Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE
Corso di Laurea in Informatica per il Management

Implementazione di un algoritmo di clustering

Relatore:
Chiar.mo Prof.
Lorenzo Donatiello

Presentata da:
Gabriel Riccardo Alsina

Sessione III
Anno Accademico 2021/2022

Indice

Abstract

Negli ultimi anni, la crescente quantità di dati generati dagli utenti ha portato allo sviluppo di sofisticati algoritmi di profilazione e raccomandazione. Questi algoritmi mirano a comprendere il comportamento e le preferenze degli utenti al fine di personalizzare l'esperienza utente. L'obiettivo principale di questa tesi è presentare un algoritmo innovativo per la memorizzazione delle preferenze utente basato sui numeri primi. Dopo aver analizzato in dettaglio le caratteristiche dei processi di profilazione in un contesto di economia digitale, si presentano i principali algoritmi di raccomandazione e raccolta dati, facendo enfasi su potenziali vantaggi e svantaggi di queste tecniche. In particolare, le performance dell'algoritmo proposto vengono confrontate con le performance della matrice delle preferenze e l'ArrayList utente. Il confronto viene eseguito sia a livello teorico (costo computazionale) che a livello pratico (quantità di memoria effettivamente utilizzata).
Da questo studio è emerso che l'algoritmo basato sui numeri primi abbia ottime performance rispetto gli altri algoritmi studiati per quanto riguarda lo spazio di archiviazione dei dati. Però presenta prestazioni peggiori per quanto riguarda le performance sulla velocità. C'è anche da dire che queste performance sono soprattutto dovute al linguaggio di programmazione usato e alla macchina sulla quale sono stati svolti i test. In altri linguaggi e con altri elaboratori più potenti le performance potrebbero essere migliori.

Capitolo 0

Introduzione

L'economia di oggi, basata su una sovrabbondanza di informazioni, è sempre più guidata dall'attenzione[8]. La merce del Mondo attuale (rara, scarsa e quindi preziosa) è diventata l'attenzione degli utenti. Che esista un'economia dell'attenzione è chiaro a tutti coloro che si occupano di marketing, spesso infatti si parla anche di *attention marketing* o marketing dell'attenzione. Hendricks e Vestergaard[11] spiegano come l'attenzione sia estremamente preziosa per chiunque abbia qualcosa da vendere, sia esso un prodotto, il proprio brand o pubblicità. Nell'era dell'informazione e della proliferazione di piattaforme online, ecco che i dati sono strettamente collegati all'attenzione. Permettono infatti di capire il tempo di permanenza su una story su Instagram, per quanti secondi una persona rimane a guardare un video su TikTok o ascoltare una canzone su Spotify.
In questo contesto, la personalizzazione dell'informazione è passata da essere un vantaggio competitivo a essere un *must-have* di vitale importanza per aziende e media. La personalizzazione dell'informazione è un modo per adattare i contenuti o le offerte a seconda delle preferenze e dei bisogni di ciascun utente. Si basa sull'analisi dei dati raccolti sulle abitudini e i gusti degli utenti, per offrire loro esperienze più rilevanti e coinvolgenti. La personalizzazione è quindi possibile grazie alla profilazione utente e al recupero efficiente delle informazioni, che viene eseguito per personalizzare uno scenario, mantenendo i profili utente separati per singolo utente. La personalizzazione delle informazioni ha portato i sistemi di raccomandazione a un livello molto alto. Con la personalizzazione, questi sistemi possono generare consigli specifici per l'utente in modo sempre più accurato[7]. La potenza e l'importanza di questi algoritmi è ormai un fatto conosciuto alla massa, non solo a tecnici e appassionati. Basti pensare che solo qualche anno fa Netflix ha offerto $1.000.000 per chi potesse aumentare del 10% la precisione del loro sistema. Negli ultimi anni, la crescente quantità di dati generati dagli utenti ha portato allo sviluppo di sofisticati algoritmi di profilazione e raccomandazione degli utenti[2]. Questi algoritmi mirano a comprendere il comportamento e le preferenze degli utenti al fine di personalizzare l'esperienza dell'utente e migliorare l'efficacia della pubblicità online. Secondo un rapporto di Accenture del 2018[4], dopo aver intervistato

8.000 consumatori tra Nord America ed Europa è risultato che il 91% dei consumatori è più propenso ad acquistare dai *brand* che li riconoscono, li ricordano e offrono loro offerte e raccomandazioni rilevanti. Tuttavia, la raccolta, l'elaborazione e l'analisi di grandi set di dati sollevano importanti preoccupazioni etiche e sulla privacy. Secondo un report di McKinsey [3], il 75% delle nostre scelte su Netflix è dovuto alle raccomandazioni. È proprio questo, in realtà, che fa sorgere alcuni dubbi etici e legati alla privacy. Innanzitutto, come spiegato in precedenza, questi algoritmi sono studiati con l'obiettivo di mantenere la clientela, catturandone l'interesse e creando una sorta di "dipendenza". Nel 2017 durante la presentazione dei risultati agli investitori, Reed Hastings, il CEO di Netflix, ha dichiarato che "il loro competitor principale è solo il bisogno umano di chiudere gli occhi per un terzo della giornata". La questione etica si pone anche quando si pensa al tempo che molti adolescenti passano su social a causa di un continuo flusso di raccomandazioni perfettamente rispondenti ai loro gusti. Inoltre, i contenuti offerti all'utente possono contribuire alla polarizzazione ideologica. Questo è dovuto al cosiddetto effetto *camera dell'eco* o echo-chamber, ovvero la creazione di contesti e condizioni che, sui media, portano a uno stato di isolamento ideologico degli individui dovuto alla natura confermativa dei nuovi contenuti proposti.
Questa tesi mira a presentare un nuovo algoritmo per clusterizzare i dati sulle preferenze degli utenti in modo innovativo. Obiettivo è anche fornire una panoramica completa dello stato dell'arte di questo campo, in particolare il capitolo 1 introduce il contesto attuale dei social network e social media; il capitolo 2 analizza in dettaglio la profilazione degli utenti; il capitolo 3 descrive i principali algoritmi di raccomandazione e raccolta dati, presentando i potenziali vantaggi e svantaggi di queste tecniche sia per gli utenti che per le imprese e fornendo raccomandazioni per la ricerca futura in questo settore. Il capitolo 4, dunque, presenta un nuovo algoritmo di clustering basato sui numeri primi, confrontandolo con quelli presentati in precedenza. Il confronto avviene sia a livello teorico che a livello pratico, mostrando il codice Java e analizzando i risultati ottenuti dagli esperimenti. Infine, il capitolo 5 evidenzia le principali sfide tecniche e le considerazioni etiche coinvolte in questo campo e come vengono gestite dalla comunità europea.

Capitolo 1

Reti sociali

1.1 Definizioni

Con il termine rete sociale, o social network, si identifica un gruppo di individui connessi tra di loro attraverso legami sociali. Questi legami possono essere di diversi tipi, come ad esempio: vincoli familiari, rapporti di lavoro o conoscenza di tipo superficiale. Questo gruppo in qualche modo connesso rappresenta quindi una rete ("*network*" appunto) di persone che sono unite da un "interesse" di varia natura che li accomuna e vogliono, per questo, costruire una comunità (o *community*).
I servizi di social networking, comunemente chiamati social media, sono gli strumenti che permettono collegamenti sociali e comunicazioni interattive tra gli utenti scambiandosi varie tipologie di contenuti: testuali, audio e video.
Il social media rappresenta quindi lo strumento (l'app o il sito web) mentre il social network è rappresentato dalle persone che utilizzano il social media per creare la propria rete di comunità online. Queste reti sociali sono spesso rappresentate seguendo la teoria dei grafi, dove i nodi sono gli utenti e i collegamenti rappresentano le relazioni sociali, che di fatto possono essere sia esplicite, come parentela e compagni di classe, sia implicite, ad esempio amicizia e interesse comune. Per una grande rete ben collegata, la maggior parte dei nodi può raggiungere ogni altro nodo attraverso un piccolo numero di collegamenti. L'idea di sei gradi di separazione suggerisce che, in media, ogni due persone sono collegate da sei salti.

1.2 Funzionalità dei social network e social media

I social media hanno alcuni o tutti questi sette blocchi funzionali: *identità, conversazioni, condivisione, presenza, relazioni, reputazione* e *gruppi*. Diverse forme di social media hanno diversi punti di messa a fuoco. Ad esempio, i progetti collaborativi come Wikipedia si preoccupano principalmente della condivisione e della reputazione, invece

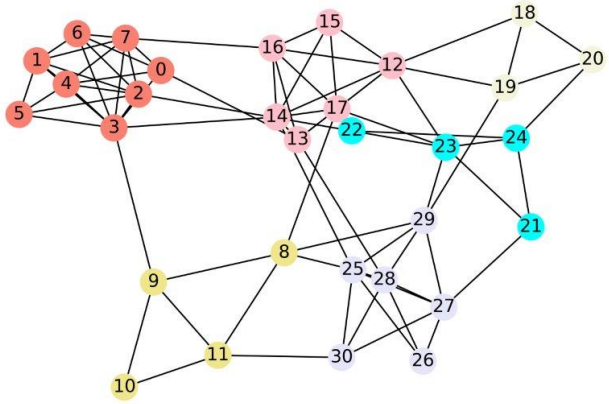


Figura 1.1: Rappresentazione di un social network tramite teoria dei grafi

nel mondo dei gioco virtuali, l'identità, la presenza, la reputazione e i gruppi sono i più importanti.[13] I social media consentono alle persone di partecipare alle attività online e abbattono la barriera per gli utenti online di condividere e consumare informazioni in qualsiasi luogo e in qualsiasi momento. Gli utenti possono essere sia consumatori passivi di contenuti che produttori attivi di contenuti e generare dati a una velocità senza precedenti. I social media sono diversi dai media tradizionali, come giornali, libri e televisione, in quanto quasi chiunque può pubblicare e accedere alle informazioni in modo economico.
Questa facilità nel condividere informazioni fa si che i social media devono immagazzinare ed elaborare moli di dati al secondo, i dati dei social media sono grandi, rumorosi, incompleti, altamente destrutturati e legati alle relazioni sociali. Queste proprietà uniche dei dati dei social media suggeriscono che l'applicazione ingenua delle tecniche esistenti può fallire o portare a una comprensione inappropriata dei dati.
Progressi nella tecnologia dell'informazione e della comunicazione hanno portato all'evidente necessità di disporre di sistemi informativi personalizzati, il cui obiettivo è adattare la funzionalità di scambio di informazioni all'interesse specifico dei loro utenti. La ricerca sulla personalizzazione degli utenti è un campo di studio attuale che si è disperso in vari domini come l'intelligenza artificiale, il data mining e la scienza dell'informazione, una delle applicazioni degne di nota della personalizzazione dell'utente è il sistema di raccomandazione.

Capitolo 2

Profilazione utente

La profilazione degli utenti è un processo che consiste nella raccolta e nell'analisi dei dati personali degli individui, al fine di comprenderne le preferenze, i comportamenti e i bisogni. Questi dati possono essere utilizzati per personalizzare l'esperienza degli utenti su diversi servizi online. La raccolta dei dati può avvenire in modo esplicito come un form di registrazione oppure in modo implicito con il social media mining. Alla fine di questo capitolo parliamo di cosa si può ottenere da questa profilazione, ovvero il news Feed.

2.1 Come ottenere i dati

Kanoje[5] ha descritto la profilazione dell'utente come un mezzo per determinare i dati di interesse dell'utente che si basa sulla conoscenza dell'utente e sul recupero accurato della soddisfazione dell'utente da parte del sistema. I sistemi precedenti erano più preoccupati di ottenere i dati direttamente dagli utenti, ovvero il sistema chiedeva esplicitamente agli utenti i dati necessari. Ma questo metodo non è considerato efficiente in quanto l'utente non è mai interessato a fornire direttamente l'input, quindi la ricerca di oggi è più focalizzata sulla profilazione dei dati dell'utente implicitamente sulla base di alcune azioni eseguite dall'utente, può anche essere definita *profilazione comportamentale dell'utente*. Sono state condotte molte ricerche su questo e possiamo identificare due approcci principali alla profilazione degli utenti:

- **Profilazione esplicita dell'utente**: La profilazione esplicita, anche detta statica, è un processo di analisi delle caratteristiche statiche e prevedibili degli utenti. In questo approccio il comportamento degli utenti è previsto analizzando i dati disponibili dell'utente. Questi dati provengono solitamente dalla compilazione di moduli online o da sondaggi. Ci sono alcuni problemi quando si dipende solo dall'uso della profilazione esplicita poiché gli utenti non sono interessati a rivelare le proprie informazioni a nessuno poiché sono preoccupati per la loro privacy e

il processo di compilazione del modulo potrebbe essere noioso tanto che l'utente potrebbe cercare di evitarlo. Quindi l'accuratezza dell'utilizzo di questo tipo di profilazione diminuisce in base al periodo di tempo di compilazione richiesto.

- **Profilazione implicita dell'utente**: La profilazione implicita, anche detta dinamica, invece di concentrarsi sulle informazioni attuali che si hanno sull'utente, punta a raccogliere costantamente informazioni sul comportamento dell'utente durante l'utilizzo della piattaforma. Tale tipo di sistema è anche chiamato profilazione comportamentale e oggigiorno è più indicato come profilazione ontologica dell'utente.

- **Profilazione utente ibrida**: Questo tipo di profilazione dell'utente combina i vantaggi della profilazione sia implicita che esplicita. Cioè prende in considerazione sia le caratteristiche statiche di un utente sia le informazioni comportamentali riguardanti l'utente. Questo approccio aiuta la creazione di profili in modo più efficiente e mantiene l'accuratezza delle informazioni temporali man mano che le informazioni vengono aggiornate temporaneamente.

