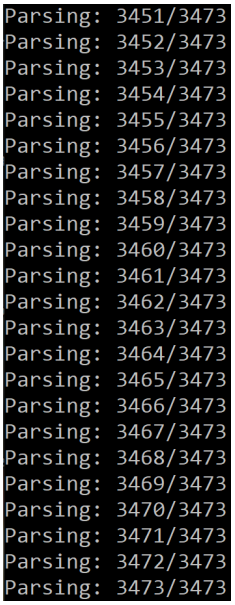


Figura 4.7: Parsing dei video scaricati *Fonte: autoprodotta*

Al termine di questa attività, verrà generato un file csv all'interno della cartella "authors", posizionata dentro alla cartella "dataset". Il file si chiamerà "pubLiked_ITookANap" e avrà l'aspetto riportato nella Figura 4.8.

	A	B	C
1	author_id	author_secUid	author_uniqueId
2	95287896911298500	MS4wLjABAAAAfQF7s0Sht2dGxZyYaO9ngUAJTr4lvuGtG2YVSuulKaX	djangeljohal
3	6711277954175480000	MS4wLjABAAAAxr53QkYoDV6xDkw6Y9HKEJXFCIRpkwY3mYTA9Mu	the.diesel.pup
4	6814632795857380000	MS4wLjABAAAACarn6jeikK-GM-OrOzC4_N68KmdfPk_uUJ24cPmax	erinelizabethrn
5	6812630647296910000	MS4wLjABAAAAJ9m1g9voBWgI2vAicdM2qZTgveFqRoebtmP8ISgsIN	zarathechihuahua
6	60986910046695400	MS4wLjABAAAAqmiu-EWjd9w-nLlIIMcRFaM2NQh3WQBqh5rE8hffc	kimchallan
7	6776416970512020000	MS4wLjABAAAAueLxIcUfyhxY--3lqSQTxNiLkfq-Bby7vCjzU-H69-0RSX	beckysanders13
8	6765330343584400000	MS4wLjABAAAASlMhFOHsZ2s1ZcMrMpSDZLdPZSrSY_cHIIQGcbG-X	memphiszoo
9	6806761667830000000	MS4wLjABAAAApYJgAa2INmKvDkSqoCHyGx-1B32aVmIr_gd8P4wnr	dougiesaintbernard
10	6813813337140110000	MS4wLjABAAAAtLx60RKSKcEO8vO1pG8L0zJ4yv3QLq9VBcd8BNldW	ferdithefrenchie
11	116776367744028000	MS4wLjABAAAAKd-xGjsjN9yAtdQTH2BHzbmn2MlTqoIUUJzgw7iHl_	happygirlm1063
12	6894262376439790000	MS4wLjABAAAARJgAQpdZtd3IY1KdZAoaIHKFV1qkQlceQdHHUngTn	loonieboonies
13	6835413248940060000	MS4wLjABAAAAJUj3rQiAwaVdY4gGiGFR7xfHvoech2L6K6uUb8N3W	cattyshackcatcafe
14	6773840526186900000	MS4wLjABAAAA8OfuHNl7DOd3Go6-77XJKdXrp_ti_MAYxIFd7RtWn	purejillma5
15	6885130083720390000	MS4wLjABAAAAoEaoPFlvlTH6BhRYVysJYmt70rn3Kva8JLH1PobUflO	mateomoments
16	6819792281123060000	MS4wLjABAAAAbZ3xoR138mTHgIIRiG7dlre9dm0esf7iJczHnUQuwX	iwillnotbedefeated
17	6810833031160660000	MS4wLjABAAAAlfMLLOuDihnnssUbRghNjricCnzeVG9Zi9LziKUjd3LH	thevital_np
18	6734401688663090000	MS4wLjABAAAAMwhXmnykOpIrtiHe971LQCtjsjOHzwM1iGMbRR-V	golden_chappy
19	6761558420308210000	MS4wLjABAAAATt2iexlGSksCSlxM5cTKElkLxjg19eDEtw01sl0TEl66ijJ	jessiecat221
20	6811850847095010000	MS4wLjABAAAACcDHitwAIvFVCIPNHf5Lt0-zB4uvtB0fG0l9NMXV-PT	mr_mrs_sausage
21	6808258876791290000	MS4wLjABAAAA7oazDuVTa3pQBokNz0ropklhdR_viayeo7xB8i_FhZ	pebblesandkobe
22	6725142980292770000	MS4wLjABAAAA4-88P1gB99B8GNkJlK47RlgBwL8bQbc1hAjHZ5NHm	starlit_seeker
23	6805735243750090000	MS4wLjABAAAA87QmUjTVUQW_fCRmyX5Q8_jcmV7_tJHd1-YF57H	trey.serna

Figura 4.8: File csv "pubLiked_ITookANap *Fonte: autoprodotta*

Di seguito, partirà in modo automatico la terza funzione, con l'obiettivo di creare connessioni tra gli utenti che hanno pubblicato il video e coloro che hanno messo like. Dal prompt dei comandi comparirà una *Progress Bar* che mostrerà l'andamento del processo, poiché questa fase richiederà diverso tempo (Figura 4.9).

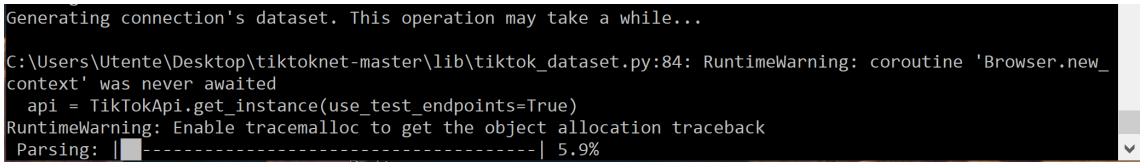
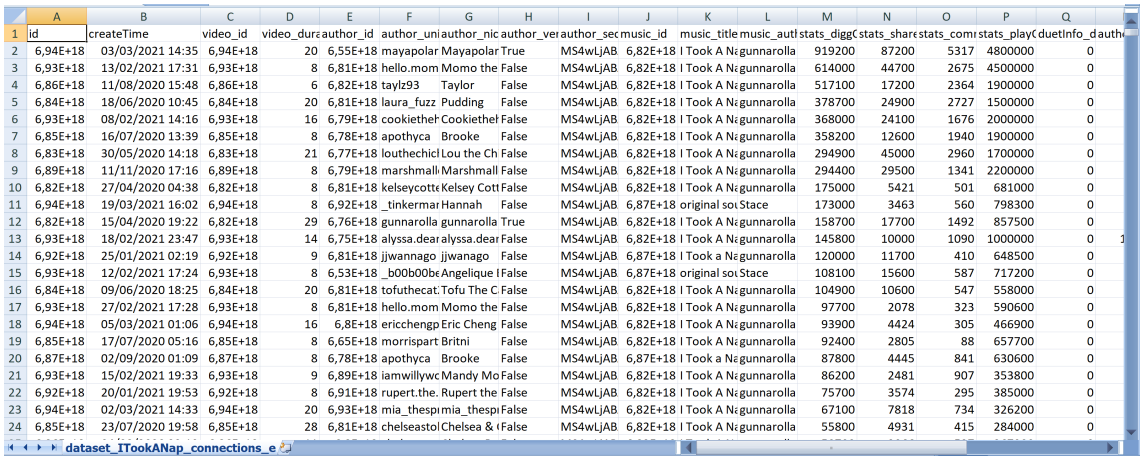


Figura 4.9: Visualizzazione del progresso dal prompt dei comandi *Fonte: autoprodotta*

Quando avrà terminato, verrà salvato un nuovo csv all'interno della cartella "dataset", denominato "dataset_ITookANap_connections". Si tratterà di un file contenente molti campi, di cui alcuni inutili, poiché non ancora sottoposto ad ETL. Al termine dell'elaborazione dei dati, verrà prodotto non solo questo file, ma anche il file conclusivo, ovvero quello a cui sono state applicate le tecniche *Extract, Transform and Load*.

Si otterrà un file denominato "dataset_ITookANap_connections_etl" e avrà la struttura riportata in Figura 4.10.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1	id	createTime	video_id	video_dura	author_id	author_uni	author_nic	author_ver	author_sec	music_id	music_title	music_autl	stats_digg(	stats_share	stats_com	stats_play(	duetInfo_d
2	6,94E+18	03/03/2021 14:35	6,94E+18	20	6,55E+18	mayapolar	Mayapolar	True	MS4wLjAB	6,82E+18	I Took A Na	gunnarolla	919200	87200	5317	4800000	0
3	6,93E+18	13/02/2021 17:31	6,93E+18	8	6,81E+18	hello.mom	Momo the	False	MS4wLjAB	6,82E+18	I Took A Na	gunnarolla	614000	44700	2675	4500000	0
4	6,86E+18	11/08/2020 15:48	6,86E+18	6	6,82E+18	taylz93	Taylor	False	MS4wLjAB	6,82E+18	I Took A Na	gunnarolla	517100	17200	2364	1900000	0
5	6,84E+18	18/06/2020 10:45	6,84E+18	20	6,81E+18	laura_fuzz	Pudding	False	MS4wLjAB	6,82E+18	I Took A Na	gunnarolla	378700	24900	2727	1500000	0
6	6,93E+18	08/02/2021 14:16	6,93E+18	16	6,79E+18	cookiethel	Cookiethel	False	MS4wLjAB	6,82E+18	I Took A Na	gunnarolla	368000	24100	1676	2000000	0
7	6,85E+18	16/07/2020 13:39	6,85E+18	8	6,78E+18	apothyca	Brooke	False	MS4wLjAB	6,82E+18	I Took A Na	gunnarolla	358200	12600	1940	1900000	0
8	6,83E+18	30/05/2020 14:18	6,83E+18	21	6,77E+18	louthechicl	Lou the Ch	False	MS4wLjAB	6,82E+18	I Took A Na	gunnarolla	294900	45000	2960	1700000	0
9	6,89E+18	11/11/2020 17:16	6,89E+18	8	6,79E+18	marshmall	Marshmall	False	MS4wLjAB	6,82E+18	I Took A Na	gunnarolla	294400	29500	1341	2200000	0
10	6,82E+18	27/04/2020 04:38	6,82E+18	8	6,81E+18	kelseycott	Kelsey Cott	False	MS4wLjAB	6,82E+18	I Took A Na	gunnarolla	175000	5421	501	681000	0
11	6,94E+18	19/03/2021 16:02	6,94E+18	8	6,92E+18	_tinkermar	Hannah	False	MS4wLjAB	6,87E+18	original sou	Stace	173000	3463	560	798300	0
12	6,82E+18	15/04/2020 19:22	6,82E+18	29	6,76E+18	gunnarolla	gunnarolla	True	MS4wLjAB	6,82E+18	I Took A Na	gunnarolla	158700	17700	1492	857500	0
13	6,93E+18	18/02/2021 23:47	6,93E+18	14	6,75E+18	alyssa.dear	alyssa.dear	False	MS4wLjAB	6,82E+18	I Took A Na	gunnarolla	145800	10000	1090	1000000	0
14	6,92E+18	25/01/2021 02:19	6,92E+18	9	6,81E+18	jjwannago	jjwanago	False	MS4wLjAB	6,87E+18	I Took a Na	gunnarolla	120000	11700	410	648500	0
15	6,93E+18	12/02/2021 17:24	6,93E+18	8	6,53E+18	_b00b00be	Angelique I	False	MS4wLjAB	6,87E+18	original sou	Stace	108100	15600	587	717200	0
16	6,84E+18	09/06/2020 18:25	6,84E+18	20	6,81E+18	tofuthecat	Tofu The C	False	MS4wLjAB	6,82E+18	I Took A Na	gunnarolla	104900	10600	547	558000	0
17	6,93E+18	27/02/2021 17:28	6,93E+18	8	6,81E+18	hello.mom	Momo the	False	MS4wLjAB	6,82E+18	I Took A Na	gunnarolla	97700	2078	323	590600	0
18	6,94E+18	05/03/2021 01:06	6,94E+18	16	6,8E+18	ericchengp	Eric Cheng	False	MS4wLjAB	6,82E+18	I Took A Na	gunnarolla	93900	4424	305	466900	0
19	6,85E+18	17/07/2020 05:16	6,85E+18	8	6,65E+18	morrispart	Britni	False	MS4wLjAB	6,82E+18	I Took A Na	gunnarolla	92400	2805	88	657700	0
20	6,87E+18	02/09/2020 01:09	6,87E+18	8	6,78E+18	apothyca	Brooke	False	MS4wLjAB	6,87E+18	I Took a Na	gunnarolla	87800	4445	841	630600	0
21	6,93E+18	15/02/2021 19:33	6,93E+18	9	6,89E+18	iamwillywc	Mandy Mo	False	MS4wLjAB	6,82E+18	I Took A Na	gunnarolla	86200	2481	907	353800	0
22	6,92E+18	20/01/2021 19:53	6,92E+18	8	6,91E+18	rupert.the.	Rupert the	False	MS4wLjAB	6,82E+18	I Took A Na	gunnarolla	75700	3574	295	385000	0
23	6,94E+18	02/03/2021 14:33	6,94E+18	20	6,93E+18	mia_thespi	mia_thespi	False	MS4wLjAB	6,82E+18	I Took A Na	gunnarolla	67100	7818	734	326200	0
24	6,85E+18	23/07/2020 19:58	6,85E+18	28	6,81E+18	chelseastol	Chelsea & (	False	MS4wLjAB	6,82E+18	I Took A Na	gunnarolla	55800	4931	415	284000	0

Figura 4.10: File csv "dataset_ITookANap_connections_etl" *Fonte: autoprodotta*

Come precedentemente spiegato, il file si compone di 27 colonne, ciascuna delle quali rappresenta un campo ripulito ed utile all'analisi che ne seguirà.

5. Creazione delle reti a supporto dell'analisi

5.1 Reti

Con i dati scaricati è possibile costruire le reti di collegamento tra gli autori che hanno pubblicato video relativi alla stessa challenge, così da riuscire a mettere in luce i collegamenti tra gli utenti che hanno soltanto caricato i video e coloro che hanno sia messo like, sia pubblicato un video a loro volta. La rete che si verrà a creare sarà costituita da un insieme di nodi, che rappresenteranno i singoli utenti appunto, ed un insieme di archi, che collegheranno gli utenti che hanno interagito tra loro relativamente ad una stessa challenge. È stata utilizzata la libreria NetworkX, un pacchetto Python che permette la creazione, la manipolazione, lo studio della struttura e della dinamica delle reti create. Sarà necessario specificare l'insieme dei nodi e l'insieme degli archi, per poi permettere la creazione del grafo, con conseguente calcolo degli indici principali.

5.2 Struttura e funzionamento dell'algoritmo

All'interno della working directory troviamo diversi file che utilizzeremo nel corso della creazione della rete e per tutta l'analisi seguente. Sono presenti, infatti, i seguenti file:

- "main.py";
- "network.py";
- "pre_analysis";
- "lifespan_analysis";
- "intervals_analysis".

All'interno della cartella "lib" troviamo:

- "challenges";
- "tiktok_dataset";
- "tiktok_network";
- "utils".

Il file "main.py" è stato precedentemente analizzato, insieme a "tiktok_dataset", e viene utilizzato per l'estrazione dei dati dalla piattaforma di TikTok, con la conseguente creazione del dataset. Il file "network.py" sarà quello che approfondiremo in questa sezione, poiché è il file

principale che permette di costruire la rete, unitamente al file presente all'interno della cartella "lib" denominato "tiktok_network.py" (Figura: 5.1):

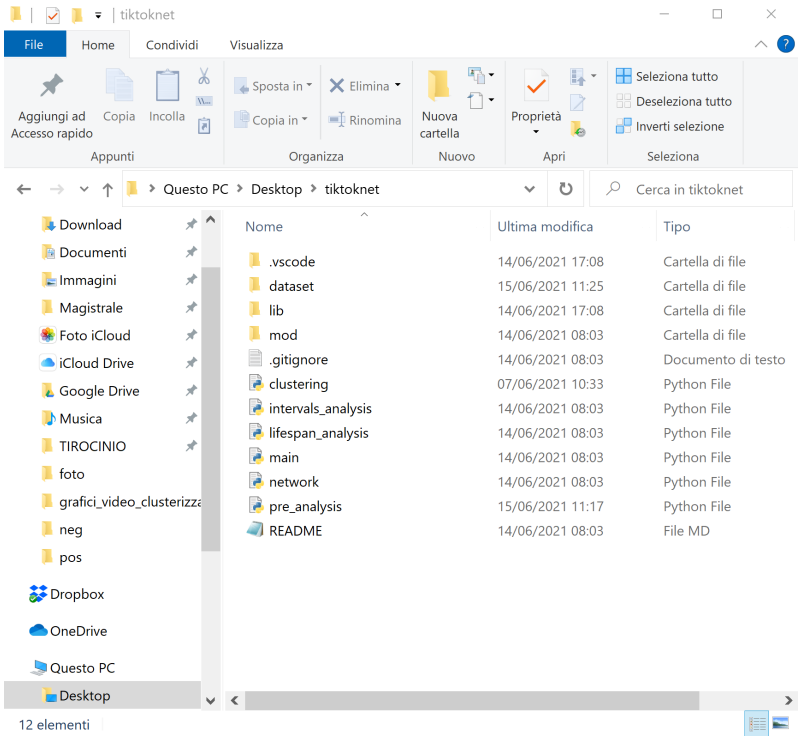


Figura 5.1: Cartella tiktoknet, *Fonte: autoprodotta*

Il file "network.py" inizia con l'importazione di due librerie: "networkX" e "matplotlib.pyplot", oltre che con l'importazione del file "tiktok_network.py", presente all'interno della cartella "lib". Viene, quindi, creata la rete in questione, considerando l'hashtag relativo alla challenge considerata. Viene richiamata, infatti, la funzione "graphCalculation" a cui viene passato il nome dell'hashtag e il parametro di colorazione dei nodi, che, in questo caso, è relativo al tempo di pubblicazione, chiamato "createTime".

La funzione "graphCalculation", che si trova in "tiktok_network.py", prende come parametri in input: "dataset", ovvero il nome dell'hashtag, e "colorCriteria", cioè il criterio di colorazione. All'interno della funzione viene creato l'insieme dei nodi e degli archi, vuoti inizialmente. Successivamente viene letto il csv, creato in seguito all'applicazione delle tecniche di ETL, ovvero "dataset_hashtag_connections_etl.csv". Vengono rimossi i nodi sconnessi, eliminando le righe in cui la colonna "likedBy_uniqueId" assume valore pari ad un trattino (-). Attraverso un ciclo "for", si considerano tutti gli id degli utenti e si assegnano come nodi. Per quello che riguarda gli archi, si specificano due valori: source e target. Come source vengono considerati i valori di "likedBy_id", mentre per target ci si riferisce agli "author_id". Con "likedBy_id" si fa riferimento all'utente che ha messo like al video considerato, mentre con "author_id" ci si riferisce all'utente che ha pubblicato il video. Come etichette, ovvero come label, viene assegnato lo "uniqueId" relativo agli utenti. Subito dopo, si va a confrontare il valore assegnato a "colorCriteria": se assume valore pari a "createTime", si va a considerare l'orario e la data della creazione del video, in modo da assegnare una diversa colorazione ad ogni nodo. Se il video è contrassegnato come originale, allora il nodo avrà colorazione rossa, se l'utente ha soltanto messo like e ha pubblicato un video assumerà una colorazione sfumata dal nero al bianco, in base all'orario di pubblicazione. In particolare, i nodi che rappresentano gli utenti che hanno pubblicato i video più vecchi in ordine temporale saranno

neri; tutti gli altri assumeranno un colore compreso nella scala di grigi, fino a diventare bianchi per i più recenti. In ultima istanza troviamo dei nodi di colorazione gialla, che rappresentano quegli utenti che hanno pubblicato un video relativo alla challenge considerata ed hanno messo like sia a se stessi, sia ad un altro utente. Siccome sono stati eliminati loop sui nodi, sono stati rimossi i collegamenti aventi autolike. Di conseguenza, il video è stato rimosso dal dataset e gli utenti gialli vengono considerati soltanto come utenti che hanno messo mi piace ad altri.

Un grafo di questo tipo avrebbe assunto una distribuzione dei nodi differente da quella voluta, con i nodi gialli al centro e tutti gli altri posizionati verso l'esterno. L'intenzione era quella di ottenere una distribuzione più logica, che evolvesse verso l'esterno secondo l'ordine temporale di pubblicazione. È stata, quindi, creata una funzione denominata "position_1", tramite la quale è stato possibile assegnare delle coordinate polari ad ogni nodo disegnato. Si considera un raggio pari al valore del colore, che si riferisce all'ordine temporale; dopodiché si genera un numero casuale da 0 a 360, che verrà assegnato al valore dell'angolo. Le coordinate x e y del punto da disegnare saranno pari a:

$$x = raggio * cos(angolo)$$
$$y = raggio * sen(angolo)$$

La funzione "position_1" prende in input il valore del colore dei nodi e li distribuisce secondo gradazione, considerando di aumentare il valore di 10 unità ogni volta che si effettua un ciclo. Infine tutti i parametri vengono ritornati dalla funzione stessa e vengono assegnati alle variabili richiamate nel file "network.py", considerando la costruzione di un grafo direzionato. Verrà, poi, richiamata la funzione "graphStats", sempre posizionata all'interno del file "tiktok_network.py", grazie alla quale vengono calcolati gli indici principali:

- *"number of nodes"*: numero di nodi;
- *"number of edges"*: numero di archi;
- *"mean indegree"*: numero medio di archi diretti ad un nodo considerando un grafo orientato;
- *"mean indegree std"*: deviazione standard considerando gli archi diretti in un vertice all'interno di un grafo orientato;
- *"mean outdegree"*: numero medio di archi che escono da un nodo, considerando un grafo orientato;
- *"mean outdegree std"*: deviazione standard relativo agli archi che escono da un nodo, considerando un grafo orientato;
- *"average clustering coefficient"*: misura del grado in cui i nodi di un grafo tendono ad essere connessi fra loro;
- *"density"*: probabilità che una qualsiasi coppia di nodi sia adiacente.

Infine, attraverso il comando "plt.show", viene mostrato sullo schermo il grafo creato. Questo rappresenta le connessioni tra gli utenti che hanno avuto delle interazioni tra loro.

5.3 Challenge esemplificativa: ITookANap

Per chiarezza, verrà esposto l'intero procedimento da un punto di vista pratico, attraverso l'utilizzo di una challenge esemplificativa. Si percorrerà passo passo ogni step necessario per la creazione delle reti a supporto dell'analisi. È stata scelta una delle challenge positive utilizzate durante il lavoro, ovvero "ITookANap".

Per avviare la creazione della rete sarà sufficiente inserire il nome della challenge all'interno del file "network.py", alla linea 6, come input alla funzione "graphCalculation", presente nel file "tiktok_network.py". Il tutto assume l'aspetto mostrato in Figura 5.2.

```python
import networkx as nx
import matplotlib.pyplot as plt
import lib.tiktok_network as ntx

plt.figure(figsize=(15,15))
graph, labels, colors, pos = ntx.graphCalculation("ITookANap",colorCriteria="createTime")
ntx.graphStats(graph)
nx.draw_networkx_nodes(graph,pos,node_color=colors,node_size=60)
nx.draw_networkx_labels(graph, pos, labels,font_size=5)
nx.draw_networkx_edges(graph,pos,arrows=True,arrowsize=3,arrowstyle="-|>",alpha=0.5)#edge_color='lightgrey'
ax= plt.gca()
ax.collections[0].set_edgecolor("#000000")
plt.show()
```

Figura 5.2: Codice presente nel file "network.py", *Fonte: autoprodotta*

Se viene mandata in esecuzione la creazione della rete relativa a questa challenge, otterremo una distribuzione casuale dei nodi, come è possibile vedere in Figura 5.3.

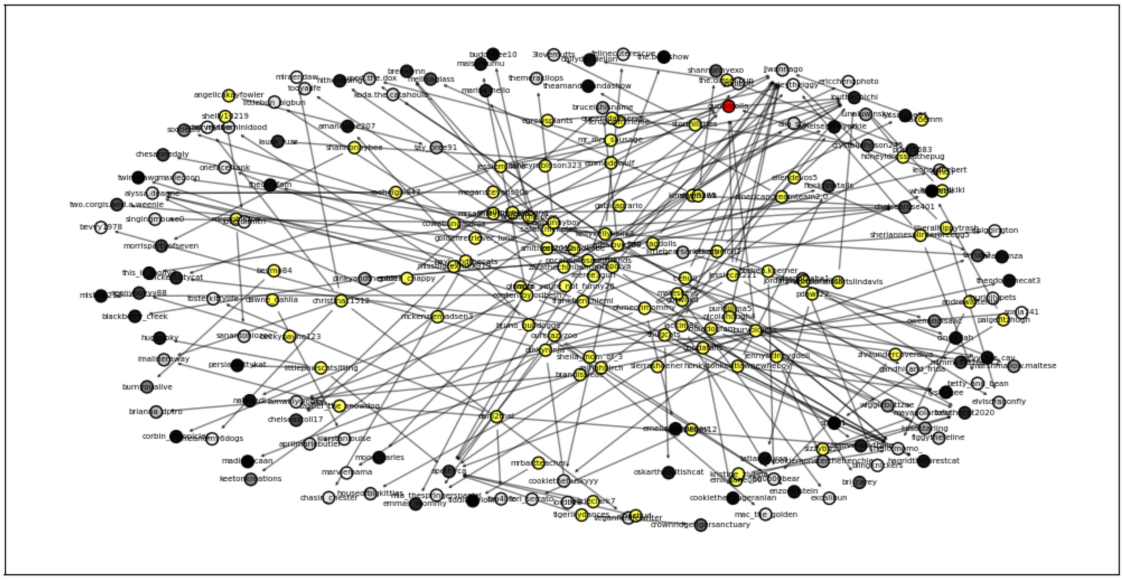


Figura 5.3: Prima rete prodotta avente distribuzione casuale, *Fonte: autoprodotta*

Com'è possibile notare dal grafo, il nodo rosso rappresenta il video etichettato come originale ed ha una posizione casuale, non logica rispetto agli altri nodi, prendendo in considerazione il momento di pubblicazione. L'intenzione è stata quindi quella di modificare la posizione dei nodi, ponendoli in ordine temporale e, quindi, di colore. La prima prova effettuata è stata quella di porli lungo una retta, dando lo stesso valore per le coordinate x e y di ogni nodo. La funzione generata è mostrata in Figura 5.4.

```python
def position_1(radius):
    x_coord = radius
    y_coord = radius
    return (x_coord, y_coord)
```

Figura 5.4: Funzione per generare la distribuzione lineare, *Fonte: autoprodotta*

Il risultato ottenuto è stato il grafico riportato nella Figura 5.5.

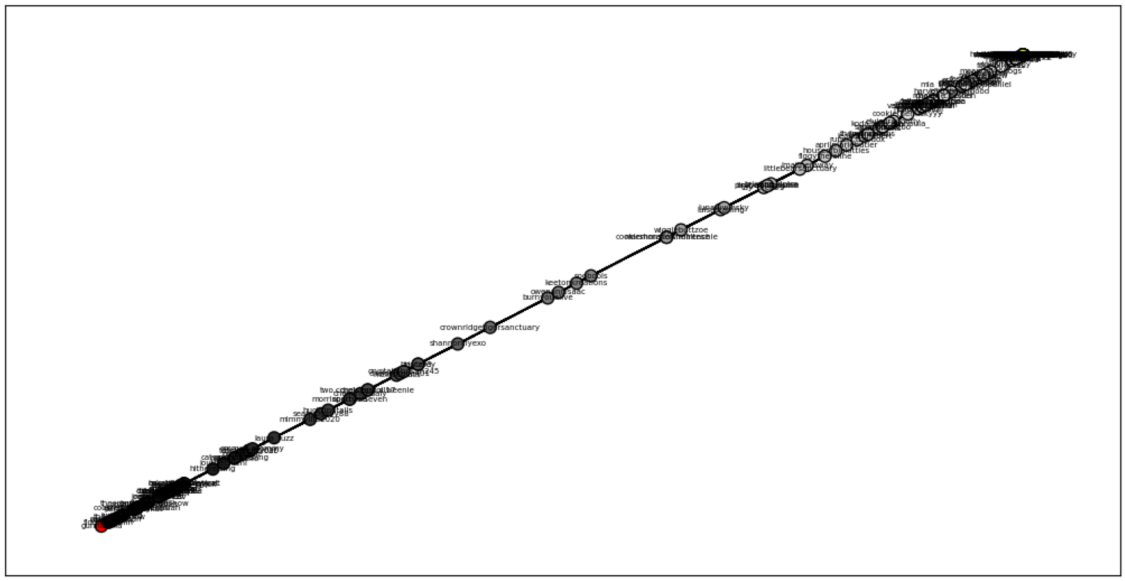


Figura 5.5: Seconda rete prodotta avente distribuzione lineare, *Fonte: autoprodotta*

Com'è chiaro dall'immagine, in questa tipologia di distribuzione i nodi sono posti in ordine temporale, andando dal basso a sinistra verso l'alto a destra. Il nodo rosso, originale, è, appunto, il primo nodo che troviamo in basso a sinistra, tutti gli altri nodi dal nero al bianco sono posizionati lungo la linea, mentre in alto a destra troviamo un'agglomerazione di nodi, tra cui quelli gialli (coloro che hanno solo messo like senza considerare il video pubblicato, poiché hanno messo autolike, e l'unico collegamento sarebbe stato con se stessi). Non è, comunque, una visualizzazione soddisfacente, poiché si tratta di una linea in cui vengono agglomerati tutti i nodi. Risulta incomprensibile e poco utile all'analisi, per questo è stata creata un'altra modalità di rappresentazione del grafo, utilizzando le coordinate polari. In questo modo è stato possibile ottenere una visualizzazione più distribuita dei nodi, garantendo una migliore visualizzazione, volta all'analisi. La funzione utilizzata è mostrata in Figura 5.6.

```
def position_1(radius):
    angolo = randrange(361) #calcolo un numero casuale da 0 a 360
    x_coord = radius*(math.cos(angolo))
    y_coord = radius*(math.sin(angolo))
    return (x_coord, y_coord)
```

Figura 5.6: Funzione per generare la distribuzione temporale, *Fonte: autoprodotta*

Grazie a questa funzione, che prende in input il raggio, si calcola un numero casuale da 0 a 360 per identificare l'angolo. Dopodiché, applicando la definizione di coordinate polari, si attribuisce alla x il valore dato dal prodotto tra il raggio, preso in input dalla funzione, e il coseno dell'angolo calcolato. Per la y vale la stessa definizione, sostituendo il seno al coseno. In seguito all'applicazione di questa funzione si otterrà, quindi, la distribuzione mostrata in Figura 5.7.

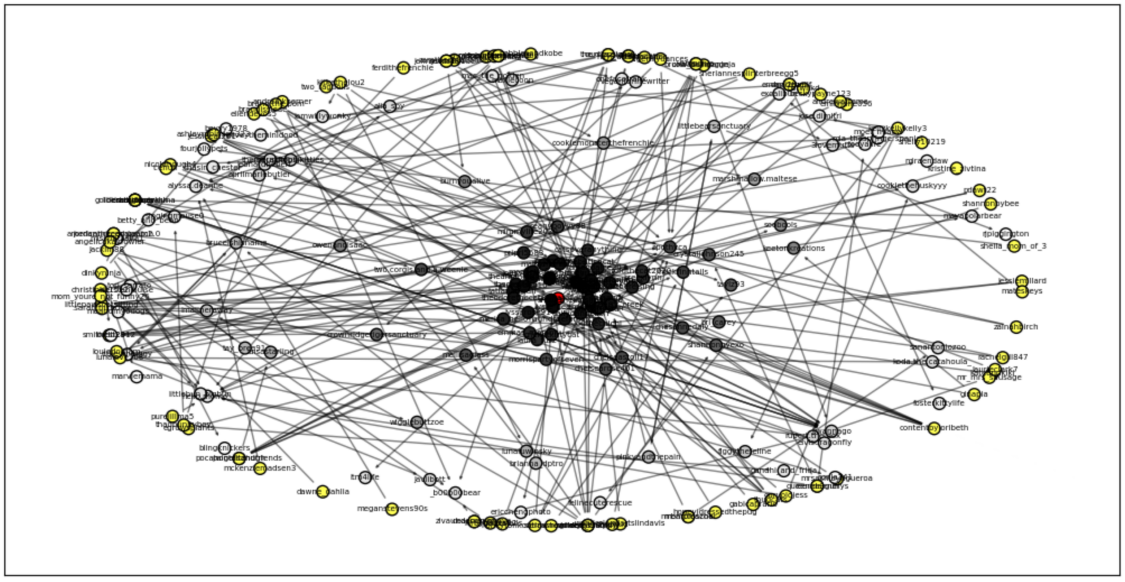


Figura 5.7: Rete finale avente distribuzione temporale, *Fonte: autoprodotta*

Abbiamo, quindi, ottenuto la distribuzione desiderata, considerando l'utente che ha pubblicato il video originale al centro, contornato da altri utenti che hanno pubblicato il video a poca distanza temporale, per poi aumentare la distanza all'aumentare del tempo (e allo schiarirsi del colore). Infine, nella parte più esterna troviamo i nodi gialli, che, come spiegato in precedenza, rappresentano coloro che hanno pubblicato un video e si sono messi autolike, oltre ad aver messo like ad un altro utente, così, non considerando il video e l'autolike, rappresentano soltanto l'aver messo un like ad un video pubblicato da un altro utente.

5.4 Reti delle challenge scelte

Sono state scaricate diverse challenge, tra cui ne sono state scelte sette positive e sette negative. In generale, è stato più semplice trovare challenge positive, piuttosto che negative, poiché queste ultime sono piuttosto "nascoste" e sono caratterizzate da più nomi. Sono state scelte le challenge che avevano una maggiore quantità di nodi ed archi, in modo da avere una grado più alto di dettaglio da poter analizzare.

Ricapitolando, le sette challenge positive sono le seguenti:

- "bussitchallenge";
- "copinesdancechallenge";
- "emojichallenge";
- "colpiditesta";
- "boredinthehouse";
- "ITookANap";
- "plankchallenge".

Le sette challenge negative, invece, sono le seguenti:

- "silhouettechallenge";
- "bugsbunnychallenge";
- "strippatiktok";
- "firewroks";
- "fightchallenge";
- "sugarbaby";

- "updownchallenge".

Per ogni challenge è stata creata la rete e sono state generate le grandezze principali, per una prima analisi. Come precedentemente visto, durante la creazione della rete per la challenge esemplificativa, vengono generate diverse grandezze per ogni rete, ovvero:

- "numero di nodi";
- "numero di archi";
- "indegree medio": numero medio di archi diretti ad un nodo considerando un grafo orientato;
- "indegree medio std": deviazione standard considerando gli archi diretti in un vertice all'interno di un grafo orientato;
- "outdegree medio": numero medio di archi che escono da un nodo, considerando un grafo orientato;
- "outdegree medio std": deviazione standard relativa agli archi che escono da un nodo, considerando un grafo orientato;
- "clustering coefficient medio": misura del grado con cui i nodi di un grafo tendono ad essere connessi fra loro;
- "densità": probabilità che una qualsiasi coppia di nodi sia adiacente.

5.4.1 Reti di challenge positive

- *Bussitchallenge.* Principalmente ricreata da ragazze, in cui si presentano senza trucco e in tenuta casalinga, per poi passare improvvisamente ad essere vestite al meglio, sistemate in trucco e parrucco, il tutto enfatizzato dall'utilizzo delle transitions. (Figura 5.8). I valori delle grandezze considerate sono i seguenti:
  - Numero di nodi: 618;
  - Numero di archi: 708;
  - Indegree medio (numero di archi diretti verso un nodo): 1.145631067961165;
  - Indegree medio std (deviazione standard): 1.2286317302650416;
  - Outdegree medio (numero di archi che escono da un nodo):1.145631067961165;
  - Outdegree medio std (deviazione standard): 4.745444622633178;
  - Clustering coefficient medio: 0.004697137417399196;
  - Densità: 0.0018567764472628282.