I social media raccolgono una vasta quantità di dati per la profilazione degli utenti. Ecco alcuni esempi di dati che possono essere raccolti:

- **Informazioni personali**: nome, data di nascita, luogo di residenza, sesso, foto del profilo, informazioni sul lavoro e sulla scuola, relazioni personali, etc.

- **Informazioni sulle attività online**: ricerche, interessi, post sui social media, interazioni con altri utenti (come ad esempio commenti, "mi piace", condivisioni, etc.), pubblicazioni, etc.

- **Dati demografici**: età, sesso, lingua, nazionalità, etc.

- **Informazioni sui dispositivi**: tipo di dispositivo, sistema operativo, localizzazione, informazioni sul browser, etc.

- **Informazioni sulle preferenze**: gusti musicali, gusti cinematografici, gusti letterari, etc.

La maggior parte di queste informazioni, come le informazioni personali o i dati demografici vengono prese implicitamente, molto spesso all'iscrizione del social media; le altre, come le informazioni sull'attività online o informazioni sulle preferenze, vengono prese in modo implicito mentre utilizziamo il social media: più utilizziamo l'applicazione più diamo nostre informazioni e più miglioriamo la nostra profilazione, di conseguenza anche i contenuti che ci vengono consigliati.

Le informazioni sulle attività online includono tutte le azioni che gli utenti compiono sui social media e su altri siti web. Ad esempio:

- **Post**: i post che gli utenti pubblicano sulle loro pagine o sui loro profili, inclusi testo, immagini, video, etc.
- **Interazioni con gli altri utenti**: ad esempio commenti, "mi piace", condivisioni, etc.
- **Ricerche**: i termini di ricerca che gli utenti immettono nel motore di ricerca del sito o nel motore di ricerca del browser.
- **Navigazione**: le pagine che gli utenti visitano, il tempo trascorso su ogni pagina, il percorso di navigazione, etc.
- **Acquisti**: i prodotti che gli utenti acquistano online, il prezzo, la quantità, etc.

2.2 Social media mining

Il social media mining è un processo di estrazione, analisi e interpretazione dei dati generati dai social media. Questo approccio utilizza tecniche di data mining e di machine learning per analizzare i dati di social media come post sui social network, commenti e recensioni, con lo scopo di comprendere le opinioni, le tendenze, i comportamenti e le preferenze degli utenti. Il social media mining è spesso utilizzato in diversi ambiti, tra cui la ricerca di mercato, la politica, la pubblicità e la gestione della crisi, per aiutare le aziende e le organizzazioni a prendere decisioni consapevoli e a migliorare la loro presenza sui social media.
La prima cosa da considerare quando si ha a che fare con la personalizzazione è produrre una rappresentazione accurata delle informazioni dell'utente (preferenze e interessi dell'utente), solitamente memorizzate nel profilo dell'utente. Secondo l'articolo Mining Social Media with Social Theories: A Survey [9], ci sono tre tipi di "oggetti" nei dati dei social media: utenti, relazioni sociali e contenuti generati dagli utenti, che consentono di classificare approssimativamente le attività di social media mining in tre gruppi basati su questi oggetti: *user-based tasks*, *relation-based tasks* e *content-based tasks*.

2.2.1 User-based tasks

Una migliore comprensione dei propri social network può aiutare a condividere e raccogliere informazioni affidabili in modo più efficace ed efficiente. Per i fornitori di servizi di social media, una migliore comprensione dei propri clienti può aiutarli a fornire servizi migliori. Le attività relative agli utenti forniscono mezzi necessari ed efficaci per comprendere gli utenti dei social media. Attività chiave relative agli utenti sono:

Rilevamento della comunità

Le comunità nei social media possono essere esplicite come i gruppi Facebook. Tuttavia, in molti siti di social media, le comunità sono implicite e oscure ai loro membri. Il rilevamento della comunità viene proposto per trovare queste comunità implicite nei social media identificando gruppi di utenti che sono più densamente connessi tra loro rispetto al resto della rete. Il rilevamento di comunità implicite può avvantaggiare molte attività di social media mining come il *social targeting* e la *personalizzazione*. Formalmente, per un social network $G(U, S)$, il rilevamento della comunità consiste nel trovare un insieme di comunità C in cui gli utenti sono più densamente connessi all'interno di una comunità rispetto al resto degli utenti. L'omofilia è un concetto della sociologia che descrive la tendenza degli individui ad associarsi e legarsi con altri simili, questo suggerisce che è probabile che utenti simili siano collegati e che si influenzeranno a vicenda diventando più simili.

Classificazione degli utenti

La classificazione dell'utente è progettata per dedurre le informazioni sugli utenti dagli altri utenti nella stessa rete. La teoria della correlazione sociale suggerisce che le etichette degli utenti collegati dovrebbero essere correlate, motivo principale per cui i ricercatori ritengono che le etichette di U possano essere previste con la struttura della rete e con gli utenti parzialmente etichettati.
Un tipico algoritmo di classificazione degli utenti include parti di tre componenti:

- **Un classificatore locale**: Utilizzato per l'assegnazione iniziale dell'etichetta.
- **Un classificatore relazione**: Impara a classificare dall'etichette dei suoi vicini fino a suggerire etichette all'utente
- **Un classificatore collettivo**: Applica un classificatore relazionale a ciascun nodo in modo iterativo fino a quando l'incoerenza tra etichette vicine non viene ridotta al minimo.

Rilevamento dei social spammer

I social media sono diventati un modo importante ed efficace per diffondere informazioni. Data la sua popolarità e ubiquità, i social spammer creano molti account falsi e inviano contenuti commerciali non richiesti. Gli spammer sociali sono diventati dilaganti e il volume dello spam è aumentato drasticamente. Ad esempio, l'83% degli utenti dei social network ha ricevuto almeno una richiesta di amicizia o un messaggio indesiderato. Ciò non solo causa un uso improprio della larghezza di banda di comunicazione, dello spazio di archiviazione e della potenza di calcolo, ma fa anche perdere tempo agli utenti e viola

i loro diritti alla privacy. Pertanto, lo sviluppo di efficaci tecniche di rilevamento degli spammer sociali è di fondamentale importanza per migliorare l'esperienza dell'utente e influenzare positivamente il valore complessivo dei servizi di social media.

2.2.2 Relation-based tasks

Le attività relative alle relazioni si concentrano sull'estrazione di relazioni tra gli utenti e mirano a rivelare una visione dettagliata e completa delle relazioni sociali.

Previsione dei collegamenti

È fondamentale che i siti di social media forniscano servizi per incoraggiare più interazioni dell'utente con una migliore esperienza, come l'espansione del proprio social network. Un modo efficace è consigliare automaticamente le connessioni poiché è difficile per gli utenti capire chi è disponibile sui siti di social media. Il problema essenziale della raccomandazione di un amico è noto come *link prediction* (previsione del collegamento in italiano).

Previsione del legame sociale

Diversi tipi di relazioni possono influenzare le persone in modi diversi. Ad esempio, lo stile di lavoro di un utente può essere principalmente influenzato dai suoi colleghi; mentre le abitudini di vita quotidiana possono essere fortemente influenzate dalla sua famiglia. È necessario e importante rivelare questi diversi tipi di relazioni sociali, quindi ci chiediamo se possiamo dedurre automaticamente i tipi di relazioni sociali.

Previsione della forza del legame

Gli utenti dei social media possono avere centinaia di relazioni sociali. Il basso costo della formazione dei link nei social media può portare a reti con forze relazionali eterogenee (ad esempio, conoscenti e migliori amici mescolati insieme). È probabile che le coppie di utenti con forti punti di forza condividano una maggiore somiglianza rispetto a quelle con punti di forza deboli; pertanto una migliore comprensione dei punti di forza delle relazioni sociali può aiutare i siti di social media a servire meglio i propri utenti con: raccomandazioni migliori e strumenti di gestione degli amici più efficaci. Questo pone il problema sulla previsione della forza di legame.
Guidati dalla teoria della correlazione sociale, quattro diverse categorie di caratteristiche (vale a dire: somiglianza degli attributi, connettività topologica, connettività transazionale e connettività transazionale di rete) sono estratte da fonti tra cui collegamenti di amicizia, informazioni sul profilo, post in bacheca, post di immagini e le appartenenze ai gruppi. Quindi vengono addestrati vari classificatori per prevedere la forza del collegamento dalle informazioni transazionali sulla base di queste caratteristiche estratte.

2.2.3 Content-based tasks

L'uso pervasivo dei social media genera enormi quantità di dati a un ritmo senza precedenti e il problema del sovraccarico di informazioni diventa sempre più utile per gli utenti dei social media. La raccomandazione ha dimostrato di essere efficace nel mitigare il problema del sovraccarico di informazioni, grazie alle numerose tecniche sviluppate nell'ultimo decennio per vari compiti di content mining come l'analisi dei dati testuali, scraping di dati, analisi del sentiment, analisi di clustering e analisi di associazione.

Raccomandazione sociale

È probabile che gli utenti nel mondo fisico chiedano suggerimenti ai loro amici prima di prendere una decisione di acquisto e gli amici degli utenti forniscono costantemente buoni consigli, abbiamo osservazioni simili nei mondi online. Ad esempio, il 66% delle persone sui siti social ha chiesto ad amici o follower di aiutarli a prendere una decisione e l'88% dei link su cui hanno fatto clic i ragazzi di 14-24 anni sono stati inviati loro da un amico e il 78% dei consumatori si fida dei consigli dei colleghi piuttosto che degli annunci e le SERP[1] di Google. La teoria della correlazione sociale indica che la preferenza di un utente dovrebbe essere simile al suo social network. I metodi Ensemble prevedono un valore mancante per un dato utente come combinazione lineare di valutazioni dell'utente e del suo social network, basato sul tradizionale metodo di fattorizzazione della matrice CF, con l'intuizione che gli utenti e i loro social network dovrebbero avere valutazioni simili sugli stessi contenuti.

Selezioni delle caratteristiche

Una caratteristica dei contenuti generati dagli utenti nei social media è l'elevata dimensione, ad esempio ci sono decine di migliaia di termini nei tweet o nei pixel per le foto in Instagram. Le tradizionali attività di data mining come la classificazione e il clustering possono fallire a causa della "maledizione" della dimensionalità. La selezione delle caratteristiche, ovvero il processo di riduzione degli ingressi per l'elaborazione e l'analisi o l'individuazione delle caratteristiche maggiormente significative rispetto alle altre, ha dimostrato di essere un modo efficace per gestire dati ad alta dimensione per un data mining efficiente. Come accennato in precedenza, il contenuto generato dall'utente è collegato a causa della disponibilità di relazioni sociali e pone sfide ai tradizionali algoritmi di selezione delle caratteristiche che sono tipicamente progettati per i dati IID (in cui le variabili hanno tutte la stessa distribuzione di probabilità).

[1]SERP, dall'inglese search engine results page, è la schermata dei risultati prodotta dal motore di ricerca in risposta a una richiesta dell'utente

Sentiment Analysis

Al giorno d'oggi i servizi di social media come Twitter e Facebook sono sempre più utilizzati dagli utenti online per condividere e scambiare opinioni, fornendo ricche risorse per comprendere le opinioni pubbliche. Ad esempio, un semplice modello che sfrutta il sentimento e il contenuto di Twitter supera i predittori basati sul mercato in termini di previsione dei ricavi al botteghino per i film. L'umore pubblico, misurato da una raccolta su larga scala di tweet, ottiene un'accuratezza dell'86,7% nel prevedere le variazioni giornaliere dei valori di chiusura del DJIA[2]. Pertanto, negli ultimi anni l'analisi del sentiment per tali dati ricchi di opinioni sui social media ha attirato una crescente attenzione.

2.3 Il feed

Il feed news (o semplicemente feed) è una funzionalità comune sui social media che mostra agli utenti una selezione personalizzata di post, foto, video e altri contenuti che potrebbero interessare loro. Questa selezione viene effettuata tramite un algoritmo di personalizzazione che utilizza informazioni sugli interessi, le interazioni e il comportamento degli utenti per presentare contenuti rilevanti. Il social media mining entra in gioco utilizzando i dati sugli interessi degli utenti, come ad esempio i loro post più recenti o le interazioni con altri contenuti, per selezionare il contenuto da mostrare nel feed news. Un sistema di raccomandazione fornisce suggerimenti per gli elementi più pertinenti per un particolare utente. In genere, i suggerimenti si riferiscono a vari processi decisionali, ad esempio quale prodotto acquistare, quale musica ascoltare o quali notizie online leggere. I sistemi di raccomandazione sono particolarmente utili quando un individuo ha bisogno di scegliere un elemento da un numero potenzialmente enorme di elementi che un servizio può offrire.
Il primo algoritmo di raccomandazione per determinare la visibiiltà di un post nel feed è EdgeRank di Facebook, creato nel 2009 e utilizzato fino al 2013. EdgeRank si basava su tre fattori "affinity" (affinità), "weight" (peso) e "decay" (tempo di decadimento). Dal 2013 Facebook utilizza un nuovo algoritmo e i tre fattori di EdgeRank sono ancora oggi applicati assieme ad altri 100.000.