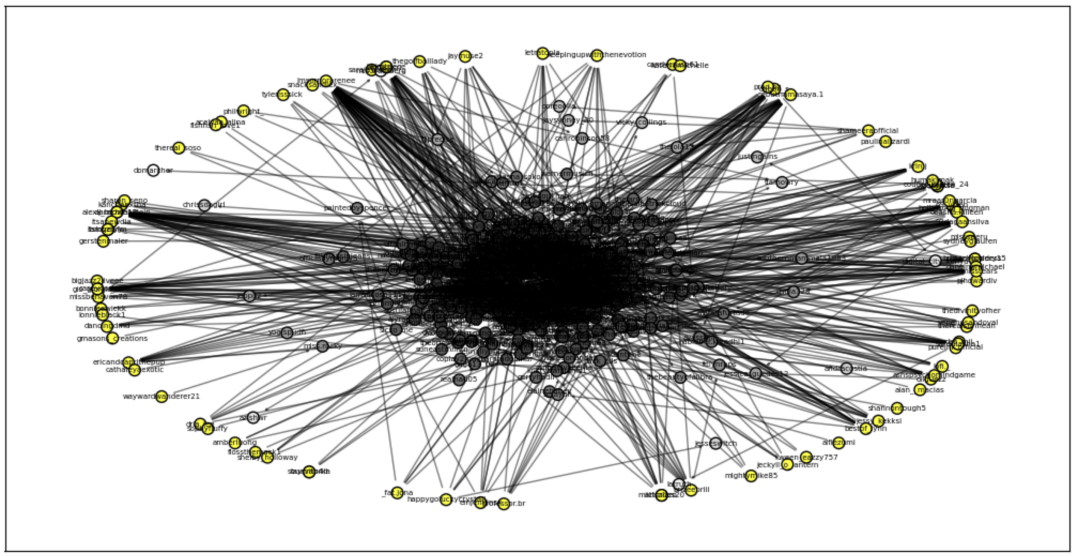


Figura 5.8: Bussitchallenge, *Fonte: autoprodotta*

- *Copinesdancechallenge.* Si esegue un balletto, diventato un trend associato alla musica utilizzata. Solitamente esiste una coreografia che gli utenti imparano e ripropongono. È considerata una challenge positiva perchè non ha alcun risvolto potenzialmente pericoloso, e l'unico obiettivo della challenge è l'*entertainment*, cioè il puro divertimento (Figura 5.9). I valori delle grandezze considerate sono i seguenti:
  - Numero di nodi: 237;
  - Numero di archi: 226;
  - Indegree medio (numero di archi diretti verso un nodo): 0.9535864978902954;
  - Indegree medio std (deviazione standard): 1.8943503072029657;
  - Outdegree medio (numero di archi che escono da un nodo): 0.9535864978902954;
  - Outdegree medio std (deviazione standard): 1.9230880707211988;
  - Clustering coefficient medio: 0.0;
  - Densità: 0.00404062075377243.

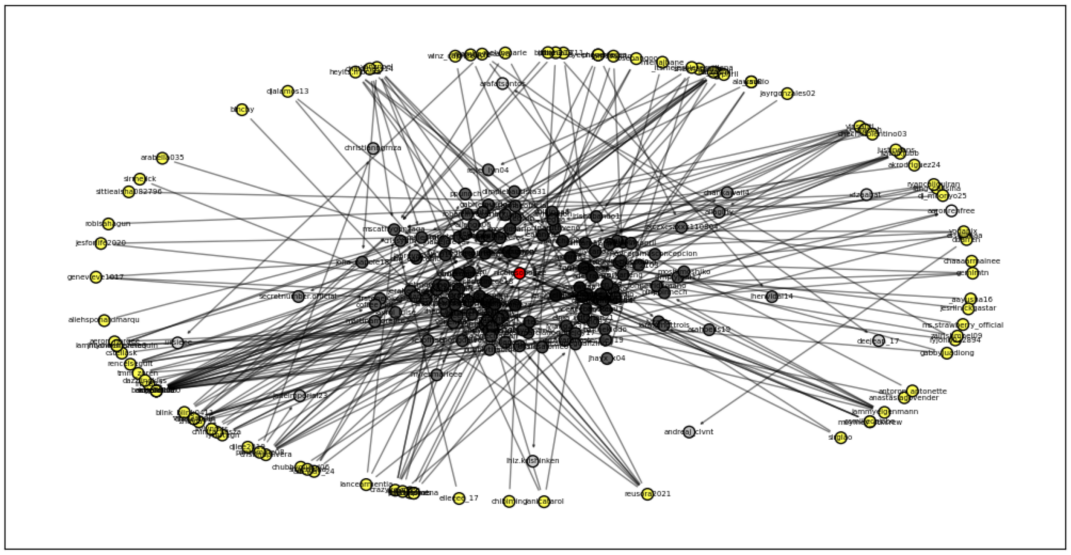


Figura 5.9: Copinesdancechallenge, *Fonte: autoprodotta*

- *Emojichallenge.* Sfida di grande successo, in cui l'obiettivo è quello di simulare le emoji che utilizziamo sulle chat attraverso le espressioni facciali. Solitamente la sfida consiste nel replicare un certo numero di emoji in un certo lasso di tempo; infatti vengono applicate delle emoji nella parte alta del video e l'utente le imita. Questa challenge è considerata positiva poiché non comporta nessun tipo di rischio per l'utente (Figura 5.10). I valori delle grandezze considerate sono i seguenti:
  - Numero di nodi: 440;
  - Numero di archi: 498;
  - Indegree medio (numero di archi diretti verso un nodo): 1.1318181818181818;
  - Indegree medio std (deviazione standard): 1.0937368653284538;
  - Outdegree medio (numero di archi che escono da un nodo): 1.1318181818181818;
  - Outdegree medio std (deviazione standard): 5.259267695622165;
  - Clustering coefficient medio: 0.0053093283770125465;
  - Densità: 0.0025781735348933527.

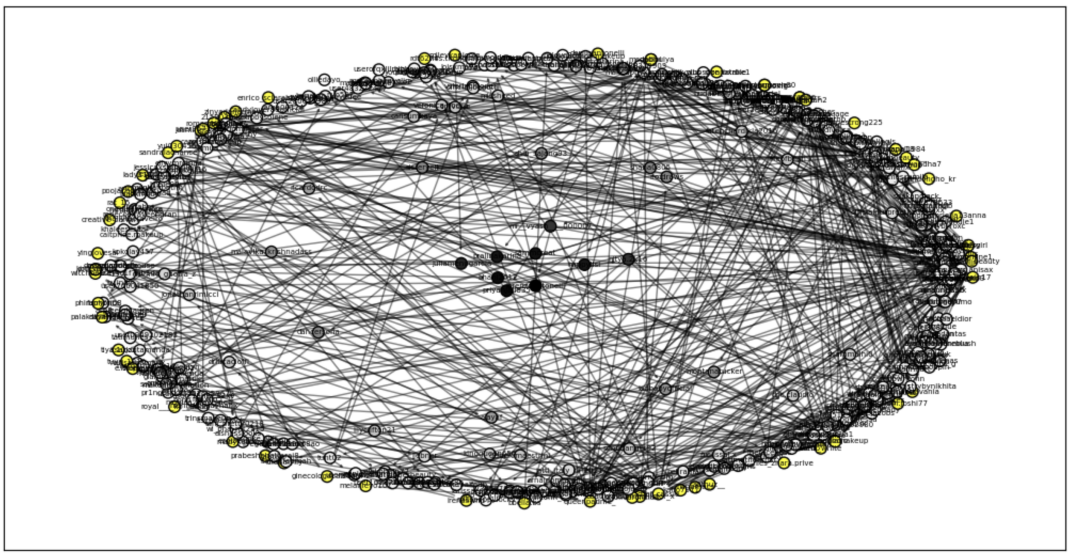


Figura 5.10: Emojichallenge, *Fonte: autoprodotta*

- *Colpiditesta.* Si tratta di un gioco in cui la persona che viene ripresa deve cercare di colpire un pallone virtuale con la propria testa, simulando dei veri e propri palleggi. Ricorda il funzionamento di un videogioco e viene considerata una challenge positiva perché non rappresenta un pericolo per cose o persone (Figura 5.11). I valori delle grandezze considerate sono i seguenti:
  - Numero di nodi: 691;
  - Numero di archi: 843;
  - Indegree medio (numero di archi diretti verso un nodo): 1.2199710564399422;
  - Indegree medio std (deviazione standard): 0.9806849546867517;
  - Outdegree medio (numero di archi che escono da un nodo): 1.2199710564399422;
  - Outdegree medio std (deviazione standard): 3.5861456356576413;
  - Clustering coefficient medio: 0.014694590715182419;
  - Densità: 0.0017680739948404958.

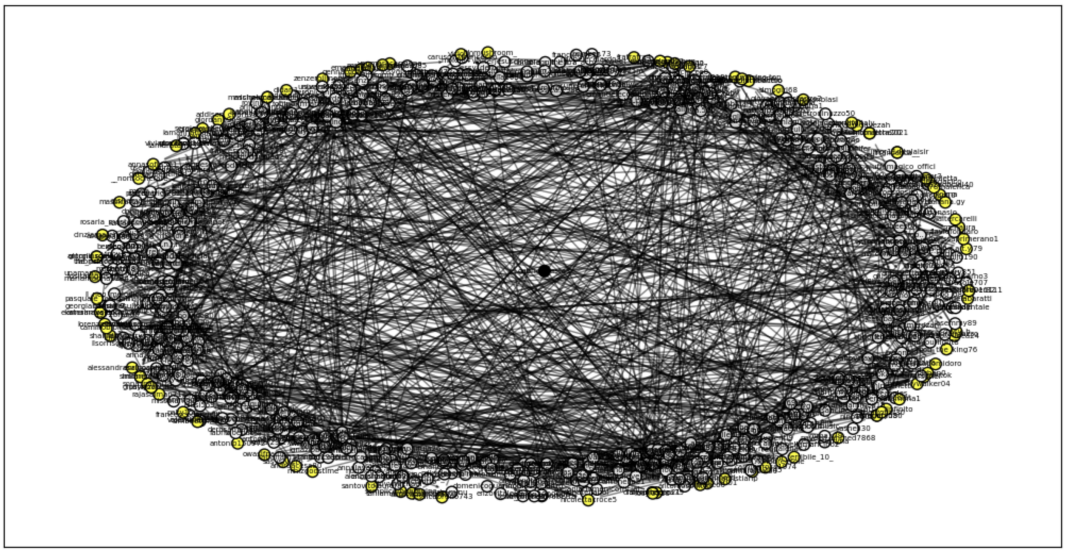


Figura 5.11: Colpiditesta, *Fonte: autoprodotta*

- *Boredinthehouse.* Challenge che ha acquisito grande successo durante il periodo di quarantena, in cui le persone si mostravano annoiate in casa, intente ad effettuare azioni quotidiane. Queste sfide molto spesso sono incentrate su atti piuttosto basici, come il fatto di fissare la telecamera oppure effettuare dei minimi movimenti, per questo non sono considerate per nulla pericolose (Figura 5.12). I valori delle grandezze considerate sono i seguenti:
  - Numero di nodi: 306;
  - Numero di archi: 309;
  - Indegree medio (numero di archi diretti verso un nodo): 1.0098039215686274;
  - Indegree medio std (deviazione standard): 1.1588965173922292;
  - Outdegree medio (numero di archi che escono da un nodo): 1.0098039215686274;
  - Outdegree medio std (deviazione standard): 2.8485580506791846;
  - Clustering coefficient medio: 0.0018339484870925034;
  - Densità: 0.003310832529733205.

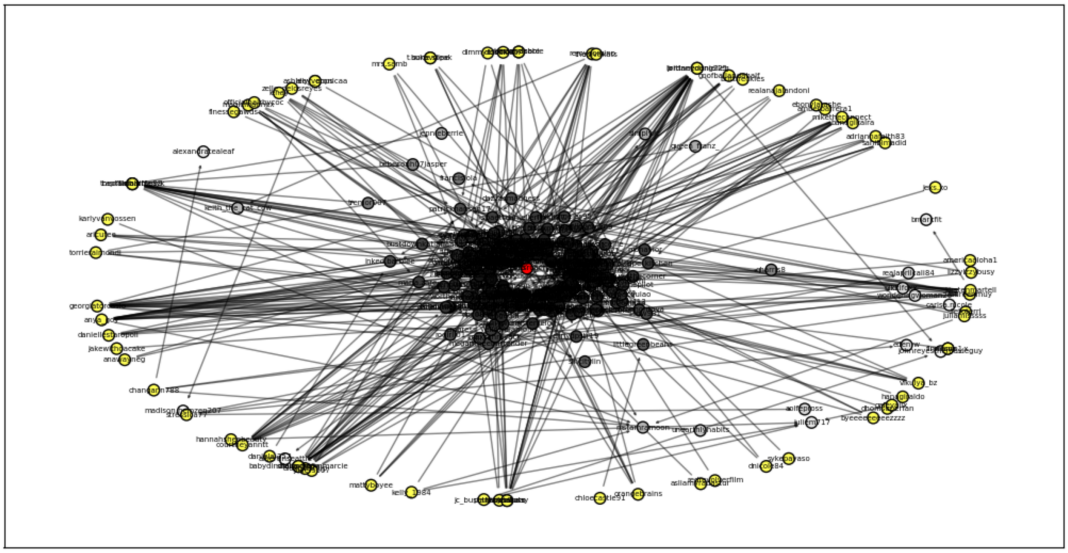


Figura 5.12: Boredinthehouse, *Fonte: autoprodotta*

- *ITookANap.* Si riprende una persona o un animale domestico mentre sta facendo un pisolino. Questi video sono tendenzialmente divertenti e l'unico obiettivo è quello di intrattenere e di far ridere l'utente spettatore. Rientra in assoluto all'interno delle challenge positive perché rappresenta quella parte più scherzosa e divertente della piattaforma, senza il bisogno di ricadere in azioni pericolose per attirare l'attenzione (Figura 5.13). I valori delle grandezze considerate sono i seguenti:
  - Numero di nodi: 219;
  - Numero di archi: 201;
  - Indegree medio (numero di archi diretti verso un nodo): 0.9178082191780822;
  - Indegree medio std (deviazione standard): 1.53575431460188;
  - Outdegree medio (numero di archi che escono da un nodo): 0.9178082191780822;
  - Outdegree medio std (deviazione standard): 1.4501150134596954;
  - Clustering coefficient medio: 0.0;
  - Densità: 0.004210129445771019.

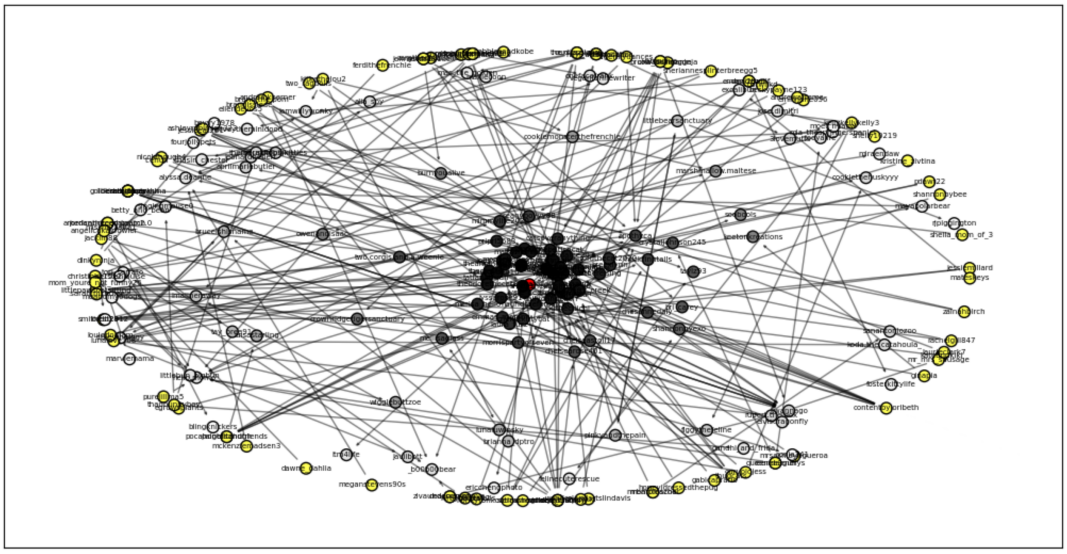


Figura 5.13: ITookANap, *Fonte: autoprodotta*

- *Plankchallenge.* Si tratta di una sfida di resistenza fisica, in cui si devono effettuare molti plank, tipici esercizi di palestra. Ha trovato largo appoggio da parte degli utenti sportivi che si sono sfidati sia in termini di resistenza fisica, sia in termini di forza. Non è pericolosa, in quanto i meno esperti hanno semplicemente effettuato un numero inferiore di plank (Figura 5.14). I valori delle grandezze considerate sono i seguenti:
  - Numero di nodi: 271;
  - Numero di archi: 266;
  - Indegree medio (numero di archi diretti verso un nodo): 0.981549815498155;
  - Indegree medio std (deviazione standard): 0.8304796752846221;
  - Outdegree medio (numero di archi che escono da un nodo): 0.981549815498155;
  - Outdegree medio std (deviazione standard): 3.4475017689307146;
  - Clustering coefficient medio: 0.007947455618785472;
  - Densità: 0.003635369687030204.

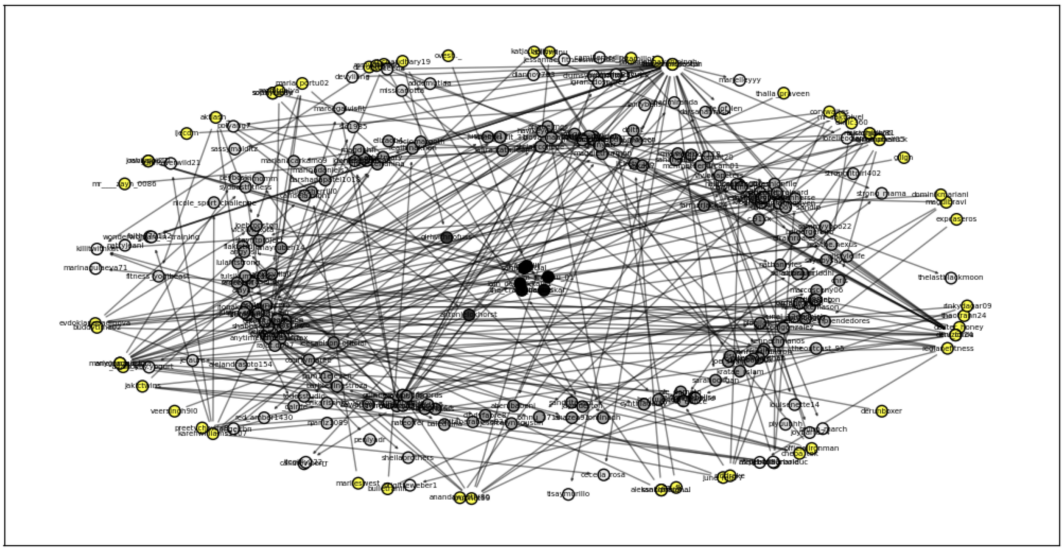


Figura 5.14: Plankchallenge, *Fonte: autoprodotta*

5.4.2 Reti di challenge negative

- *Fightchallenge.* È la challenge in cui adolescenti, ragazzi o adulti ne riprendono altri mentre si picchiano. Questa challenge è pericolosa e diseducativa (Figura 5.15). I valori delle grandezze considerate sono i seguenti:
  - Numero di nodi: 409;
  - Numero di archi: 339;
  - Indegree medio (numero di archi diretti verso un nodo): 0.8288508557457213;
  - Indegree medio std (deviazione standard): 1.3268153683786494;
  - Outdegree medio (numero di archi che escono da un nodo): 0.8288508557457213;
  - Outdegree medio std (deviazione standard): 1.2951098167389872;
  - Clustering coefficient medio: 0.0008964955175224124;
  - Densità: 0.002031497195455199.

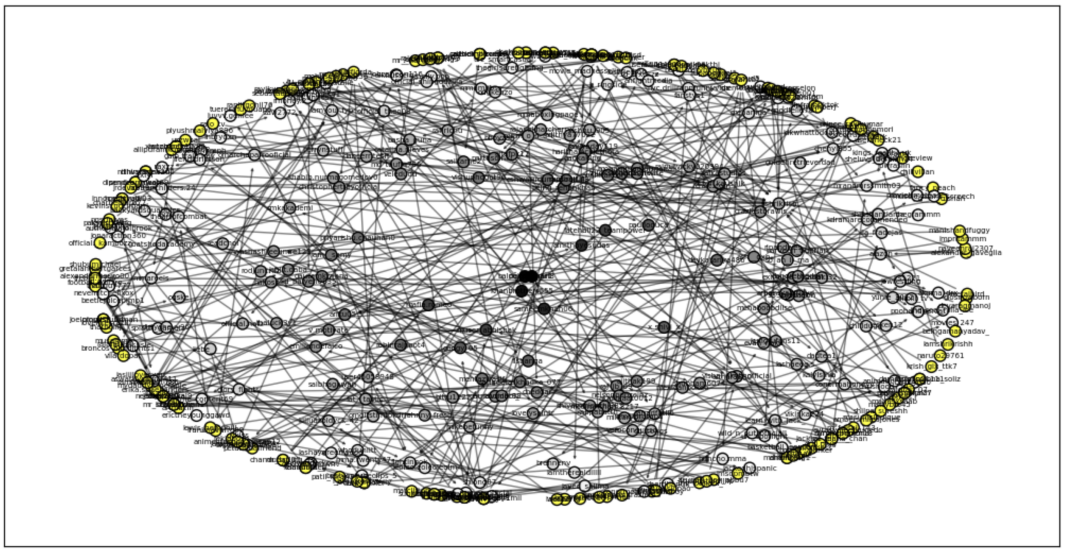


Figura 5.15: Fightchallenge, *Fonte: autoprodotta*

- *Silhouettechallenge.* Si tratta di una challenge in cui ci si mostra senza troppi vestiti, mostrando l'ombra della silhouette del corpo. Il pericolo consiste nel fatto che è stato scoperto un modo che permette di eliminare il filtro. Sebbene il trend sia nato per combattere ogni forma di discriminazione legata al peso e alla forma dei corpi, si potrebbe mettere a rischio la privacy di centinaia di migliaia di utenti (Figura 5.16). I valori delle grandezze considerate sono i seguenti:
  - Numero di nodi: 262;
  - Numero di archi: 259;
  - Indegree medio (numero di archi diretti verso un nodo): 0.9810606060606061;
  - Indegree medio std (deviazione standard): 1.1853034757648517;
  - Outdegree medio (numero di archi che escono da un nodo): 0.9810606060606061;
  - Outdegree medio std (deviazione standard): 2.562037321888803;
  - Clustering coefficient medio: 0.0;
  - Densità: 0.0037302684641087685.

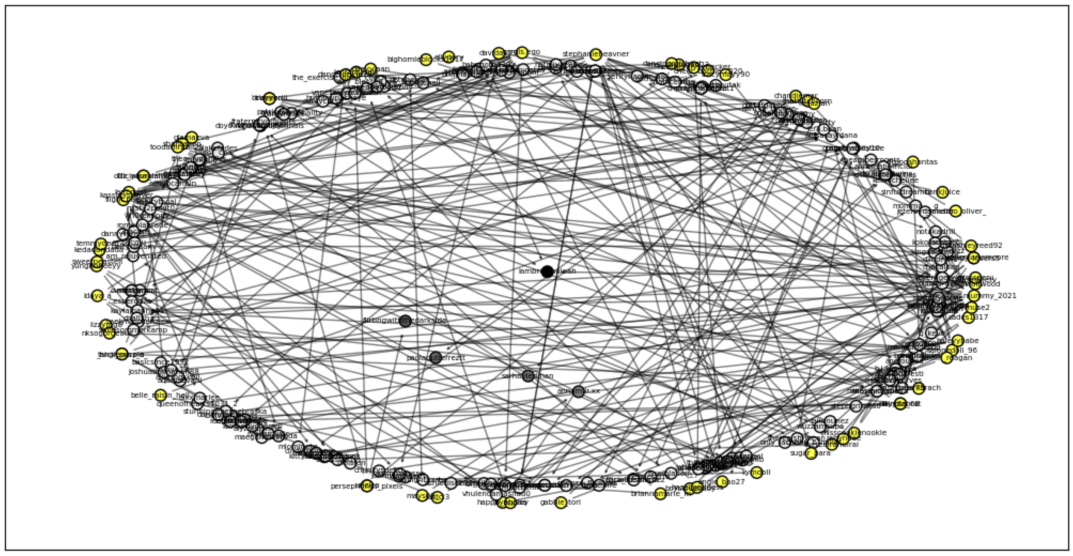


Figura 5.16: Silhouettechallenge, *Fonte: autoprodotta*

- *Bugsbunnychallenge.* Le ragazze simulano di avere le orecchie da coniglio, utilizzando i piedi e una prospettiva di ripresa che permette di ottenere la sagoma corretta. La negatività di questa challenge sta nel fatto che diversi utenti la effettuino con pochi indumenti, permettendola visione di parti intime (Figure 5.17). I valori delle grandezze considerate sono i seguenti:
  - Numero di nodi: 212;
  - Numero di archi: 239;
  - Indegree medio (numero di archi diretti verso un nodo): 1.1273584905660377;
  - Indegree medio std (deviazione standard): 2.1029643506640237;
  - Outdegree medio (numero di archi che escono da un nodo): 1.1273584905660377;
  - Outdegree medio std (deviazione standard): 2.5657038836651287;
  - Clustering coefficient medio: 0.0;
  - Densità: 0.005342931234910132.

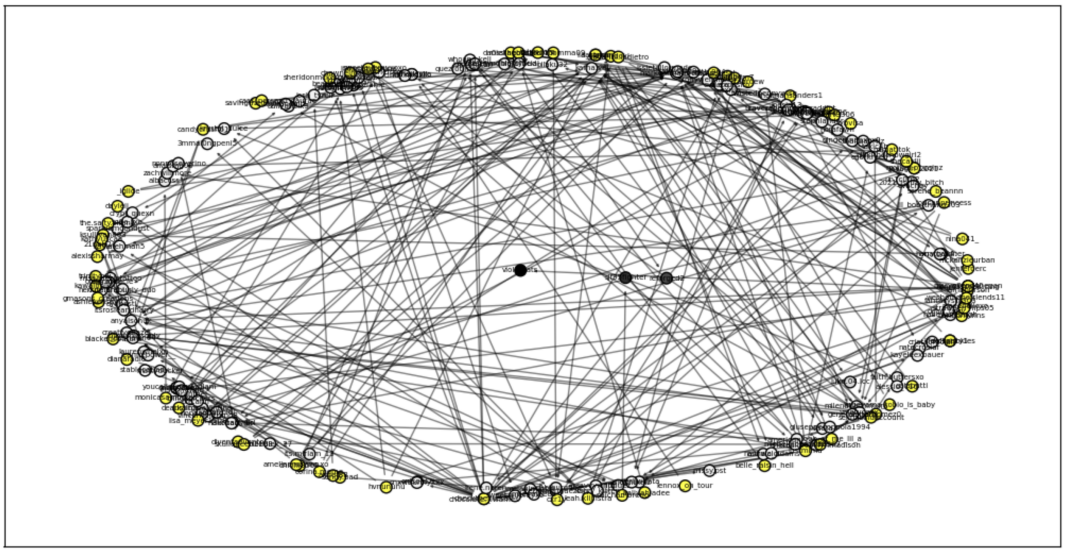


Figura 5.17: Bugsbunnychallenge, *Fonte: autoprodotta*

- *Strippatiktok.* In questa challenge diverse stripper professioniste mostrano la loro vita e i loro guadagni, inducendo possibili utenti adolescenti all'emulazione. Non ci sono azioni che mettono in pericolo la salute di una persona, ma sono sicuramente video negativi, poiché rappresentano una cattiva influenza per gli adolescenti (Figura:5.18). I valori delle grandezze considerate sono i seguenti:
  - Numero di nodi: 297;
  - Numero di archi: 519;
  - Indegree medio (numero di archi diretti verso un nodo): 1.7474747474747474;
  - Indegree medio std (deviazione standard): 2.0597641377678086;
  - Outdegree medio (numero di archi che escono da un nodo): 1.7474747474747474;
  - Outdegree medio std (deviazione standard): 5.928888530941131;
  - Clustering coefficient medio: 0.0024798581222891585;
  - Densità: 0.005903630903630904.

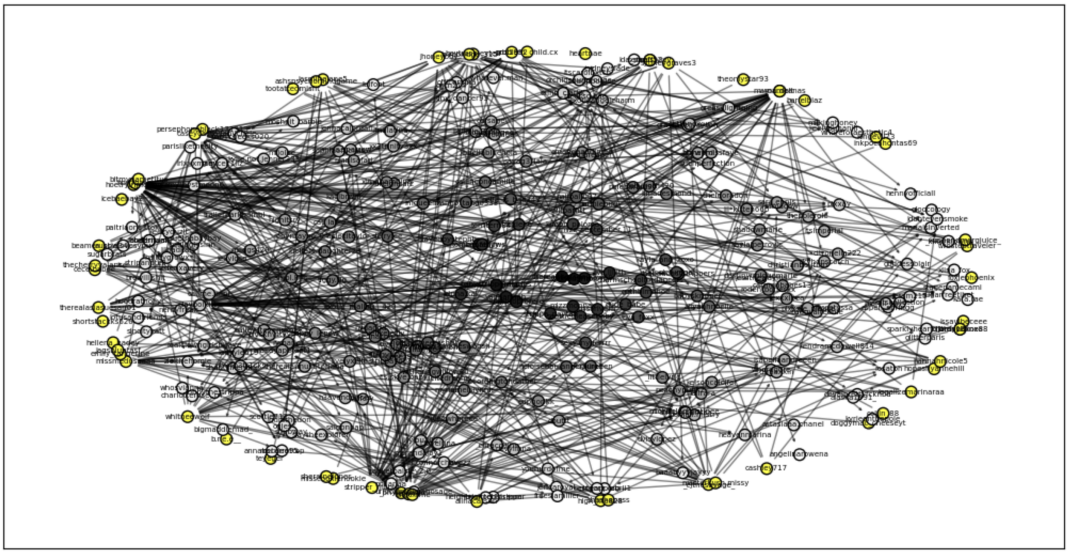


Figura 5.18: Strippatiktok, *Fonte: autoprodotta*

- *Firewroks.* Consiste nel far esplodere fuochi d'artificio in prossimità di soggetti ed oggetti a rischio. Questo può essere potenzialmente dannoso per chi si trova vicino ad un'esplosione indotta da persone non professioniste, poco consce del pericolo della loro azione (Figura 5.19). I valori delle grandezze considerate sono i seguenti:
  - Numero di nodi: 141;
  - Numero di archi: 111;
  - Indegree medio (numero di archi diretti verso un nodo): 0.7872340425531915;
  - Indegree medio std (deviazione standard): 0.7700834498235685;
  - Outdegree medio (numero di archi che escono da un nodo): 0.7872340425531915;
  - Outdegree medio std (deviazione standard): 1.7002649382133657;
  - Clustering coefficient medio: 0.008274231678486997;
  - Densità: 0.005623100303951368.

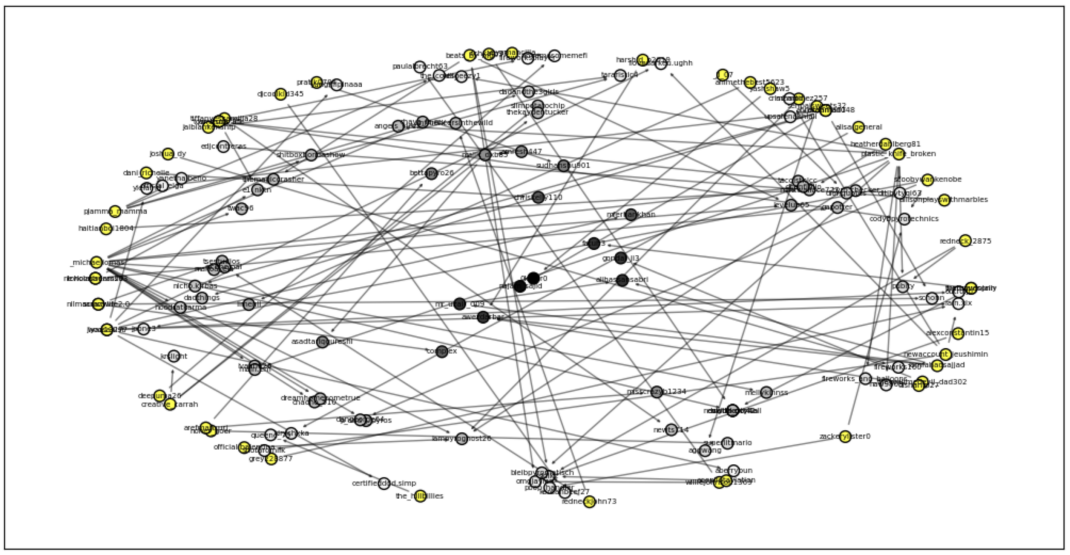


Figura 5.19: Firewroks, *Fonte: autoprodotta*

- *Updownchallenge.* È un gioco di coppia in cui si afferranno mani e piedi e si cerca di ribaltare la posizione attuale da terra. Il rischio è quello di sbattere la testa nel momento in cui ci si capovolge e c'è il rischio di farsi davvero male (Figura 5.20). I valori delle grandezze considerate sono i seguenti:
  - Numero di nodi: 243;
  - Numero di archi: 199;
  - Indegree medio (numero di archi diretti verso un nodo): 0.8189300411522634;
  - Indegree medio std (deviazione standard): 1.2701591258508924;
  - Outdegree medio (numero di archi che escono da un nodo): 0.8189300411522634;
  - Outdegree medio std (deviazione standard): 1.1898627403128363;
  - Clustering coefficient medio: 0.0102880658436214;
  - Densità: 0.003384008434513485.

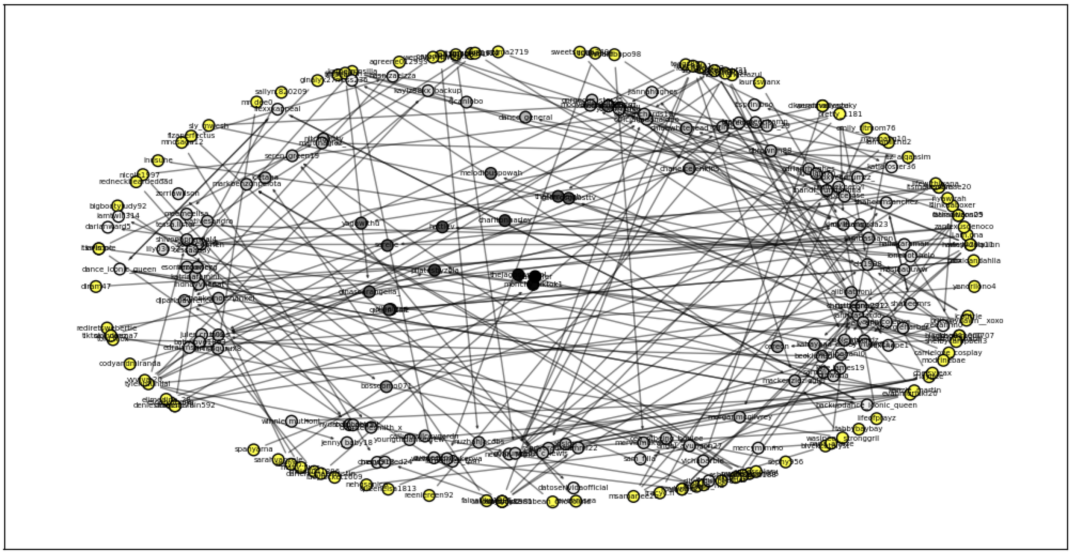


Figura 5.20: Updownchallenge, *Fonte: autoprodotta*

- *Sugarbaby.* Alcune persone mostrano la vita e gli svaghi che si concedono grazie a particolari lavori, come fare le accompagnatrici di personaggi benestanti. Viene considerata negativa per l'influenza che potrebbe avere sugli adolescenti e sulle nuove generazioni (Figura: 5.21). I valori delle grandezze considerate sono i seguenti:
  - Numero di nodi: 151;
  - Numero di archi: 143;
  - Indegree medio (numero di archi diretti verso un nodo): 0.9470198675496688;
  - Indegree medio std (deviazione standard): 1.0084942514632496;
  - Outdegree medio (numero di archi che escono da un nodo): 0.9470198675496688;
  - Outdegree medio std (deviazione standard): 2.298251795927956;
  - Clustering coefficient medio: 0.0035492769110754474;
  - Densità: 0.00631346578366446.