[2]Il Dow Jones Industrial Average (DJIA), noto anche come Dow Jones o semplicemente il Dow, è un indice azionario americano che rappresenta la media delle performance delle 30 principali aziende quotate al NYSE

Capitolo 3

Algoritmi di raccomandazione

Gli algoritmi di raccomandazione sono una componente chiave della tecnologia che fornisce suggerimenti personalizzati agli utenti, basati sulle loro preferenze e comportamenti. Secondo uno studio di Google, il 60% dei video visti su YouTube vengono dai consigliati. Il processo di raccomandazione, come abbiamo visto in precedenza, inizia con la raccolta e l'analisi dei dati degli utenti, in questo capitolo analizziamo come vengono usati questi dati e gli algoritmi per trovare i migliori elementi da consigliare. Prendendo come ispirazione il corso di Machine Learning di Google[2], nei successivi paragrafi viene presentata una panoramica sugli argomenti che ci serviranno per capire al meglio questi algoritmi.

3.1 Incorporamento dello spazio

I filtri presentati in precedenza mappano gli elementi e i contenuti attraverso un vettore di incorporamento in uno spazio di incorporamento comune $E = d$ Elementi *simili* si avvicinano nello spazio di incorporamento, come ad esempio i film che di solito vengono visti dallo stesso utente, il concetto di "vicinanza" è definito da una misura matematica di somiglianza, descritta in dettaglio nel paragrafo successivamente. Supponiamo di assegnare a ogni contenuto un valore in una scala da [-1, 1], dove -1 e 1 sono i poli opposti di una determinata caratteristica. In questa scalare, diamo un valore ad ogni utente basato sui contenuti con cui ha interagito. Per esempio, in un sistema di raccomandazione di film, assegneremo ad ogni film uno scalare S compreso tra -1 e 1, dove -1 rappresenta i film di animazione per bambini e 1 rappresenta i film di maggiore impatto per adulti. Assegneremo poi ad ogni utente un valore I, chiamato prodotto di incorporamento, calcolato come il prodotto vettoriale degli S dei film che ha guardato, come mostrato in esempio in Figura 3.1. Il prodotto dell'incorporamento del film e quello dell'utente devono essere più alti (più vicini a 1) per i film che prevediamo che l'utente apprezzi. Il primo e il quarto utente hanno preferenze ben spiegate da questa funzionalità: il primo utente preferisce i film per i bambini e il quarto utente preferisce i film per gli adulti.

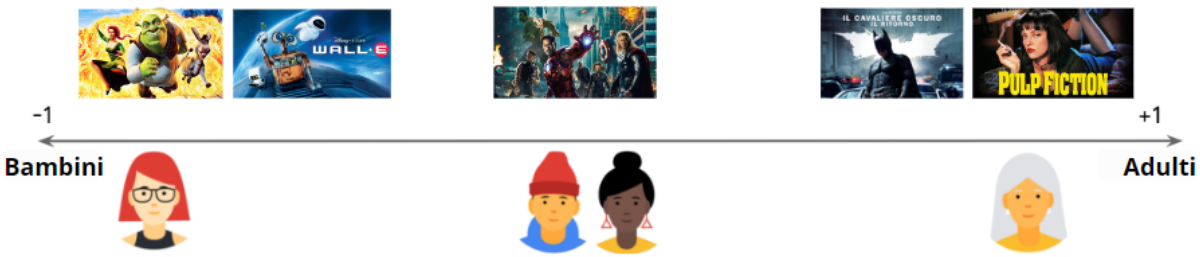


Figura 3.1: Esempio di assegnazione di uno scalare a ogni contenuto e calcolo del prodotto di incorporamento ad ogni utente in 1D

Tuttavia, le preferenze del secondo e del terzo utente non sono spiegate correttamente da questa singola funzionalità, in quanto sembra che abbiano gusti simili, ma osservando la matrice delle preferenze 3.2 ci si può accorgere che hanno gusti totalmente diversi. Quin-

Figura 3.2: Esempio di matrice delle preferenze, per semplicità mostra valori booleani

di non basta una funzionalità per spiegare le preferenze degli utenti, per superare questo problema aggiungiamo un'altra funzionalità rappresentata da un'altra dimensione. Per esempio, rappresentiamo con una misura se un film è un grande successo commerciale (blockbuster) o un film d'essai[1]. Queste due funzionalità le rappresentiamo con il seguente spazio bidimensionale.in Figura 3.3. Ora possiamo spiegare meglio le preferenze degli utenti, non ci sono limiti nelle dimensioni dello spazio d'incorporamento, tuttavia in letteratura si consiglia di di tenere lo spazio di incorporamento in dimensioni ridotte, ovvero, d è molto più piccolo delle dimensioni delle caratteristiche del contenuto.

[1]Il termine "film d'essai" si riferisce a un tipo di cinema che è destinato a un pubblico più ristretto e intellettualmente curioso rispetto ai film commerciali

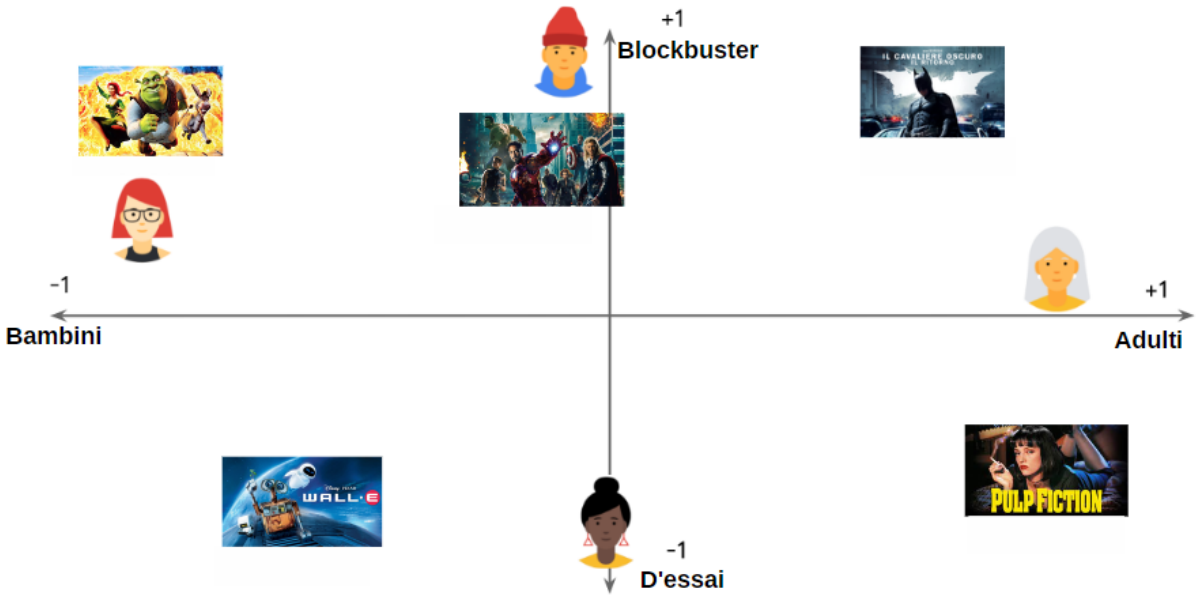


Figura 3.3: Esempio di assegnazione di uno scalare in un sistema a due dimensioni

3.2 Misure di somiglianza

Una misura di similitudine è una funzione $s : E \times E \to R$ che prende una coppia di incorporamenti e restituisce una scala che misura la loro somiglianza. Gli incorporamenti possono essere utilizzati per la generazione dei candidati come segue: in presenza di incorporamenti di query $q \in E$, il sistema cerca incorporamenti di elementi $x \in E$ che sono vicini a q, ovvero incorporazioni con elevata somiglianza $s(q, x)$. Per determinare il grado di somiglianza, la maggior parte dei sistemi di consigli si basa su uno o più dei seguenti elementi:

- **Coseno**: Questa metrica utilizza il prodotto scalare tra i due vettori e le loro lunghezze per calcolare un valore compreso tra -1 e 1 che indica il grado di somiglianza tra i vettori. Il valore 1 indica che i due vettori sono perfettamente simili o paralleli, mentre il valore -1 indica che i vettori sono perfettamente dissimili o ortogonali. Un valore compreso tra -1 e 1 indica un grado di somiglianza che varia tra la perfetta dissomiglianza e la perfetta somiglianza.

- **Prodotto con più punti**: Questa metrica utilizza il prodotto punto (che è un altro termine per descrivere il prodotto scalare) tra i due vettori per calcolare un valore compreso tra $-\infty$ e $+\infty$ che indica il grado di somiglianza tra i vettori. Un valore maggiore di 0 indica che i due vettori sono simili o hanno una direzione simile, mentre un valore minore di 0 indica che i vettori sono dissimili o hanno una direzione opposta. Un valore maggiore di 0 indica un grado di somiglianza più elevato, mentre

un valore più piccolo indica un grado di somiglianza più basso. Questo prodotto si calcola $s(q,x) = \langle q,x \rangle = \sum_{i=1}^{d} q_i x_i$. oppure da $s(q,x) = \|x\|\|q\| \cos(q,x)$

- **Distanza euclidea**: Questa metrica utilizza la distanza euclidea tra i due vettori per calcolare un valore compreso tra 0 e $+\infty$ che indica il grado di dissomiglianza tra i vettori. Il valore 0 indica che i due vettori sono perfettamente simili o coincidenti, mentre un valore maggiore di 0 indica un grado di dissomiglianza che aumenta con l'aumentare della distanza tra i vettori. Questa distanza si calcola come $s(q,x) = \|q-x\| = \left[\sum_{i=1}^{d}(q_i - x_i)^2\right]^{\frac{1}{2}}$

Considerando l'esempio in figura 3.4 il vettore nero illustra l'incorporamento della query mentre gli altri tre vettori di incorporamento (elemento A, elemento B, elemento C) rappresentano gli elementi candidati.

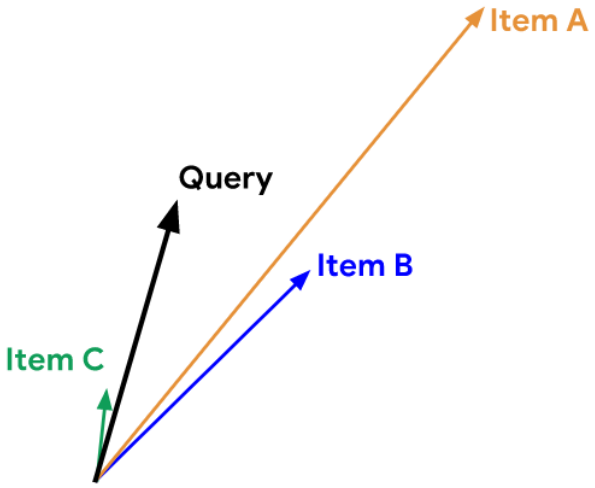


Figura 3.4: Esempio di vettori per rappresentare il grado di somiglianza.

A seconda della misura di somiglianza utilizzata il ranking degli elementi può essere diverso:

- **Coseno**: Item A > Item B > Item C
- **Prodotto con più punti**: Item C > Item A > Item B
- **Distanza euclidea**: Item B > Item C > Item A

Queste tecniche possono essere utilizzate singolarmente o in combinazione per calcolare la somiglianza tra elementi e fornire le raccomandazioni più appropriate per un utente. La scelta della tecnica o delle tecniche da utilizzare dipenderà dal tipo di dati e dalla specifica applicazione del sistema di raccomandazione. Ora che abbiamo spiegato l'idea di fondo, spieghiamo come avvengono le raccomandazioni.

3.3 Filtro basato sul contenuto

Nozioni di base

Le raccomandazioni basate sul contenuto usano informazioni degli elementi per apprendere le preferenze degli utenti, e consigliano elementi che condividono proprietà con gli elementi con cui un utente ha precedentemente interagito.

I dati in un sistema di raccomandazione basato sul contenuto possono essere salvati in diverse forme, ma solitamente vengono utilizzati database relazionali o NoSQL per gestire questi dati. In un database relazionale i dati vengono organizzati in tabelle con colonne e righe. Ad esempio, una tabella potrebbe contenere informazioni sulle opere, come ad esempio il titolo, l'autore, il genere, la data di uscita, e così via. Ogni opera avrà una riga nella tabella e le informazioni sull'opera saranno memorizzate come valori nella riga corrispondente. Nell'immagine seguente rappresentiamo vari film dividendoli per genere.