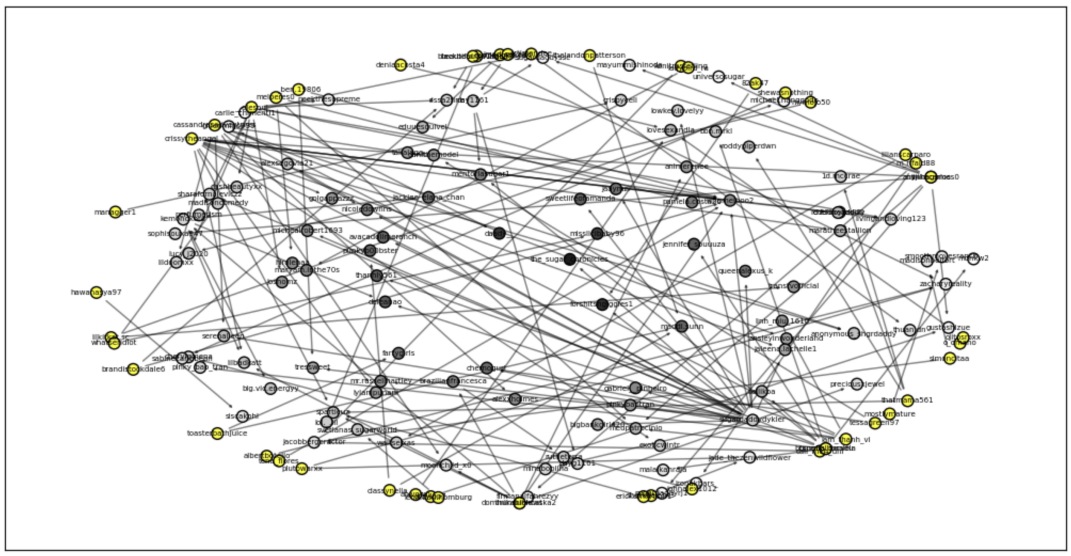


Figura 5.21: Sugarbaby, *Fonte: autoprodotta*

5.5 Analisi per tipologia di challenge

In questa sezione verranno analizzate le challenge nel loro insieme, considerando una visione più globale delle stesse e suddividendole per tipologia, ovvero considerando un'analisi globale per le positive e una per le negative. Questa parte di analisi è stata creata mediante il codice che si trova all'interno del file "pre_analysis.py".

In questo file sono state considerate tutte le challenge positive e negative; dopodiché sono state importate tutte le informazioni relative al grafo mediante la funzione "graphCalculation", ovvero la funzione utilizzata per costruire la rete, che si trova all'interno del file "tiktok_network.py". Successivamente sono state stampate sul video diverse informazioni relative alla globalità delle challenge considerate, suddivise per tipologia:

- "Mean number of nodes": il numero medio di nodi;
- "Mean number of edges": il numero medio di archi;
- "Mean indegree": l'*indegree* medio;
- "Mean outdegree": l'*outdegree* medio;
- "Mean clustering coefficient": il valore medio del *clustering coefficient*;
- "Mean lifespan (days)": la media di *lifespan*, ovvero la media del tempo di vita, relativamente a quella tipologia di challenge, espressa in giorni;

- "Mean 5percent lifespan number of nodes": numero di nodi nel primo 5% di lifespan;
- "Mean 25percent lifespan number of nodes": numero di nodi nel primo 25% di lifespan;
- "Mean 50percent lifespan number of nodes": numero di nodi nel primo 50% di lifespan;
- "Mean 75percent lifespan number of nodes": numero di nodi nel primo 75% di lifespan;
- "Mean density": densità media per quella tipologia di challenge;
- "Mean video duration": durata media dei video;
- "Mean shares count": media del numero di condivisioni;
- "Mean likes count": media del numero di like;
- "Mean comments count": numero medio di commenti;
- "Mean views count": numero medio di visualizzazioni;
- "Mean different music count": numero medio di musiche utilizzate;
- "Mean likes given by the author count": numero medio di like messi dagli utenti che hanno pubblicato il video;
- "Mean author following count": numero medio di persone seguite dagli autori dei video;
- "Mean likes received by the author count": numero medio di like ricevuti dagli autori;
- "Mean author follower count": numero medio di follower degli autori;
- "Mean published videos by the author count": numero medio di video pubblicati dagli autori.

Considerando tutte le challenge positive, ed operando una media tra tutti i valori, sono stati ottenuti i risultati medi presenti nella Figura 5.22, che rappresenta uno screen dell'output ottenuto da terminale, in seguito all'avvio dello script presente nel file "pre_analysis".

```
**********POSITIVE**********
Mean number of nodes: 397.42857142857144 (std: 176.7022169339925)
Mean number of edges: 435.85714285714283 (std: 235.38140997807218)
Mean indegree: 1.0514526800506354 (std: 0.10541502122085629)
Mean outdegree: 1.0514526800506354 (std: 0.10541502122085629)
Mean clustering coefficient: 0.004926065802210305 (std: 0.004823241528141782)
Mean lifespan (days): 405.0 (std: 360.53808992353953)
Mean 5percent lifespan number of nodes: 0.02577867411761835 (std: 0.023422375762886326)
Mean 25percent lifespan number of nodes: 0.27588664782412164 (std: 0.2658177462998551)
Mean 50percent lifespan number of nodes: 0.3431705111023088 (std: 0.31967014089551427)
Mean 75percent lifespan number of nodes: 0.4418870524525513 (std: 0.3023960510211521)
Mean density: 0.003057139484757649 (std: 0.0009279240443667967)
Mean video duration: 20.126312984878307 (std: 5.027297500377051)
Mean likes count: 155516.2971533283 (std: 119351.77149364064)
Mean comments count: 1542.5055355291242 (std: 1404.9443200149535)
Mean views count: 1350133.9581025508 (std: 1129098.882266855)
Mean different music count: 162.57142857142858 (std: 155.33085163414177)
Mean likes given by the author count: 14402.298945195758 (std: 9722.36972347388)
Mean author following count: 1164.564972507375 (std: 784.6585970635088)
Mean likes received by the author count: 7004933.38601145 (std: 5421016.823545114)
Mean author follower count: 385590.4287211372 (std: 262090.0932376342)
Mean published videos by the author count: 351.51570030489205 (std: 174.88248681854546)
******************************************************
```

Figura 5.22: Metriche delle challenge positive, *Fonte: autoprodotta*

Come si può notare dalla figura appena riportata, la prima cosa che viene stampata sullo schermo è la dicitura "positive", che sta ad indicare il fatto che vengono considerate tutte le challenge positive. Dopodiché vengono riportati tutti i parametri medi calcolati. La stessa operazione è stata fatta anche per le challenge negative. Anche in questo caso, sono state considerate tutte le metriche calcolate singolarmente per ogni challenge, per poi effettuare una media. Sono state ottenute le metriche successive (Figura 5.23).

```
**********NEGATIVE**********
Mean number of nodes: 245.66666666666666 (std: 91.92327718749418)
Mean number of edges: 268.3333333333333 (std: 134.93043475147564)
Mean indegree: 1.0698331016583287 (std: 0.3223366997444435)
Mean outdegree: 1.0698331016583287 (std: 0.3223366997444435)
Mean clustering coefficient: 0.0025333103715623356 (std: 0.0028725049564332205)
Mean lifespan (days): 550.1666666666666 (std: 318.53802319695245)
Mean 5percent lifespan number of nodes: 0.008506127670750882 (std: 0.0050803082278690695)
Mean 25percent lifespan number of nodes: 0.02983721894408339 (std: 0.026060019933023974)
Mean 50percent lifespan number of nodes: 0.0916296884409476 (std: 0.07759287815326026)
Mean 75percent lifespan number of nodes: 0.23586615337107433 (std: 0.16622271036399658)
Mean density: 0.004824148980953472 (std: 0.0014880794436646749)
Mean video duration: 20.624773026465082 (std: 4.965933214648054)
Mean shares count: 6263.079085952716 (std: 4342.868560822319)
Mean likes count: 234757.6804569841 (std: 123295.26817854658)
Mean comments count: 2527.1630145092636 (std: 951.3799573789522)
Mean views count: 1887515.5643671707 (std: 1219301.2989547413)
Mean different music count: 120.66666666666667 (std: 76.6869538375915)
Mean likes given by the author count: 11275.020709972972 (std: 1774.4371819143864)
Mean author following count: 641.8561971321036 (std: 172.201828161197)
Mean likes received by the author count: 10786164.94111401 (std: 7985241.823490642)
Mean author follower count: 406975.8600137855 (std: 272406.0132533508)
Mean published videos by the author count: 254.80776062380195 (std: 54.28336721732613)
**********
```

Figura 5.23: Metriche delle challenge negative, *Fonte: autoprodotta*

Per meglio confrontarle, le stesse metriche sono state inserite all'interno delle Tabelle 5.24, 5.25 e 5.26.

	CHALLENGES	
METRICHE (ANALISI QUANTITATIVA RETE)	**POSITIVE**	**NEGATIVE**
Numero medio nodi	397.42857142857144 (std: 176.7022169339925)	264.14285714285717 (std: 96.3897276261155)
Numero medio archi	435.85714285714283 (std: 235.38140997807218)	272.0 (std: 125.24376231972593)
Degree in medio	1.0514526800506354 (std: 0.10541502122085629)	1.0289998014214246 (std: 0.3147413316992674)
Degree out medio	1.0514526800506354 (std: 0.10541502122085629)	1.0289998014214246 (std: 0.3147413316992674)
Densita' media	0.003057139484757649 (std: 0.0009279240443667967)	0.004434450081458973 (std: 0.0016760762997961 13)
Clustering coefficient medio	0.004926065802210305 (std: 0.004823241528141782)	0.0037714088899105736 (std: 0.004033591130850253)
Lifespan medio in giorni	405.57142857142856 (std: 360.12730175507073)	790.2857142857143 (std: 339.39261473811644)
Numero medio di nodi nel primo 5% del lifespan (normalizzato)	0.03407898418412558 (std: 0.03360736415464353)	0.004734745805942603 (std: 0.0015208034393389578)
Numero medio di nodi nel primo 25% del lifespan (normalizzato)	0.3816137950942098 (std: 0.33532646066788824)	0.012617989098519645 (std: 0.009477136272783496)
Numero medio di nodi nel primo 50% del lifespan (normalizzato)	0.4834940045663992 (std: 0.41259283051009393)	0.05602328567930863 (std: 0.04901102230109351)
Numero medio di nodi nel primo 75% del lifespan (normalizzato)	0.6218294229258083 (std: 0.3866255928173815)	0.2554617671877901 (std: 0.20820688964327472)

Figura 5.24: Analisi quantitativa della rete, *Fonte: autoprodotta*

	CHALLENGES	
METRICHE (ANALISI QUANTITATIVA VIDEO)	**POSITIVE**	**NEGATIVE**
Durata media dei video	19.913924040291388 (std: 4.667971548666577)	19.650845868120744 (std: 4.525320681299974)
Numero medio di musiche utilizzate per video	195.57142857142858 (std: 178.05926552925106)	172.28571428571428 (std: 113.81026100578138)
Numero medio di likes	127970.48371056521 (std: 102029.4318979586)	140851.61548785152 (std: 86342.78005390946)
Numero medio di commenti	1302.278495121724 (std: 1241.8369960425507)	1516.953251052944 (std: 818.5186934849705)
Numero medio di condivisioni	4098.141309036778 (std: 4064.3674893922475)	3700.7301212097072 (std: 2827.089360252258)
Numero medio di views	1118085.921905425 (std: 944554.4688768962)	1117946.118077494 (std: 778954.2713391208)

Figura 5.25: Analisi quantitativa dei video, *Fonte: autoprodotta*

	CHALLENGES	
METRICHE (ANALISI QUANTITATIVA AUTORI)	**POSITIVE**	**NEGATIVE**
Numero medio di likes messi dagli autori	14746.383890596222 (std: 9614.174340545957)	14001.363118809499 (std: 2039.3912309143482)
Numero medio di following degli autori	1219.7271627756743 (std: 811.75799106804)	1004.1241780347457 (std: 196.78384958106136)
Numero medio di likes ricevuti dagli autori	6311099.805816695 (std: 5082668.524184106)	6885791.910503639 (std: 4770140.289347319)
Numero medio di follower degli autori	343151.98215558525 (std: 248888.0159774165)	275155.6317362132 (std: 154290.0027294992)
Numero medio di video pubblicati dagli autori	372.67256481159023 (std: 196.38430792959468)	305.33493562778597 (std: 57.45177217743928)

Figura 5.26: Analisi quantitativa degli autori, *Fonte: autoprodotta*

5.5.1 Considerazioni

Per quello che riguarda una visione più globale, è possibile notare che, in media, le challenge positive hanno un numero maggiore di nodi e di archi, anche se la differenza non è eccessiva. Hanno entrambe dei bassi valori di densità e di *clustering coefficient* medio. Se si considera il *lifespan*, ovvero la vita media delle challenge in giorni, è possibile notare come le negative abbiano mediamente una durata maggiore. Sicuramente, le challenge negative sono quelle che hanno un impatto più profondo sugli adolescenti e, per questo, saranno sicuramente ricondivise più volte e più a lungo nel protrarsi del tempo.

Relativamente ai video, secondo l'analisi quantitativa effettuata, è possibile notare come la durata media sia molto simile tra le due tipologie di challenge, sebbene la differenza maggiore risieda nel numero di interazioni. Il numero medio di like delle challenge negative, infatti, è pressoché il doppio rispetto a quello delle positive, così come il numero medio di commenti. Anche le visualizzazioni medie delle negative sono nettamente superiori. È piuttosto comprensibile

un risultato di questo tipo, poiché, come precedentemente spiegato, i video negativi o pericolosi tendono ad attrarre maggiormente l'attenzione delle persone, soprattutto se si parla della generazione Z. Questo aspetto si ritrova anche nel momento in cui si va ad affrontare un'analisi quantitativa relativamente agli autori. È immediato notare come gli autori di video relativi a challenge negative abbiano un numero nettamente maggiore di like ricevuti, a parità di follower. Questo significa che molte persone avranno visualizzato quel determinato video e avranno cliccato "like", oltre agli utenti che già seguivano l'autore. Sicuramente la pericolosità dell'azione crea una suspence nello spettatore, che è portato ad aumentare il tempo di visualizzazione del video e il conseguente interesse verso quella tipologia di challenge.

6. Studio del ciclo di vita

6.1 Analisi media delle challenge positive e negative

Il lavoro successivo è stato incentrato sulla rappresentazione degli andamenti delle curve relative alle challenge positive e negative. Questa operazione è stata attuata mediante il file "pre_analysis.py", posizionato all'interno della working directory, graficando i dati ottenuti. È stato considerato sull'asse delle ascisse il *lifespan*, normalizzato tra 0 e 100, mentre sull'asse delle ordinate è stato riportato il numero medio di nodi, normalizzato tra 0 e 1 (Figura 6.1).

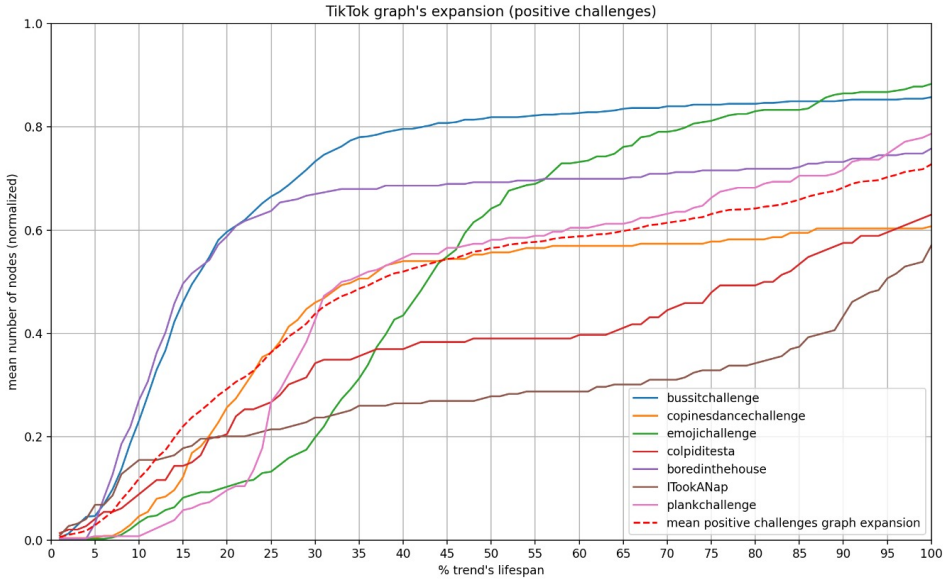


Figura 6.1: Andamento delle challenge positive, *Fonte: autoprodotta*

Il numero di nodi, come spiegato in precedenza, rappresenta il numero di utenti che ha pubblicato un video relativo alla challenge in questione. Per ogni 5% della progressione del lifespan è stato riportato il numero di nodi presenti in quell'intervallo, accumulandoli ai valori degli intervalli precedenti, se presenti. Come risultato si ottengono le curve cumulative di ogni challenge, che considerano il numero di nodi ricavati dopo un certo tempo trascorso dall'inizio del lifespan.

Com'è possibile notare dalla Figura 6.1, le challenge positive hanno sommariamente un andamento di tipo polinomiale, caratterizzate da un primo momento di rapida crescita, un momento di stabilizzazione e un'ultima parte di nuova crescita. È stata, poi, calcolata la curva media derivante da tutte le curve rappresentative delle challenge ed è stata riportata sul grafico mediante una linea rossa tratteggiata.

Nella parte superiore, a sinistra, è presente la legenda, che riporta i colori con cui sono state disegnate le curve delle challenge, con lo scopo di ottenere una migliore identificazione delle stesse. Inoltre è riportata la media tra tutte le curve, costituita da un tratteggio per meglio distinguerla dalle altre.

Lo stesso grafico è stato riportato anche per le challenge negative in Figura 6.2.

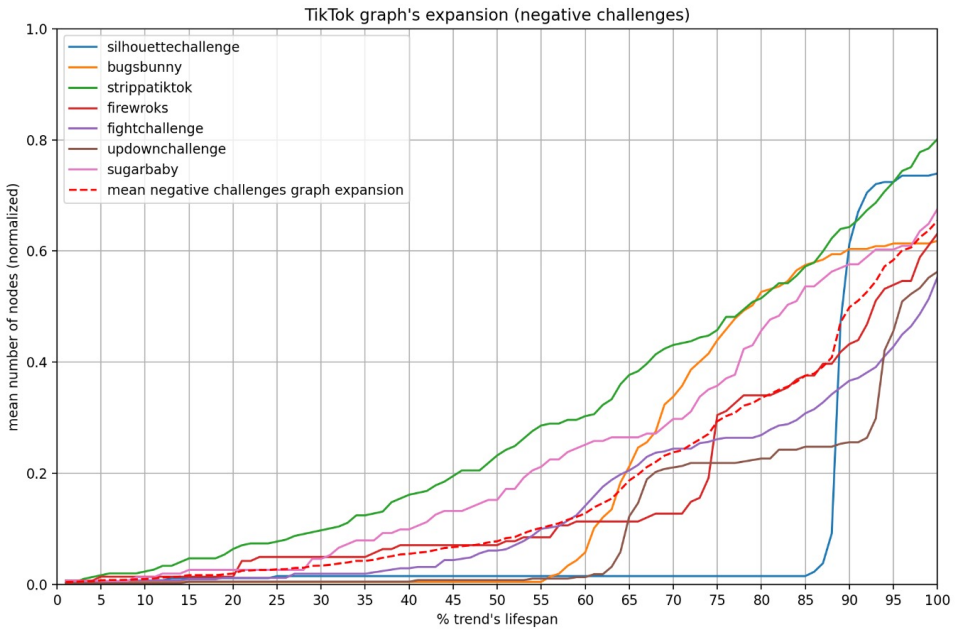


Figura 6.2: Andamento delle challenge negative, *Fonte: autoprodotta*

Le challenge negative hanno un andamento sempre crescente, si potrebbe dire esponenziale, la cui media, anche in questo caso, è rappresentata dalla linea tratteggiata in rosso. Una challenge in particolare, ovvero *silhouettechallenge*, presenta un andamento piuttosto estremo rispetto alle altre. Dopo aver verificato che la sua rimozione non portava pressoché alcun cambiamento al risultato medio, non è stata definita come outlier, ma come una challenge avente un andamento piuttosto repentino, sebbene segua comunque l'evoluzione delle altre challenge di categoria.

Per avere un confronto migliore delle due medie, queste ultime sono state rappresentate all'interno dello stesso grafico, colorando le positive mediante un colore blu e le negative con colore arancione, così come riportato in Figura 6.3.

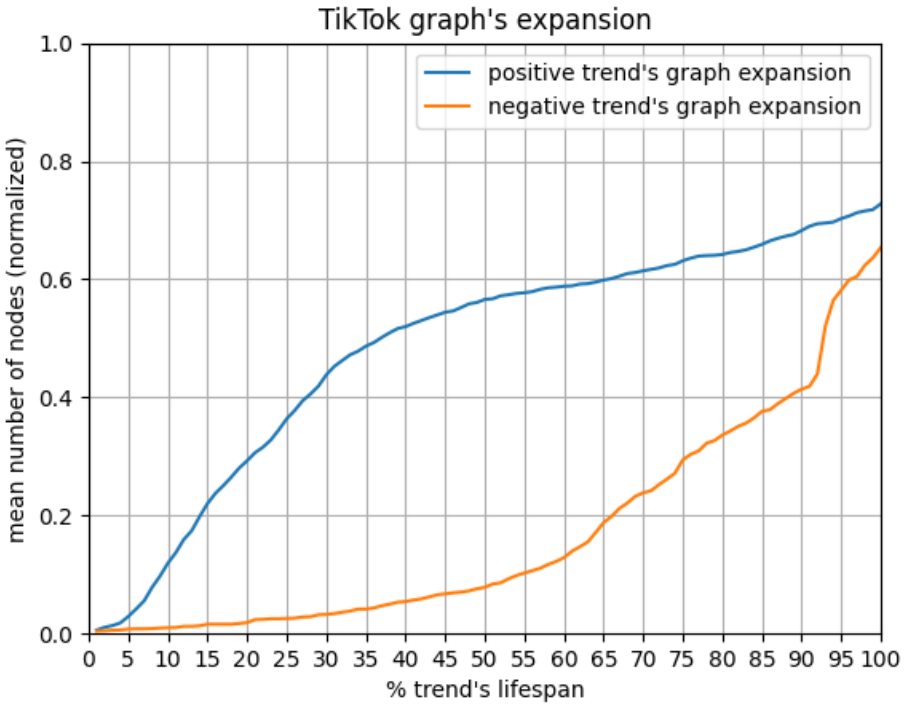


Figura 6.3: Andamento medio delle challenge a confronto, *Fonte: autoprodotta*

Mettendo a confronto gli andamenti delle due medie, ottenute dai valori di tutte le challenge positive e di tutte quelle negative, si nota una netta differenza tra le due diverse tipologie di challenge. Questo rappresenta, di per sé, un ottimo risultato, poiché conferma l'ipotesi di partenza, ovvero che ci fosse una diversità di propagazione tra le challenge positive e quelle negative, cercando di mettere in luce le eventuali differenze trovate.

È piuttosto semplice notare che, in media, le challenge positive hanno una crescita immediatamente più elevata, fino ad un lifespan che si aggira attorno a 30-40%. Subito dopo si stabilizzano attorno ad un certo valore per un minor lasso di tempo rispetto alla fase di crescita precedente. Solitamente, intorno ad un lifespan pari a 70-75%, si ha nuovamente una crescita finale, meno ripida rispetto alla prima. A differenza delle positive, invece, quelle negative hanno uno scarso impatto iniziale, poiché si verifica una crescita molto lenta, pressoché nulla, per poi esplodere in una seconda fase e mantenere la crescita piuttosto costante fino al termine del lifespan.

È stata, quindi, effettuata un'approssimazione delle medie, mediante una curva di tipo polinomiale per le positive e una di tipo esponenziale per le negative, in modo da poterne successivamente studiare gli andamenti con maggiore precisione. Come si nota dal grafico riportato nella Figura 6.4, le curve approssimano molto bene gli andamenti. Per la media delle challenge positive con andamento polinomiale è stata considerata un'approssimazione mediante una curva di quarto grado, mentre per la media delle challenge negative è stata utilizzata un'esponenziale.

Anche in questo caso, in alto a destra, è presente la legenda che suddivide le curve per colore, dove la curva reale che rappresenta la medie delle challenge positive è blu, mentre la sua approssimata è arancione. Per quello che riguarda le negative, la curva reale che rappresenta le medie delle challenge negative è colorata di verde, mentre la sua approssimata è di colore rosso.

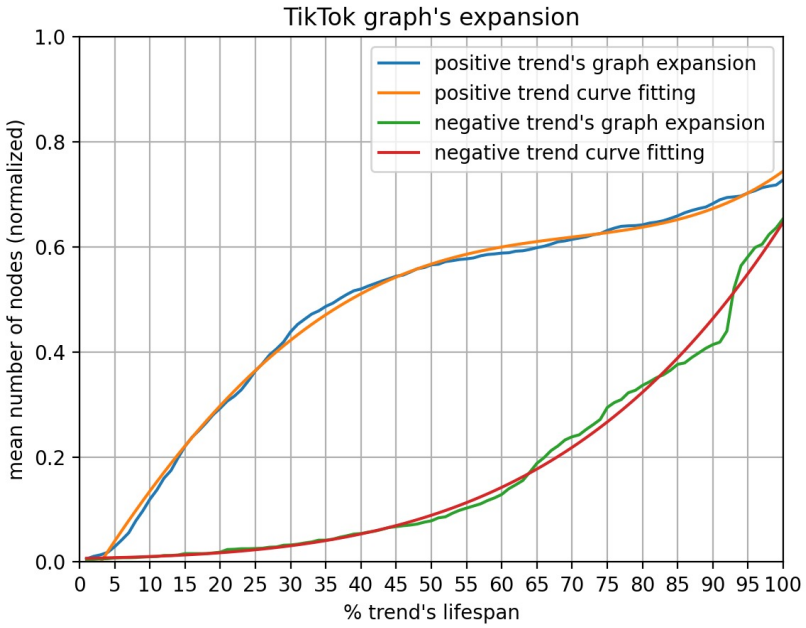


Figura 6.4: Approssimazione delle curve medie, *Fonte: autoprodotta*

Considerando che la derivata rappresenta il tasso di cambiamento di una funzione rispetto ad una variabile, ovvero la misura di quanto il valore di una funzione cambia al variare del suo argomento, è possibile dire che la derivata misura la crescita (o decrescita) che avrebbe una funzione in uno specifico punto spostandosi di pochissimo dal punto considerato. Per questo è stato generato anche un grafico relativo alle derivate prime delle curve medie riportato in Figura 6.5.

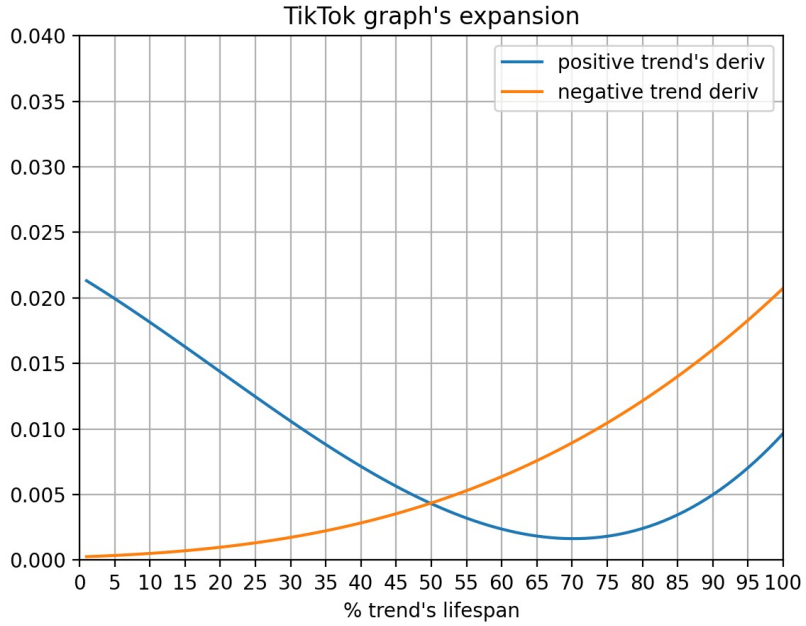


Figura 6.5: Derivata delle curve medie, *Fonte: autoprodotta*

Anche da questo grafico, ovviamente, è possibile notare il grande divario che intercorre tra le due tipologie di challenge: le positive hanno una derivata che decresce fino ad un minimo per poi risalire, mentre le negative hanno un andamento che continua ad essere esponenziale.

OSSERVAZIONE

L'andamento medio delle challenge positive ricorda l'andamento della *massa critica*. La massa critica è il valore massimo raggiunto da una rete, oltre al quale si ha una stabilizzazione, come vediamo dal seguente grafico:

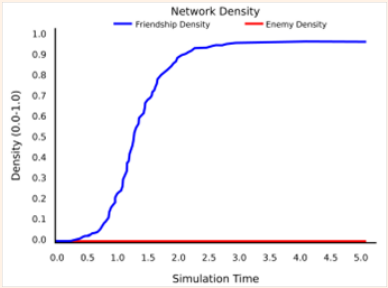


Figura 6.6: Massa critica, *Fonte: autoprodotta*

6.2 Identificazione Intervalli

L'obiettivo del lavoro seguente si incentra sulla determinazione degli intervalli che compongono il ciclo di vita delle challenge considerate. In particolare, si è cercato di identificare un metodo tramite il quale, data una nuova curva, si riesca a calcolare la polinomiale più vicina alla curva stessa, per poi determinare i diversi intervalli in cui l'andamento della curva cambia. Per l'identificazione degli intervalli sono stati considerati i punti di annullamento della derivata prima, che identificano i punti di massimo e di minimo della curva di partenza. Chiaramente, un punto di massimo determinerà un intervallo precedente in cui la curva sarà crescente e, successivamente, decrescente; viceversa per i punti di minimo.

Finora sono stati considerati grafici cumulativi, ovvero grafici in cui ogni utente che pubblicava un video relativo alla challenge in un determinato momento veniva sommato ai precedenti utenti che avevano pubblicato un video della stessa challenge in momenti precedenti del lifespan. Questo era stato fatto per ottenere una visualizzazione che mostrasse l'andamento totale della challenge stessa e la sua evoluzione nel tempo. Per svolgere questo studio, però, è stato necessario considerare non più l'andamento cumulativo degli utenti, ma l'andamento puntuale, ovvero il numero di utenti che pubblicano un video in ogni avanzamento percentuale del 5% del lifespan. In particolare, per ogni step del 5% sull'asse delle x (che rappresenta il *lifespan*), è stato riportato il numero di utenti che hanno pubblicato un video in quell'intervallo di tempo. È stata effettuata questa operazione perché, altrimenti, la curva cumulativa avrebbe avuto un andamento sempre crescente a priori, o tutt'al più costante (nel caso in cui nessun utente avesse caricato video), mentre volevamo individuare i punti di crescita e di decrescita della curva, con l'obiettivo di distinguere i momenti di maggiore interazione da quelli di minore interesse.

Questo procedimento è stato attuato mediante lo script contenuto nel file "lifespan_analysis", che verrà riportato in appendice.

Le curve ottenute per ciascuna challenge sono state interpolate con una curva di approssimazione, in modo da poterne calcolare la derivata prima. Dopo aver posto la derivata prima pari a 0, sono stati individuati i punti di massimo e minimo e sono stati definiti gli intervalli. Questo è stato possibile

poiché il massimo o il minimo di una curva indica un conseguente cambiamento dell'andamento della curva stessa, quindi è sicuramente un buon indicatore per l'identificazione degli intervalli.

Si riportano, di seguito, i grafici ottenuti per ogni challenge, con i relativi intervalli, partendo dalle challenge positive:

- *boredinthehouse*, definita da 3 punti e 4 intervalli compresi tra [0, 15], [15, 60], [60, 90], [90, 100] (Figura 6.7).

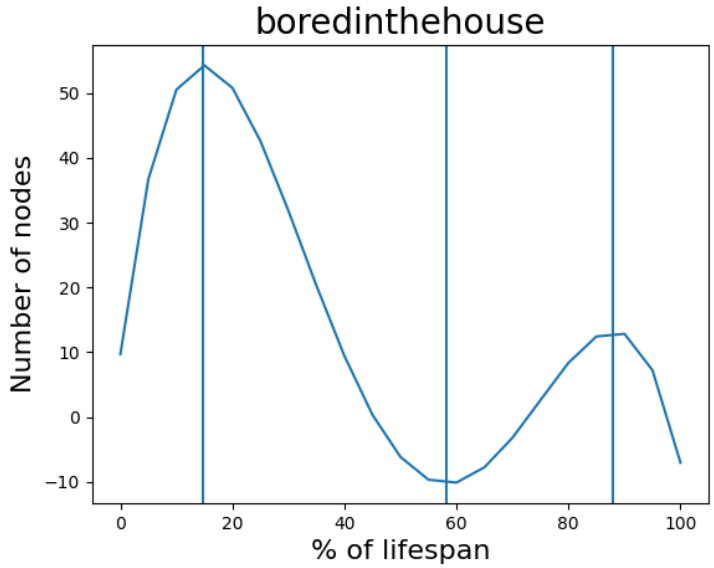


Figura 6.7: Challenge positiva: *boredinthehouse*, *Fonte: autoprodotta*

- *bussitchallenge*, definita da 3 punti e 4 intervalli compresi tra [0, 15], [15, 60], [60, 90], [90, 100] (Figura 6.8).

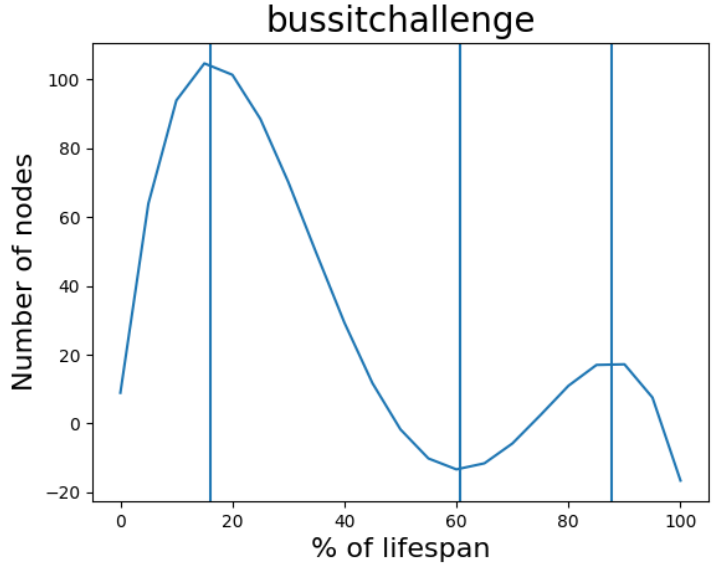


Figura 6.8: Challenge positiva: *bussitchallenge*, *Fonte: autoprodotta*

- *colpiditesta*, definita da 3 punti e 4 intervalli compresi tra [0, 15], [15, 45], [45, 95], [95, 100] (Figura 6.9).