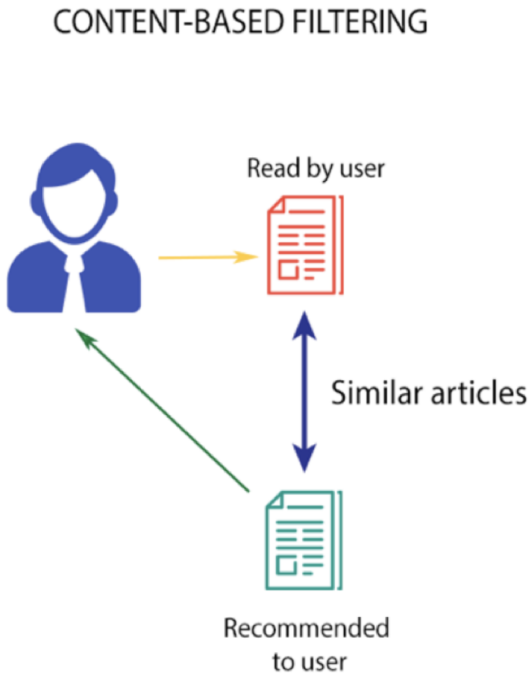


Figura 3.5: Esempio grafico del filtraggio basato sul contenuto

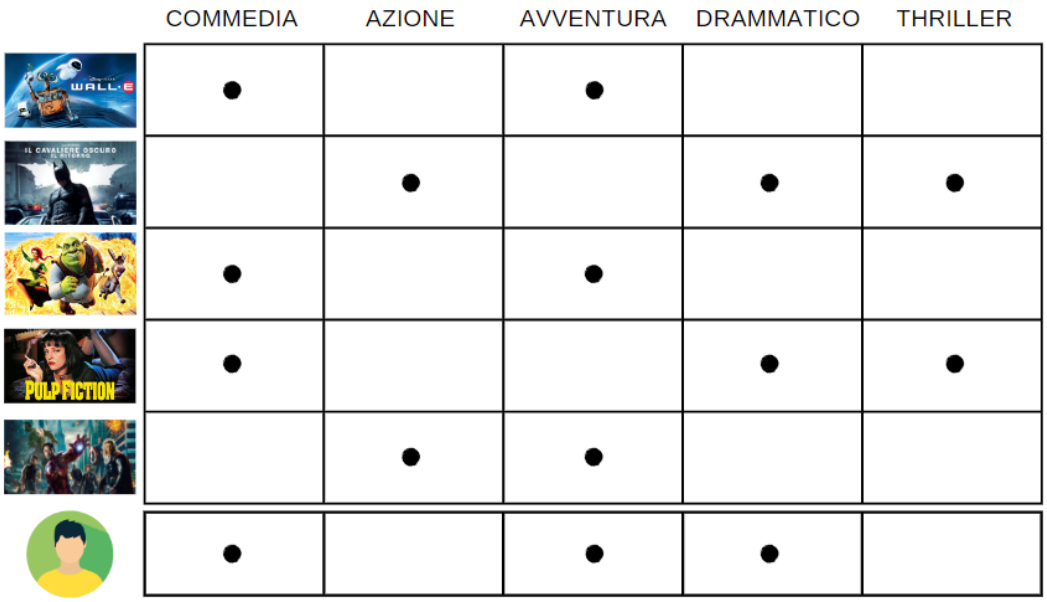


Figura 3.6: Esempio di come possono essere rapprensetate le preferenze dell'utente

In una tabella NoSQL i dati vengono organizzati in documenti che possono contenere informazioni sulle opere e sugli utenti, come ad esempio i loro gusti, preferenze, e le valutazioni. Questi documenti possono essere organizzati in diverse collezioni, a seconda delle esigenze del sistema. Per ogni elemento vengono calcolati dei descrittori o vettori di caratteristiche, che descrivono le proprietà dell'elemento stesso. Questi descrittori vengono poi utilizzati per calcolare le somiglianze tra gli elementi e, a sua volta, per effettuare le raccomandazioni.

Vantaggi e svantaggi

Vantaggi:

- Il modello non ha bisogno di dati sugli altri utenti, dato che i consigli sono specifici per questo utente. Questo consente di scalare più facilmente a un vasto numero di utenti.
- Il modello può catturare gli interessi specifici di un utente e può consigliare elementi di nicchia a cui solo pochi altri utenti sono interessati.

Svantaggi:

- Poiché la rappresentazione degli elementi è in qualche modo personalizzata, questa tecnica richiede molte conoscenze relative al dominio. Di conseguenza, il modello può avere solo lo stesso aspetto delle funzionalità progettate a mano.
- Il modello può fornire suggerimenti solo in base agli interessi esistenti dell'utente. In altre parole, il modello ha una capacità limitata di ampliare gli interessi esistenti degli utenti.

3.4 Filtro collaborativo

Nozioni di base

Il sistema basato sul filtraggio collaborativo identifica modelli simili nel comportamento degli utenti e fornisce raccomandazioni su contenuti con cui hanno già interagito altri clienti simili. Ad esempio, se l'utente X è simile all'utente Y e all'utente Y piace il video 1, il sistema può consigliare il video 1 all'utente X anche se l'utente X non ha visto video simili al video 1. In un sistema di raccomandazione basato sul filtraggio collaborativo, i dati degli utenti vengono solitamente salvati in una tabella, o in un database, che associa ogni utente a un insieme di prodotti o elementi che hanno valutato o interagito con in precedenza. Questa tabella viene utilizzata per calcolare le similitudini tra gli utenti e identificare le raccomandazioni più appropriate

per ciascun utente. Per esempio, se un sistema di raccomandazione sta raccomandando film, potrebbe essere costituito da una tabella che associ ogni utente ai film che ha valutato o visto. Queste informazioni vengono quindi utilizzate per raccomandare film simili a quelli valutati positivamente da un utente a altri utenti con gusti simili.

Vantaggi e svantaggi

Vantaggi:

- Non è richiesta alcuna conoscenza del dominio perché gli incorporamenti vengono appresi automaticamente.
- Il modello può aiutare gli utenti a scoprire nuovi interessi. Il sistema di machine learning potrebbe non sapere se l'utente è interessato a un determinato articolo, ma il modello potrebbe comunque consigliarlo perché utenti simili sono interessati all'elemento.

Svantaggi:

- Difficoltà a suggerire elementi nuovi, in quanto l'elemento appena aggiunto non ha feedback dagli utenti e quindi non può essere consigliato.
- Difficoltà a includere le funzionalità secondarie per la query

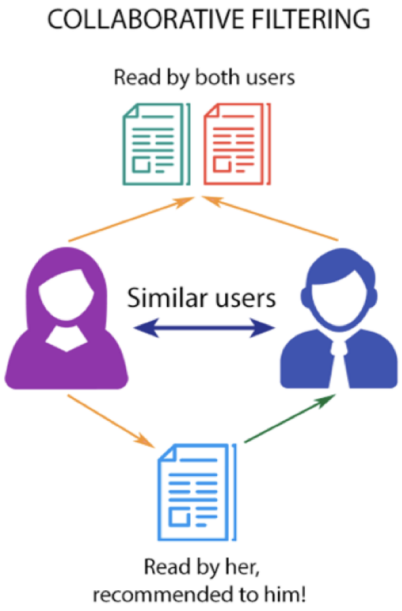


Figura 3.7: Esempio grafico dell'algoritmo del filtraggio collaborativo

3.5 Problemi

Queste metodologie, oltre agli svantaggi che hanno
singolarmente, hanno degli svantaggi comuni, come per esempio:

- Problemi di memoria: salvare tutti i dati in una matrice cosi grande, farà si che la maggior parte dei valori dell'utente saranno valori nulli o predefiniti. Tuttavia, ci sono alcuni modi per ridurre questo spreco di memoria:
  - **Compressione dei dati**: è possibile utilizzare tecniche di compressione dei dati, come la codifica run-length, per ridurre la quantità di memoria necessaria per memorizzare i dati.

  - **Matrix Factorization**: invece di memorizzare l'intera matrice, è possibile utilizzare tecniche di fattorizzazione della matrice, come la SVD o l'NMF, per rappresentare la matrice con un numero minore di valori.
  - **Sparse Matrix Representation**: se la maggior parte delle interazioni o delle valutazioni degli utenti sono mancanti, è possibile rappresentare la matrice in modo sparso, ovvero utilizzando solo i valori non nulli e memorizzarli in una struttura dati più efficiente come una lista di coppie chiave-valore.

- Problemi di scalabilità: man mano che il numero di utenti e elementi cresce, gli algoritmi di raccomandazione tradizionali subiranno seri problemi di scalabilità. Ad esempio, con decine di milioni di clienti $O(M)O(M)$ e milioni di articoli$O(N)SU$, un algoritmo di raccomandazione con complessità di nn è già troppo grande. Inoltre, molti sistemi devono reagire immediatamente ai requisiti online e formulare raccomandazioni per tutti gli utenti indipendentemente dai loro milioni di utenti, con la maggior parte dei calcoli che avviene in macchine di memoria molto grandi.

3.6 Metodo ibrido

Infine l'ultimo approccio alla creazione di sistemi di raccomandazione consiste nel combinare il filtro basato sul contenuto e collaborativo. Questo sistema consiglia elementi basati sulle classificazioni utente e sulle informazioni sugli elementi. L'approccio ibrido offre i vantaggi sia del filtro collaborativo che della raccomandazione basata sul contenuto[7].

3.7 Il caso Spotify - Teoria dei Bandits

Uno dei più rinomati algoritmi di raccomandazione è l'algoritmo di Spotify. Secondo una statistica fornita dalla stessa aziende Svedese, più di un terzo delle nuove scoperte di artisti avviene durante le sessioni Made for you, ovvero playlist personalizzate create da Spotify. L'algoritmo di raccomandazione di Spotify combina una vasta gamma di tecniche di machine learning e analisi dei dati per fornire raccomandazioni personalizzate e accurate agli utenti. La piattaforma continua a migliorare il suo algoritmo attraverso l'analisi costante dei dati e l'apprendimento automatico, per fornire un'esperienza di ascolto sempre migliore.
Nel documento "Explore, Exploit, and Explain: Personalizing Explainable Recommendations with Bandits"[6], gli ingegneri di Spotify trattano il problema della personalizzazione delle raccomandazioni e propongono un approccio basato sulla *teoria dei bandits*. Questa teoria combina esplorazione (scoperta di nuove opzioni) e sfruttamento (scelta delle opzioni che hanno dimostrato di essere efficaci in passato) per ottenere raccomandazioni ottimali per ogni utente.

La teoria dei bandits è una branca dell'intelligenza artificiale che si concentra sui problemi di ottimizzazione in situazioni in cui le informazioni sulle opzioni disponibili sono limitate o incerte. Il nome "bandits" si riferisce a una metafora delle macchine da gioco in cui l'obiettivo è quello di selezionare la macchina da gioco che paga il jackpot più alto. Questi problemi sono spesso descritti come *"problemi di bandit"*. In un problema di bandit, un agente deve scegliere tra diverse azioni per ottenere una ricompensa, che può essere positiva o negativa. L'agente sceglie un'azione alla volta, riceve la relativa ricompensa e utilizza queste informazioni per migliorare le sue scelte future. L'obiettivo è quello di massimizzare la somma totale delle ricompense ottenute nel tempo. Il paper "Explore, Exploit, and Explain" non menziona specifiche tecniche di ottimizzazione, tuttavia nella teoria dei bandits sono utilizzate diverse tecniche di ottimizzazione per combinare esplorazione e sfruttamento, tra cui:

- **Upper Confidence Bound** (UCB): utilizza la varianza delle valutazioni delle opzioni per determinare quale opzione esplorare. In questo modo, l'algoritmo UCB può scegliere di esplorare opzioni che sembrano promettenti ma che non sono state valutate a sufficienza, ma allo stesso tempo sfruttare le informazioni sulle preferenze passate dell'utente per fare raccomandazioni più mirate
- **Thompson Sampling**: utilizza la distribuzione delle valutazioni delle opzioni per determinare quale opzione scegliere.
- **Linear Upper Confidence Bound**: utilizza una combinazione di tecniche di regressione per combinare esplorazione e sfruttamento.

Le tecniche di ottimizzazione servono a trovare la soluzione ottimale in un contesto specifico. Nel caso delle raccomandazioni personalizzate basate sulla teoria dei bandits, le tecniche di ottimizzazione servono a combinare l'esplorazione e lo sfruttamento delle informazioni sulle preferenze dell'utente per fare raccomandazioni il più possibile personalizzate e accurate.
L'obiettivo durante la fase di esplorazione (explore) è quello di raccogliere informazioni sulle preferenze dell'utente e sulla qualità delle diverse opzioni, in modo da poter utilizzare queste informazioni per fare raccomandazioni più mirate in futuro. Per farlo l'algoritmo utilizza tecniche di ottimizzazione per testare e valutare diverse opzioni per l'utente. Ad esempio, può presentare all'utente opzioni casuali o scegliere di presentare opzioni che sembrano promettenti in base alle informazioni sulle preferenze passate dell'utente o su altre informazioni disponibili. Inoltre, durante la fase di esplorazione, le spiegazioni (explain) sono utilizzate per fornire informazioni sui motivi per cui sono state presentate determinate opzioni all'utente. Ad esempio, l'algoritmo potrebbe spiegare che un'opzione viene presentata perché è molto popolare tra gli utenti con preferenze simili o perché è stata valutata come altamente qualitativa in base ai dati disponibili. Questo aumenta la trasparenza dell'algoritmo (che non viene più visto come una scatola nera) e

la fiducia degli utenti nelle raccomandazioni. Durante la fase di sfruttamento (exploit), le tecniche di ottimizzazione possono aiutare a scegliere l'opzione più adatta in base alle informazioni sulle preferenze dell'utente raccolte fino a quel momento.
In sintesi, le tecniche di ottimizzazione sono utilizzate per trovare la soluzione ottimale per il problema di raccomandazione, tenendo conto delle informazioni disponibili sulle preferenze dell'utente e altri fattori rilevanti. Immaginiamo che si stia utilizzando un'app di streaming musicale che utilizza la teoria dei bandits per personalizzare le raccomandazioni agli utenti. Durante la fase di esplorazione, l'app potrebbe utilizzare una tecnica di ottimizzazione come l'algoritmo Upper Confidence Bound (UCB) per determinare quali brani presentare per valutare le preferenze di un singolo utente. L'algoritmo UCB potrebbe scegliere di presentare brani di artisti che non sono ancora stati ascoltati dall'utente, ad esempio, perché non troppo popolari. Durante la fase di sfruttamento, l'app potrebbe utilizzare una tecnica di ottimizzazione come l'algoritmo Thompson Sampling per scegliere quali brani raccomandare in base alle valutazioni passate. L'algoritmo Thompson Sampling utilizza la distribuzione delle valutazioni passate dell'utente per scegliere brani che hanno maggiore probabilità di piacere all'utente. In questo modo, le tecniche di ottimizzazione aiutano l'app di streaming musicale a combinare esplorazione e sfruttamento delle preferenze per offrire raccomandazioni sempre più personalizzate e accurate. Questo può aumentare la soddisfazione degli utenti con l'app e il tempo che viene trascorso ad ascoltare la musica su di essa.
Volendo analizzare in dettaglio il funzionamento dell'algoritmo Upper Confidence Bound, possiamo immaginare un utente appena iscritto al servizio di streaming; all'inizio, l'app presenterà casualmente diverse opzioni di generi musicali per valutare le sue preferenze. Ad esempio, potrebbe ascoltare un po' di pop, rock, hip-hop, jazz, etc. L'app terrà traccia delle sue valutazioni per ogni genere musicale e utilizzerà l'algoritmo UCB per calcolare l'intervallo di confidenza per ogni genere musicale. L'intervallo di confidenza è una stima della sua preferenza per un genere musicale basata sulle valutazioni passate. L'algoritmo UCB sceglierà quindi il genere musicale con l'intervallo di confidenza più alto e lo presenterà per la valutazione. Ad esempio, se l'intervallo di confidenza per il pop è molto alto rispetto agli altri generi musicali, l'app presenterà brani di pop per la valutazione. L'algoritmo UCB continuerà a presentare generi musicali con l'intervallo di confidenza più alto fino a quando non avrà sufficienti informazioni sulle preferenze dell'utente per generi musicali. A quel punto, l'app potrà utilizzare le informazioni sulle preferenze per raccomandare brani di generi musicali che pensa possano piacere di più.

Capitolo 4

Algoritmo di clustering dei dati basato sui numeri primi

4.1 Spiegazione dell'algoritmo

L'algoritmo di clustering dei dati basato sui numeri primi, chiamato successivamente algoritmo primi, si fonda sul concetto di *fattorizzazione unica* dei numeri primi. Un numero intero positivo, infatti, può essere scritto come un prodotto di numeri primi. Questa combinazione di numeri primi è univoca per ciascun numero intero. Ad esempio, il numero 35 può essere scritto come il prodotto fra 7 e 5. Così come 2023 è il risultato di 7 * 17 * 17.
L'idea chiave dell'algoritmo primi si basa sull'associazione delle varie categorie ad un numero primo. Le preferenze degli utenti vengono quindi salvata come il prodotto dei numeri primi delle categorie con cui l'utente ha affinità. Per fare un esempio pratico con i generi dei film, si potrebbe associare ad ogni genere di film un numero primo, come si evince in Figura 4.13, dove Commedia = 2, Azione = 3, Avventura = 5 e così via. I generi di film preferiti di un utente vengono salvati come la moltiplicazione dei numeri primi associati alle sue preferenze. Tornando alla Figura 4.13, se un utente esprime le proprie preferenze per i generi Azione e Avventura, gli sarà associato il numero 15.
Con questo metodo risulta computazionalmente semplice controllare se un utente ha affinità con una determinata categoria. Per farlo infatti, basta prendere la variabile associata all'utente e fare una divisione con il modulo con il numero primo della categoria ricercata. Se questa da resto 0, vuol dire che il numero primo è un fattore del numero associato all'utente, quindi all'utente piace quel genere. Cosa inversa, se invece il resto è diverso da zero vuol dire che all'utente non dato preferenza a quella categoria. Tornando all'esempio di Figura 4.13, se si volesse controllare se all'utente piace il genere Commedia, basterebbe prendere il numero 15 e controllare il resto della divisione con 2. Essendo diverso da 0, l'utente non ha espresso preferenza per le commedie. Per eseguire

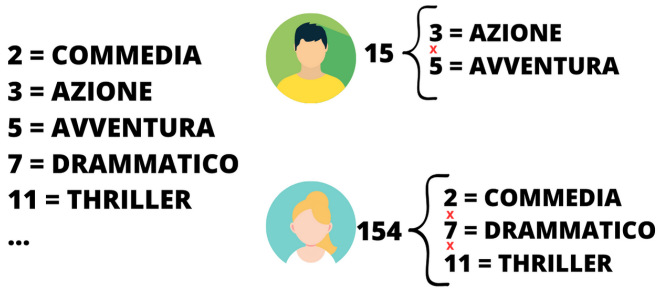


Figura 4.1: Esempio grafico di come vengono salvati i dati

questa operazione un computer ci mette solo tempo 1, in quanto è solo un operazione matematica.
È fondamentale che i numeri primi associati alle preferenze inizino da 2, questo perché se il numero primo associato ad un determinato genere fosse 1, questo genere risulterebbe sempre tra i preferiti di tutti, in quanto qualsiasi sia il numero dell'utente risulterà sempre divisibile per 1.

4.2 Confronto con altri algoritmi

Con l'obiettivo di analizzare le performance dell'algoritmo primi lo confrontiamo con i due algoritmi più comunemente usati in letteratura, ovvero la *matrice delle preferenze* e la memorizzazione delle categorie in un arrayList di preferenze dentro un "oggetto" utente (chiamata per comodità in questa tesi *ArrayList utente*).

4.2.1 Implementazioni delle classi

In seguito viene spiegato ogni algoritmo incluso il codice, scritto in Java, per spiegare nel dettaglio il funzionamento.

Algoritmo primi

Il metodo basato sui numeri primi si basa su due classi: *DizionarioNumeriPrimi* e *GestorePreferenze*. DizionarioNumeriPrimi gestisce l'associazione categoria - numero primo, salvando questa associazione in un dizionario con il numero primo come chiave e la categoria come valore. Questo permette nel caso di strutture dati efficenti come Array dinamici o HashMap l'accesso alla struttura dati in tempo $O(1)$.

La classe *DizionarioNumeriPrimi* viene inizializzata con un dizionario che collega il numero primo con l'elemento collegato, con Object questo può essere un numero, una stringa oppure un intero elemento. L'ArrayList *numeriLiberi* serve per raccogliere i numeri primi quando viene eliminata una categoria, infatti, quando questo avviene il numero primo associato alla categoria viene aggiunto a numeriLiberi.

```
1   public class DizionarioNumeroPrimi {
2       //Dove salvo gli elementi
3       protected HashMap<Integer, Object> categorie;
4       //Array con numeri primi
5       protected int[] numeriPrimi;
6       //Indice del numero primo massimo assegnato
7       protected int indexMaxPrime;
8       //Array di numeri liberi che posso usare
9       protected ArrayList<Integer> numeriLiberi;
10
11      //Costruttore
12      public DizionarioNumeroPrimi(int numCategorieMax){
13          categorie = new HashMap<>();
14          numeriPrimi = new int[numCategorieMax];
15          calcolaPrimi(numeriPrimi, numCategorieMax);
16          numeriLiberi = new ArrayList<>();
17          indexMaxPrime = -1;
18      }
19
20      //Per riempire l'array di numeri primi
21      public void calcolaPrimi(int[] numeriPrimi, int nCat){
22          numeriPrimi[0] = calculateNextPrime(numeriPrimi[1]);
23          for(int i=1; i<nCat; i++){
24              numeriPrimi[i] = calculateNextPrime(numeriPrimi[i-1]);
25          }
26      }
27
28      //Calcola il prossimo numero primo
29      public int calculateNextPrime(int prime){
30          prime++;
31          for (int i = 2; i < prime; i++) {
32             if(prime%i == 0) {
33                prime++;
34                i=2;
35             } else {
36                continue;
37             }
38          }
```

```
            maxPrime = prime;
            return prime;
        }
    }
```

Con questi metodi è possibile aggiungere un elemento al dizionario, l'aggiunta viene fatta con il metodo *addElement*, questo richiama *getPrimeNumber* che controlla se ci sono numeri primi da riciclare, se questo c'è allora usa quello altrimenti prende il prossimo dall'Array numeriPrimi

```
   //Aggiungi elemento al traduttore
    public int addElement(Object elemento){
        categorie.put(getPrimeNumber(), elemento);
        return maxPrime;
    }

    //Controlla se ci sono numeri primi liberi in mezzo dovuti
    //a cancellazioni
    public int getPrimeNumber(){
        if(numeriLiberi.size()!=0){
            return numeriLiberi.remove(0);
        } else {
            indexMaxPrime++;
            return numeriPrimi[indexMaxPrime];
        }
    }

    //Rimuovi elemento al traduttore da numero primo
    public Object removeByPrimeNumber(int primeNumber){
        numeriLiberi.add(primeNumber);
        return categorie.remove(primeNumber);
    }
```

Per avere il dizionario associato, il metodo *getAll* restituisce l'intera struttura dati.

```
    //GetAll
    public HashMap<Integer, Object> getAll(){
        return categorie;
    }
```

Per cercare un elemento è possibile utilizzare usare il metodo *getByPrimeNumber* che restituisce un Object da un numero primo. E il metodo *getPrimeNumberOf* che restituisce il numero primo di un elemento. Il secondo metodo è ovviamente più lento del primo in quanto il primo utilizza la caratteristica del dizionario dell'accesso diretto sui dati, invcece il secodno scorre tutti gli elementi per cercare il corrispondente.

```
    //Ritorna l'elemento con quel numero primo
    public Object getByPrimeNumber(int primeNumber){
        return categorie.get(primeNumber);
    }

    //Ritorna numero primo di un elemento
    public int getPrimeNumberOf(Object element){
        for (HashMap.Entry<Integer, Object> entry :
                                        categorie.entrySet()) {
            if (entry.getValue() == element) {
                return entry.getKey();
            }
        }
        return 1;
    }
```

GestorePreferenze si occupa di gestire la preferenza dell'utente: (i) memorizza il numero associato; (ii) aggiunge e rimuove preferenze; e (iii) controlla la presenza di uno o più generi. Inizializziamo la classe con un DizionarioNumeriPrimi, che abbiamo visto in precedenza, chiamandolo traduttore e teniamo salvato il numero associato all'utente, in modo da non doverlo chiedere ogni volta.

```
    public class gestorePreferenze {
    public DizionarioNumeroPrimi traduttore;
    public long user;

        public gestorePreferenze(DizionarioNumeroPrimi traduttore){
            this.traduttore = traduttore;
            this.user = 1;
        }
    }
```

Con questi metodi è possibile aggiungere o rimuovere una preferenza all'utente, per aggiungere una preferenza basta moltiplicare il numero associato all'utente per il numero primo associato alla categoria che vogliamo aggiungere. Per la rimozione invece basta dividere il numero per il numero primo che vogliamo togliere.

```
   //Aggiungi
    public long addCategory(long category){
        if(!presenceCheck(category)){
            return user *= category;
        }
        return user;
    }

```

```
    //Rimuovi
    public long removeCategory(long category){
        if(!presenceCheck(category)){
            return user /= category;
        }
        return user;
    }
```

Utilizziamo il metodo presenceCheck per controllare la presenza di una categoria, per farlo basta controllare il modulo tra il numero associato all'utente e il numero che vogliamo controllare la presenza, se questa da come risultato un numero diverso da zero vuol dire che l'utente non ha affinità con quella categoria.

```
   //Controlla presenza
    public boolean presenceCheck(long category){
        return user % category == 0;
    }
```

Questo metodo serve per controllare se un Object ha affinità con l'utente, prima controlla i divisori dell'oggetto e poi restituisce una lista di elementi con cui l'utente ha affinità.

```
    //Controllo affinita
    public ArrayList<Elemento> checkObjInUser(long obj){
        ArrayList<Elemento> list = new ArrayList<>();
        for(Entry<Integer, Object> entry :
                            traduttore.categorie.entrySet()){
            if(obj < entry.getKey()){
                return list;
            } else {
                if(obj % entry.getKey() == 0){
                    if(presenceCheck(entry.getKey())){
                        list.add(new Elemento(entry.getKey(),
                                              entry.getValue()));
                    }
                    obj /= entry.getKey();
                }
            }
        }
        return list;
    }
```

Restituisce tutte le preferenze dell'utente.

```
   //Restituisci tutte le preferenze
    public ArrayList<Integer> getAllPreference(){
        long userValue = user;
```

```
        ArrayList<Integer> list = new ArrayList<>();
        for(Entry<Integer, Object> entry :
                              traduttore.categorie.entrySet()){
            if(userValue < entry.getKey()){
                return list;
            } else {
                if(presenceCheck(entry.getKey())){
                    list.add(entry.getKey());
                    userValue /= entry.getKey();
                }
            }
        }
        return list;
    }
```

Matrice delle preferenze

La matrice delle preferenze si basa sulla creazione di una matrice dinamica realizzata con un ArrayList fatto di Arraylist di *boolean*. Gli utenti vengono rappresentati come righe della matrice e le categorie come colonne. Quando si crea un nuovo utente viene creato un nuovo elemento dell'arrayList (riga) inizializzando ogni categoria a *Null* (colonne). Qualora si volesse aggiungere una nuova categoria, sarà necessario aggiungere un nuovo elemento (colonna) per ogni arrayList collegato ad ogni utente (riga). Questo risulta essere uno svantaggio in termini di spazio di memoria utilizzato, ma fondamentale per la ricerca in quanto permette l'accesso diretto sui dati in tempi $O(1)$. La matrice delle preferenze viene inizializzata in questa maniera, con una lista di liste di boolean per gestire la matrice dinamicamente.

```
    public class MatricePreferenze {
        //Matrice di preferenze: un lista di liste
        public List<List<Boolean>> matrice;

        //Costruttore
        public MatricePreferenze() {
            matrice = new ArrayList<>();
        }
    }
```