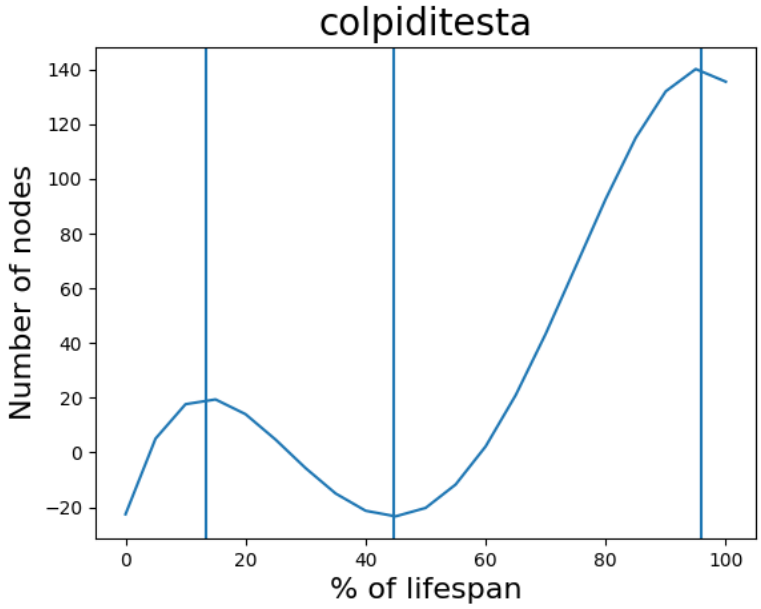


Figura 6.9: Challenge positiva: *colpiditesta*, *Fonte: autoprodotta*

- *copinesdancechallenge*, definita da 3 punti e 4 intervalli compresi tra [0, 20], [20, 65], [65, 90], [90, 100] (Figura 6.10).

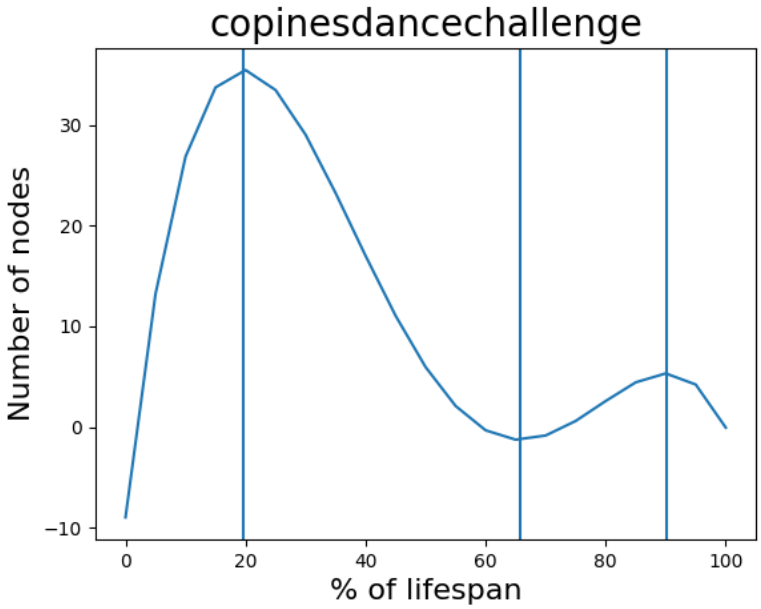


Figura 6.10: Challenge positiva: *copinesdancechallenge*, *Fonte: autoprodotta*

- *emojichallenge*, definita da 3 punti e 4 intervalli compresi tra [0, 10], [10 ,35], [35, 65], [65, 100] (Figura 6.11).

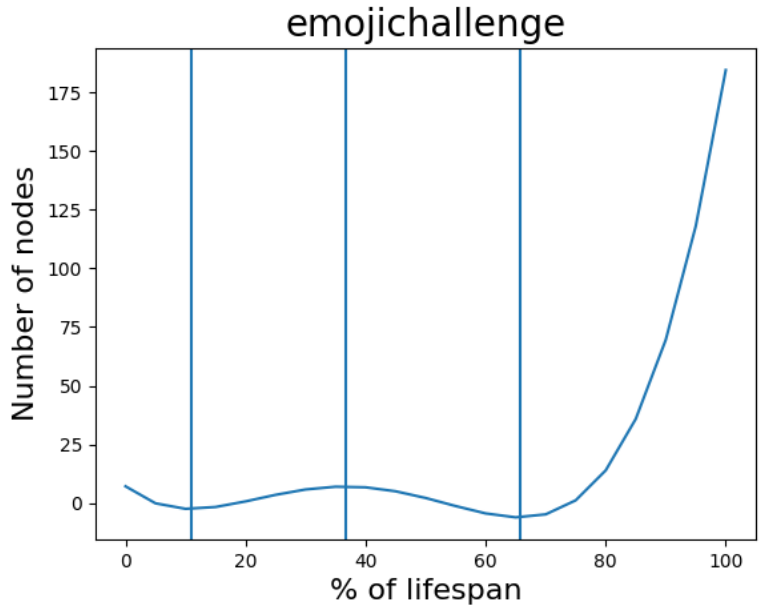


Figura 6.11: Challenge positiva: *emojichallenge*, *Fonte: autoprodotta*

- *ITookANap*, definita da 2 punti e 3 intervalli compresi tra [0, 10], [10, 55], [55, 100] (Figura 6.12).

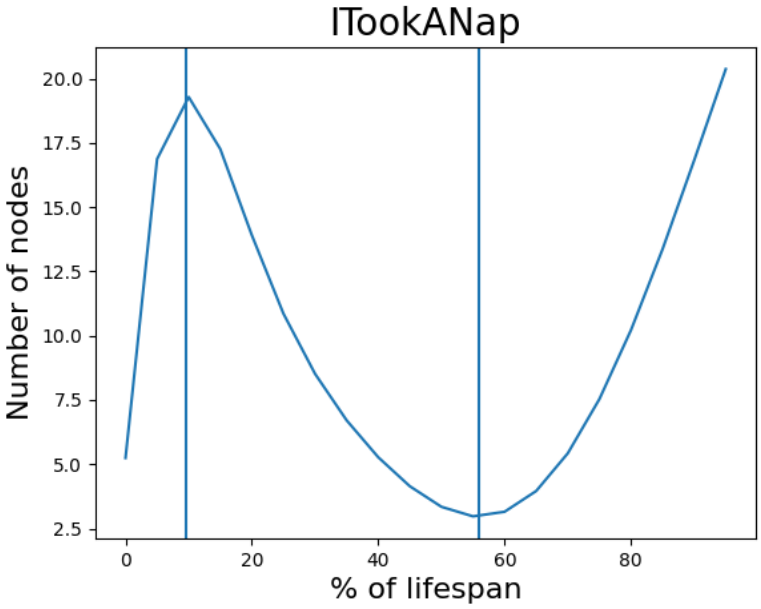


Figura 6.12: Challenge positiva: *ITookANap*, *Fonte: autoprodotta*

- *plankchallenge*, definita da 2 punti e 3 intervalli compresi tra [0, 15], [15, 70], [70, 100] (Figura 6.13).

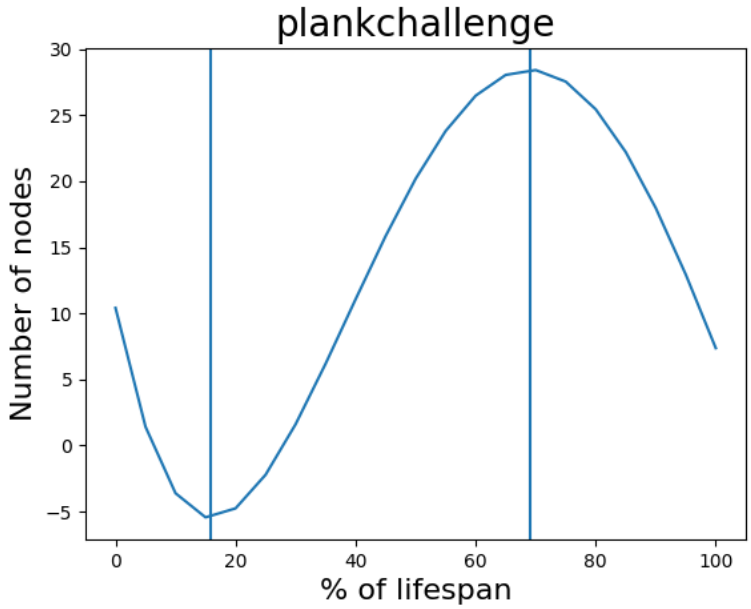


Figura 6.13: Challenge positiva: *plankchallenge*, *Fonte: autoprodotta*

Per quello che riguarda le challenge negative sono stati ottenuti i seguenti grafici:

- *bugsbunnychallenge*, definita da 3 punti e 4 intervalli compresi tra [0,10], [10,40], [40,70], [70,100] (Figura 6.14).

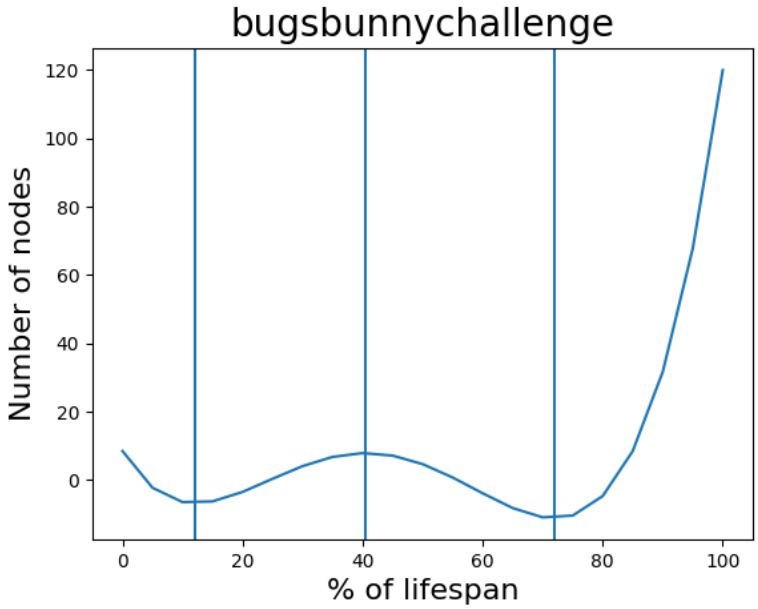


Figura 6.14: Challenge negativa: *bugsbunnychallenge*, *Fonte: autoprodotta*

- *fightchallenge*, definita da 3 punti e 4 intervalli compresi tra [0,10], [10,60], [60,75], [75,100] (Figura 6.15).

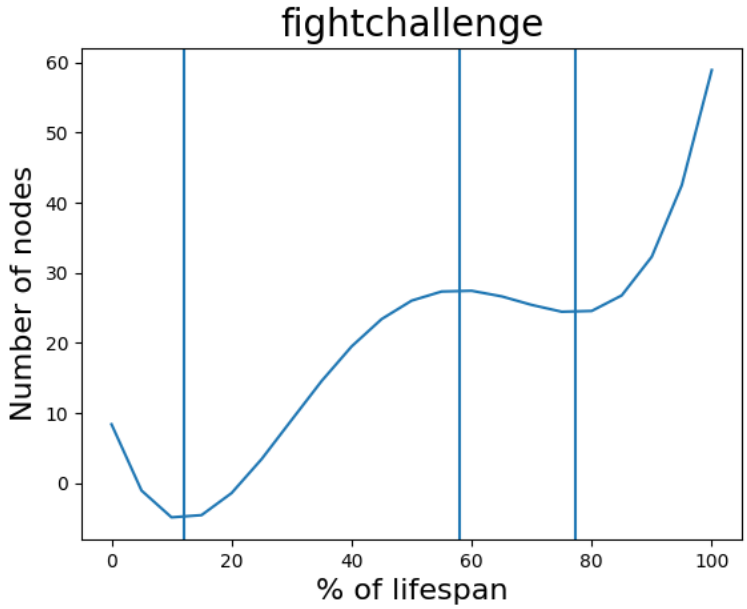


Figura 6.15: Challenge negativa: *fightchallenge*, *Fonte: autoprodotta*

- *firewroks*, definita da 2 punti e 3 intervalli compresi tra [0,15], [15,25], [25,100] (Figura 6.16).

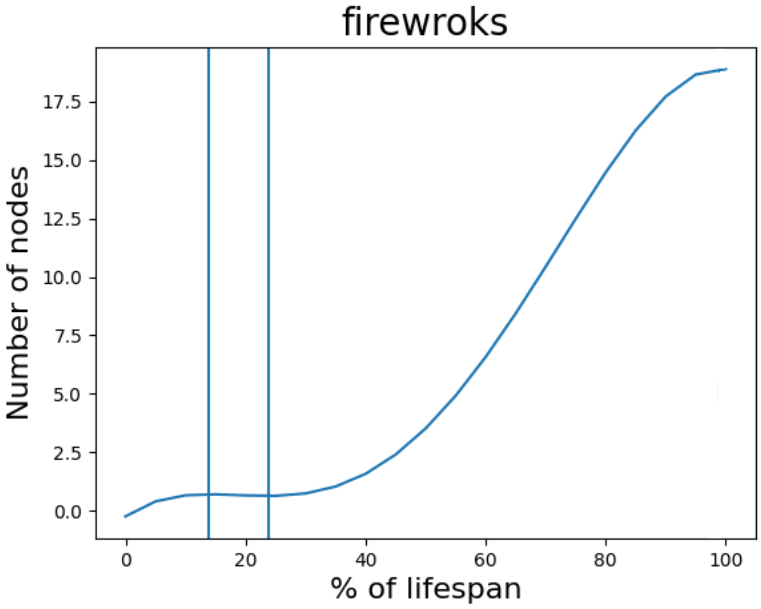


Figura 6.16: Challenge negativa: *firewroks*, *Fonte: autoprodotta*

- *silhouettechallenge*, definita da 2 punti e 3 intervalli compresi tra [0,20], [20,55], [55,100] (Figura 6.17).

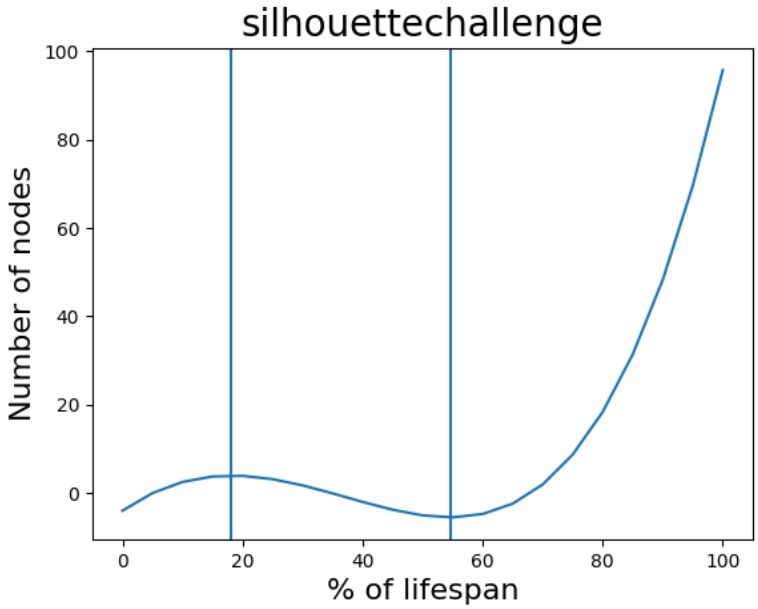


Figura 6.17: Challenge negativa: *silhouettechallenge*, *Fonte: autoprodotta*

- *strippatiktok*, definita da 2 punti e 3 intervalli compresi tra [0,5], [5,25], [25,100] (Figura 6.18).

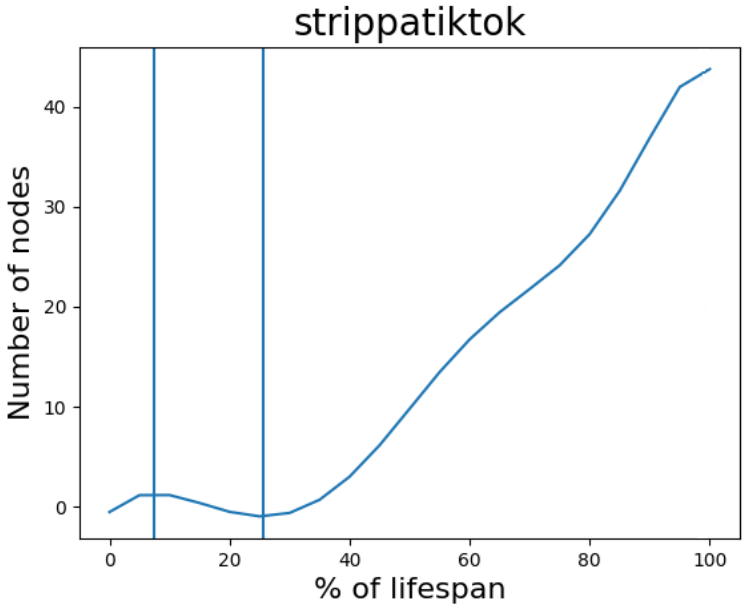


Figura 6.18: Challenge negativa: *strippatiktok*, *Fonte: autoprodotta*

- *sugarbaby*, definita da 3 punti e 4 intervalli compresi tra [0,10], [10,35], [35,60], [60,100] (Figura 6.19).

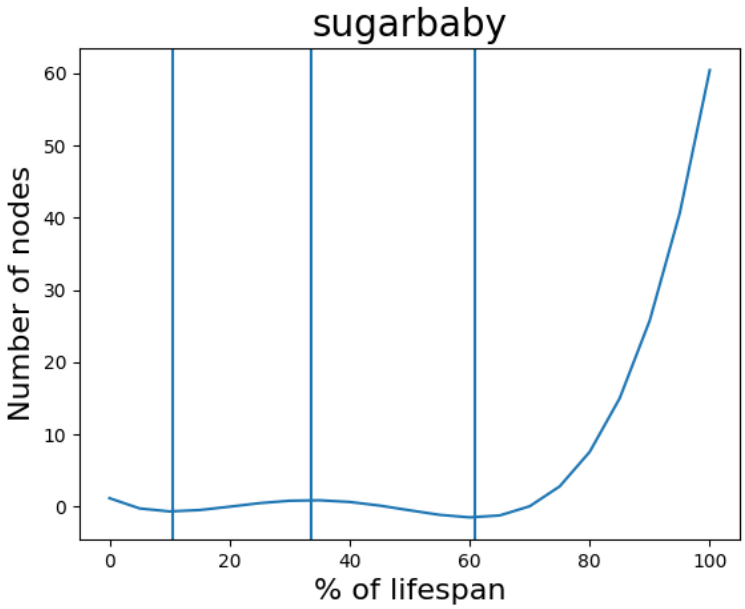


Figura 6.19: Challenge negativa: *sugarbaby*, *Fonte: autoprodotta*

- *updownchallenge*, definita da 2 punti e 3 intervalli compresi tra [0,10], [10,30], [30,100] (Figura 6.20).

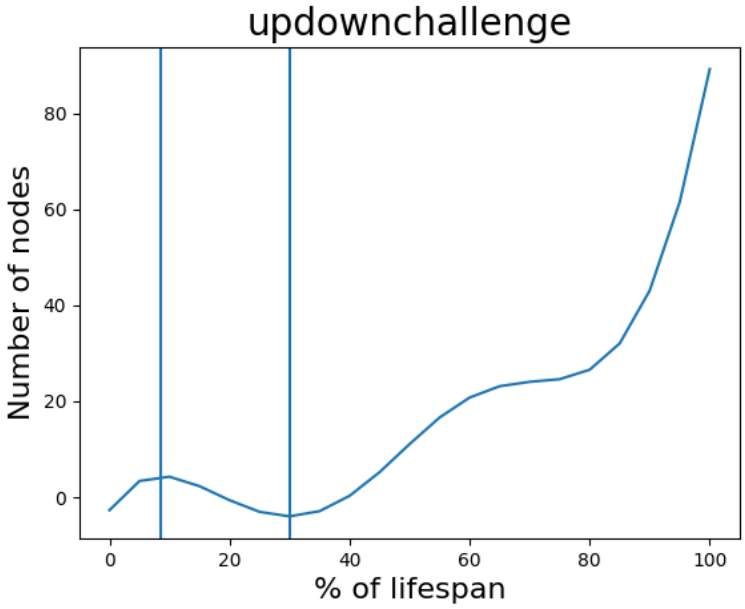


Figura 6.20: Challenge negativa: *updownchallenge*, *Fonte: autoprodotta*

Sono stati raccolti tutti i dati ottenuti all'interno di una tabella, specificando la challenge considerata, i punti che identificano gli intervalli e la pendenza della curva che si trova in ciascun intervallo. Quest'ultima è stata calcolata sottraendo il valore del primo punto al valore dell'ultimo punto della curva in quel determinato intervallo. Chiaramente, se si ottengono valori positivi, la curva sarà crescente; altrimenti, per valori negativi, la curva sarà decrescente (Tabella 6.21).

CHALLENGE	NUMERO INTERVALLI	PUNTI	INTERVALLI	PENDENZA
bussitchallenge	4	[15,60,90]	[0,15] [15,60] [60,90] [90,100]	95.99761169137744, -118.27642465066745, 31.442752095506865, -34.65612810666666
copinesdancechallenge	4	[20,65,90]	[0,20] [20,65] [65,90] [90,100]	44.42518137697745, -36.73127888077218, 6.595148986191536, -5.383536707316777
emojichallenge	4	[10,35,65]	[0,10] [10,35] [35,65] [65,100]	-9.638018338059004, 9.546001823519797, -13.173647721950804, 190.69084774863003
colpiditesta	4	[15,45,95]	[0,15] [15,45] [45,95] [95,100]	42.363435391181014, -43.14062187380682, 163.76026344047963, -4.857724049892198
boredinthehouse	4	[15,60,90]	[0,15] [15,60] [60,90] [90,100]	44.5623965568164, -64.56145728021626, 23.551015781769152, -20.30237854792317
ITookANap	3	[10,55]	[0,10] [10,55] [55,100]	14.065591288101796, -16.346575071141746, 20.79717459700647
plankchallenge	3	[15,70]	[0,15] [15,70] [70,100]	-15.892189426059272, 33.91760912813959, -21.067956875052293
silhouettechallenge	3	[20,55]	[0,20] [20,55] [55,100]	7.936578520088091, -9.456174703125733, 101.19774412205517
bugsbunnychallenge	4	[10,40,70]	[0,10] [10,40] [40,70] [70,100]	-15.282581453174771, 14.697732977239173, -19.100222219090277, 131.21810288023198
strippatiktok	3	[5,25]	[0,5] [5,25] [25,100]	1.8382030350663812, -2.2790505167584825, 44.86095887059749, -0.15737187013107246
firewroks	3	[15,25]	[0,15] [15,25] [25,100]	0.9443402064857265, -0.07128642549841413, 18.261101016359742, -0.019718508997719653
fightchallenge	4	[10,60,75]	[0,10] [10,60] [60,75] [75,100]	-13.57020642429758, 32.704269037893404, -3.2177874916522597, 34.62202715548912
sugarbaby	4	[10,35,60]	[0,10] [10,35] [35,60] [60,100]	-1.8355530693248934, 1.5590573088233972, -2.3854554155505863, 61.93731349489266
updownchallenge	3	[10,30]	[0,10] [10,30] [30,100]	7.093317678683478, -8.383086843081944, 93.15708510156927

Figura 6.21: Tabella riassuntiva degli intervalli, *Fonte: autoprodotta*

6.2.1 Feature degli intervalli

Per poter classificare gli intervalli è necessario scegliere delle *feature* ed analizzarle intervallo per intervallo. Dopo aver esaminato i dati a disposizione sono state scelte le seguenti *feature*:

- numero medio di nodi;

- numero medio di archi;
- densità media;
- numero totale di like;
- numero medio di like;
- numero medio di condivisioni;
- numero medio di commenti;
- numero medio di visualizzazioni;
- numero medio di like dati dall'autore;
- numero medio di following dell'autore;
- numero medio di like ricevuti dall'autore;
- numero medio di follower dell'autore;
- degree centrality;
- eigenvector centrality;
- numero di video;
- numero di utenti verificati;
- numero di componenti connesse;
- massimo numero di nodi delle componenti connesse;
- lifespan;
- indegree medio;
- pagerank;
- closeness centrality;
- betweenness centrality;
- average clustering;
- raggio della massima componente connessa;
- diametro della massima componente connessa;
- differenza del numero di video con l'intervallo precedente;
- numero di ore medio che intercorre tra la pubblicazione di due video nello stesso intervallo;
- percentuale iniziale del lifespan;
- percentuale finale del lifespan.

A tutti i valori medi è stata aggiunta la deviazione standard. In seguito, le feature sono state implementate, calcolate e salvate su un csv denominato "intervals.csv", grazie al file "lifespan_analysis" (vedi appendice).

Dopo aver calcolato i valori delle feature, questi sono stati inseriti all'interno di una matrice di correlazione, per valutare la relazione che intercorre tra di esse. La matrice di correlazione è una tabella quadrata (cioè con lo stesso numero di righe e colonne) che riporta, nelle intestazioni di riga e di colonna, l'elenco delle variabili su cui si vuole valutare la correlazione. Nelle singole celle, all'interno della tabella, sono, invece, indicati i singoli indici di correlazione bivariata.

Per analizzare più nel dettaglio questa matrice possiamo suddividerla in tre parti: una diagonale principale e due triangoli, uno in basso a sinistra ed uno in alto a destra. Le correlazioni sulla diagonale che va dall'angolo in alto a sinistra a quello in basso a destra (ovvero sulla diagonale principale della matrice di correlazione) sono tutte uguali ad 1. Questi 1 sulla diagonale principale, semplicemente, indicano che la correlazione di una variabile con se stessa per definizione è sempre massima.

All'interno di una matrice di correlazione, l'indice di correlazione tra ciascuna coppia di variabili appare sempre due volte. I valori presenti nel triangolo inferiore sinistro della matrice di correlazione sono, infatti, gli stessi riportati nelle celle presenti nel triangolo superiore destro della stessa matrice. Questo perché l'indice di correlazione è una misura statistica di tipo simmetrico che non tiene conto dell'ordine con cui le variabili sono inserite all'interno della formula.

In Figura 6.22 viene riportata la matrice di correlazione risultante dalle feature considerate per

ogni intervallo.

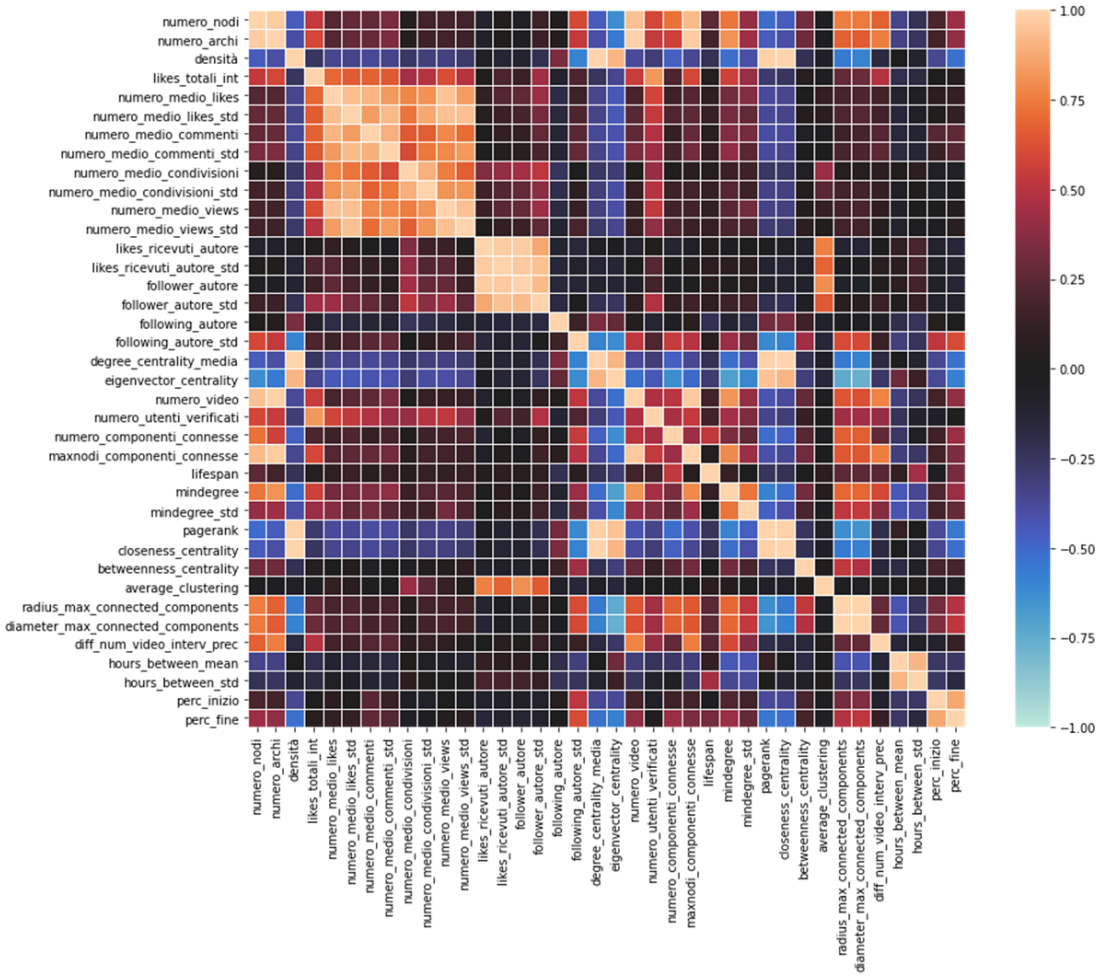


Figura 6.22: Matrice di correlazione tra le feature, *Fonte: autoprodotta*

I valori assunti dalle correlazioni vanno da -1 a 1 e assumono colori che vanno dall'arancione al celeste, come mostrato nella barra verticale a lato della matrice di correlazione. Sulla diagonale abbiamo tutti valori pari ad 1, come spiegato precedentemente. Per studiare tutti i valori ottenuti, essi sono stati riportati all'interno di un csv e sono state considerate solo le grandezze con una correlazione inferiore a -0,5%. Ciò è stato fatto per isolare le feature meno correlate tra loro, scegliendo quelle che descrivono meglio l'intervallo, in modo da poter avere un'idea completa di come possono variare i parametri all'interno degli intervalli. Si ottiene, così, una matrice ben poco correlata, formata da 8 feature (Figura 6.23); queste sono:

- numero medio di like;
- following dell'autore;
- numero di video;
- lifespan;
- betweenness centrality;
- average clustering coefficient;
- numero di ore medio che intercorre tra la pubblicazione di due video nello stesso intervallo;
- percentuale iniziale del lifespan.

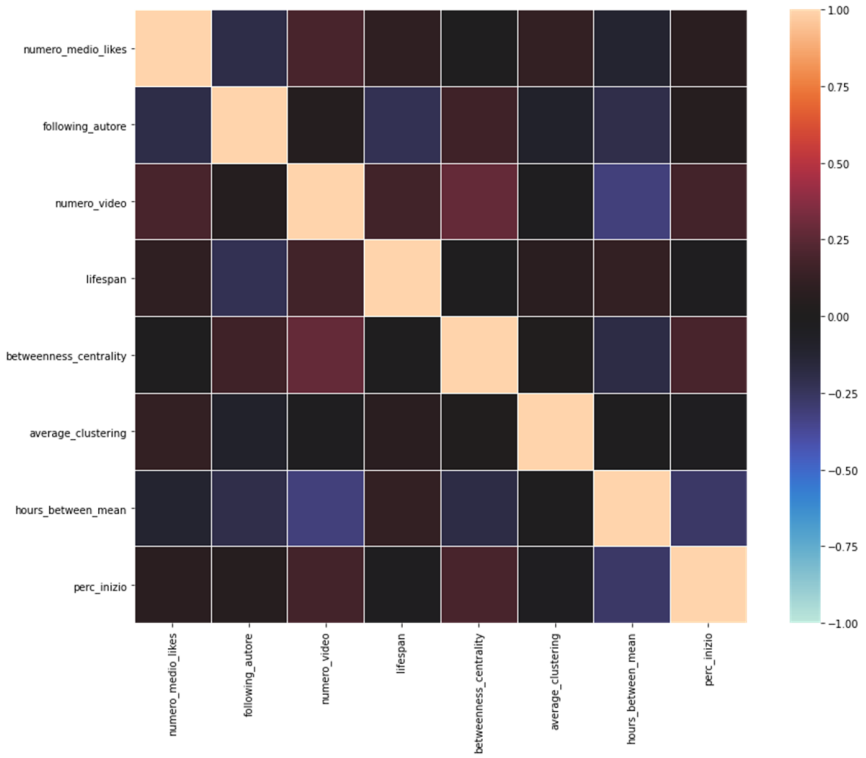


Figura 6.23: Matrice di correlazione tra feature poco correlate tra loro, *Fonte: autoprodotta*

Come si può notare dalla matrice di correlazione, si ha una colorazione prevalentemente scura, che indica una bassa correlazione tra le feature scelte, come si evince dalla legenda a lato della matrice.

Questi valori verranno, in seguito, utilizzati per caratterizzare gli intervalli e raggrupparli in cluster, in modo da poter effettuare un'analisi sugli stessi e identificare eventuali knowledge pattern.

7. Estrazione dei time pattern

7.1 Clusterizzazione

Il clustering è una tecnica che consente di raggruppare oggetti in modo non supervisionato, cioè senza la possibilità di sfruttare esempi da utilizzare come base di apprendimento. Ogni raggruppamento, o cluster, rappresenta una classe di appartenenza degli elementi. Non avendo a disposizione esempi da cui imparare, il clustering sfrutta delle similarità tra i dati che deve analizzare e che possono essere di varia natura ma che, in sostanza, definiscono una distanza tra i punti del dataset (solitamente una distanza euclidea).

K-means è uno degli algoritmi di clustering più diffuso e più performante. Nonostante ciò è un algoritmo semplicissimo da implementare ed utilizzare. Esso si basa sulle medie, ovvero punti appartenenti allo spazio delle feature, che mediano le distanze tra tutti i dati appartenenti al cluster ad esso associato. Tali medie rappresentano, quindi, una sorta di baricentri dei cluster ed, in generale, proprio per le loro caratteristiche, non sono punti del dataset.

K-means è molto semplice e si suddivide in diverse fasi. Non conoscendo le classi presenti nel dataset di ingresso, la prima cosa da fare è decidere il numero di classi (o meglio cluster, in questo caso) in cui si vuole suddividere il dataset stesso. Questo numero è detto K, da cui il nome del metodo K-means (il termine means sottintende l'uso delle medie). Si scelgono in modo casuale K punti appartenenti allo spazio delle feature. L'unica condizione è che non siano coincidenti, anzi, solitamente ci si assicura che siano abbastanza distanti tra loro, perché, in caso contrario, l'algoritmo potrebbe avere problemi a convergere. Si calcola, successivamente, la distanza di ogni punto del dataset rispetto ad ogni media; il punto viene associato al cluster collegato alla media più vicina. Si ricalcola la posizione di ogni media facendo la media delle posizioni di tutti i punti del cluster associato e si itera nuovamente fino a quando non ci sarà più alcun ingresso che cambia cluster.

In generale, potrebbe non essere così immediato individuare il numero di cluster da utilizzare. È però possibile utilizzare un metodo maggiormente oggettivo per decidere questo numero: il cosiddetto *elbow method*, o metodo del gomito. In pratica si itera K-means per diversi valori di K ed ogni volta si calcola la somma delle distanze al quadrato tra ogni media ed i punti del corrispettivo cluster. Si graficano, poi, i valori di K (asse orizzontale) e i valori della somma delle distanze al

quadrato (asse verticale). L'obiettivo è quello di individuare, all'interno della curva ottenuta, un gomito, in modo tale da poter identificare il numero ottimale di k.

Per quanto riguarda il nostro contesto di riferimento, per poter applicare questo metodo di identificazione numerica ottimale di cluster, è stata generata la curva, normalizzando i valori delle feature tra 0 ed 1, come riportato in Figura 7.1.