I metodi *addUser* e *removeUser* servono per aggiungere o rimuovere righe della matrice, che corrispondono agli utenti.

```
    //Aggiungere un nuovo utente
    public void addUser() {
        ArrayList<Boolean> nuovaRiga = new ArrayList<>();
```

```
        int numColonne = matrice.size() > 0 ?
                                matrice.get(0).size() : 0;
        for(int j = 0; j < numColonne; j++) {
            nuovaRiga.add(null);
        }
        matrice.add(nuovaRiga);
    }

    //Rimuovere utente
    public void removeUser(int indiceRiga) {
        matrice.remove(indiceRiga);
    }
```

I metodi *addCategory* e *removeCategory* servono per aggiungere o rimuovere colonne della matrice, queste corrispondono alle categorie.

```
    //Aggiungi categoria
    public void addCategory() {
        for(List<Boolean> riga : matrice) {
            riga.add(false);
        }
    }

    //Rimuovi categoria
    public void removeCategory(int indiceColonna) {
        for(List<Boolean> riga : matrice) {
            riga.remove(indiceColonna);
        }
    }
```

Con questi metodi è possibile ottenere o impostare un valore di un determinato utente per una determinata categoria.

```
//Controlla presenza di una categoria per un utente
public boolean getElemento(int indiceRiga, int indiceColonna) {
    return matrice.get(indiceRiga).get(indiceColonna);
}

//Set categoria per un utente
public void setElemento(int indiceRiga, int indiceColonna,
    boolean valore) {
    matrice.get(indiceRiga).set(indiceColonna, valore);
}
```

Restituisce l'intera riga di un utente indipendente del valore ad esso associato.

```
    public ArrayList<Boolean> getUser(int index){
```

```
        return (ArrayList<Boolean>) matrice.get(index);
    }
```

Raccoglie tutte le categorie con cui un utente ha affinità e le mette dentro un ArrayList denominata *gusti*.

```
    //Restituisce tutte le categorie che ha un utente
    public ArrayList<Integer> getAllPreferences(int indiceRiga) {
        ArrayList<Integer> gusti = new ArrayList<Integer>();
        for (int i = 0; i < matrice.get(indiceRiga).size(); i++) {
            if (matrice.get(indiceRiga).get(i)) {
                gusti.add(i);
            }
        }
        return gusti;
    }
```

Il metodo *checkObjInUser* controlla se più categorie insieme piacciono all'utente.

```
    //Cerca piu categorie in un utente
    public ArrayList<Boolean> checkObjInUser(ArrayList<Integer>
                        oggetto, ArrayList<Boolean> gustiUtente){
        ArrayList<Boolean> risp = new ArrayList<>();
        oggetto.forEach((categoria) -> {
            if(gustiUtente.get(categoria)){
                risp.add(true);
            } else {
                risp.add(false);
            }
        });
        return risp;
    }
```

ArrayList utente

L'arrayList per ogni utente permette di salvare le preferenze degli utenti dentro la loro ipotetica classe Utente. Questo fornisce il vantaggio di non salvare tutti gli elementi (salvando spazio in memoria), ma ha lo svantaggio di non avere accesso diretto ai dati, costringendo a memorizzare un indirizzamento alla categoria. Quindi non viene salvata semplicemente l'eventuale affinità, come accade nella matrice delle preferenze. La classe ArrayListUser ha un'ArrayList di preferenze per salvare le preferenze.

```
   public class ArrayListUser {
    private ArrayList<Integer> preferenze;

```

```
    public ArrayListUser(){
        preferenze = new ArrayList<>();
    }
}
```

Il metodo *addCategory* controlla prima la presenza dell'elemento dentro l'arrayList con il metodo *presenceCheck*, se questo non c'è lo aggiunge all'arrayList. Invece, il metodo *removeCategory* rimuove la preferenza con il metodo del costrutto ArrayList *remove*, questo se lo rimuove restituisce l'oggetto, per questo motivo si fa anche un controllo != null, per controllare se tale categoria era presente, infatti il metodo *removeCategory* restituisce True se è stato effettivamente tolto o False se questo non era presente e quindi non si è tolto niente.

```
//Aggiungere una categoria ad un utente
    public boolean addCategory(int preferenza){
        if(!presenceCheck(preferenza)){
            preferenze.add(preferenza);
            return true;
        } else {
            return false;
        }
    }

    //Rimuovi una categoria ad un utente
    public boolean removeCategory(int preferenza){
        return preferenze.remove(preferenza) != null;
    }
```

Ritorna tutte le preferenze dell'utente semplicemente ritornando l'arrayList preferenze.

```
//Ritorna tutte le preferenze
    public ArrayList<Integer> getAllPreferences(){
        return preferenze;
    }
```

Il metodo *presenceCheck* controlla se nell'arrayList è presente una determinata categoria, invece il metodo *checkObjInUser* controlla la presenza di più categorie tra le preferenze dell'utente.

```
//Controllo presenza
    public boolean presenceCheck(int preferenza){
        return preferenze.indexOf(preferenza) != -1;
    }

    //Controllare affinita' con un oggetto
    public ArrayList<Boolean> checkObjInUser(ArrayList<Integer>
```

```
8                                                          preferenze){
9           ArrayList<Boolean> risp = new ArrayList<>();
10          preferenze.forEach((preferenza) -> {
11              risp.add(presenceCheck(preferenza));
12          });
13
14          return risp;
15      }
```

4.2.2 Confronto a livello teorico

Tutti e tre gli algoritmi presentano gli stessi metodi per eseguire le seguenti operazioni:

- **addCategory**: aggiungere una categoria possibile
- **removeCategory**: rimuovere una categoria possibile
- **addPreference**: Aggiungere una nuova preferenza all'utente
- **removePreference**: Rimuove una nuova preferenza all'utente
- **presenceCheck**: Controllare se un utente ha affinità con una categoria
- **GetAll**: ritornare tutti i valori che all'utente ha affinità
- **CheckObjInUser**: controllare se più categorie hanno affinità con l'user

In questo paragrafo viene confrontato il costo computazionalmente dei metodi che implementano:

Metodo	**NumeriPrimi**	**MatricePreferenze**	**ArrayListUtente**
addCategory	$O(1)$	$O(U)$	-
removeCategory	$O(U)$	$O(U * a)$	-
addPreference	$O(1)$	$O(1)$	$O(1)$
removePreference	$O(1)$	$O(1)$	$O(a)$
presenceCheck	$O(1)$	$O(1)$	$O(n)$
GetAll	$O(C)$	$O(C)$	$O(1)$
CheckObjInUser	$O(obj.lenght)$	$O(obj.length)$	$O(n^*obj.lenght)$

O rappresenta il costo computazionale; U indica il numero degli utenti; C il numero di categorie; a Costo per modificare l'arrayList; *obj.length* numero di categorie che l'oggetto che stiamo cercando

Ogni metodo ha diverse implementazioni che lo rendono più o meno performante computazionalmente.

addCategory e removeCategory

Per i metodi *addCategory* e *removeCategory* si può notare che l'algoritmo ArrayListUtente non ha un costo computazionale, in quanto la classe proposta in questa tesi non implementa la gestione dell'inserimento di nuove categorie. L'algoritmo matrice delle preferenze ha una forte correlazione con il numero degli utenti U, in quanto la creazione di nuova categoria prevede l'aggiunta di un elemento in coda all'arrayList per ogni utente. Per quanto riguarda la rimozione invece, oltre alla rimozione bisogna anche considerare l'aggiustamento dell'indirizzamento dell'arrayList. Questa operazione ha un costo computazionale di n e viene eseguita per ogni U. Infine, si può vedere come l'algoritmo primi abbia un costo $O(1)$ per l'aggiunta di un elemento al dizionario, avendo però un costo elevato per la rimozione di categorie. Questo accade in quanto l'eliminazione dalla struttura dati ha un costo costante, ma bisogna eliminare tale cifra da ogni utente, in modo da poter riutilizzare il numero primo ed evitare equivoci.

addPreference, removePreference e presenceCheck

Per i metodi sugli utenti ovvero *addPreference, removePreference* e *presenceCheck* si può notare che la matrice delle preferenze ha prestazioni migliori a livello computazionale, avendo un costo computazionale di $O(1)$. La spiegazione deriva dal fatto che la grande struttura dati che implementa questo algoritmo permette facilmente l'accesso diretto ai dati della matrice. ArrayListUtente ha ottime performance quando bisogna aggiungere un elemento, in quanto è sufficiente aggiungerlo infondo alla lista. Tuttavia, questo algoritmo peggiora quando bisogna togliere o cercare un elemento, in quanto non ordina le categorie e per cercare una specifica categoria ha bisogno di scorrere tutta la struttura dati finché non trova l'elemento.

GetAll e CheckObjInUser

Infine, si analizzano le prestazioni negli ultimi due metodi: *GetAll* che si occupa di prendere tutti i gusti dell'utente salvati e *CheckObjInUser* che controlla un oggetto con più caratteristiche. ArrayListUtente ha ottime prestazioni per GetAll in quanto è proprio quello che memorizza, quindi basta ritornare solamente l'ArrayList memorizzato. Viceversa l'algoritmo ha pessime prestazioni per controllare più caratteristiche, in quanto bisogna scorrere l'intero ArrayList per ogni categoria che contiene l'oggetto. Situazione

simile avviene per la matrice delle preferenze per quanto riguarda il metodo checkObjInUser, però il controllare l'affinità con più categorie costa la quantità di categorie richieste. Stesso discorso vale per l'algoritmo primi, che è costretto a scorrere l'intero set di numeri primi per cercare corrispondenza nei gusti dell'utente. Discorso molto diverso avviene per checkObjInUser, in quanto sono possibili due scenari: (i) se l'obiettivo è capire se l'intero *obj* è presente tra le preferenze dell'utente, questo ha un costo come*removePreference* $O(1)$, in quanto si ha solo una divisione da compiere; e (ii) se l'obiettivo è capire quali categorie fanno parte per l'utente, il costo computazionale sarà anche in questo caso costante.

Conclusione

Da questo studio possiamo notare che l'algoritmo primi è fortemente influenzato dal numero di categorie che vengono salvate dentro l'utente, in quanto causa operazioni più lunghe dovute a moltiplicazioni e divisioni con numeri molto grandi ma comunque per assunzione costanti. Però in casi ottimali pareggia le migliori performance degli altri due algoritmi.

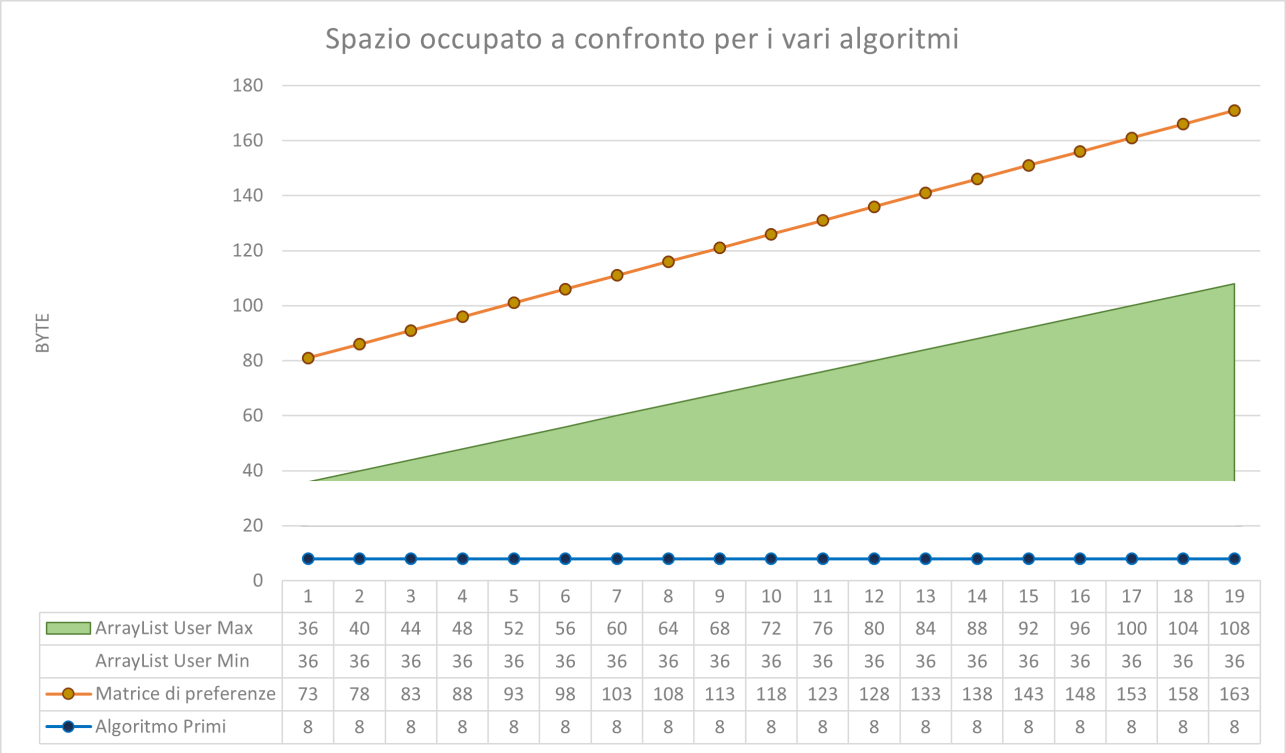
4.2.3 Confronto a livello pratico

Le performance degli algoritmi a livello pratico possono essere analizzate focalizzandosi sulla quantità di memoria che effettivamente utilizzano e sulla velocità per eseguire più volte determinate operazioni.

Spazio di archiviazione

Il grafico 4.2 mostra lo spazio di archiviazione utilizzato dai singoli algoritmi, considerando i dati di un solo utente e per un numero di categorie da 1 a 19.
A differenza della matrice delle preferenze e dell'algoritmo primi, lo spazio di archiviazione usato da *ArrayList User* è rappresentato da un area. Questo perché lo spazio di archiviazione è sensibile alla quantità di dati salvata dall'utente, poiché l'aggiunta di un elemento all'ArrayList ha un costo di 4Byte. Questo non accade invece con la matrice delle preferenze poiché vengono salvati tutti i dati in ogni caso. Situazione simile, ma diversa, accade per l'algoritmo primi, in quanto le preferenze vengono salvate in una variabile primitiva di tipo *long*; questo tipo di dato permette la memorizzazione di un numero intero minore di $2^{63} - 1$ su 2^{64} bit, quindi 8byte di memoria. Un' altra cosa da notare è che la matrice delle preferenze e ArrayList utente hanno un numero alto di byte utilizzato anche per un utente e una categoria. Questo è dovuto dall'*overhead*[1] della struttura dati che utilizzano per salvare i dati, ovvero gli ArrayList.

[1]L'overhead rappresenta la quantità di risorse o spazio di memoria aggiuntivi richiesti per gestire un'operazione o una struttura dati.



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
ArrayList User Max	36	40	44	48	52	56	60	64	68	72	76	80	84	88	92	96	100	104	108
ArrayList User Min	36	36	36	36	36	36	36	36	36	36	36	36	36	36	36	36	36	36	36
Matrice di preferenze	73	78	83	88	93	98	103	108	113	118	123	128	133	138	143	148	153	158	163
Algoritmo Primi	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8

Figura 4.2: Salvataggio dati con un dato di tipo *long*

Esiste un limite in questo metodo di memorizzazione per salvare le preferenze. Come accennato in precedenza, il tipo di dato che viene utilizzato può contenere numeri fino a $2^{63} - 1$, che in numero decimale corrisponde a 9.223.372.036.854.775.807 o più semplicemente circa $9 * 10^{18}$. Per quanto grande possa sembrare, questo limite è facilmente superabile quando arriviamo alla ventesima categoria. Il prodotto dei primi 19 numeri primi (escludo l'1), infatti, è circa $1 * 10^{18}$. Moltiplicando questo numero per il ventesimo numero primo, 71, si ottiene circa $71 * 10^{18}$, sforando così il tetto massimo ammissibile. Per ovviare a questo problema si è implementata la classe *BigInteger* di Java, utilizzata per implementare numeri molto elevati. A rispecchio di quanto detto, come possiamo vedere nel grafico 4.3, la memoria utilizzata dall'algoritmo primi aumenta, ma rimane comunque inferiore a quella utilizzata dagli altri algoritmi.

Velocità

Un ulteriore indice di performance utilizzato è quello della velocità. Per determinare una misura quantitativa della velocità misuriamo i metodi che vengono utilizzati. Questa operazione viene eseguita 100 volte per poter catturare la varianza della velocità, in

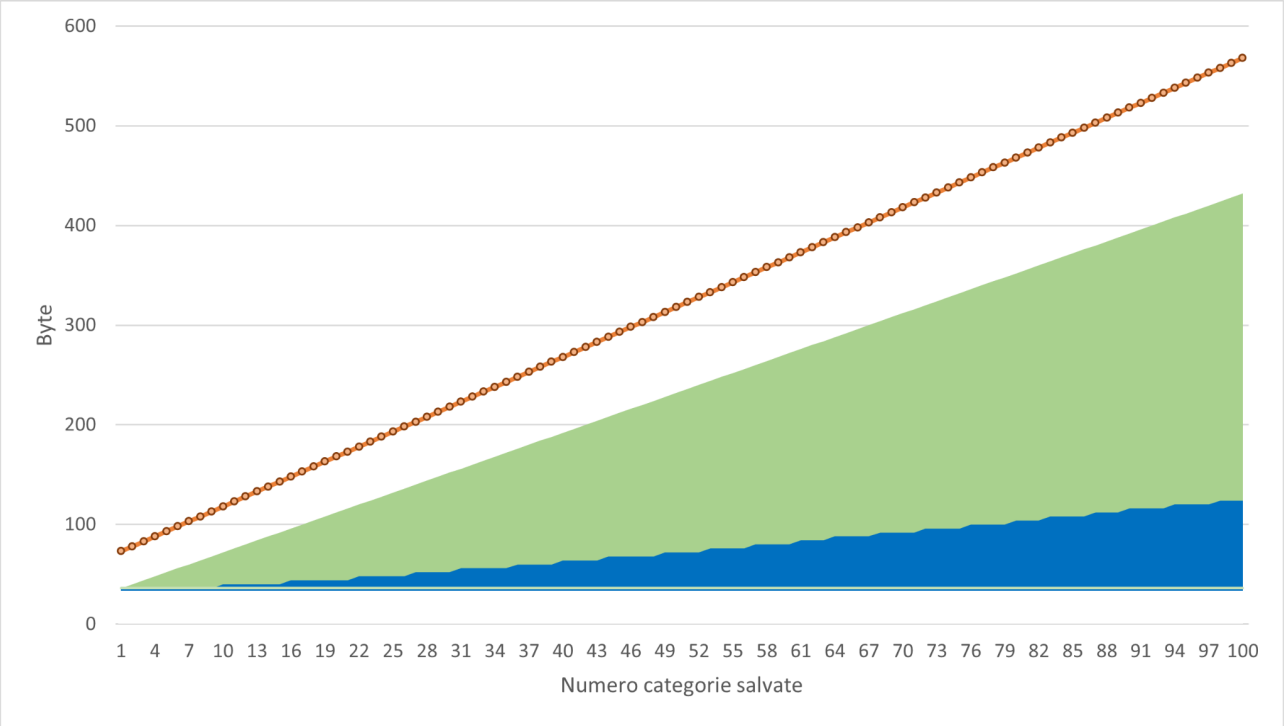


Figura 4.3: Salvataggio dati con un dato di tipo *BigInteger*

quanto le performance sono comunque dovute alla macchina utilizzata per eseguire i test.
Per misurare la velocità si sono creati due scenari:

- **Manipolazione del gestore delle categorie**, aggiungere e rimuovere un determinato numero di categorie che possono essere aggiunte ad un utente;
- **Gestione delle categorie per l'utente**, aggiungere, rimuove e cercare determinate categorie all'interno di un utente.

C'è da dire che inoltre lo scenario numero 1 viene eseguito molte meno volte dello scenario numero 2. In quanto le categorie dovrebbero essere scelte all'inizio, con ovviamente qualche modifica in corso d'opera. Invece, le operazioni sugli utenti, scenario 2, potrebbero essere fatte in ogni momento.
Nello studio della manipolazione delle categorie, si confronteranno unicamente le performance dell'algoritmo primi e della matrice delle preferenze, in quanto ArrayList utente non gestisce la memorizzazione generale delle preferenze. Per confrontare i due algoritmi si sono individuati tre scenari:

1. Situazione ideale per entrambi: 19 categorie per 1 utente

2. Situazione peggiore per l'algoritmo primi: 15 categorie e 10000 utente

3. Situazione peggiore per la matrice delle preferenze: 100 categorie e 1 utente

4. Situazione peggiore per entrambi: 100 categorie e 10000 utenti

1) La situazione numero 1 è l'ideale per entrambi gli algoritmi. Ideale per l'algoritmo primi in quanto vengono salvate poche categorie, quindi potrà lavorare con numeri piccoli. Ed ideale per la matrice perché deve gestire solo un'utente, quindi eseguirà le operazioni soltanto una volta.

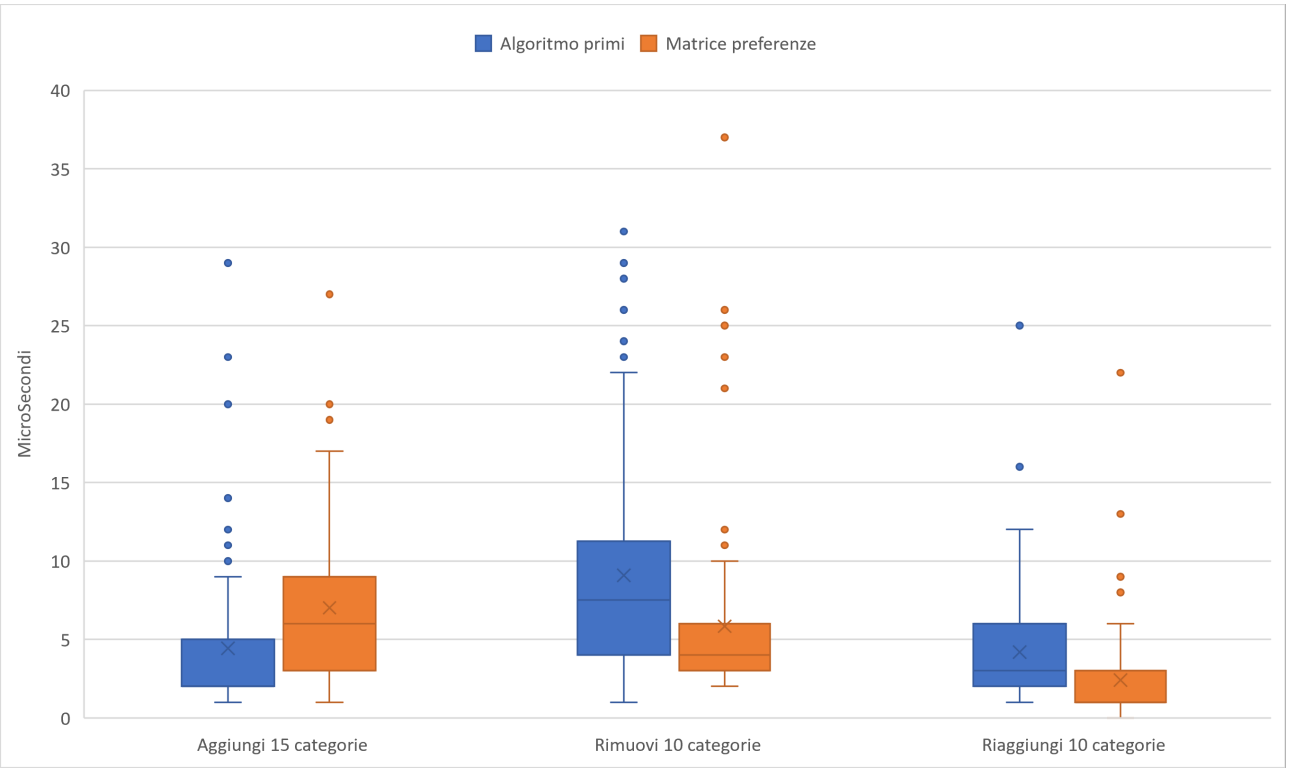


Figura 4.4: Aggiunta, rimozione e riaggiunta in caso ottimale per entrambi

I risultati mostrano che l'algoritmo Primi è leggermente migliore per quanto riguarda l'aggiunta di nuove categorie, però è nettamente peggiore nel rimuovere categorie dato dal fatto che deve anche controllare tutti gli utenti (1 in questo caso) per verificare la presenza di questa categoria nelle sue preferenze ed eventualmente toglierla. Inoltre, quando togliamo una categoria, dobbiamo mettere il numero primo associato dentro un'ArrayList per poterlo riutilizzare in futuro per un'altra categoria. Per questo motivo possiamo notare nella terza parte che le prestazioni dell'algoritmo Primi scendo in quanto deve prendere il dato da un ArrayList e non dall'array di numeri primi.

2) La situazione peggiore per la matrice delle preferenze avviene quando si devono gestire più utenti, questo perché bisogna ripetere le operazioni di aggiunta e rimozione per ogni utente. Questo però è anche

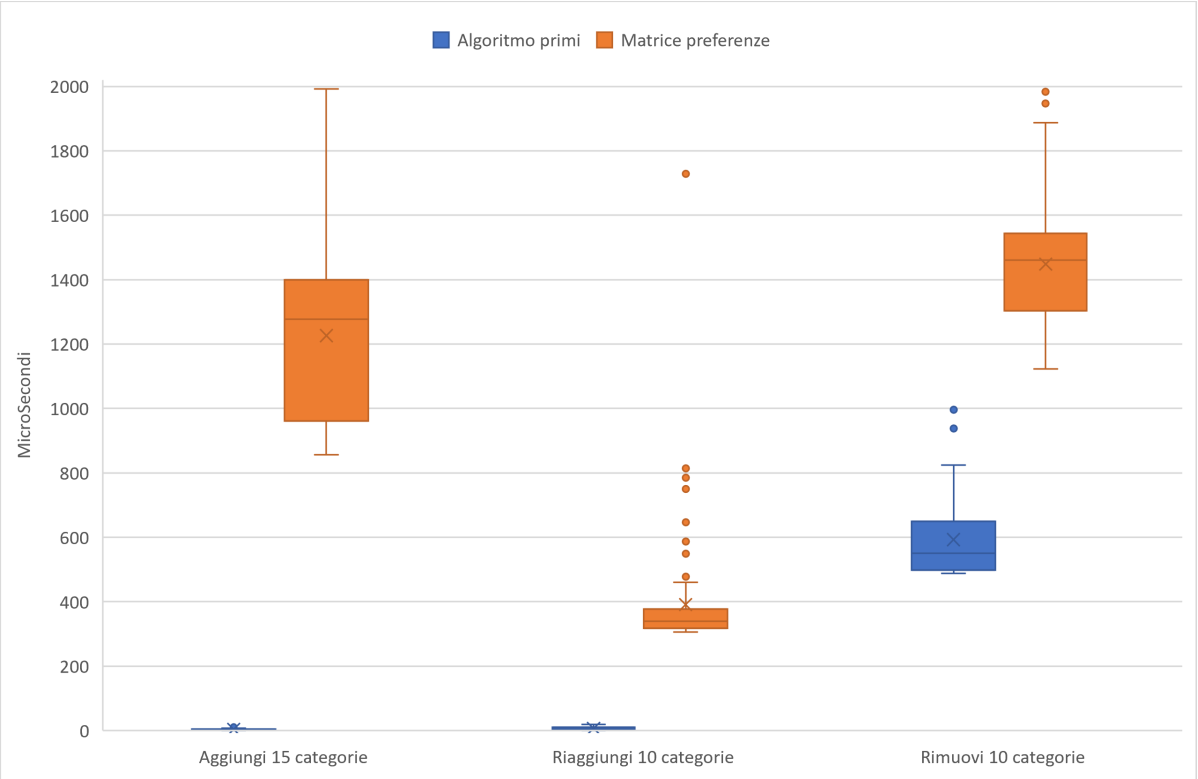


Figura 4.5: Aggiunta, rimozione e riaggiunta in caso ottimale per Primi

Questa situazione si può notare nel grafico 4.5, dove si vede chiaramente che le prestazioni peggiori dell'algoritmo primi sono chiaramente migliori delle prestazioni della matrice delle preferenze per quanto riguarda l'aggiunta e la riaggiunta di categorie, tanto da non vedersi neanche nel grafico 4.5 ma si possono vedere meglio nel Grafico 4.6. Però nelle rimozione continua ad aver prestazioni migliori rispetto alla matrice di preferenze, però si vede che inizia a peggiorare di performance.