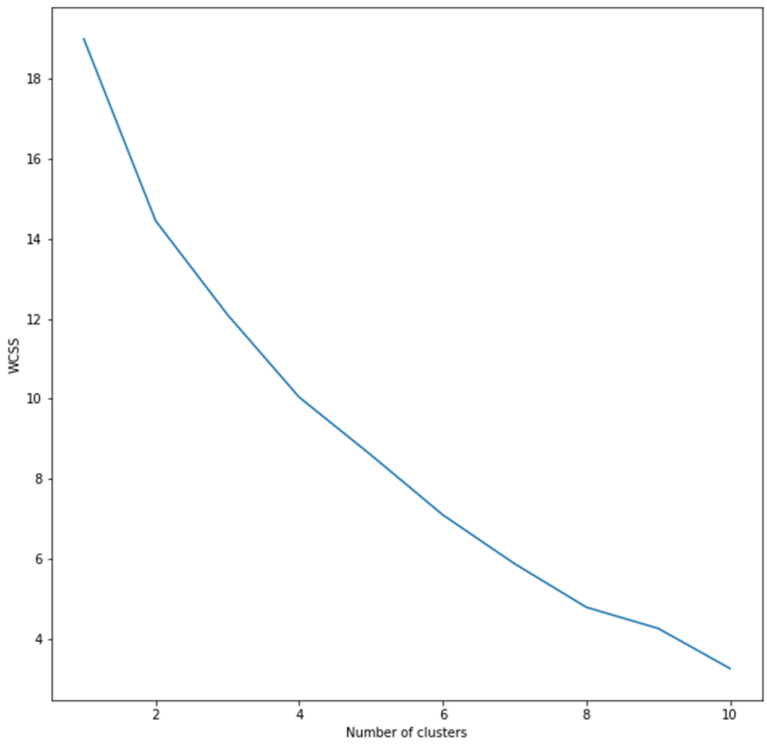


Figura 7.1: Elbow method (o metodo del gomito), *Fonte: autoprodotta*

In questo caso la curva ottenuta è piuttosto regolare e non ci sono punti in cui si presenta un "gomito". Per questo, dopo aver effettuato diversi esperimenti, si è scelto di considerare un K pari a 5, poiché sembra descrivere bene la situazione reale.

È stata poi applicata la PCA, ovvero l'analisi delle componenti principali, metodo che rientra nei problemi di trasformazione lineare e che viene ampiamente utilizzata in diversi campi, soprattutto per l'estrazione delle caratteristiche e la riduzione della dimensionalità. La PCA permette di trovare le direzioni della massima varianza nei dati ad alta dimensione e di proiettarle su un nuovo sottospazio con dimensioni uguali o inferiori a quello originale. Utilizzando la proiezione matematica, il set di dati originale, che potrebbe aver coinvolto molte variabili, viene ad essere interpretato solo da poche variabili (dette componenti principali). L'output della PCA sono proprio queste componenti principali, il cui numero è inferiore o uguale al numero di variabili originali. In definitiva, l'analisi delle componenti principali ci aiuta a identificare modelli nei dati in base alla correlazione tra funzionalità. Infatti, spesso accade che uno studio del set di dati di dimensioni ridotte consenta all'utente di individuare tendenze, modelli e valori anomali nei dati molto più facilmente di quanto sarebbe stato possibile senza eseguire l'analisi delle componenti principali.

Le componenti principali possiedono alcune proprietà utili che sono elencate di seguito:

- sono essenzialmente le combinazioni lineari delle variabili originali; il vettore dei pesi in questa combinazione è, in realtà, l'autovettore trovato che, a sua volta, soddisfa il principio dei minimi quadrati;

- sono ortogonali tra di loro;
- la variazione presente tra di esse diminuisce man mano che ci spostiamo dalla prima componente all'ultima.

L'obiettivo è, dunque, quello di ridurre la dimensionalità del nostro spazio delle funzioni, ovvero proiettare lo spazio delle funzioni su un sottospazio più piccolo, tramite PCA, in cui gli autovettori formeranno gli assi di questo nuovo sottospazio di caratteristiche. Tuttavia, gli autovettori definiscono solo le direzioni del nuovo asse, poiché hanno tutte la stessa unità di lunghezza 1. Quindi, al fine di decidere quale autovettore vogliamo eliminare per il nostro sottospazio di dimensione inferiore, dobbiamo considerare i corrispondenti autovalori. Gli autovalori (in ordine decrescente di valore) rappresentano l'ammontare della variabilità totale osservata sulle variabili originarie, espressa da ciascuna componente principale; gli autovettori, invece, rappresentano le corrispondenti direzioni (ortogonali) di massima variabilità estratte dalle componenti principali. Riassumendo, gli autovettori con gli autovalori più bassi recano il minor numero di informazioni sulla distribuzione dei dati; sono proprio questi che vogliamo eliminare. L'approccio comune è quello di classificare gli autovalori, dal più alto al più basso, e scegliere i primi K autovettori corrispondenti. Questa tecnica, oltre a semplificare il lavoro di manipolazione delle caratteristiche, aiuta a migliorare i risultati del classificatore (riducendo il rumore), motivo per cui non è raro utilizzarla insieme ad altri algoritmi di machine learning.

Nel caso specifico che stiamo trattando, l'applicazione della PCA ha ridotto il dataset a due componenti, come riportato in Figura 7.2.

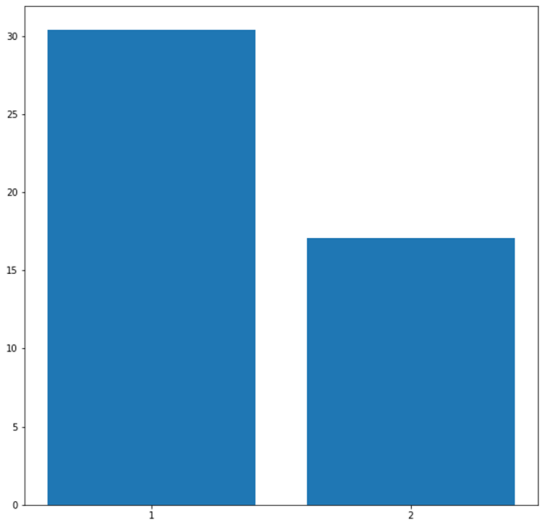


Figura 7.2: Riduzione delle componenti con PCA, *Fonte: autoprodotta*

La figura mostra che le varianze delle due componenti principali estratte descrivono bene il dataset; infatti l'auspicio nell'applicazione della PCA è che le varianze campionarie delle prime Componenti Principali (PC) siano grandi, mentre le varianze delle altre componenti siano abbastanza piccole da poter considerare trascurabili le corrispondenti PC. Omettere PC a bassa variabilità campionaria, e porre tutta l'attenzione sulle PC con varianza più elevata, può essere visto come un modo semplice e "sensato" di ridurre la dimensionalità (numero di colonne) del dataset.

In seguito all'applicazione della PCA sono stati testati sia K-Means, sia EM (Expectation - Maximization, chiamato anche Gaussian-Mixture). EM rappresenta un'evoluzione di K-Means ed è un metodo iterativo per trovare la massima verosimiglianza (locale), o la stima massima a posteriori (MAP) dei parametri nei modelli statistici, in cui il modello dipende da variabili latenti

non osservate. L'iterazione EM alterna l'esecuzione di un passaggio di *Expectation* (E), che crea una funzione per l'aspettativa della verosimiglianza logaritmica, valutata utilizzando la stima corrente per i parametri, e un passaggio di *Maximization* (M), che calcola i parametri massimizzando la probabilità trovata allo step di *expectation* E. Queste stime dei parametri vengono, quindi, utilizzate per determinare la distribuzione delle variabili latenti nella fase E successiva.

Tra il risultato ottenuto con K-Means ed EM con Gaussian Mixture è stato scelto EM, perché restituisce valori più coerenti e meno casuali. È possibile che avvenga questo fenomeno, poiché EM può essere considerato un'evoluzione di K-Means. Il clustering ottenuto è riportato nella Figura 7.3.

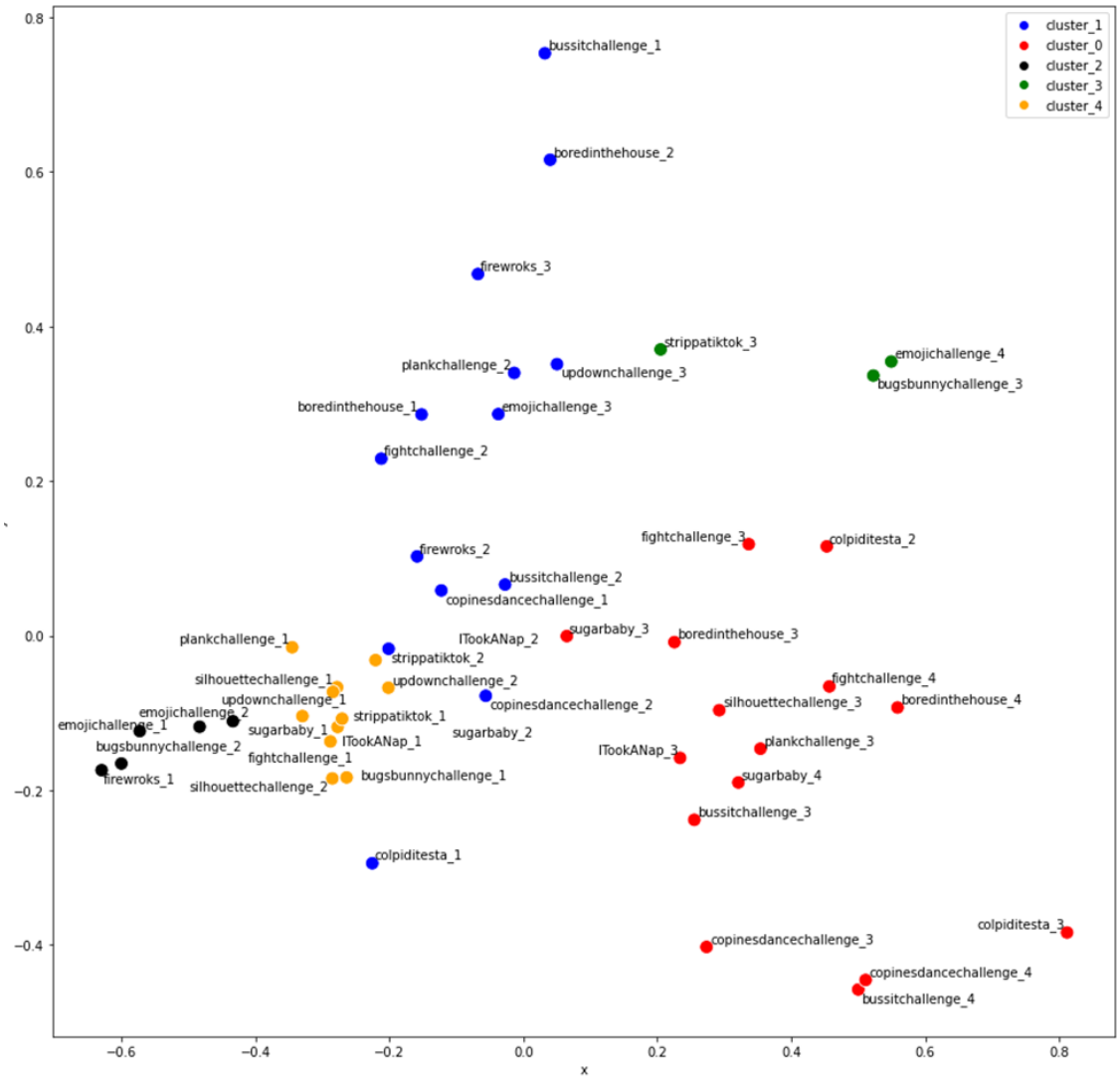


Figura 7.3: Clustering, *Fonte: autoprodotta*

È possibile notare i 5 diversi cluster, colorati in modo differente, ciascuno dei quali fa riferimento ad un cluster preciso, numerato e riportato nella legenda posizionata in alto a destra.

7.1.1 Caratterizzazione dei cluster

È stata effettuata un'analisi qualitativa relativamente alle feature degli intervalli che vengono considerati all'interno di un unico cluster. Questa analisi ha l'obiettivo di evidenziare eventuali

similarità o differenze tra i diversi cluster.

In particolare, gli intervalli presenti all'interno della clusterizzazione identificata con il numero zero, di colore rosso, rappresentano gli intervalli finali del lifespan, in cui la challenge tende ad avere meno interazioni rispetto ad altri intervalli. Essi sono caratterizzati da meno like in media, oltre ad avere autori di video meno influenti rispetto ad altri cluster (infatti, è presente un minor numero di utenti verificati, caratterizzati da molti following e da pochi follower). Inoltre passano più ore tra la pubblicazione di un video e l'altro, sinonimo di una dilatazione temporale che intercorre tra le pubblicazioni dei video. La rete derivante da questi intervalli è più connessa rispetto agli altri ed ha valori di centralità più elevati, a riprova del fatto che ci troviamo in una fase ormai consolidata della challenge.

Per quello che riguarda il cluster uno, di colorazione blu, è possibile dire che questi intervalli corrispondono ad una fase di *boom*, una crescita che successivamente tende a diminuire. Essi sono caratterizzati in media da molti like e da molti video. Vi sono utenti importanti, molti dei quali sono utenti verificati, con molti più follower e un minor numero di following. Inoltre, passano poche ore tra un video e l'altro, per cui si può dedurre che, durante questa fase, ci siano diversi utenti che pubblicano dei video. La rete derivante dall'intervallo è meno connessa rispetto a quella di altri intervalli, e i valori di centralità sono più bassi. Questo potrebbe significare una fase più caotica.

Il cluster numero due, di colorazione nera, identifica intervalli iniziali del lifespan, in cui il numero medio di like è più basso rispetto agli altri intervalli, poiché si sta trattando un cluster che contiene le prime fasi di sviluppo di una challenge. Nonostante il confronto con altri cluster evidenzi un numero medio di like più basso, si può notare che esso resta comunque piuttosto elevato, sinonimo del fatto che la challenge è appena stata creata, sta destando curiosità e, probabilmente, esploderà in un momento successivo. Gli appartenenti a questo cluster sono caratterizzati da un basso numero medio di video, pubblicati principalmente da utenti non verificati, ma che possiedono un ingente numero di follower. Per questo si ha un vasto numero di visualizzazioni e di condivisioni. Passa diverso tempo tra la pubblicazione di un video e l'altro. La rete non è connessa, sinonimo del fatto che, in questa fase, le pubblicazioni dei video vengono effettuate da persone non connesse tra loro.

Il cluster numero tre, di colorazione verde, contiene pochi intervalli, i quali iniziano nella seconda metà del lifespan e sono caratterizzati da un elevato numero medio di like. Gli utenti che pubblicano video in questa fase sono per lo più utenti verificati, con un enorme numero di follower e di interazioni. Il numero medio di video pubblicati raggiunge il suo apice, considerando che intercorrono poche ore tra la pubblicazione di un video ed un altro. La rete è maggiormente connessa rispetto ad altri intervalli.

L'ultimo cluster, il numero quattro, assume una colorazione gialla. La partenza media di questi intervalli si trova all'inizio del lifespan, gli appartenenti a tale cluster sono caratterizzati dal più basso numero medio di like tra tutti i cluster considerati. Il numero medio di video pubblicati è piuttosto basso, sebbene il più basso in assoluto sia da attribuire al cluster 2. Nonostante gli appartenenti a tale cluster abbiano un numero maggiore di video pubblicati rispetto al cluster 2, si nota, come in questo caso, ci sia un elevato numero di visualizzazioni, ma una minore interazione da parte degli utenti, in particolare facendo riferimento ai commenti e alle condivisioni. Questo potrebbe essere dovuto al fatto che chi pubblica i video in questa fase ha meno follower rispetto agli autori del cluster 2, quindi meno persone fidelizzate che ricondividono il contenuto. Passano, inoltre, diverse ore tra la pubblicazione dei video e la rete è sconnessa.

7.1.2 Considerazioni

Confrontando i vari cluster tra loro è possibile ipotizzare che i cluster due e quattro siano dei raggruppamenti di intervalli iniziali, con la differenza che nel cluster quattro c'è un maggior numero di video pubblicati rispetto al cluster due. Questi video, però, hanno autori meno influenti

rispetto a quelli presenti all'interno degli intervalli che compongono il cluster due. Riassumendo, il cluster due ha un minor numero di video, i quali sono pubblicati da utenti più influenti e, conseguentemente, apportano un maggior numero di interazioni, mentre il cluster quattro ha un maggior numero di video, pubblicati, però, da utenti con meno follower, quindi aventi meno interazioni.

La fase di esplosione di una challenge è espressa dai cluster uno e tre. In particolare, il cluster tre potrebbe rappresentare la parte di crescita che segue il *boom* del cluster uno.

Infine, il cluster zero riunisce gli intervalli finali di una challenge, i quali hanno superato la fase di crescita e si stabilizzano (Tabella 7.1).

momenti di vita della challenge	**cluster**
fase iniziale	cluster 2; cluster 4
fase di esplosione	cluster 1; cluster 3
fase finale	cluster 0

Tabella 7.1: Cluster per momenti di vita delle challenge

7.2 Sequenze degli intervalli all'interno di ogni challenge

In questa fase è necessario spostare l'attenzione sull'analisi di ogni singola curva per analizzare le sequenze di cluster di ogni challenge. Il procedimento è stato applicato sia per le challenge positive, sia per quelle negative. L'obiettivo è quello di mettere in luce eventuali sequenze ripetute, volte a formare dittonghi che possano essere considerati veri e propri design pattern.

7.2.1 Challenge positive

Le sette challenge positive sono identificate dalle seguenti sequenze:

- *boredinthehouse:* composta dalla sequenza 1-1-0-0, come si può notare in Figura 7.4.

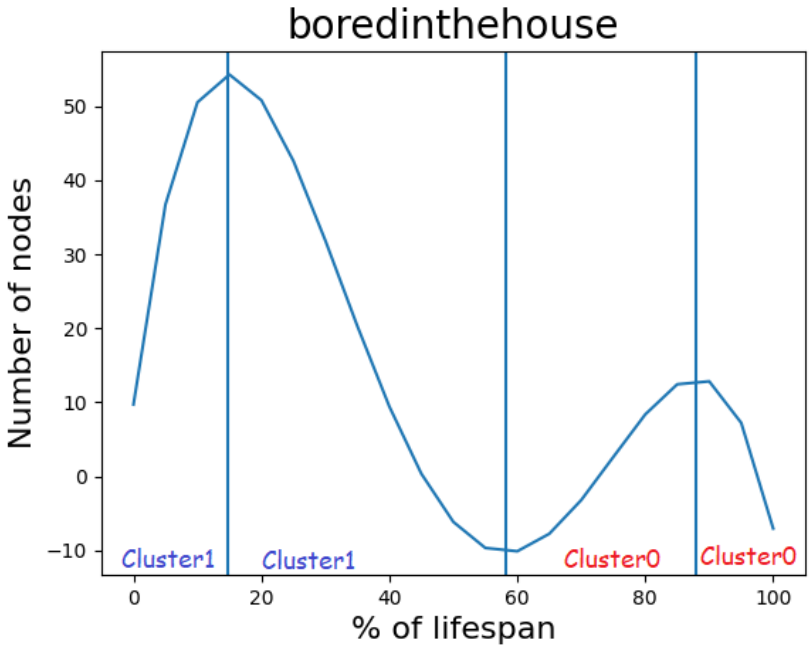


Figura 7.4: Sequenza di cluster per *boredinthehouse*, *Fonte: autoprodotta*

- *bussitchallenge:* composta dalla sequenza 1-1-0-0, come si può notare in Figura 7.5.

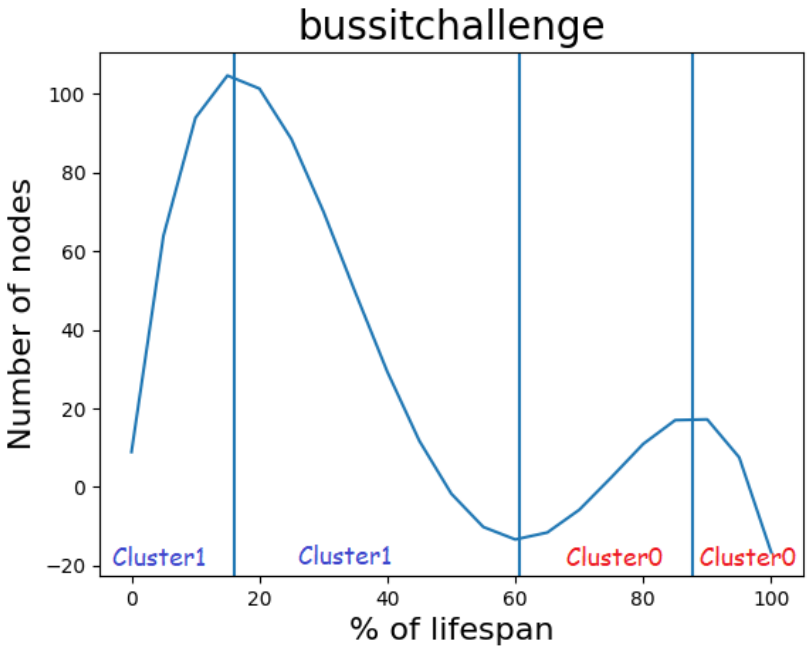


Figura 7.5: Sequenza di cluster per *bussitchallenge*, *Fonte: autoprodotta*

- *colpiditesta:* composta dalla sequenza 1-1-0-0, come si può notare in Figura 7.6.

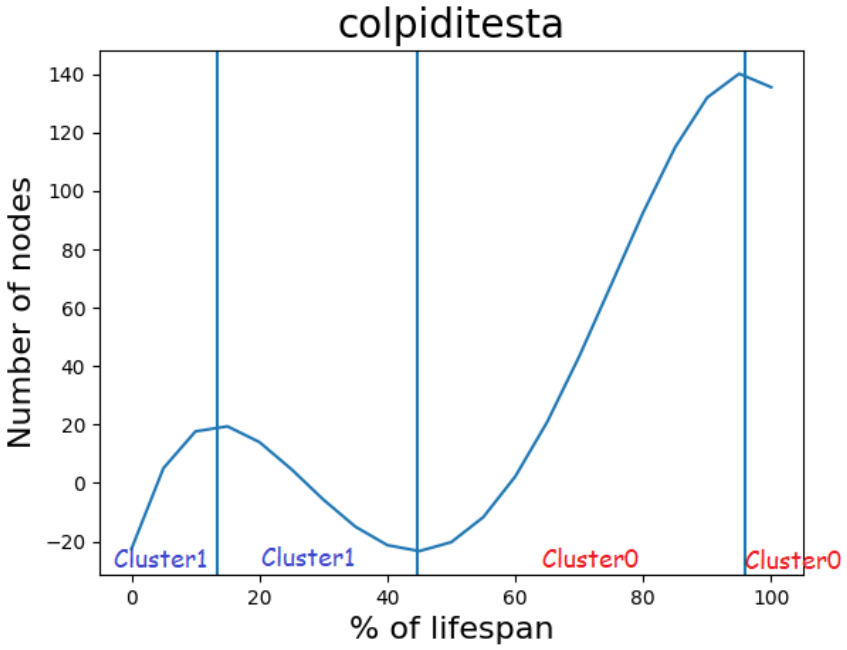


Figura 7.6: Sequenza di cluster per *colpiditesta*, *Fonte: autoprodotta*

- *copinesdancechallenge:* composta dalla sequenza 1-1-0-0, come si può notare in Figura 7.7.

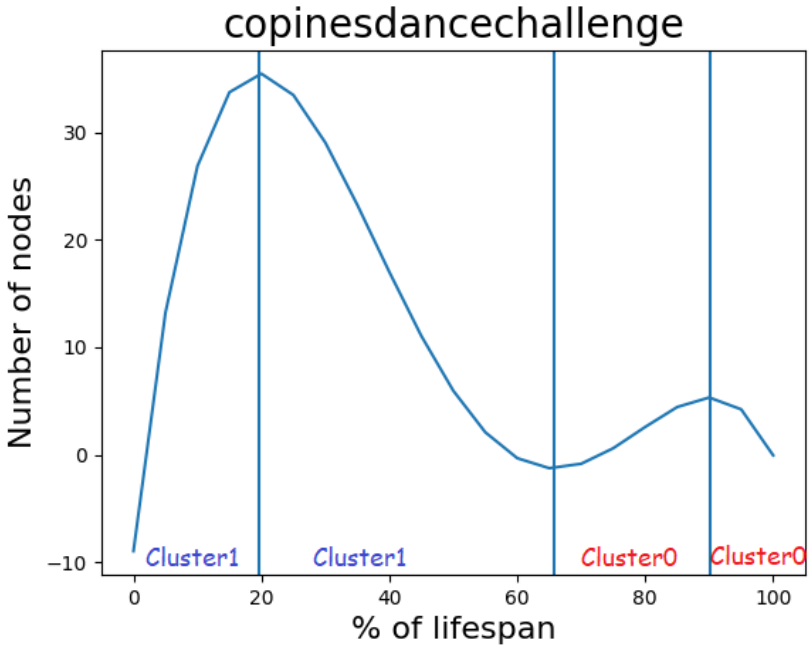


Figura 7.7: Sequenza di cluster per *copinesdancechallenge*, *Fonte: autoprodotta*

- *emojichallenge:* composta dalla sequenza 2-2-1-3, come si può notare in Figura 7.8.

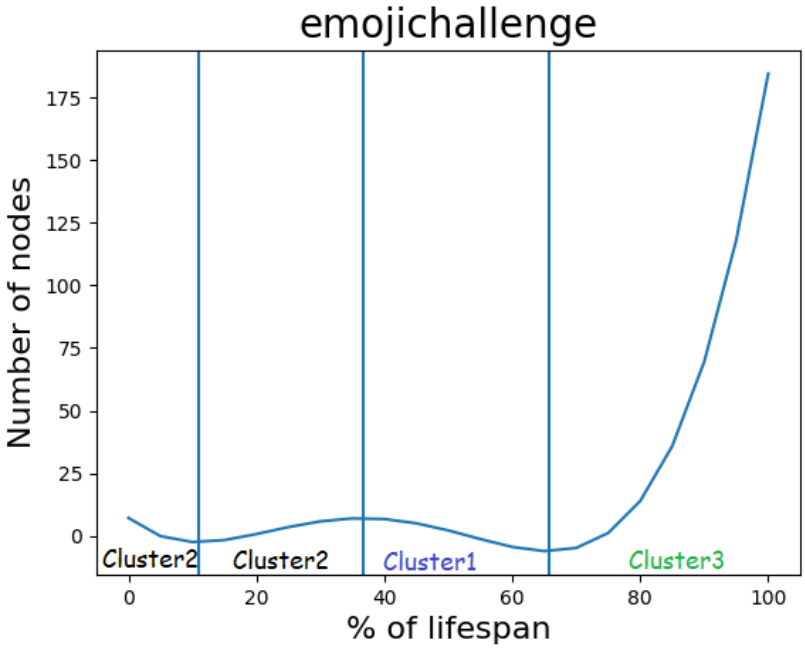


Figura 7.8: Sequenza di cluster per *emojichallenge*, *Fonte: autoprodotta*

- *ITookANap:* composta dalla sequenza 4-1-0, come si può notare in Figura 7.9.

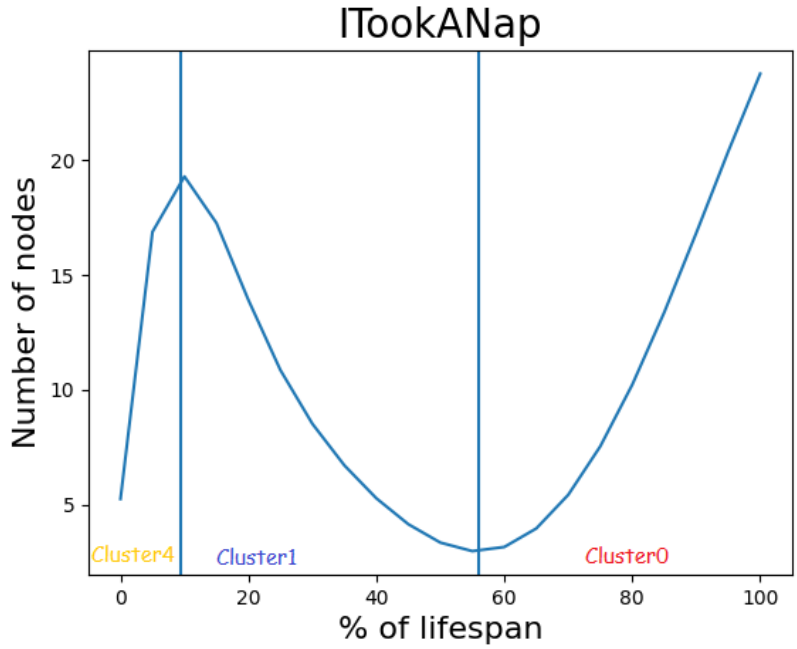


Figura 7.9: Sequenza di cluster per *ITookANap*, *Fonte: autoprodotta*

- *plankchallenge:* composta dalla sequenza 4-1-0, come si può notare in Figura 7.10.

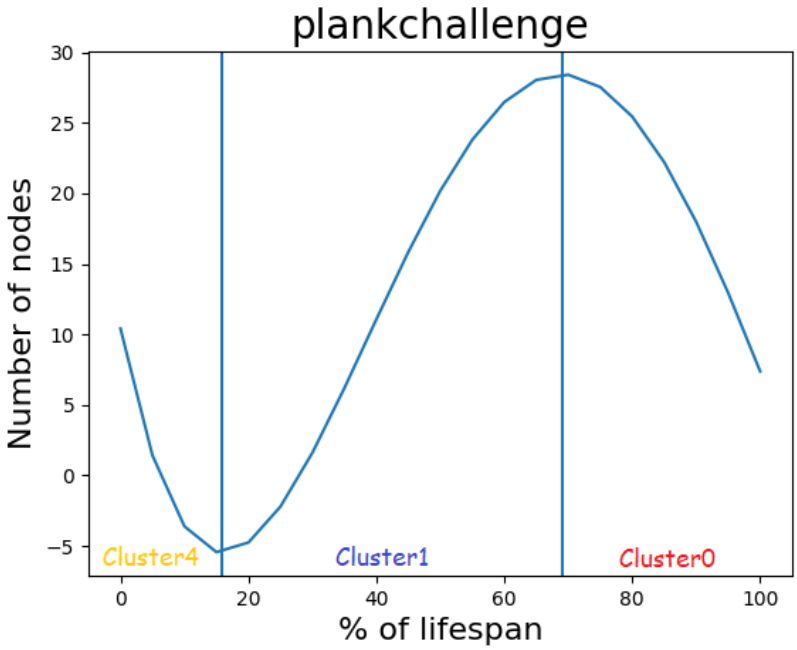


Figura 7.10: Sequenza di cluster per *plankchallenge*, *Fonte: autoprodotta*

7.2.2 Challenge negative

Le sette challenge negative sono identificate dalle seguenti sequenze:

- *bugsbunnychallenge:* composta dalla sequenza 4-2-2-3, come si può notare in Figura 7.11.

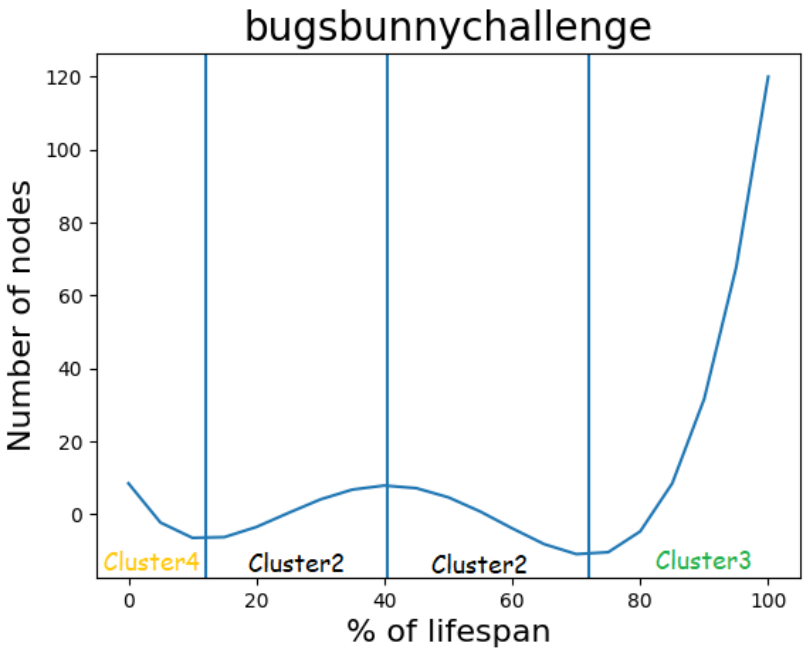


Figura 7.11: Sequenza di cluster per *bugsbunnychallenge*, *Fonte: autoprodotta*

- *fightchallenge:* composta dalla sequenza 2-1-0-0, come si può notare in Figura 7.12.

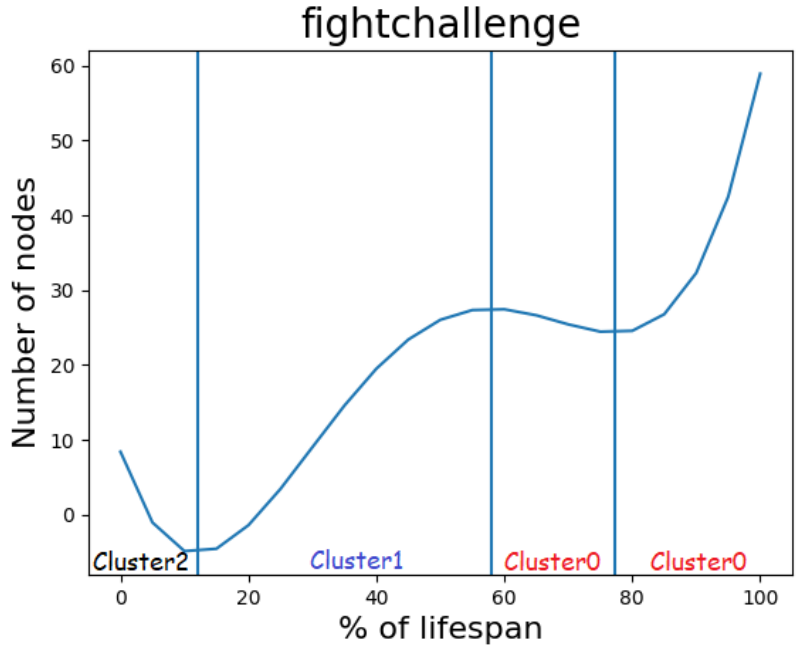


Figura 7.12: Sequenza di cluster per *fightchallenge*, *Fonte: autoprodotta*

- *firewroks:* composta dalla sequenza 2-1-1, come si può notare in Figura 7.13.

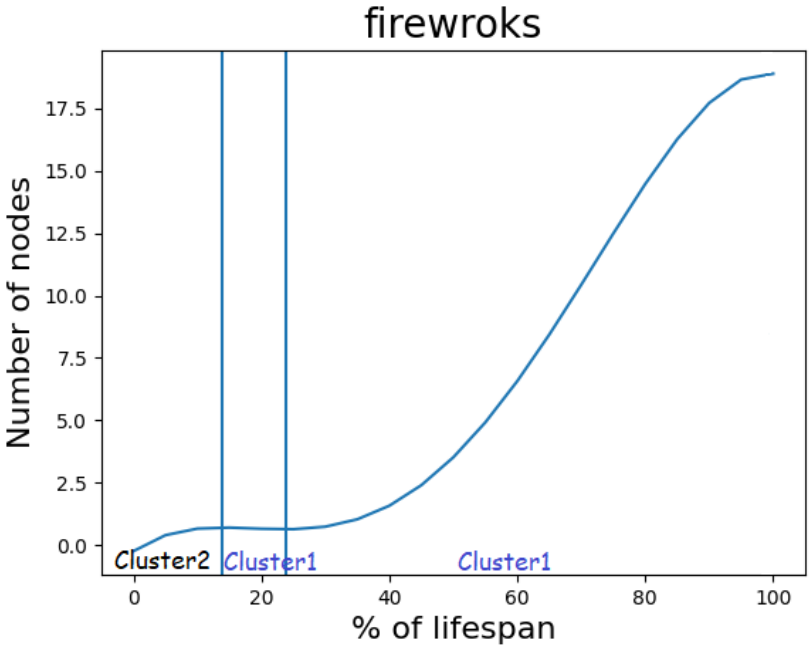


Figura 7.13: Sequenza di cluster per *firewroks*, *Fonte: autoprodotta*

- *silhouettechallenge:* composta dalla sequenza 0-0-1, come si può notare in Figura 7.14.

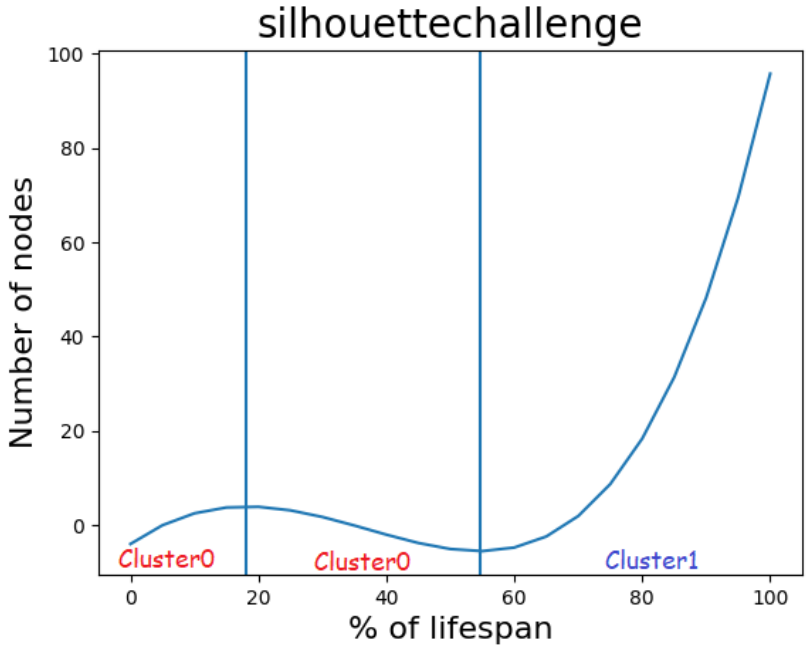


Figura 7.14: Sequenza di cluster per *silhouettechallenge*, *Fonte: autoprodotta*

- *strippatiktok:* composta dalla sequenza 4-4-3, come si può notare in Figura 7.15.

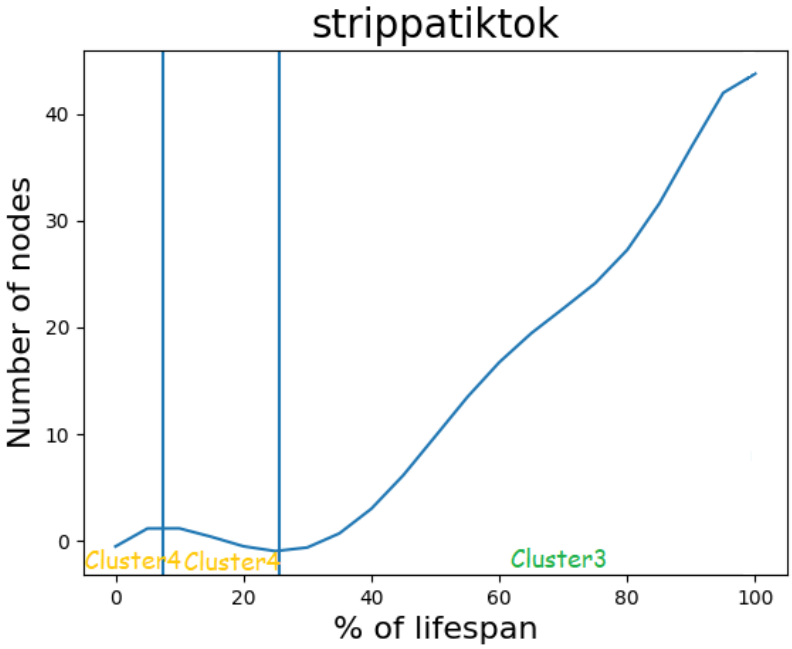


Figura 7.15: Sequenza di cluster per *strippatiktok*, *Fonte: autoprodotta*

- *sugarbaby:* composta dalla sequenza 4-4-0-0, come si può notare in Figura 7.16.

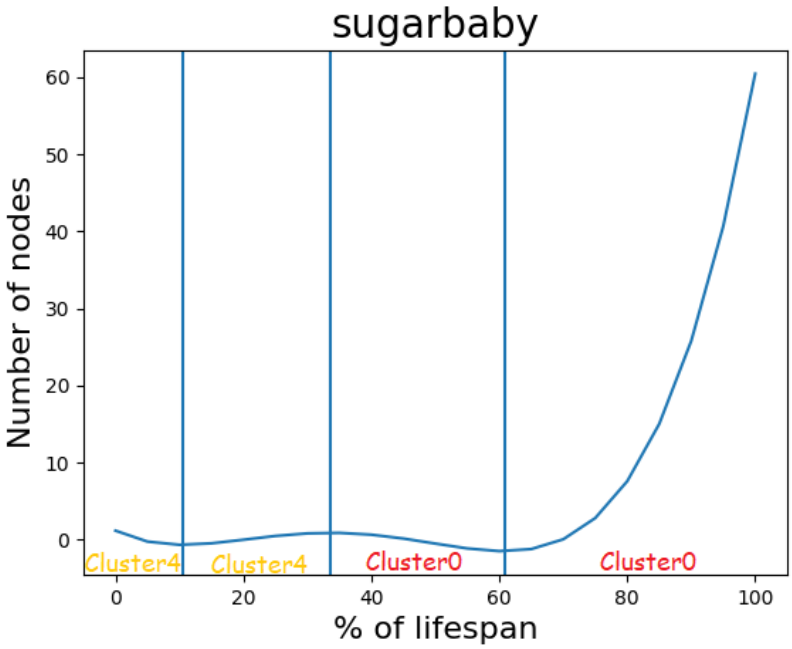


Figura 7.16: Sequenza di cluster per *sugarbaby*, *Fonte: autoprodotta*

- *updownchallenge:* composta dalla sequenza 4-4-1, come si può notare in Figura 7.17.

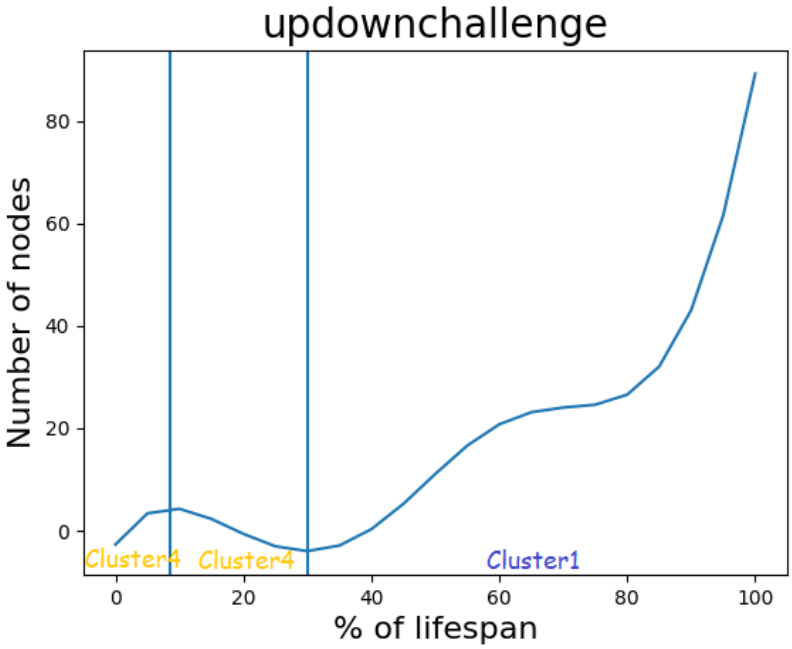


Figura 7.17: Sequenza di cluster per *updownchallenge*, *Fonte: autoprodotta*

Per ogni challenge sono stati identificati i cluster di appartenenza di ciascun intervallo. Si può notare, ad un primo sguardo, che le challenge positive hanno delle similarità tra di loro, mentre le negative sembrano essere più caotiche. Infatti, per quello che concerne le positive, ci sono diverse sequenze ricorrenti che si presentano in molte challenge, mentre, per quanto riguarda quelle negative, non è immediato individuare delle ricorrenze.

7.2.3 Analisi delle sequenze

Dopo l'analisi svolta finora, grazie alla quale sono state individuate delle sequenze, si è cercato di studiare la presenza di eventuali design pattern ricorrenti. Dai grafici si notano delle ricorrenze, ma è necessario studiarle in maniera più approfondita, con l'obiettivo di astrarre eventuali sequenze di cluster e portare l'intera analisi verso un'ottica a granularità maggiore. Le sequenze ottenute sono riportate nelle Tabelle 7.2 e 7.3.

challenge positive	**sequenza**
boredinthehouse	1-1-0-0
bussitchallenge	1-1-0-0
colpiditesta	1-1-0-0
copinesdancechallenge	1-1-0-0
emojichallenge	2-2-1-3
Itookanap	4-1-0
plankchallenge	4-1-0

Tabella 7.2: Sequenze delle challenge positive

challenge negative	sequenza
bugsbunnychallenge	4-2-2-3
fightchallenge	2-1-0-0
firewroks	2-1-1
silhouettechallenge	4-4-0
strippatiktok	4-4-3
sugarbaby	4-4-0-0
updownchallenge	4-4-1

Tabella 7.3: Sequenze delle challenge negative

7.3 Astrazione ed identificazione di time pattern

Dopo aver identificato le sequenze di intervalli per ogni challenge, è stata operata un'astrazione, in modo da identificare eventuali time pattern. Le astrazioni sono state effettuate considerando lettere che rappresentino dittonghi di intervalli che si ripetono con una certa frequenza. Sono state definite delle ipotesi di partenza, che verranno verificate in un secondo momento mediante un'analisi statistica. Come primo passo, sono stati effettuati i seguenti raggruppamenti per le challenge positive:

- A = 1-1 = 1
- B = 0-0 = 0
- C = 2-2 = 2
- D = 3-3 = 3
- E = 4-4 = 4

Questa astrazione permette di raggruppare intervalli che appartengono allo stesso cluster e che, quindi, rappresentano la stessa tipologia di interazione. Questa generalizzazione porta alle sequenze riportate nella Tabella 7.4.

challenge positive	sequenza
boredinthehouse	A-B
bussitchallenge	A-B
colpiditesta	A-B
copinesdancechallenge	A-B
emojichallenge	C-A-D
Itookanap	E-A-B
plankchallenge	E-A-B

Tabella 7.4: Astrazione delle sequenze delle challenge positive

A questo punto è necessario formulare due ipotesi, per cui cluster simili tra loro possono essere considerati come un'unica tipologia di intervallo e, quindi, come un unico macro-settore.

Prima ipotesi: Se D == B, ovvero se 3 == 0, si otterrebbero le sequenze riportate nella Tabella 7.5.

challenge positive	sequenza
boredinthehouse	A-D
bussitchallenge	A-D
colpiditesta	A-D
copinesdancechallenge	A-D
emojichallenge	C-A-D
Itookanap	E-A-D
plankchallenge	E-A-D

Tabella 7.5: Astrazione delle sequenze delle challenge positive

Seconda Ipotesi: se C == E, ovvero se 2 == 4, si otterrebbero le sequenze riportate nella Tabella 7.6.

challenge positive	sequenza
boredinthehouse	A-D
bussitchallenge	A-D
colpiditesta	A-D
copinesdancechallenge	A-D
emojichallenge	C-A-D
Itookanap	C-A-D
plankchallenge	C-A-D

Tabella 7.6: Astrazione delle sequenze delle challenge positive

Dimostrando, quindi, le due ipotesi per cui D == B, ovvero 3 == 0, e per cui C == E, ovvero 2 == 4, si può affermare che la sequenza generica delle challenge positive sia data da C-A-D.

Per le challenge negative, invece, la situazione è più caotica. Assumendo come vere le ipotesi fatte per le challenge positive, è possibile astrarre nel modo riportato in Tabella 7.7.

challenge negative	sequenza
bugsbunnychallenge	E-C-D
fightchallenge	C-A-B
firewroks	C-A
silhouettechallenge	E-B
strippatiktok	E-B
sugarbaby	E-B
updownchallenge	E-A

Tabella 7.7: Sequenze delle challenge negative

Per la seconda ipotesi, però, C == E, quindi tutti i dittonghi C-E possono essere raggruppati e definiti semplicemente come C. Stesso ragionamento per D == B, come riportato nella prima ipotesi effettuata. In questo modo, si otterranno le sequenze riportate nella Tabella 7.8.

challenge negative	sequenza
bugsbunnychallenge	C-D
fightchallenge	C-A-D
firewroks	C-A
silhouettechallenge	C-D
strippatiktok	C-D
sugarbaby	C-D
updownchallenge	C-A

Tabella 7.8: Sequenze delle challenge negative

Nel caso delle challenge negative, è possibile riconoscere due comportamenti differenti: alcune sono caratterizzate dalla sequenza C-A, quindi vengono interrotte durante la fase di esplosione della challenge, altre sono identificate dalla sequenza C-D, ovvero nascono e muoiono, senza avere una vera e propria fase di *boom*. L'unica eccezione è data da *fightchallenge*, caratterizzata dalla sequenza C-A-D, che possiede anche la fase di espansione.

7.3.1 Dimostrazione delle ipotesi

L'ultima fase è relativa alla dimostrazione delle ipotesi effettuate per generalizzare le sequenze. In primo luogo, è stata svolta un'analisi qualitativa dei cluster 3 e 0, che vengono raggruppati con la prima ipotesi, dove si assume che D == B. Infatti, tramite i dati presenti all'interno dei csv precedentemente prodotti, è possibile verificare che, relativamente alle interazioni, gli intervalli appartenenti al cluster 0 hanno pochi like e poche interazioni con i video (ovvero poche visualizzazioni, condivisioni e commenti), mentre il cluster 3, sebbene sia caratterizzato da molti like, è formato anch'esso da intervalli in cui ci sono pochi commenti e condivisioni dei video, e questa è una prima similarità. Per quello che riguarda il numero medio di video pubblicati, si può affermare che gli intervalli che appartengono al cluster 0 sono caratterizzati da un elevato numero di video pubblicati a distanza di ore uno dall'altro. La stessa cosa accade anche per gli intervalli appartenenti al cluster 3. Una differenza tra i due è da individuare nella durata del lifespan, infatti il cluster 0 ha un lifespan piccolo, mentre il cluster 3 ha un lifespan grande. Entrambi, però, hanno inizio nella seconda metà della vita delle challenge. Entrambe le reti, inoltre, hanno densità elevata.

Riassumendo, dall'analisi qualitativa effettuata, si può notare come sia il cluster 3 che il cluster 0 siano cluster finali di un intervallo, sebbene il cluster 0 arrivi subito ad una situazione di stabilità, mentre il cluster 3 è più lento e, quindi, genera un numero maggiore di like. I due cluster hanno caratteristiche pressoché simili; l'unica vera differenza sta nel numero di like, appunto, poiché il cluster 3 ha molti più like del cluster 0. Questo potrebbe essere dovuto soprattutto alla lunghezza del lifespan maggiore che caratterizza il cluster 3 rispetto al cluster 0.

Possiamo, quindi, dire che sia il cluster 3 che il cluster 0 rappresentano lo stesso fenomeno, sebbene con qualche differenza di evoluzione.

Per confermare tale supposizione è stata calcolata la percentuale di similarità, effettuando un test statistico. È stato necessario verificare se i due campioni avessero varianza uguale oppure no, in modo da poter scegliere se applicare il *t-test* (in caso di varianza uguale), oppure il *t-test di Welch* (in caso di varianze differenti).

Per poter effettuare questo passaggio è stato eseguito il *test di Bartlett*, in quanto erano presenti campioni di lunghezza di differente. Questo test viene utilizzato per verificare se più campioni provengono da popolazioni con varianze uguali. La soglia è stata posta pari a 0.05 ed esso ha riportato un p-value pari a 0.45, quindi è possibile considerare i due campioni a varianza simile e applicare il *t-test*.

Il test t (o, dall'inglese, t-test) è un test statistico di tipo parametrico con lo scopo di verificare se il valore medio di una distribuzione si discosta significativamente da un certo valore di riferimento. Questo test consente di confrontare con un certo livello di confidenza la media di un campione di dati con un valore noto. Tale valore può essere o la media della popolazione oppure un valore teorico. Esso è definito parametrico in quanto verifica l'ipotesi su una distribuzione lavorando su un suo parametro, in questo caso la media. I passi da effettuare sono:

- Definire l'ipotesi e scegliere il livello di significatività α: il primo step consiste nel definire il tipo di test di ipotesi (two tailed or one tailed) che si vuole utilizzare, le ipotesi (nulla H0 ed alternativa Ha) ed il livello di significatività.
- Definire la media e la deviazione standard del campione.
- Calcolare il numero di gradi di libertà del campione. Ricordiamo che questi ultimi sono pari al numero di dati indipendenti conosciuti. Poiché conosciamo la media del campione, il numero di dati indipendenti è pari a n-1.
- Calcolare il valore t limite (ovvero il valore oltre il quale, con il livello di significatività scelto, può essere rigettata l'ipotesi H0) dipende dal numero di gradi di libertà. Il calcolo della variabile t limite viene fatto mediante l'utilizzo delle tavole t.

Se la variabile t calcolata assume un valore oltre il t limite, allora l'ipotesi H0 può essere rigettata. In caso contrario, non è possibile rigettare l'ipotesi H0.

Nell'attuale caso di studio, H0 rappresenta l'ipotesi per cui gli intervalli di tipo 0 e 3 siano differenti, mentre il rigetto dell'ipotesi H0 stessa identifica una similarità statistica tra di essi. Imponendo la soglia di accettazione di H0 pari a 0.05, è stato eseguito il *t-test*, ottenendo un p-value pari a 0.33. Con questo dato è possibile rigettare H0 ed affermare che gli intervalli di tipo 0 e 3 sono statisticamente simili.

Per quanto riguarda, invece, i cluster 2 e 4, è possibile affermare, da un punto di vista qualitativo, che sono molto simili, in quanto hanno in media lo stesso numero di like, lo stesso numero medio di video, caratteristiche delle reti simili e sono entrambi intervalli iniziali. Possiamo dire che i cluster 2 e 4 rappresentano lo stesso fenomeno iniziale del lifespan di una challenge.

Anche in questo caso, per confermare tale supposizione, abbiamo calcolato la percentuale di similarità, effettuando un test statistico. Per poterlo applicare, sono stati ripercorsi tutti i passi effettuati con la dimostrazione precedente.

È stato nuovamente applicato il *test di Bartlett*, ponendo la soglia a 0.05, ed è stato ottenuto un p-value pari a 0.13. Per questo, è possibile considerare i due campioni a varianza simile. Alla luce di questo fatto, è stato applicato il *t-test* con una soglia di accettazione pari allo 0.05 e un'ipotesi H0 per cui gli intervalli di tipo 2 e 4 sono statisticamente differenti. Il p-value ottenuto è pari a 0.09 e questo ci permette di rigettare H0 e considerare gli intervalli di tipo 2 e 4 statisticamente simili.

7.3.2 Time pattern

In conclusione, otteniamo che le challenge positive hanno un'evoluzione di tipo C-A-D, ovvero uno stato iniziale in cui vengono pubblicati i primi video, si creano le prime connessioni ma ancora la realtà è piuttosto caotica e il tutto avviene durante la fase C. Dopodichè c'è una fase di boom, ovvero la fase A, in cui vengono pubblicati molti video e vi sono molte interazioni. Aumenta il numero di utenti verificati che pubblicano video e nella rete si vengono a creare sempre più connessioni. La fase A è, poi, caratterizzata da una conseguente fase discendente, in cui il boom tende a ridimensionarsi e la rete, in genere, a stabilizzarsi. Infine, abbiamo la fase D, ovvero la fase morente. Questa può essere di due tipi, o veloce (intervalli di tipo 0), in cui in breve tempo la challenge raggiunge una fase di stabilità, con pochi like, pochi video ed una rete ben connessa, oppure lenta (intervalli di tipo 3), in cui si ha un tempo maggiore, ancora molti like ai video, ma le interazioni sono poche e la rete risulta ben connessa.

Per le challenge pericolose, invece, abbiamo sostanzialmente 2 pattern differenti. Tolta l'unica eccezione, ovvero *fightchallenge* che sembra seguire il pattern delle non pericolose, abbiamo:

- Una fase nascente di tipo C, come nel caso delle non pericolose, ma poi subito una fase morente di tipo D, quindi nessuna fase di boom.
- Una fase nascente di tipo C, come nel caso delle non pericolose, la prima fase del boom, la crescita, di tipo A, ma nessuna fase di decrescita. La challenge muore nella fase di picco, probabilmente perché bloccata in qualche modo dalla piattaforma.

8. Confronto con approcci correlati

8.1 Stato dell'arte

Sono diversi i lavori pubblicati in merito all'analisi di aspetti concernenti il social network più diffuso del momento. Essi trattano i temi più disparati e sono presenti in gran quantità, ma è necessario puntualizzare che differiscono dall'attuale lavoro per obiettivi ed approcci.

In generale, all'interno del paper "Research on the Causes of the "Tik Tok" App Becoming Popular and the Existing Problems" di Li Xu, Xiaohui Yan e Zhengwu Zhang, sono stati analizzati i motivi per cui TikTok è diventato popolare. Dal 2017, infatti, l'industria relativa ai video brevi ha visto il suo apogeo e, proprio all'interno di questo mercato, ha trovato grande risalto TikTok, il principale social di creazione e pubblicazione di video brevi. All'interno di questa ricerca vengono analizzati i vantaggi e le minacce dell'applicazione, oltre a fornire un'analisi relativa alle strategie utilizzate, come quelle di promozione e di *advertising*, l'uso dell'Intelligenza Artificiale e la soddisfazione di utilizzo da parte dell'utente. Di contro, vengono identificati diversi svantaggi, come la povertà di contenuti, la presenza di fake news e la crescente dipendenza degli utenti verso la piattaforma.

Proprio a questo proposito è stato pubblicato un esperimento denominato "Part of the Hazards of TikTok", con cui sono stati selezionati novanta studenti e sono stati divisi in tre gruppi:

- il primo gruppo ha utilizzato TikTok per 60 minuti;
- il secondo gruppo ha utilizzato TikTok per 30 minuti;
- il terzo gruppo non ha utilizzato TikTok.

Dopodiché tutti gli studenti sono stati sottoposti ad un quiz composto da cinque domande logiche, considerando un timing da rispettare. Da questo esperimento è emerso che l'utilizzo prolungato della piattaforma ha un grande impatto sulle capacità logiche delle persone, in particolare sulla generazione Z, che sembra essere la più colpita dall'influenza di TikTok.

Un altro problema derivante dall'utilizzo smodato dei social, in particolare di TikTok, appunto, è relativo alla manipolazione del pensiero adolescenziale, in quanto alcuni video possono spingere verso comportamenti depressivi, che possono sfociare in anoressia o, addirittura, nel suicidio. Questa ricerca è stata pubblicata da Giuseppe Logrieco, Maria Rosaria Marchili, Mario Roversi e

Alberto Villani, all'interno del paper "The paradox of TikTok anti-pro-anorexia videos: how social media can promote non-suicidal self injury and anorexia".

È stato analizzato, anche, cosa spinge i ragazzi della generazione Z a partecipare ad una challenge, all'interno della pubblicazione "It's time to TikTok: exploring generation Z's motivations to participate in challenges" di Ahlse Johannes, Nilsson Felix e Sandstrom Nina. La conclusione ha mostrato come i motivi principali siano da ricercare nel divertimento e nella continua ricerca della viralità.

Questo aspetto è stato analizzato anche all'interno del paper "The Influence of Personality Traits and User Motivation on TikTok Mobile Video Usage" di Omar e Dequan, che mostra i risultati di un sondaggio online per reclutare 385 utenti TikTok, utilizzando la tecnica di campionamento della rete online. I risultati suggeriscono che sono state le motivazioni degli utenti ad avere un'influenza significativa sull'uso di TikTok, piuttosto che i tratti della personalità. Le motivazioni degli utenti, infatti, sono da ricercare tra l'espressione di sé, la voglia di interazione sociale e la curiosità nei confronti dei profili altrui. Questi elementi sono predittori significativi dei comportamenti di utilizzo di TikTok, ma differiscono per livelli ed influenza. Questo studio contribuisce alla comprensione, sia teorica che empirica, dell'uso dei media generati dagli utenti.

All'interno del paper "Dancing to the Partisan Beat: A First Analysis of Political Communication on TikTok" di Medina Serrano, Juan Carlos, Papakyriakopoulos, Orestis, Hegelich e Simon, si tratta dell'importanza di TikTok anche dal punto di vista politico. Sebbene la piattaforma sia nota per avere utenti che pubblicano video in cui ballano, sincronizzano le labbra o mostrano altri talenti, i video degli utenti che esprimono opinioni politiche hanno visto un recente impeto. Questo studio effettua una valutazione primaria della comunicazione politica su TikTok. Vengono raccolti una serie di video repubblicani e democratici degli Stati Uniti per indagare su come gli utenti comunicano tra loro su questioni politiche. Con l'aiuto della visione artificiale, dell'elaborazione del linguaggio naturale e degli strumenti statistici, viene dimostrato che la comunicazione politica su TikTok è molto più interattiva rispetto ad altre piattaforme di social media, con gli utenti che combinano più canali di informazione per diffondere i loro messaggi. Viene mostrato che la comunicazione politica avviene sotto forma di alberi di comunicazione, poiché gli utenti generano rami di risposte ai contenuti esistenti. In termini di dati demografici, emerge che gli utenti appartenenti a entrambi i partiti statunitensi sono giovani e si comportano in modo simile sulla piattaforma. Tuttavia, gli utenti repubblicani hanno generato più contenuti politici e i loro video hanno ricevuto più risposte; d'altro canto, gli utenti democratici si sono impegnati molto di più nelle discussioni trasversali.

Sempre da un punto di vista politico, il paper "Social Media and Fake News in the 2016 Election" di Hunt Allcott e Matthew Gentzkow, tratta alcune fake news circolate in merito alle elezioni presidenziali americane del 2016. In seguito alle elezioni presidenziali statunitensi del 2016, infatti, molti hanno espresso preoccupazione per gli effetti di storie false ("fake news"), diffuse in gran parte attraverso i social media. All'interno di questa ricerca viene approfondita la costruzione di notizie false e vengono presentati nuovi dati sulla loro pubblicazione prima delle elezioni. Attingendo ai dati di navigazione sul web, agli archivi dei siti web di verifica dei fatti e ai risultati di un nuovo sondaggio online, vengono tratte diverse conclusioni. La prima si basa sul fatto che i social media sono una fonte importante, ma non dominante, di notizie elettorali, con il 14% degli americani che chiama i social media la loro fonte "più importante"; in secondo luogo sono state analizzate alcune delle notizie false maggiormente virali apparse nei tre mesi precedenti le elezioni. Quelle a favore di Trump sono state condivise complessivamente 30 milioni di volte su Facebook, mentre quelle a favore della Clinton sono state condivise 8 milioni di volte. Infine, si osserva che le persone hanno molte più probabilità di credere alle storie che favoriscono il loro candidato preferito, specialmente se hanno reti di social media ideologicamente isolate.

Nella pubblicazione "Trends in the diffusion of misinformation on social media" di Hunt Allcott, Matthew Gentzkow e Chuan Yu si parla di disinformazione. Negli ultimi anni, si è

diffusa la preoccupazione che la disinformazione sui social media stia danneggiando le società e le istituzioni democratiche. In risposta, le piattaforme di social media hanno annunciato azioni per limitare la diffusione di contenuti falsi. Sono state misurate le tendenze nella diffusione di contenuti da 569 siti Web di notizie false e 9540 storie di notizie false su Facebook e Twitter tra gennaio 2015 e luglio 2018. Le interazioni degli utenti con contenuti falsi sono aumentate costantemente sia su Facebook che su Twitter. Dal 2016, tuttavia, le interazioni con contenuti falsi sono diminuite drasticamente su Facebook, mentre continuano a crescere su Twitter, con il rapporto tra le interazioni di Facebook e le condivisioni di Twitter in calo del 60%. In confronto, le interazioni con altri siti di notizie, affari o cultura hanno seguito tendenze simili su entrambe le piattaforme.

All'interno della ricerca "Cyberbullying in the world of teenagers and social media: A literature review", di Alim S., viene trattato il tema del cyberbullismo. Quest'ultimo è diventato un problema importante tra gli adolescenti, a causa del loro ingente utilizzo dei social media. Le precedenti indagini sulla letteratura non hanno trattato in dettaglio gli studi sul cyberbullismo e i fattori di rischio dello stesso. Questa revisione della letteratura esplora le aree di ricerca sul cyberbullismo, come l'uso dei social media da parte degli adolescenti, i fattori di rischio e il modo in cui gli adolescenti stessi affrontano gli incidenti di cyberbullismo. Gli attuali studi su questo tema hanno evidenziato determinati problemi, come l'elevato volume di incidenti di cyberbullismo a scuola, la maggiore divulgazione di informazioni personali sui social media, il condizionamento da parte degli *influencer* e la sicurezza dell'ambiente scolastico, sia per il bullo che per la vittima. Gli studi incentrati sui fattori di rischio del cyberbullismo hanno sollevato dibattiti su fattori relativi alla probabilità per maschi e femmine di essere vittime/cyberbulli. Affrontare il cyberbullismo richiede consapevolezza, educazione per gli attori coinvolti nel cyberbullismo, sviluppo di software per rilevarlo e per includere gli attori nel monitoraggio dello stesso.

Un altro ambito di ricerca è stato applicato all'interno del paper "Communicating COVID-19 information on TikTok: a content analysis of TikTok videos from official accounts featured in the COVID-19 information hub" di Yachao Li, Mengfei Guan, Paige Hammond e Lane E Berrey. Durante la pandemia di COVID-19, TikTok ha creato un centro di informazioni per fornire agli utenti comunicazioni coinvolgenti e autorevoli su questo tema. Questo studio indaga il formato video, il tipo, il contenuto dei video relativi al COVID-19 su TikTok e il modo in cui gli attributi dei video sono correlati agli indicatori quantitativi del coinvolgimento degli utenti, inclusi il numero di visualizzazioni, di like, di commenti e di condivisioni. Un'analisi dei contenuti ha esaminato 331 video di account ufficiali presenti nell'hub di informazioni COVID-19. Al 5 maggio 2020, i video hanno ricevuto 907'930'000 visualizzazioni, 29'640'000 like, 168'880 commenti e 781'862 condivisioni. Circa un video su tre aveva i sottotitoli, che erano correlati positivamente al numero di condivisioni. Quasi tutti i video includevano un hashtag, e un numero maggiore di hashtag era correlato a più like. I tipi di video includevano recitazione, infografica animata, documentari, notizie, discorsi orali, presentazioni pittoriche e balletti. I video di danza hanno avuto il maggior numero di condivisioni. I video che trasmettevano emozioni di allarme, preoccupazione, suscettibilità e gravità relativamente al COVID-19 hanno avuto un maggiore coinvolgimento degli utenti. Questo studio potrebbe aiutare le agenzie di sanità pubblica ad essere maggiormente consapevoli dell'opportunità che rappresenta TikTok nella comunicazione sanitaria e crea una comunicazione del rischio incentrata sul pubblico per coinvolgere e informare i membri della comunità.

All'interno della ricerca "spreading hate on TikTok" di Gabriel Weimann e Natalie Masri, si parla della crescente presenza di gruppi di estremisti sulle piattaforme dei social media, diventata più importante e massiccia. Tuttavia, mentre la maggior parte dell'attenzione degli studiosi si è concentrata su piattaforme leader, come Twitter, Facebook o Instagram, l'atteggiamento estremista su altre piattaforme come TikTok non è passato inosservato. Questo studio è un'analisi descrittiva

basata sul contenuto dei video TikTok, pubblicati all'inizio del 2020. I risultati rivelano la presenza inquietante dell'estremismo di estrema destra nei video, commenti, simboli e immagini inclusi nei post di TikTok. Quest'ultimo ha caratteristiche uniche; innanzitutto, a differenza di tutti gli altri social media, gli utenti di TikTok sono quasi tutti bambini piccoli, che sono più ingenui, soprattutto quando si tratta di contenuti dannosi. In secondo luogo, TikTok è la piattaforma più recente, quindi si trova ad essere in ritardo rispetto ai suoi rivali in termini di sicurezza, che hanno avuto più tempo per capire come proteggere i propri utenti da contenuti inquietanti e dannosi.

Questi sono soltanto alcuni degli studi e delle analisi relative a questa piattaforma, che trattano diversi fenomeni sotto ogni punto di vista. Tuttavia, è importante sottolineare che in letteratura non è presente alcun approccio simile a quello utilizzato in questo lavoro.

8.2 Innovazione dell'approccio utilizzato

L'approccio utilizzato in questo lavoro può definirsi innovativo, poiché permette la realizzazione di un metodo che, data una nuova challenge, è in grado di definire gli intervalli e raggrupparli in cluster in base alle feature considerate. Ciò ha portato alla definizione di design pattern che ricorrono nelle varie tipologie di challenge e che differiscono da challenge positive a negative. L'intero processo è reso possibile dall'avvio degli script creati, nell'ordine in cui sono stati esposti all'interno di questo lavoro, in modo da garantire la creazione dei file CSV necessari per la corretta esecuzione di ogni file presente all'interno della *working directory*.

9. Conclusioni

9.1 Conclusioni

In questo elaborato è stato presentato un approccio innovativo per l'individuazione e per la caratterizzazione delle diverse fasi di sviluppo che si presentano all'interno di una challenge.

Nella prima parte del lavoro si sono presentati l'ambito di ricerca e gli obiettivi della stessa. Successivamente è stato presentato il modo in cui l'algoritmo sviluppato estrapola i dati dalla piattaforma, con conseguente attività di ETL, per poi passare alla costruzione delle reti. In seguito è stata effettuata un'analisi preliminare su ciascuna challenge, così da poter effettuare un'analisi media delle challenge positive e negative per mettere in luce le differenze tra le due tipologie. Successivamente, sono stati individuati gli intervalli di ciascuna challenge e sono stati raggruppati mediante *cluster*. Infine, è stata operata un'astrazione e sono stati individuati i time pattern esistenti.

9.2 Sviluppi futuri

I possibili sviluppi futuri, relativamente al lavoro corrente, possono incentrarsi sulla predizione del profilo di utenti che pubblicano video nell'intervallo di tempo successivo a quello considerato. Questo lavoro è reso possibile dai dati a disposizione riguardanti gli utenti in ogni intervallo.

Un altro possibile sviluppo consiste nell'identificazione di eventuali comunità e della loro evoluzione temporale, considerando un avanzamento del 5% o del 10% del lifespan, come nel caso di studio corrente. Questo permetterebbe di effettuare un'analisi sulla propagazione della challenge nel tempo e di identificare possibili strutture e comunità presenti all'interno della rete.

Appendice

Appendice1: "main.py"

```
from TikTokApi import TikTokApi
import lib.tiktok_dataset as td
import lib.challenges as challenge
import argparse

parser = argparse.ArgumentParser(description = 'TikTok dataset generator.')
parser.add_argument('-c', action = 'store, type = str,
                    help = 'Challenge hashtag. Available challenges:
                    ↪  \n'+challenge.getChallengeList()+'.',
                    metavar = '<challenge>',
                    default = "itookanap")
parser.add_argument('-l', action = 'store', type = int,
                    help = 'Check for more tiktoks (keep searching for users with public
                    ↪  liked tiktok\'s list).',
                    metavar = '<loop>',
                    default = 0)
args = parser.parse_args()

challenge = challenge.getChallenge(args.c)

api = TikTokApi.get_instance(use_test_endpoints=True)

td.buildDatasetByHashtag(api, challenge["name"], challenge["url"], 10000)
td.pubAuthList(api, challenge["name"])
rows = td.checkConnections(api, challenge["name"])
while rows != 0 and args.l: # conditional loop
    count = td.pubAuthList(api, challenge["name"], rows)
    if count == 0:
        break
    else:
        rows = td.checkConnections(api, challenge["name"], count)
td.ETL(challenge["name"])
```

Appendice2: "challenges.py"

```
challenges = { "itookanap":
                    {"name": "ITookANap",
                    "url":
                    ↪  "https://www.tiktok.com/@gunnarolla/video/6816020939759815942"}, #
                    ↪  original video
                "elpepe":
                    {"name":"elpepe",
                    "url": ""},
                "ohnanachallenge":
                    {"name":"ohnanachallenge",
                    "url":
                    ↪  "https://www.tiktok.com/@saffronbarker/video/6778137328499084549"},
                "makarenachallenge":
                    {"name":"makarenachallenge",
                     "url":""},
                "weirdsoundchallenge":
                    {"name":"weirdsoundchallenge",
                    "url":
                    ↪  "https://www.tiktok.com/@angelbrown_19/video/6806305248223759622"},
                "copinesdancechallenge":
                    {"name":"copinesdancechallenge",
                    "url":
                    ↪  "https://www.tiktok.com/@nicolesopogee/video/6906089727640816897"},
                "walkamile":
                    {"name":"walkamile",
                     "url": ""},
                "silhouettechallenge":
                    {"name":"silhouettechallenge",
                     "url": ""},
                "badchallenge":
                    {"name":"badchallenge",
                     "url": ""},
                "eatingchallenge":
                    {"name":"eatingchallenge",
                     "url": ""},
                "kissyourpetchallenge":
                    {"name":"kissyourpetchallenge",
                    "url": "https://www.tiktok.com/@foopydrip/video/6922985431424437510"},
                "emojichallenge":
                    {"name":"emojichallenge",
                     "url": ""},
                "flexibilitychallenge":
                    {"name":"flexibilitychallenge",
                    "url":
                    ↪  "https://www.tiktok.com/@ayushishukla098/video/6546886448757347328"},
                "bussitchallenge":
                    {"name":"bussitchallenge",
                     "url": ""},
                "dominochallenge":
                    {"name":"dominochallenge",
                    "url":
                    ↪  "https://www.tiktok.com/@stilestefanbae/video/6905779244845092098"},
                "popcornkaraoke":
                    {"name":"popcornkaraoke",
                     "url": ""},
                "boredinthehouse":
                    {"name":"boredinthehouse",
                    "url":
                    ↪  "https://www.tiktok.com/@curtisroach/video/6800471860761971974"},
```

```
                "beautifulpeople":
                    {"name":"beautifulpeople",
                    "url":
                     ↪  "https://www.tiktok.com/@daniloantonelli/video/6716795433986952454"},
                "savagechallenge":
                    {"name":"savagechallenge",
                     "url": ""},
                "savagelove":
                    {"name":"savagelove",
                     "url": ""},
                "plankchallenge":
                    {"name":"plankchallenge",
                     "url": ""},
                "levelupchallenge":
                    {"name":"levelupchallenge",
                    "url": "https://www.tiktok.com/@jelinuh/video/6580863052575411461"},
                "makemomsmile":
                    {"name":"MakeMomSmile",
                    "url": ""},
                "makeupchallenge":
                    {"name":"makeupchallenge",
                     "url": ""},
                "thatjustmybabydoggy":
                    {"name":"thatjustmybabydoggy",
                     "url": "https://www.tiktok.com/@stemjmu/video/6826015024713059590"},
                "cerealchallenge":
                    {"name":"cerealchallenge",
                     "url": ""},
                "facefilterchallenge":
                    {"name":"facefilterchallenge, facefilter",
                     "url": ""},
                "obstaclechallenge":
                    {"name":"obstaclechallenge",
                     "url": ""},
                "ohnono":
                    {"name":"ohnono",
                     "url": ""},
                "colpiditesta":
                    {"name":"colpiditesta",
                     "url": ""},
                "achiassomiglio":
                    {"name":"achiassomiglio",
                     "url": ""},
                "bugsbunny":
                    {"name":"bugsbunnychallenge,bugsbunny,bugsbunnychallange",
                    "url": ""},
                "dontbreathe":
                    {"name":
                     ↪  "dontbreathe,dontbreathechallenge,dontbreath,dontbreathchallenge",
                    "url": ""},
                "firewroks":
                    {"name": "firewroks,fireworks",
                    "url": ""},
                "drinkchallenge":
                    {"name": "drinkchallenge,drinkingchallenge,tiktokdrinkchallenge",
                    "url": ""},
                "nakedchallenge":
                    {"name": "nakedchallenge, nakedtiktok, alwaysnaked, fullynaked,
                    ↪  nakedchallange, walkinnaked, nakedchalleng, thenakedchallenge,
                    ↪  halfnaked",
                    "url": ""},
```

```
                "burnhair":
                    {"name": "burnhair, hairburn, burnyourhair, burnmyhair,
                    ↪ hairburnchallenge, burnhairstupid",
                    "url": ""},
                "eatingchalk":
                    {"name": "eatingchalk, eatingchalkasmr, eatchalk, wetchalkeating,
                    ↪ asmrchalkeating, chalkeatingchallenge, chalkeatinglovers,
                    ↪ loveeatingchalk, chalkpencileating",
                    "url":""},
                "knifechallenge":
                    {"name": "knifechallenge, knifechallange, knifechallanege",
                    "url":""},
                "strippatiktok":
                    {"name": "strippatiktok, striptok, striptiktok, strippa,
                    ↪ strippaoftiktok, strippatik, strippeertiktok, strippatiktok2021,
                    ↪ strippatok",
                    "url": ""},
                "fightchallenge":
                    {"name": "fightchallenge, fighting, fights, fightscene,
                    ↪ fightscenechallenge, fightvideo, fightvideos,
                    ↪ fightnightchallenge",
                    "url": ""},
                "sugarbaby":
                    {"name": "sugarbaby, sugarbabytips, sugarbabylife, sugarbabies,
                    ↪ suggarbaby, sugarbabytiktok, suggarbabby, babysugar, sugarbabby,
                    ↪ sugardaddy, sugardarddy, sugardady, suggardaddy, sugardadddy,
                    ↪ sugardaddies, daddysugar, sugardadaddy, sugardad",
                    "url": ""},
                "foodcoloringchallenge":
                    {"name": "foodcoloringchallenge, foodcoloring, foodcolouringchallenge,
                    ↪ foodcoloringinwater, foodcoloringandmilk, redfoodcoloring,
                    ↪ greenfoodcoloring, bluefoodcoloring",
                    "url": ""},
                "updownchallenge":
                    {"name": "updownchallenge, upanddown, UpDown, upanddownchallenge,
                    ↪ updownupdownchallenge",
                    "url": ""},
                "basketballbeerchallenge":
                    {"name": "basketballbeerchallenge, basketballbeer,
                    ↪ beerbasketballchallenge, basketballbeerfail,
                    ↪ beerchallengebasketball, basketballbeerchallengefail,
                    ↪ beerbasketball",
                    "url": ""}
                # insert challenge here
            }

def getChallenge(challenge):
    return challenges[challenge.lower()]

def getChallengeList():
    return ', '.join(list(challenges.keys()))
```

Appendice3: "tiktok_dataset.py"

```python
from TikTokApi import TikTokApi
from TikTokApi import browser
import pandas as pd
import lib.utils as utils
import os
import uuid
import logging

def ETL(dataset, dateConv = True):
    dataset = dataset.split(",")[0]
    df = pd.read_csv("./dataset/dataset_"+dataset+"_connections.csv", sep=";")
    df = df[['id', 'createTime', 'video_id', 'video_duration', 'author_id',
    ↪  'author_uniqueId', 'author_nickname', 'author_verified', 'author_secUid',
    ↪  'music_id', 'music_title', 'music_authorName', 'stats_diggCount',
    ↪  'stats_shareCount', 'stats_commentCount', 'stats_playCount',
    ↪  'duetInfo_duetFromId', 'authorStats_diggCount', 'authorStats_followingCount',
    ↪  'authorStats_followerCount', 'authorStats_heartCount', 'authorStats_videoCount',
    ↪  'duetEnabled', 'originalVideo', 'likedBy_id', 'likedBy_secUid',
    ↪  'likedBy_uniqueId', 'idcopy']] # extract columns
    df['id'] = df['id'].astype("string")
    df['originalVideo'] = df['originalVideo'].fillna(0)
    try:
        if dateConv:
            df['createTime'] = pd.to_datetime(df['createTime']*1000, unit='ms') # convert
            ↪  to datetime
    except:
        print("createTime column already converted")
    df.loc[df['idcopy'] != '-','id'] = df['idcopy']
    df.drop(['idcopy'], axis=1, inplace=True)
    df.to_csv("dataset/dataset_"+dataset+"_connections_etl.csv", sep=';', index=False)
    ↪  #save filtered dataset

def checkConnections(api, dataset, count=0): #dataset is the hashtag
    list_of_tags = dataset.split(",")
    dataset = list_of_tags[0]
    logger = logging.getLogger()
    pd.options.mode.chained_assignment = None
    print("Generating connection's dataset. This operation may take a while...\n")
    if count == 0:
        df = pd.read_csv("./dataset/dataset_"+dataset+".csv", sep=";")
        df['id'] = df['id'].astype("string")
        df['likedBy_id'] = "-"
        df['likedBy_secUid'] = "-"
        df['likedBy_uniqueId'] = "-"
        df['idcopy'] = "-"
    else:
        df = pd.read_csv("./dataset/dataset_"+dataset+"_connections.csv", sep=";",
        ↪  dtype="string")
    rowsAdded = 0
    if os.path.exists("./dataset/authors/pubLiked_"+dataset+".csv"):
        df1 = pd.read_csv("./dataset/authors/pubLiked_"+dataset+".csv", sep = ";",
        ↪  dtype="string")
        if count != 0:
            df1 = df1.tail(count)
    else:
        df1 = pd.read_csv("./dataset/dataset_"+dataset+".csv", sep=";", dtype="string") #
        ↪  default
    logger.disabled = True
    utils.printProgressBar(0, df1.shape[0], prefix = ' Parsing:', length = 40)
```

```
48      for index, row in df1.iterrows(): # users with open liked list
49          try:
50              tiktoks = api.userLiked(row['author_id'],row['author_secUid'],count=10000)
51              if(len(tiktoks) > 0):
52                  df2 = utils.datasetHelper(tiktoks)
53
                    #df2.to_csv("./dataset/test_"+str(uuid.uuid4().hex)+".csv",sep=";",index=False)
54                  for index1, row1 in df2.iterrows(): # liked tiktoks
55                      if any(row1["desc"].lower().find("#"+elem_.lower()) != -1 for elem_ in
                        list_of_tags) and (row1["id"] not in list(df["id"])):
56                          row1['likedBy_id'] = "-"
57                          row1['likedBy_secUid'] = "-"
58                          row1['likedBy_uniqueId'] = "-"
59                          row1['idcopy'] = "-"
60                          df = df.append(row1, ignore_index=True)
61                          rowsAdded += 1
62                      if row1["id"] in list(df["id"]):
63                          if df.loc[df['id'] == row1["id"], 'likedBy_id'].iloc[:].values[0]
                            == '-': # if cell equals "-" (empty)
64                              df.loc[df['id'] == row1["id"], 'likedBy_id'] =
                                row["author_id"]
65                              df.loc[df['id'] == row1["id"], 'likedBy_secUid'] =
                                row["author_secUid"]
66                              df.loc[df['id'] == row1["id"], 'likedBy_uniqueId'] =
                                row["author_uniqueId"]
67                          else:
68                              df_temp = df.loc[df['id'] == row1["id"]] # a tiktok is liked
                                by more than a user
69                              df_temp['idcopy'] = df_temp['id']
70                              df_temp['id'] = df_temp['id'] + "_" + str(uuid.uuid4().hex)
71                              df_temp["likedBy_id"] = row["author_id"]
72                              df_temp["likedBy_secUid"] = row["author_secUid"]
73                              df_temp["likedBy_uniqueId"] = row["author_uniqueId"]
74                              df = df.append([df_temp], ignore_index=True)
75                              rowsAdded += 1
76              utils.printProgressBar(index+1, df1.shape[0], prefix = ' Parsing:', length =
                40)
77          except:
78              browser.clean_playwright()
79              api.clean_up() # cleans api and restart it
80              api = TikTokApi.get_instance(use_test_endpoints=True)
81      logger.disabled = False
82      print("\n\nSaving data...")
83      df.to_csv("dataset/dataset_"+dataset+"_connections.csv", sep=';', index=False)
84      return rowsAdded # restituisci il numero di righe aggiunte ad ogni iterazione
85
86  def buildDatasetByHashtag(api, hashtag, url_orig, _count=10000):
87      df = pd.DataFrame()
88      for tag in hashtag.split(","):
89          try:
90              tiktoks = api.byHashtag(tag.lower(), count=_count, custom_verifyFp="")
91              print("Downloaded "+str(len(tiktoks)) + " for '"+tag+"'")
92              df = df.append(utils.datasetHelper(tiktoks))
93          except Exception as ex:
94              print("Api failed downloading data for hashtag: '"+tag+"'\nPlease
                wait...checking for more tags...")
95      _id = ""
96      if url_orig:
97          tiktok = api.getTikTokByUrl(url_orig)
98          _id = tiktok["itemInfo"]["itemStruct"]["id"]
99          if _id not in list(df["id"]):
```

```
            tiktokL = [tiktok["itemInfo"]["itemStruct"]]
            df = df.append(utils.datasetHelper(tiktokL), ignore_index = True)
    df['originalVideo'] = 0
    df.loc[df['id'] == _id, 'originalVideo'] = 1 # search for the original video and flag it
    df.drop_duplicates(subset="id", keep="first", inplace=True)
    df.to_csv("dataset/dataset_"+hashtag.split(",")[0]+".csv", sep=';', index=False)

def pubAuthList(api, dataset, rows=0): # returns dataset of users with public liked tiktok's list
    dataset = dataset.split(",")[0]
    if rows == 0:
        df = pd.read_csv("./dataset/dataset_"+dataset+".csv", sep=";")
    else:
        df = pd.read_csv("./dataset/dataset_"+dataset+"_connections.csv", sep=";")
        df = df.tail(rows)
    if os.path.exists("./dataset/authors/pubLiked_"+dataset+".csv"):
        df1 = pd.read_csv("./dataset/authors/pubLiked_"+dataset+".csv", sep = ";", dtype="string")
    else:
        df1 = pd.DataFrame({'author_id': [], 'author_secUid': [], 'author_uniqueId': []}, dtype="string")
    logger = logging.getLogger()
    count = 0
    logger.disabled = True
    counter = 1
    size = df.shape[0]
    for index, row in df.iterrows():
        try:
            print("Parsing: "+str(counter)+"/"+str(size))
            counter += 1
            if len(api.userLiked(row['author_id'],row['author_secUid'],count=1)) > 0:
                tempDict = {'author_id': row['author_id'], 'author_secUid': row['author_secUid'], 'author_uniqueId': row['author_uniqueId']}
                if row['author_uniqueId'] not in df1['author_uniqueId'].iloc[:].values:
                    df1 = df1.append(tempDict, ignore_index=True)
                    if rows != 0:
                        count += 1
            #if df1.shape[0] == 2: # subset of users
                #break
        except:
            print("Recovery mode.")
            browser.clean_playwright()
            api.clean_up() # cleans api and restart it
            api = TikTokApi.get_instance(use_test_endpoints=True)
    logger.disabled = False
    #df1.drop_duplicates(subset='author_uniqueId', keep='first', inplace=True)
    df1.to_csv("dataset/authors/pubLiked_"+dataset+".csv", sep=";", index=False)
    return count
```

Appendice4: "network.py"

```
import networkx as nx
import matplotlib.pyplot as plt
import lib.tiktok_network as ntx

plt.figure(figsize=(15,15))
graph, labels, colors, pos, _, _, _ =
↪  ntx.graphCalculation("bussitchallenge",colorCriteria="createTime", lifespanCond=None,
↪  intervals=[95,100], remAutoLikes=False)
ntx.graphStats(graph)
nx.draw_networkx_nodes(graph,pos,node_color=colors,node_size=60)
nx.draw_networkx_labels(graph, pos, labels,font_size=5)
nx.draw_networkx_edges(graph,pos,arrows=True,arrowsize=3,arrowstyle="-|>",alpha=0.5)
ax= plt.gca()
ax.collections[0].set_edgecolor("#000000")
plt.show()
```

Appendice5: "tiktok_network.py"

```
import pandas as pd
import networkx as nx
import numpy as np
from random import randrange
import math

def ETL2(dataFrame, remAutoLikes):
    dataFrame = dataFrame[dataFrame["likedBy_uniqueId"]!= "-"]
    if remAutoLikes:
        dataFrame = dataFrame[dataFrame["likedBy_uniqueId"]!=dataFrame["author_uniqueId"]]
        ↪  # remove unconnected nodes
    return dataFrame

def position_1(radius):
    angle = randrange(361) #calcolo un numero casuale da 0 a 360
    x_coord = radius*(math.cos(angle))
    y_coord = radius*(math.sin(angle))
    return (x_coord, y_coord)

def graphStats(graph, _print=True):
    stats = {"nnodes":graph.number_of_nodes(),
             "nedges":graph.number_of_edges(),
             "mindegree":np.mean([x[1] for x in graph.in_degree()]),
             "moutdegree":np.mean([x[1] for x in graph.out_degree()]),
             "avgclust":nx.average_clustering(graph),
             "density":nx.density(graph)}
    if _print:
        print('Number of nodes: '+str(stats["nnodes"]))
        print('Number of edges: '+str(stats["nedges"]))
        print('Mean indegree: '+str(stats["mindegree"]))
        print('Mean indegree std: '+str(np.std([x[1] for x in graph.in_degree()])))
        print('Mean outdegree: '+str(stats["moutdegree"]))
        print('Mean outdegree std: '+str(np.std([x[1] for x in graph.out_degree()])))
        print('Average clustering coefficient: '+str(stats["avgclust"]))
        print('Density: '+str(stats["density"]))
    return stats

def graphCalculation(dataset, colorCriteria = "createTime", lifespanCond = None, intervals
↪  = None, remAutoLikes = True):
```

```
    nodes = set()
    labels = dict()
    colors = list()
    edges = list()
    dmap = dict()
    pos = dict()
    lifespan_stats_days = 0
    nodestats = dict()
    df = pd.read_csv("./dataset/dataset_"+dataset+"_connections_etl.csv", sep=";")
    df['createTime'] = pd.to_datetime(df['createTime'])
    dtemp = df.copy()
    df = ETL2(df, remAutoLikes)
    lifespan = df['createTime'].max() - df['createTime'].min()
    if intervals is not None:
        lifespanCond = None
        lim_sup = df["createTime"].min() + (lifespan/100 * intervals[1])
        lim_inf = df["createTime"].min() + (lifespan/100 * intervals[0])
        lifespan_stats_days = (lim_sup - lim_inf).days
        df = df.loc[df["createTime"].between(lim_inf, lim_sup)]
    if lifespanCond is not None:
        df = df.loc[df['createTime'] >= (df['createTime'].min() + (lifespan/100
        ↪  *lifespanCond))]
    dfnet = df.copy()
    for index, row in df.iterrows():
        nodes.add(int(row["author_id"]))
        labels[int(row["author_id"])] = row["author_uniqueId"]
        nodes.add(int(row["likedBy_id"]))
        labels[int(row["likedBy_id"])] = row["likedBy_uniqueId"]
        edg=[]
        source=int(row['likedBy_id'])
        target=int(row['author_id'])
        edg.append(source)
        edg.append(target)
        edges.append(edg)
    if colorCriteria == 'createTime':
        df['createTime_norm'] = df['createTime'].astype(np.int64) / 10e9
        df['createTime_norm'] = round((df['createTime_norm']-df['createTime_norm'].min())/
        ↪  (df['createTime_norm'].max()-df['createTime_norm'].min()) * 255) # RGB color
        for node in nodes:
            pos[node] = "" # prepare position dictionary
            if node in list(df['author_id'].astype(np.int64)):
                nodestats[node] = {"createTime":df.loc[df['author_id'].astype(np.int64) ==
                ↪  node, 'createTime'].iloc[:].values[0],
                "music_id": str(df.loc[df['author_id'].astype(np.int64) == node,
                ↪  'music_id'].iloc[:].values[0]),
                "video_duration": df.loc[df['author_id'].astype(np.int64) == node,
                ↪  'video_duration'].iloc[:].values[0],
                "stats_diggCount":df.loc[df['author_id'].astype(np.int64) == node,
                ↪  'stats_diggCount'].iloc[:].values[0],
                "stats_shareCount":df.loc[df['author_id'].astype(np.int64) == node,
                ↪  'stats_shareCount'].iloc[:].values[0],
                "stats_commentCount":df.loc[df['author_id'].astype(np.int64) == node,
                ↪  'stats_commentCount'].iloc[:].values[0],
                "stats_playCount":df.loc[df['author_id'].astype(np.int64) == node,
                ↪  'stats_playCount'].iloc[:].values[0],
                "authorStats_diggCount":df.loc[df['author_id'].astype(np.int64) == node,
                ↪  'authorStats_diggCount'].iloc[:].values[0],
                "authorStats_followingCount":df.loc[df['author_id'].astype(np.int64) ==
                ↪  node, 'authorStats_followingCount'].iloc[:].values[0],
                "authorStats_heartCount":df.loc[df['author_id'].astype(np.int64) == node,
                ↪  'authorStats_heartCount'].iloc[:].values[0],
```

```
87                  "authorStats_followerCount":df.loc[df['author_id'].astype(np.int64) ==
                    ↪  node, 'authorStats_followerCount'].iloc[:].values[0],
88                  "authorStats_videoCount":df.loc[df['author_id'].astype(np.int64) == node,
                    ↪  'authorStats_videoCount'].iloc[:].values[0]}
89                  try:
90                      val = int(df.loc[df['author_id'].astype(np.int64) == node,
                        ↪  'createTime_norm'].iloc[:].values[0])
91                  except:
92                      val = 0
93                  if df.loc[df['author_id'].astype(np.int64) == node,
                    ↪  'originalVideo'].iloc[:].values[0] == 1:
94                      colors.append("#ff0000")
95                  else:
96                      colors.append('#%02x%02x%02x' % (val, val, val))
97              else:
98                  dfnet = dfnet.append(dtemp.loc[dtemp['author_id'].astype(np.int64) ==
                    ↪  node].iloc[:])
99                  val = 256
100                 colors.append("#ffff57")
101             try:
102                 dmap[val].append(node)
103             except:
104                 dmap[val] = [node]
105         dict_k = sorted(list(dmap.keys()))
106         for val in dict_k:
107             for node in dmap[val]:
108                 pos[node] = position_1(val)
109     graph = nx.DiGraph()
110     graph.add_nodes_from(nodes)
111     graph.add_edges_from(edges)
112     dfnet.drop_duplicates(subset ='author_id',keep = 'first', inplace = True)
113     dfinterval = df # represents the dataset of the net
114     return [graph, labels, colors, pos, nodestats, dfnet, dfinterval, lifespan_stats_days]
```

Appendice 6: "pre_analysis.py"

```
import pandas as pd
import lib.challenges as challenges
import lib.tiktok_network as nx
import numpy as np
import matplotlib.pyplot as plt
from ast import literal_eval

POS_CHALLENGES = ["bussitchallenge",
                  "copinesdancechallenge",
                  "emojichallenge",
                  "colpiditesta",
                  "boredinthehouse",
                  "ITookANap",
                  "plankchallenge"] # list of selected positive challenges

NEG_CHALLENGES = ["silhouettechallenge",
                  "bugsbunny",
                  "strippatiktok",
                  "firewroks",
                  "fightchallenge",
                  "updownchallenge",
                  "sugarbaby"] #list of selected negative challenges

POS_CHALLENGES_COND = [None, None, None, None, None, None, None]
NEG_CHALLENGES_COND = [None, None, None, None, None, None, None]
ALL_CHALLENGES = [POS_CHALLENGES, NEG_CHALLENGES]
PLOT = [[],[]]
PLOTPOS = [[] for x in POS_CHALLENGES]
PLOTNEG = [[] for x in NEG_CHALLENGES]
PLOTCOUNT = [[],[]]
PLOTPOSCOUNT = [[] for x in POS_CHALLENGES]
PLOTNEGCOUNT = [[] for x in NEG_CHALLENGES]
PLOTPOSMEANLIKESCOUNT = [[] for x in POS_CHALLENGES]
PLOTNEGMEANLIKESCOUNT = [[] for x in NEG_CHALLENGES]
PLOTPOSVIDEOCOUNT = [[] for x in POS_CHALLENGES]
PLOTNEGVIDEOCOUNT = [[] for x in NEG_CHALLENGES]
PLOTPOSFOLLOWINGCOUNT = [[] for x in POS_CHALLENGES]
PLOTNEGFOLLOWINGCOUNT = [[] for x in NEG_CHALLENGES]

def reset_stats():
    return {"span":[[] for x in range(0,100)],
            "n_nodes":[[] for x in range(0,100)],
            "mlikes":[[] for x in range(0,20)],
            "n_video":[[] for x in range(0,20)],
            "following":[[] for x in range(0,20)],
            "nnodes":[],
            "nedges":[],
            "mindegree":[],
            "moutdegree":[],
            "avgclust":[],
            "density":[],
            "lifespan":[],
            "5_lifespan":[],
            "25_lifespan":[],
            "50_lifespan":[],
            "75_lifespan":[],
            "video_duration":[],
            "stats_shareCount":[],
            "stats_commentCount":[],
```

```
            "stats_diggCount":[],
            "stats_playCount":[],
            "music_id_count":[],
            "authorStats_diggCount":[],
            "authorStats_followingCount":[],
            "authorStats_heartCount":[],
            "authorStats_followerCount":[],
            "authorStats_videoCount":[]}
def print_results(arr, _type=False):
    text = "POSITIVE"
    if _type:
        text = "NEGATIVE"
        for val in arr["span"]:
            PLOT[1].append(np.mean(val))
            for i in range(len(NEG_CHALLENGES)):
                PLOTNEG[i].append(val[i])
        for val in arr["n_nodes"]:
            PLOTCOUNT[1].append(np.mean(val))
            for i in range(len(NEG_CHALLENGES)):
                PLOTNEGCOUNT[i].append(val[i])
        for val in arr["mlikes"]:
            for i in range(len(NEG_CHALLENGES)):
                PLOTNEGMEANLIKESCOUNT[i].append(val[i])
        for val in arr["n_video"]:
            for i in range(len(NEG_CHALLENGES)):
                PLOTNEGVIDEOCOUNT[i].append(val[i])
        for val in arr["following"]:
            for i in range(len(NEG_CHALLENGES)):
                PLOTNEGFOLLOWINGCOUNT[i].append(val[i])
    else:
        for val in arr["span"]:
            PLOT[0].append(np.mean(val))
            for i in range(len(POS_CHALLENGES)):
                PLOTPOS[i].append(val[i])
        for val in arr["n_nodes"]:
            PLOTCOUNT[0].append(np.mean(val))
            for i in range(len(POS_CHALLENGES)):
                PLOTPOSCOUNT[i].append(val[i])
        for val in arr["mlikes"]:
            for i in range(len(POS_CHALLENGES)):
                PLOTPOSMEANLIKESCOUNT[i].append(val[i])
        for val in arr["n_video"]:
            for i in range(len(POS_CHALLENGES)):
                PLOTPOSVIDEOCOUNT[i].append(val[i])
        for val in arr["following"]:
            for i in range(len(POS_CHALLENGES)):
                PLOTPOSFOLLOWINGCOUNT[i].append(val[i])
    print("**********************"+text+"**********************")
    print("Mean number of nodes: " + str(np.mean(arr["nnodes"])) + " (std: " +
    ↪  str(np.std(arr["nnodes"])) + ")")
    print("Mean number of edges: " + str(np.mean(arr["nedges"])) + " (std: " +
    ↪  str(np.std(arr["nedges"])) + ")")
    print("Mean indegree: " + str(np.mean(arr["mindegree"])) + " (std: " +
    ↪  str(np.std(arr["mindegree"])) + ")")
    print("Mean outdegree: " + str(np.mean(arr["moutdegree"])) + " (std: " +
    ↪  str(np.std(arr["moutdegree"])) + ")")
    print("Mean clustering coefficient: " + str(np.mean(arr["avgclust"])) + " (std: " +
    ↪  str(np.std(arr["avgclust"])) + ")")
    print("Mean lifespan (days): " + str(np.mean(arr["lifespan"])) + " (std: " +
    ↪  str(np.std(arr["lifespan"])) + ")")
```

```
114     print("Mean 5percent lifespan number of nodes: " + str(np.mean(arr["5_lifespan"])) + "
        ↪  (std: " + str(np.std(arr["5_lifespan"])) + ")")
115     print("Mean 25percent lifespan number of nodes: " + str(np.mean(arr["25_lifespan"])) +
        ↪  " (std: " + str(np.std(arr["25_lifespan"])) + ")")
116     print("Mean 50percent lifespan number of nodes: " + str(np.mean(arr["50_lifespan"])) +
        ↪  " (std: " + str(np.std(arr["50_lifespan"])) + ")")
117     print("Mean 75percent lifespan number of nodes: " + str(np.mean(arr["75_lifespan"])) +
        ↪  " (std: " + str(np.std(arr["75_lifespan"])) + ")")
118     print("Mean density: " + str(np.mean(arr["density"])) + " (std: " +
        ↪  str(np.std(arr["density"])) + ")")
119     print("Mean video duration: " + str(np.mean(arr["video_duration"])) + " (std: " +
        ↪  str(np.std(arr["video_duration"])) + ")")
120     print("Mean shares count: " + str(np.mean(arr["stats_shareCount"])) + " (std: " +
        ↪  str(np.std(arr["stats_shareCount"])) + ")")
121     print("Mean likes count: " + str(np.mean(arr["stats_diggCount"])) + " (std: " +
        ↪  str(np.std(arr["stats_diggCount"])) + ")")
122     print("Mean comments count: " + str(np.mean(arr["stats_commentCount"])) + " (std: " +
        ↪  str(np.std(arr["stats_commentCount"])) + ")")
123     print("Mean views count: " + str(np.mean(arr["stats_playCount"])) + " (std: " +
        ↪  str(np.std(arr["stats_playCount"])) + ")")
124     print("Mean different music count: " + str(np.mean(arr["music_id_count"])) + " (std: "
        ↪  + str(np.std(arr["music_id_count"])) + ")")
125     print("Mean likes given by the author count: " +
        ↪  str(np.mean(arr["authorStats_diggCount"])) + " (std: " +
        ↪  str(np.std(arr["authorStats_diggCount"])) + ")")
126     print("Mean author following count: " +
        ↪  str(np.mean(arr["authorStats_followingCount"])) + " (std: " +
        ↪  str(np.std(arr["authorStats_followingCount"])) + ")")
127     print("Mean likes received by the author count: " +
        ↪  str(np.mean(arr["authorStats_heartCount"])) + " (std: " +
        ↪  str(np.std(arr["authorStats_heartCount"])) + ")")
128     print("Mean author follower count: " + str(np.mean(arr["authorStats_followerCount"]))
        ↪  + " (std: " + str(np.std(arr["authorStats_followerCount"])) + ")")
129     print("Mean published videos by the author count: " +
        ↪  str(np.mean(arr["authorStats_videoCount"])) + " (std: " +
        ↪  str(np.std(arr["authorStats_videoCount"])) + ")")
130     print("*******************************************************")
131
132 flag = False
133 counter = 0
134 for elem in ALL_CHALLENGES:
135     STATS = reset_stats()
136     for challenge in elem:
137         if not flag:
138             cond = POS_CHALLENGES_COND[counter]
139         else:
140             cond = NEG_CHALLENGES_COND[counter]
141         df = pd.DataFrame()
142         graph, _, _, _, nodestats, df, _, _ =
            ↪  nx.graphCalculation(challenges.getChallenge(challenge)["name"].split(",")[0],
            ↪  lifespanCond=cond)
143         gen_stats = nx.graphStats(graph, _print=False)
144         #for node in nodestats:
145         #    df = df.append(nodestats[node], ignore_index=True)
146         df["createTime"] = pd.to_datetime(df["createTime"]) # convert to datetime
147         timedelta = df["createTime"].max() - df["createTime"].min()
148         incr = 0
149         for i in range(0,100):
150             STATS["span"][i].append((df.loc[df["createTime"] <= (df["createTime"].min() +
                ↪  (timedelta/100 * (i+1)))].shape[0])/gen_stats["nnodes"])
```

```
            mask = (df["createTime"] <= (df["createTime"].min() + (timedelta/100 * (i+1)))) & (df["createTime"] >= (df["createTime"].min() + (timedelta/100 * i)))
            STATS["n_nodes"][i].append(df.loc[mask].shape[0])
        for i in range(0,100,5):
            mask = (df["createTime"] <= (df["createTime"].min() + (timedelta/100 * (i+5)))) & (df["createTime"] >= (df["createTime"].min() + (timedelta/100 * i)))
            if not np.isnan(df.loc[mask]["stats_diggCount"].mean()):
                STATS["mlikes"][int(round(i/5))].append(df.loc[mask]["stats_diggCount"].mean())
            else:
                STATS["mlikes"][int(round(i/5))].append(0)
            if not np.isnan(df.loc[mask]["authorStats_followingCount"].sum()):
                STATS["following"][int(round(i/5))].append(df.loc[mask]["authorStats_followingCount"].sum())
            else:
                STATS["following"][int(round(i/5))].append(0)
            STATS["n_video"][int(round(i/5))].append(df.loc[mask].shape[0])
        STATS["nnodes"].append(gen_stats["nnodes"])
        STATS["nedges"].append(gen_stats["nedges"])
        STATS["mindegree"].append(gen_stats["mindegree"])
        STATS["moutdegree"].append(gen_stats["moutdegree"])
        STATS["avgclust"].append(gen_stats["avgclust"])
        STATS["density"].append(gen_stats["density"])
        STATS["lifespan"].append(timedelta.days)
        STATS["5_lifespan"].append((df.loc[df["createTime"] <= (df["createTime"].min() + (timedelta/100 * 5))].shape[0])/gen_stats["nnodes"])
        STATS["25_lifespan"].append((df.loc[df["createTime"] <= (df["createTime"].min() + (timedelta/100 * 25))].shape[0])/gen_stats["nnodes"])
        STATS["50_lifespan"].append((df.loc[df["createTime"] <= (df["createTime"].min() + (timedelta/100 * 50))].shape[0])/gen_stats["nnodes"])
        STATS["75_lifespan"].append((df.loc[df["createTime"] <= (df["createTime"].min() + (timedelta/100 * 75))].shape[0])/gen_stats["nnodes"])
        STATS["video_duration"].append(df["video_duration"].mean())
        STATS["stats_shareCount"].append(df["stats_shareCount"].mean())
        STATS["stats_diggCount"].append(df["stats_diggCount"].mean())
        STATS["stats_commentCount"].append(df["stats_commentCount"].mean())
        STATS["stats_playCount"].append(df["stats_playCount"].mean())
        STATS["music_id_count"].append(df['music_id'].value_counts().count())
        STATS["authorStats_diggCount"].append(df["authorStats_diggCount"].mean())
        STATS["authorStats_followingCount"].append(df["authorStats_followingCount"].mean())
        STATS["authorStats_heartCount"].append(df["authorStats_heartCount"].mean())
        STATS["authorStats_followerCount"].append(df["authorStats_followerCount"].mean())
        STATS["authorStats_videoCount"].append(df["authorStats_videoCount"].mean())
        counter += 1
    print_results(STATS, flag)
    flag = True
    counter = 0

intervals = pd.read_csv('dataset/intervals.csv', sep=',')

#plotpos

for i in range(len(POS_CHALLENGES)):
    int_ = intervals.loc[intervals['challenge'] == POS_CHALLENGES[i]].iloc[:].values[0]
    plt.title(POS_CHALLENGES[i]+" graph's expansion (positive)")
    plt.ylabel("mean number of following")
    plt.xlabel("% trend's lifespan")
    plt.xticks(range(0,101,5))
```

```
    plt.grid("--")
    plt.gca().set_xlim(xmin=0, xmax=100)
    plt.plot(range(0,101,5), [0]+PLOTPOSFOLLOWINGCOUNT[i], label=POS_CHALLENGES[i])
    #PLOTPOSCOUNT
    for point in literal_eval(int_[2]):
        if point != 100:
            plt.axvline(x=point,color='r', linewidth=2)
    plt.show()

#plotneg

for i in range(len(NEG_CHALLENGES)):
    int_ = intervals.loc[intervals['challenge'] == NEG_CHALLENGES[i]].iloc[:].values[0]
    plt.title(NEG_CHALLENGES[i]+" graph's expansion (negative)")
    plt.ylabel("mean number of following")
    plt.xlabel("% trend's lifespan")
    plt.xticks(range(0,101,5))
    plt.grid("--")
    plt.gca().set_xlim(xmin=0, xmax=100)
    plt.plot(range(0,101,5), [0]+PLOTNEGFOLLOWINGCOUNT[i], label=NEG_CHALLENGES[i])
    #PLOTNEGCOUNT
    for point in literal_eval(int_[2]):
        if point != 100:
            plt.axvline(x=point, color='r', linewidth=2)
    plt.show()
```

Appendice 7: "intervals_analysis.py"

```
from networkx.algorithms.components.connected import number_connected_components
from networkx.classes.function import subgraph
import pandas as pd
from ast import literal_eval
import lib.tiktok_network as ntx
import networkx as nx
import matplotlib.pyplot as plt
import numpy as np

def graphStats(graph, _print=True):
    unconnected_graph = graph.to_undirected()
    Gcc = sorted(nx.connected_components(unconnected_graph), key=len, reverse=True)
    stats = {"nnodes":graph.number_of_nodes(),
             "nedges":graph.number_of_edges(),
             "density":nx.density(graph),
             "mindegree":np.mean([x[1] for x in graph.in_degree()]),
             "degree_centrality_media": sum(list(nx.degree_centrality(graph).values())) /
             ↪  len(list(nx.degree_centrality(graph).values())),
             "eigenvector_centrality": sum(list(nx.eigenvector_centrality(graph,
             ↪  max_iter=100000).values()))/len(list(nx.eigenvector_centrality(graph,
             ↪  max_iter=100000))),
             "number_connected_components":
             ↪  number_connected_components(unconnected_graph),
             "maxnodes_connected_components":
             ↪  unconnected_graph.subgraph(Gcc[0]).number_of_nodes(),
             "pagerank": sum(list(nx.pagerank(graph,
             ↪  alpha=0.9).values()))/len(list(nx.pagerank(graph, alpha=0.9).values())),
             "closeness_centrality": sum(list(nx.closeness_centrality(graph).values())) /
             ↪  len(list(nx.closeness_centrality(graph).values())),
             "betweenness_centrality":
             ↪  sum(list(nx.betweenness_centrality(graph).values())) /
             ↪  len(list(nx.betweenness_centrality(graph).values())),
             "average_clustering":nx.average_clustering(graph),
             "radius_max_connected_components":
             ↪  nx.radius(unconnected_graph.subgraph(Gcc[0])),
             "diameter_max_connected_components":
             ↪  nx.diameter(unconnected_graph.subgraph(Gcc[0]))
    }
    return stats

dataintervals={
    'nome_challenge':[],
    'intervallo':[],
    'numero_nodi':[],
    'numero_archi':[],
    'densità':[],
    'likes_totali_int':[],
    'numero_medio_likes':[],
    'numero_medio_likes_std':[],
    'numero_medio_commenti':[],
    'numero_medio_commenti_std':[],
    'numero_medio_condivisioni':[],
    'numero_medio_condivisioni_std':[],
    'numero_medio_views':[],
    'numero_medio_views_std':[],
    'likes_ricevuti_autore':[],
    'likes_ricevuti_autore_std':[],
    'follower_autore':[],
    'follower_autore_std':[],
```

```
    'following_autore':[],
    'following_autore_std':[],
    'degree_centrality_media':[],
    'eigenvector_centrality':[],
    'numero_video':[],
    'numero_utenti_verificati':[],
    'numero_componenti_connesse':[],
    'maxnodi_componenti_connesse':[],
    'lifespan':[],
    'mindegree':[],
    'mindegree_std':[],
    'pagerank':[],
    'closeness_centrality':[],
    'betweenness_centrality':[],
    'average_clustering':[],
    'radius_max_connected_components':[],
    'diameter_max_connected_components':[]}

def reset_stats():
    return {"nnodes":[],
            "nedges":[],
            "density":[],
            "stats_diggCount_tot":[],
            "stats_diggCount":[],
            "stats_commentCount":[],
            "stats_playCount":[],
            "stats_shareCount":[],
            "authorStats_diggCount":[],
            "authorStats_followingCount":[],
            "authorStats_heartCount":[],
            "authorStats_followerCount":[],
            "degree_centrality_media":[],
            "eigenvector_centrality":[],
            "numero_video":[],
            "numero_utenti_verificati":[],
            "numero_componenti_connesse":[],
            "maxnodi_componenti_connesse":[],
            "lifespan":[],
            "mindegree":[],
            "pagerank":[],
            "closeness_centrality":[],
            "betweenness_centrality":[],
            "average_clustering":[],
            "radius_max_connected_components":[],
            "diameter_max_connected_components":[]}

def print_results(arr, _type=False):
    print("Mean number of nodes: " + str(np.mean(arr["nnodes"])))
    print("Mean number of edges: " + str(np.mean(arr["nedges"])))
    print("Mean density: " + str(np.mean(arr["density"])))
    print("Total likes count: " + str(np.sum(arr["stats_diggCount_tot"])))
    print("Mean shares count: " + str(np.mean(arr["stats_shareCount"])) + " (std: " +
    ↪  str(np.std(arr["stats_shareCount"])) + ")")
    print("Mean likes count: " + str(np.mean(arr["stats_diggCount"])) + " (std: " +
    ↪  str(np.std(arr["stats_diggCount"])) + ")")
    print("Mean comments count: " + str(np.mean(arr["stats_commentCount"])) + " (std: " +
    ↪  str(np.std(arr["stats_commentCount"])) + ")")
    print("Mean views count: " + str(np.mean(arr["stats_playCount"])) + " (std: " +
    ↪  str(np.std(arr["stats_playCount"])) + ")")
```

```
104     print("Mean likes given by the author count: " +
        ↪  str(np.mean(arr["authorStats_diggCount"])) + " (std: " +
        ↪  str(np.std(arr["authorStats_diggCount"])) + ")")
105     print("Mean author following count: " +
        ↪  str(np.mean(arr["authorStats_followingCount"])) + " (std: " +
        ↪  str(np.std(arr["authorStats_followingCount"])) + ")")
106     print("Mean likes received by the author count: " +
        ↪  str(np.mean(arr["authorStats_heartCount"])) + " (std: " +
        ↪  str(np.std(arr["authorStats_heartCount"])) + ")")
107     print("Mean author follower count: " + str(np.mean(arr["authorStats_followerCount"]))
        ↪  + " (std: " + str(np.std(arr["authorStats_followerCount"])) + ")")
108     print("Degree centrality: " + str(arr["degree_centrality_media"]))
109     print("Eigenvector centrality: " + str(arr["eigenvector_centrality"]))
110     print("Number of videos: " + str(np.count_nonzero(arr["numero_video"])))
111     print("Number of verified users: "+
        ↪  str(np.count_nonzero(arr["numero_utenti_verificati"])))
112     print("Number of connected components: " + str(arr["numero_componenti_connesse"]))
113     print("Max number of nodes in connected components: " +
        ↪  str(arr["maxnodi_componenti_connesse"]))
114     print("Lifespan: " + str(np.mean(arr["lifespan"])))
115     print("Mean indegree: " + str(np.mean(arr["mindegree"])) +  " (std: " +
        ↪  str(np.std([x[1] for x in graph.in_degree()]))+ ")")
116     print("Pagerank: "+str(np.mean(arr["pagerank"])))
117     print("Closeness Centrality: " + str(np.mean(arr["closeness_centrality"])))
118     print("Betweenness Centrality: " + str(np.mean(arr["betweenness_centrality"])))
119     print("Average Clustering: " + str(np.mean(arr["average_clustering"])))
120     print("Radius max connected components: " +
        ↪  str(np.mean(arr["radius_max_connected_components"])))
121     print("Diameter max connected components: " +
        ↪  str(np.mean(arr["diameter_max_connected_components"])))
122     print("*****************************************************")
123 numero_video = 0
124 vid = []
125 std_times = []
126 mean_times = []
127 df = pd.read_csv("dataset/intervals.csv")
128 df["points"] = df["points"].apply(literal_eval)
129 for index, row in df.iterrows():
130     intervals = row['points']
131     if intervals[0] != 0:
132         intervals.insert(0, 0)
133     if intervals[-1] != 100:
134         intervals.append(100)
135     for pairs in zip(intervals, intervals[1:]):
136         plt.figure(figsize=(15,15))
137         print(pairs)
138         graph, labels, colors, pos, _, _, df_int, lfd =
            ↪  ntx.graphCalculation(row["challenge"],colorCriteria="createTime",
            ↪  lifespanCond=None, intervals=list(pairs), remAutoLikes=True)
139         print(row["challenge"])
140         challenge=(row["challenge"])
141         nx.draw_networkx_nodes(graph,pos,node_color=colors,node_size=60)
142         nx.draw_networkx_labels(graph, pos, labels,font_size=5)
143         nx.draw_networkx_edges(graph, pos, arrows=True, arrowsize=3, arrowstyle="-|>",
            ↪  alpha=0.5) #edge_color='lightgrey'
144         ax= plt.gca()
145         try:
146             ax.collections[0].set_edgecolor("#000000")
147         except:
148             continue
149         #plt.show()
```

```
150         gen_stats = graphStats(graph, _print=False)
151         df_int["createTime"] = pd.to_datetime(df_int["createTime"]) # convert to datetime
152         df_int = df_int.sort_values(by='createTime')
153         timedelta = df_int["createTime"].max() - df_int["createTime"].min()
154         STATS = reset_stats()
155         STATS["nnodes"].append(gen_stats["nnodes"])
156         STATS["nedges"].append(gen_stats["nedges"])
157         STATS["density"].append(gen_stats["density"])
158         STATS["degree_centrality_media"].append(gen_stats["degree_centrality_media"])
159         STATS["eigenvector_centrality"].append(gen_stats["eigenvector_centrality"])
160         STATS["stats_diggCount_tot"].append(df_int["stats_diggCount"])
161         STATS["stats_diggCount"].append(df_int["stats_diggCount"])
162         STATS["stats_commentCount"].append(df_int["stats_commentCount"])
163         STATS["stats_playCount"].append(df_int["stats_playCount"])
164         STATS["stats_shareCount"].append(df_int["stats_shareCount"])
165         STATS["authorStats_diggCount"].append(df_int["authorStats_diggCount"])
166         STATS["authorStats_followingCount"].append(df_int["authorStats_followingCount"])
167         STATS["authorStats_followerCount"].append(df_int["authorStats_followerCount"])
168         STATS["authorStats_heartCount"].append(df_int["authorStats_heartCount"])
169         STATS["numero_video"].append(df_int["id"])
170         STATS["numero_utenti_verificati"].append(np.where(df_int["author_verified"])[0])
171         STATS["numero_componenti_connesse"].append(
            ↪ gen_stats["number_connected_components"])
172         STATS["maxnodi_componenti_connesse"].append(
            ↪ gen_stats["maxnodes_connected_components"])
173         STATS["lifespan"].append(timedelta.days)
174         STATS["mindegree"].append(gen_stats["mindegree"])
175         STATS["pagerank"].append(gen_stats["pagerank"])
176         STATS["closeness_centrality"].append(gen_stats["closeness_centrality"])
177         STATS["betweenness_centrality"].append(gen_stats["betweenness_centrality"])
178         STATS["average_clustering"].append(gen_stats["average_clustering"])
179         STATS["radius_max_connected_components"].append(
            ↪ gen_stats["radius_max_connected_components"])
180         STATS["diameter_max_connected_components"].append(
            ↪ gen_stats["diameter_max_connected_components"])
181         dataintervals["nome_challenge"].append(challenge)
182         dataintervals["intervallo"].append(pairs)
183         dataintervals["numero_nodi"].append(gen_stats["nnodes"])
184         dataintervals["numero_archi"].append(gen_stats["nedges"])
185         dataintervals["densità"].append(gen_stats["density"])
186         dataintervals["degree_centrality_media"].append(
            ↪ gen_stats["degree_centrality_media"])
187         dataintervals["eigenvector_centrality"].append(
            ↪ gen_stats["eigenvector_centrality"])
188         dataintervals["likes_totali_int"].append(df_int["stats_diggCount"].sum())
189         dataintervals["numero_medio_likes"].append(df_int["stats_diggCount"].mean())
190         dataintervals["numero_medio_likes_std"].append(np.std(df_int["stats_diggCount"]))
191         dataintervals["numero_medio_commenti"].append(df_int["stats_commentCount"].mean())
192         dataintervals["numero_medio_commenti_std"].append(
            ↪ np.std(df_int["stats_commentCount"]))
193         dataintervals["numero_medio_condivisioni"].append(
            ↪ df_int["stats_shareCount"].mean())
194         dataintervals["numero_medio_condivisioni_std"].append(
            ↪ np.std(df_int["stats_shareCount"]))
195         dataintervals["numero_medio_views"].append(df_int["stats_playCount"].mean())
196         dataintervals["numero_medio_views_std"].append(np.std(df_int["stats_playCount"]))
197         dataintervals["likes_ricevuti_autore"].append(
            ↪ df_int["authorStats_heartCount"].mean())
198         dataintervals["likes_ricevuti_autore_std"].append(
            ↪ np.std(df_int["authorStats_heartCount"]))
```

```
199         dataintervals["follower_autore"].append(
            ↪ df_int["authorStats_followerCount"].mean())
200         dataintervals["follower_autore_std"].append(
            ↪ np.std(df_int["authorStats_followerCount"]))
201         dataintervals["following_autore"].append(
            ↪ df_int["authorStats_followingCount"].mean())
202         dataintervals["following_autore_std"].append(
            ↪ np.std(df_int["authorStats_followingCount"]))
203         dataintervals["numero_video"].append(df_int["id"].count())
204         dataintervals["numero_utenti_verificati"].append(
            ↪ np.count_nonzero(np.where(df_int["author_verified"])[0]))
205         dataintervals["numero_componenti_connesse"].append(
            ↪ gen_stats["number_connected_components"])
206         dataintervals["maxnodi_componenti_connesse"].append(
            ↪ gen_stats["maxnodes_connected_components"])
207         dataintervals["lifespan"].append(lfd)
208         dataintervals["mindegree"].append(gen_stats["mindegree"])
209         dataintervals["mindegree_std"].append(str(np.std([x[1] for x in
            ↪ graph.in_degree()])))
210         dataintervals["pagerank"].append(gen_stats["pagerank"])
211         dataintervals["closeness_centrality"].append(gen_stats["closeness_centrality"])
212         dataintervals["betweenness_centrality"].append(
            ↪ gen_stats["betweenness_centrality"])
213         dataintervals["average_clustering"].append( gen_stats["average_clustering"])
214         dataintervals["radius_max_connected_components"].append(
            ↪ gen_stats["radius_max_connected_components"])
215         dataintervals["diameter_max_connected_components"].append(
            ↪ gen_stats["diameter_max_connected_components"])
216         if pairs[0] == 0:
217             numero_video = df_int["id"].count()
218             vid.append(numero_video)
219             #times.append(0)
220         else:
221             numero_video = df_int["id"].count() - numero_video
222             vid.append(numero_video)
223             numero_video = df_int["id"].count()
224             #times.append(df_int['createTime'] - prev_time)
225
226         times = []
227         prev_time = 0
228         fl = True
229         for index1,row1 in df_int.iterrows():
230             if fl:
231                 times.append(0)
232             else:
233                 times.append((row1["createTime"]-prev_time)/ pd.Timedelta('1 hour'))
234             prev_time = row1["createTime"]
235             fl = False
236         #print(times)
237         mean_times.append(np.mean(times))
238         std_times.append(np.std(times))
239         print_results(STATS)
240 dfdataintervals=pd.DataFrame.from_dict(dataintervals, orient='index')
241 dfdataintervals=dfdataintervals.transpose()
242 dfdataintervals["diff_num_video_interv_prec"] = vid
243 dfdataintervals["hours_between_mean"] = mean_times
244 dfdataintervals["hours_between_std"] = std_times
245 dfdataintervals.to_csv("dataset/dataintervals.csv", sep=';', index=False)
```

Appendice 8: "lifespan_analysis.py"

```
import numpy as np
from scipy import interpolate
import matplotlib.pyplot as plt
import pandas as pd
import lib.tiktok_network as nx
import lib.challenges as challenges

p_challenges_list = [
    "bussitchallenge",
    "copinesdancechallenge",
    "emojichallenge",
    "colpiditesta",
    "boredinthehouse",
    "ITookANap",
    "plankchallenge"
]
n_challenges_list = [
    "silhouettechallenge",
    "bugsbunnychallenge",
    "strippatiktok",
    "firewroks",
    "fightchallenge",
    "sugarbaby",
    "updownchallenge"
]

perc_to_sum = 5
def split_in_perc(video_published):
    total_days = len(video_published)
    video_published['% lifespan'] = video_published['life_day'].apply(lambda day:
    ↪  ((day+1)*100)/total_days)
    video_published_d = {}
    for row in video_published.itertuples():
        video_count = row[2]
        lifespan_perc = row[3]
        video_published_d[lifespan_perc] = video_count
    video_published_splitted = {}
    video_published_splitted[0] = 0
    current_index = perc_to_sum
    _sum = 0
    for perc in video_published_d:
        if perc < current_index:
            try:
                video_published_splitted[current_index] += video_published_d[perc]
            except:
                video_published_splitted[current_index] = video_published_d[perc]
        else:
            try:
                video_published_splitted[current_index] += video_published_d[perc]
            except:
                video_published_splitted[current_index] = video_published_d[perc]
            current_index += perc_to_sum
    df = pd.DataFrame.from_dict(video_published_splitted, orient='index',
    ↪  columns=['video_published'])
    return df

p_changes_by_day = {}
for p in p_challenges_list:
    current = nx.graphCalculation(p)[-3]
```

```
58      current['createTime'] =  pd.to_datetime(current['createTime'], format='%Y%m%d
        ↪  %H:%M:%S')
59      video_published = current.set_index('createTime').groupby(
        ↪  pd.Grouper(freq='D'))["author_uniqueId"].count().to_frame()
60      video_published.reset_index(inplace=True)
61      video_published.columns = ['date', 'video_count']
62      video_published.drop('date', axis=1, inplace=True)
63      video_published.reset_index(inplace=True)
64      video_published.columns = ['life_day', 'video_count']
65      p_changes_by_day[p] = split_in_perc(video_published)
66
67  n_changes_by_day = {}
68  for p in n_challenges_list:
69      current = nx.graphCalculation(p)[-3]
70      current['createTime'] =  pd.to_datetime(current['createTime'], format='%Y%m%d
        ↪  %H:%M:%S')
71      video_published = current.set_index('createTime').groupby(
        ↪  pd.Grouper(freq='D'))["author_uniqueId"].count().to_frame()
72      video_published.reset_index(inplace=True)
73      video_published.columns = ['date', 'video_count']
74      video_published.drop('date', axis=1, inplace=True)
75      video_published.reset_index(inplace=True)
76      video_published.columns = ['life_day', 'video_count']
77      n_changes_by_day[p] = split_in_perc(video_published)
78
79  intervals = {}
80  intervals["challenge"] = []
81  intervals["n_intervals"] = []
82  intervals["points"] = []
83  intervals["diff_videos_intervals"] = []
84
85  for challenge in p_changes_by_day:
86      intervals["challenge"].append(challenge)
87      current = p_changes_by_day[challenge]
88      s =100000000
89      if challenge == 'ITookANap':
90          s = 410
91      function = interpolate.UnivariateSpline(current.index,current["video_published"], k=4,
        ↪  s=s)
92      first_derivate = function.derivative(n=1)
93      inf_points = first_derivate.roots()
94      intervals["n_intervals"].append(len(inf_points))
95      points = [5 * round(p/5) for p in inf_points]
96      intervals["points"].append(points)
97      punti_andamenti = [0]
98      punti_andamenti.extend(inf_points)
99      punti_andamenti.extend([100])
100     valori_punti = [function(p) for p in punti_andamenti]
101     differenze = [j-i for i, j in zip(valori_punti[:-1], valori_punti[1:])]
102     intervals["diff_videos_intervals"].append(differenze)
103     plt.title(challenge, fontsize=20)
104     xnew = np.arange(0, 105, perc_to_sum)
105     ynew = function(xnew)
106     plt.plot(xnew, ynew)
107     for inf in inf_points:
108         plt.axvline(x=inf)
109     plt.xlabel("% of lifespan", fontsize=16)
110     plt.ylabel("Number of nodes", fontsize=16)
111     plt.show()
112
113 for challenge in n_changes_by_day:
```

```
114         intervals["challenge"].append(challenge)
115         current = n_changes_by_day[challenge]
116         s =100000000
117         if challenge == 'strippatiktok':
118             s = 200
119         if challenge == 'updownchallenge':
120             s = 11700
121         function = interpolate.UnivariateSpline(current.index,current["video_published"], k=4,
            ↪  s=s)
122         first_derivate = function.derivative(n=1)
123         inf_points = first_derivate.roots()
124         intervals["n_intervals"].append(len(inf_points))
125         points = [5 * round(p/5) for p in inf_points]
126         intervals["points"].append(points)
127         punti_andamenti = [0]
128         punti_andamenti.extend(inf_points)
129         punti_andamenti.extend([100])
130         valori_punti = [function(p) for p in punti_andamenti]
131         differenze = [j-i for i, j in zip(valori_punti[:-1], valori_punti[1:])]
132         intervals["diff_videos_intervals"].append(differenze)
133         plt.title(challenge, fontsize=20)
134         xnew = np.arange(0, 105, perc_to_sum)
135         ynew = function(xnew)
136         plt.plot(xnew, ynew)
137         for inf in inf_points:
138             plt.axvline(x=inf)
139         plt.xlabel("% of lifespan", fontsize=16)
140         plt.ylabel("Number of nodes", fontsize=16)
141         plt.show()
142 
143 intervals_d = pd.DataFrame.from_dict(intervals)
144 intervals_d.to_csv('dataset/intervals.csv', index=False)
```

Appendice 9: "clustering.py"

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

intervalli = pd.read_csv("dataset/dataintervals.csv", sep=";", header=0)
intervalli["perc_inizio"] = intervalli["intervallo"].apply(lambda x: tuple(map(int,
↪  x.replace("(", "").replace(")","").split(', ')))[0])
intervalli["perc_fine"] = intervalli["intervallo"].apply(lambda x: tuple(map(int,
↪  x.replace("(", "").replace(")","").split(', ')))[1])
intervalli.drop("intervallo", axis=1, inplace=True)
intervalli.to_csv('dataintervals.csv', sep=";", index=False)
nomi = intervalli['nome_challenge']
intervalli.drop("nome_challenge", axis=1, inplace=True)
intervalli

intervalli.drop('numero_nodi', axis=1, inplace=True, errors='ignore')
intervalli.drop('degree_centrality_media', axis=1, inplace=True, errors='ignore')
intervalli.drop('maxnodi_componenti_connesse', axis=1, inplace=True, errors='ignore')
intervalli.drop('follower_autore_std', axis=1, inplace=True, errors='ignore')
intervalli.drop('numero_medio_likes_std', axis=1, inplace=True, errors='ignore')
intervalli.drop('hours_between_std', axis=1, inplace=True, errors='ignore')
intervalli.drop('numero_medio_commenti_std', axis=1, inplace=True, errors='ignore')
intervalli.drop('numero_medio_views_std', axis=1, inplace=True, errors='ignore')
intervalli.drop('likes_ricevuti_autore_std', axis=1, inplace=True, errors='ignore')
intervalli.drop('numero_medio_condivisioni_std', axis=1, inplace=True, errors='ignore')
intervalli.drop('following_autore_std', axis=1, inplace=True, errors='ignore')
intervalli.drop('likes_ricevuti_autore', axis=1, inplace=True, errors='ignore')
intervalli.drop('mindegree_std', axis=1, inplace=True, errors='ignore')
intervalli.drop('perc_fine', axis=1, inplace=True, errors='ignore')
intervalli.drop('numero_archi', axis=1, inplace=True, errors='ignore')
intervalli.drop('likes_totali_int', axis=1, inplace=True, errors='ignore')
intervalli.drop('numero_medio_commenti', axis=1, inplace=True, errors='ignore')
intervalli.drop('numero_medio_condivisioni', axis=1, inplace=True, errors='ignore')
intervalli.drop('numero_medio_views', axis=1, inplace=True, errors='ignore')
intervalli.drop('mindegree', axis=1, inplace=True, errors='ignore')
intervalli.drop('diff_num_video_interv_prec', axis=1, inplace=True, errors='ignore')
intervalli.drop('numero_componenti_connesse', axis=1, inplace=True, errors='ignore')
intervalli.drop('numero_utenti_verificati', axis=1, inplace=True, errors='ignore')
intervalli.drop('densità', axis=1, inplace=True, errors='ignore')
intervalli.drop('betweenness_centrality', axis=1, inplace=True, errors='ignore')
intervalli.drop('average_clustering', axis=1, inplace=True, errors='ignore')

f, ax = plt.subplots(figsize=(15, 11))
corr = intervalli.corr()
sns.heatmap(corr, vmax=1, vmin=-1, center=0, square=True, linewidths=.5)

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA
from sklearn import preprocessing

x = intervalli.values #returns a numpy array
min_max_scaler = preprocessing.MinMaxScaler()
x_scaled = min_max_scaler.fit_transform(x)
intervalli = pd.DataFrame(x_scaled)

wcss = []
```

```
for i in range(1,11):
    model = KMeans(n_clusters = i, init = "k-means++")
    model.fit(intervalli)
    wcss.append(model.inertia_)
plt.figure(figsize=(10,10))
plt.plot(range(1,11), wcss)
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()

pca = PCA(2)
data = pca.fit_transform(intervalli)
plt.figure(figsize=(10,10))
var = np.round(pca.explained_variance_ratio_*100, decimals = 1)
lbls = [str(x) for x in range(1,len(var)+1)]
plt.bar(x=range(1,len(var)+1), height = var, tick_label = lbls)
plt.show()

#centers = np.array(model2.cluster_centers_)
model = KMeans(n_clusters = 6, init = "k-means++")
label = model.fit_predict(data)
fig = plt.figure(figsize=(10,10))
ax = fig.add_subplot()
uniq = np.unique(label)
for i in uniq:
    plt.scatter(data[label == i , 0] , data[label == i , 1] , label = i)
#This is done to find the centroid for each clusters.
for i in range(len(data)):
  ax.text(data[i][0], data[i][1], nomi[i], fontsize=7)
print(label)
#fig.tight_layout()
plt.legend()
plt.show()
```

Appendice 10: "utils.py"

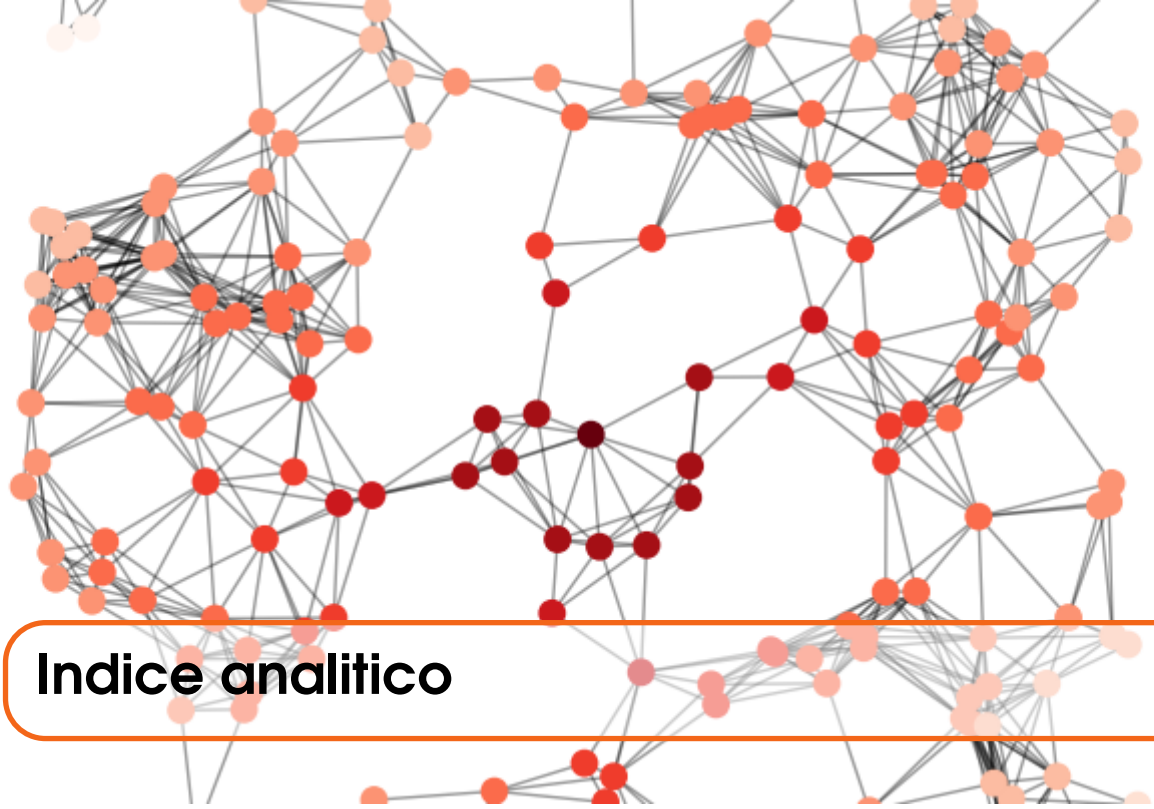
```python
from TikTokApi import TikTokApi
import pandas as pd

# Print iterations progress
def printProgressBar (iteration, total, prefix = '', suffix = '', decimals = 1, length = 100, fill = '', printEnd = "\r"):
    #print("\033[A                             \033[A")
    percent = ("{0:." + str(decimals) + "f}").format(100 * (iteration / float(total)))
    filledLength = int(length * iteration // total)
    bar = fill * filledLength + '-' * (length - filledLength)
    print(f'\r{prefix} |{bar}| {percent}\% {suffix}', end = printEnd)

def datasetHelper(tiktoks):
    rows = []
    for tiktok in tiktoks:
        _row = {}
        for key in tiktok:
            elem = tiktok.get(key)
            if isinstance(elem, dict) or isinstance(elem, list):
                if isinstance(elem, list):
                    elem = elem[0]
                for elem_k in elem:
                    _row[key + "_" + elem_k] = str(elem.get(elem_k)).replace(";", "")
            else:
                _row[key] = str(elem).replace(";", "")
        rows.append(_row)
    return pd.json_normalize(rows)

def checkAvLikedPerc(api, dataset, _iter = 0): # prints tiktok liked list availability
    counter = 0
    ct = 0
    df = pd.read_csv("./dataset/dataset_"+dataset+".csv", sep=";")
    den = df.shape[0]
    if _iter > 0: # _iter != default
        den = _iter
    for index, row in df.iterrows():
        counter += len(api.userLiked(row['author_id'],row['author_secUid'],count=1))
        ct += 1
        if ct == _iter and den == _iter:
            break # stop loop if target
    print("\n\nAvailability: " + str(round((counter/den)*100, 2)) + "\% (" + str(counter) + "/" + str(den) + ")")
```

Bibliografia

Riferimenti

- Juan Carlos Medina Serrano, Orestis Papakyriakopoulos, and Simon Hegelich. 2020. Dancing to the Partisan Beat: A First Analysis of Political Communication on TikTok. In 12th ACM Conference on Web Science (WebSci '20). Association for Computing Machinery, New York, NY, USA, 257–266.
- Alim, S. (2017). Cyberbullying in the world of teenagers and social media: A literature review. In Information Resources Management Association, Gaming and technology addiction: Breakthroughs in research and practice (pp. 520–552). Information Science Reference/IGI Global.
- Allcott, Hunt, and Matthew Gentzkow. 2017. "Social Media and Fake News in the 2016 Election." Journal of Economic Perspectives, 31 (2): 211-36.
- Li Y, Guan M, Hammond P, Berrey LE. Communicating COVID-19 information on TikTok: a content analysis of TikTok videos from official accounts featured in the COVID-19 information hub [published online ahead of print, 2021 Mar 1]. Health Educ Res. 2021.
- Gabriel Weimann & Natalie Masri (2020) Research Note: Spreading Hate on TikTok, Studies in Conflict & Terrorism, DOI: 10.1080/1057610X.2020.1780027.
- Omar, B. & Dequan, W. (2020). Watch, Share or Create: The Influence of Personality Traits and User Motivation on TikTok Mobile Video Usage. International Association of Online Engineering. Retrieved June 27, 2021 from https://www.learntechlib.org/p/216454/.
- Basch, Corey H., Grace C. Hillyer, and Christie Jaime. "COVID-19 on TikTok: harnessing an emerging social media platform to convey important public health messages." International journal of adolescent medicine and health 1.ahead-of-print (2020).
- Hayes, Clare, et al. "“Making Every Second Count”: Utilizing TikTok and Systems Thinking to Facilitate Scientific Public Engagement and Contextualization of Chemistry at Home." (2020): 3858-3866.
- Masciantonio, Alexandra, et al. "Don't put all social network sites in one basket: Facebook, Instagram, Twitter, TikTok, and their relations with well-being during the COVID-19 pandemic." PloS one 16.3 (2021): e0248384.

- Herrick, Shannon SC, Laura Hallward, and Lindsay R. Duncan. "“This is just how I cope”: An inductive thematic analysis of eating disorder recovery content created and shared on TikTok using EDrecovery." International Journal of Eating Disorders (2020).
- Zhao, Zhengwei. "Analysis on the “Douyin (Tiktok) Mania” Phenomenon Based on Recommendation Algorithms." E3S Web of Conferences. Vol. 235. EDP Sciences, 2021.
- Ravikumar, Vaishali, et al. "Is TikTok the New Instagram? Analysis of Plastic Surgeons on Social Media." Plastic and Reconstructive Surgery 147.5 (2021): 920e-922e.
- Zhu, Yumei. "The Expectation of TikTok in International Media: A Critical Discourse Analysis." Open Journal of Social Sciences 8.12 (2020): 136-148.
- Olivares-Garcia, Francisco J., and Maria Ines Mendez Majuelos. "Analysis of the main trends published on TikTok during the quarantine period by COVID-19." Revista Espanola De Comunicacion En Salud (2020): S243-S252.
- Aisa, Aufia, and Mega Kirana Dewi. "Z Genereations Perspective: Analysis of Islamic Learning through Tiktok Social Media." SCHOOLAR: Social and Literature Study in Education 1.1 (2021): 22-25.
- Gray, Joanne. "The geopolitics of ‘platforms’: the TikTok challenge." Internet Policy Review (2021).
- Klug, Daniel. "" It took me almost 30 minutes to practice this". Performance and Production Practices in Dance Challenge Videos on TikTok." arXiv preprint arXiv:2008.13040 (2020).
- Barbotti, Ilaria. TikTok Marketing: Video virali e hashtag challenge: come fare business con la Generazione Z. HOEPLI EDITORE, 2020.
- Atherton, Rachel Rose. "The ‘Nutmeg Challenge’: a dangerous social media trend." Archives of disease in childhood 106.5 (2021): 517-518.
- Ahlse, Johannes, Felix Nilsson, and Nina Sandström. "It’s time to TikTok: Exploring Generation Z’s motivations to participate in Challenges." (2020).
- Henneman, Todd. "Beyond Lip-Synching: Experimenting with TikTok Storytelling." Teaching Journalism & Mass Communication 10.2 (2020): 1-14.
- Knowledge, I. "The TikTok Strategy: Using AI Platforms to take over the world." Retrieved from INSEAD KNOWLEDGE: https://knowledge. insead. edu/entrepreneurship/the-tiktok-strategy-using-ai-platforms-to-take-over-theworld-11776 (2019).



Indice analitico

P

R

S

T

RINGRAZIAMENTI

Dedico questo lavoro ai miei genitori e al mio compagno, che mi hanno sempre sostenuta durante uno dei percorsi più formativi ed importanti della mia vita.
Ringrazio il Prof. Domenico Ursino, per la grande disponibilità dimostrata e l'indiscutibile umanità che lo contraddistingue. Grazie al suo intero team: Enrico, Gianluca e Luca. Ringrazio Lorenzo Giuliani, che ha condiviso con me il lavoro relativo al tirocinio e alla presentazione della tesi, fornendomi grande supporto.
Un grazie va ai miei compagni di corso, per aver reso tutto estremamente più leggero e divertente. Grazie a tutti gli amici e alle amiche più care.

Ancona, 16 luglio 2021