3) La situazione peggiore per l'algoritmo primi avviene quando si lavora con tante categorie e indi per cui si lavora con numeri molto alti. Però questa situazione è ottimale per la matrice di preferenze in quanto si sta lavorando su un solo utente. Nel Grafico 4.7, si può notare che per quanto riguarda l'aggiunta di una categoria i due algoritmi hanno prestazioni più o meno simili, leggermente migliore la matrice di preferenze nel caso della riaggiunta. Però come mostra il 4.8 l'algoritmo Primi ha performance nettamente peggiori per quanto riguarda la rimozione di categorie. La matrice ha prestazioni ottime

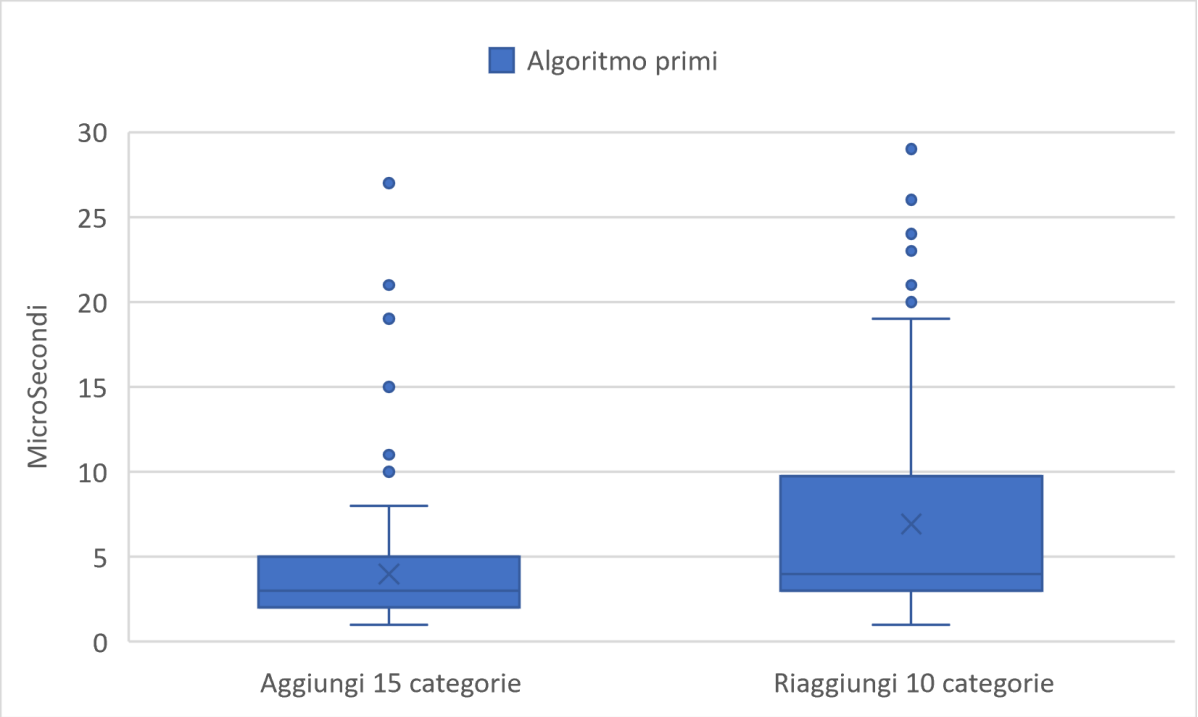


Figura 4.6: Vista in dettaglio prestazioni algoritmo primi in situazione ideale

perché si sta analizzando il suo caso ottimo, ovvero con un solo utente.

4) La situazione peggiore per entrambi gli algoritmi è che ci siano molte categorie e molti utenti. La presenza di molte categorie non rappresenta un problema per la matrice delle preferenze, ma è la presenza di molti utenti che richiede di ripetere le stesse operazioni per ogni utente, facendo si che questo sia il caso pessimo. Invece, l'algoritmo primi sente entrambi i fattori, in quanto tante categorie costringono l'algoritmo a lavorare con tanti numeri grandi e tanti utenti portano l'algoritmo a fare più controlli per eliminare le categorie dalle proprie preferenze. Nel Grafico 4.9 si può notare che le prestazioni di aggiunta e riaggiunta, per quanto riguarda l'algoritmo primi, non cambiano molto rispetto alla scenario 2), cosa che però accade per la matrice di preferenze. Invece la rimozione di categorie, per l'algoritmo Primi, ha performance drasticamente peggiori rispetto la matrice di preferenze, perché questa operazione richiede tante divisioni (1 per ogni utente) con numeri elevati 4.11.

Per quanto riguarda invece lo studio riguardante la gestione delle categorie per l'utente, si confrontano tutti e tre gli algoritmi, questi test sono svolti con 10000 utenti. Nel Grafico 4.12 sono state misurate le performance prendendo in considerazione il caso ottimo dell'algoritmo primi e dell'algoritmo ArrayList user, ovvero il caso in cui ci siano poche categorie. Come si può notare dal Grafico 4.12 le performace dell'algoritmo Primi,

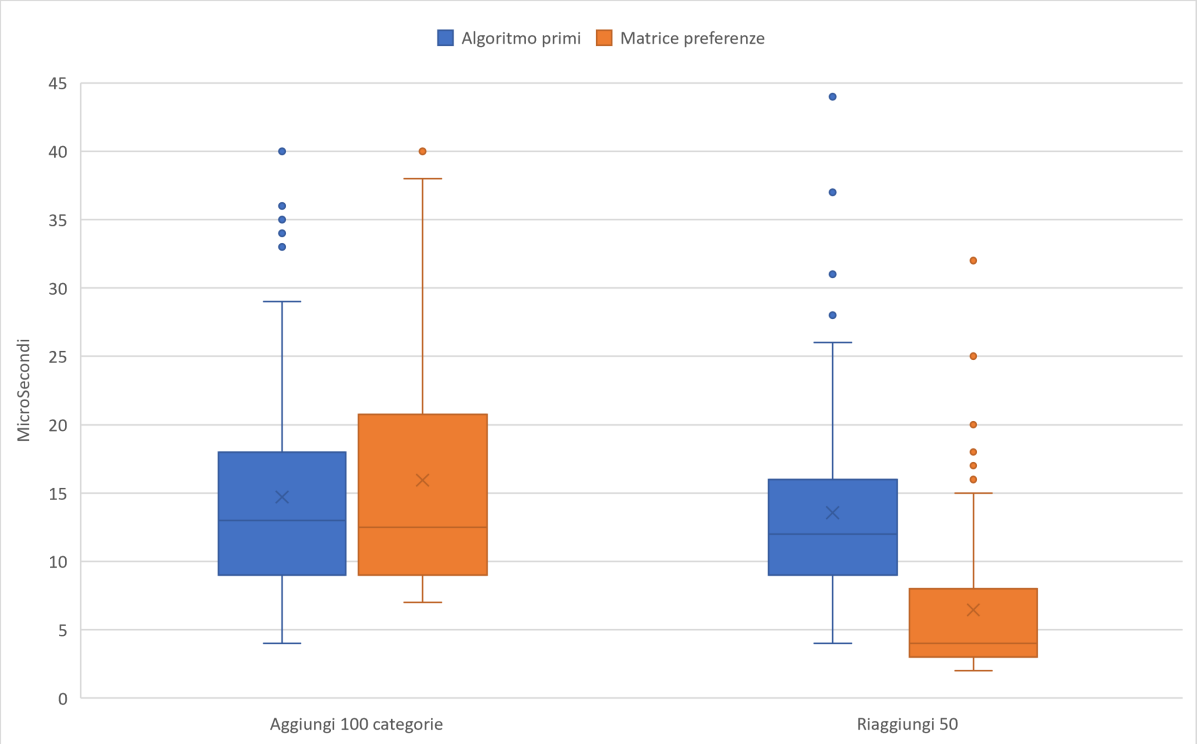


Figura 4.7: Aggiunta e riaggiunta in caso ottimale per matrice delle preferenze

nella situazione di aggiunta e rimozione di categorie nell'utente, sono nettamente migliori rispetto quelle dell'algoritmo ArrayList User. Mentre si avvicina ai risultati della matrice di preferenze, che avendo l'accesso diretto ai dati sarà sempre nel caso ottimo. Invece, nel caso della ricerca della categoria nell'utente possiamo notare che tutti hanno più o meno prestazioni simili, pur rimanendo quello della matrice il migliore.

Analizzando il caso pessimo per i due algoritmi, ovvero il caso con molte categorie (500), si può notare che l'algoritmo primi non riesce a performare all'altezza degli altri due, nonostante anche l'algoritmo ArrayList User ottenga delle prestazioni peggiori rispetto le precedenti.

Conclusione

In conclusione, si può dire che l'algoritmo Primi abbia ottime performance rispetto gli altri algoritmi studiati per quanto riguarda lo spazio di archiviazione dei dati. Però presenta prestazioni peggiori per quanto riguarda le performance sulla velocità. C'è anche

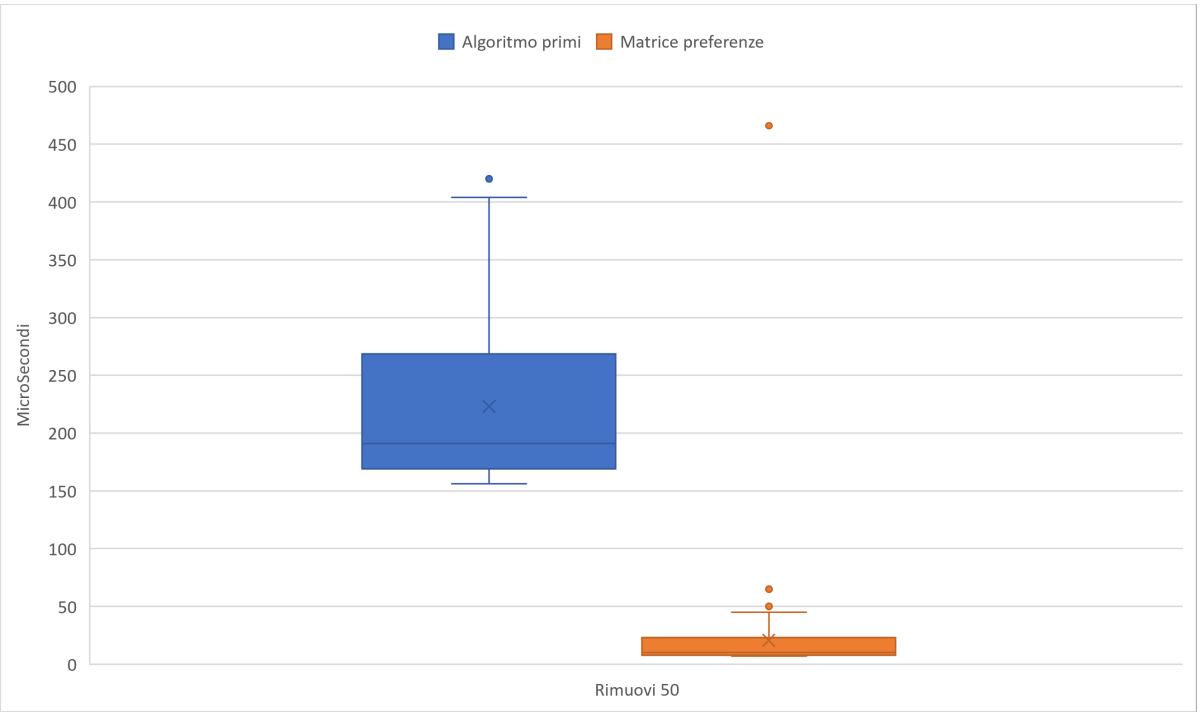


Figura 4.8: Rimozione in caso ottimale per matrice delle preferenze

da dire che queste performance sono soprattutto dovute al linguaggio di programmazione usato e alla macchina sulla quale sono stati svolti i test. In altri linguaggi e con altri elaboratori più potenti le performance potrebbero essere migliori.

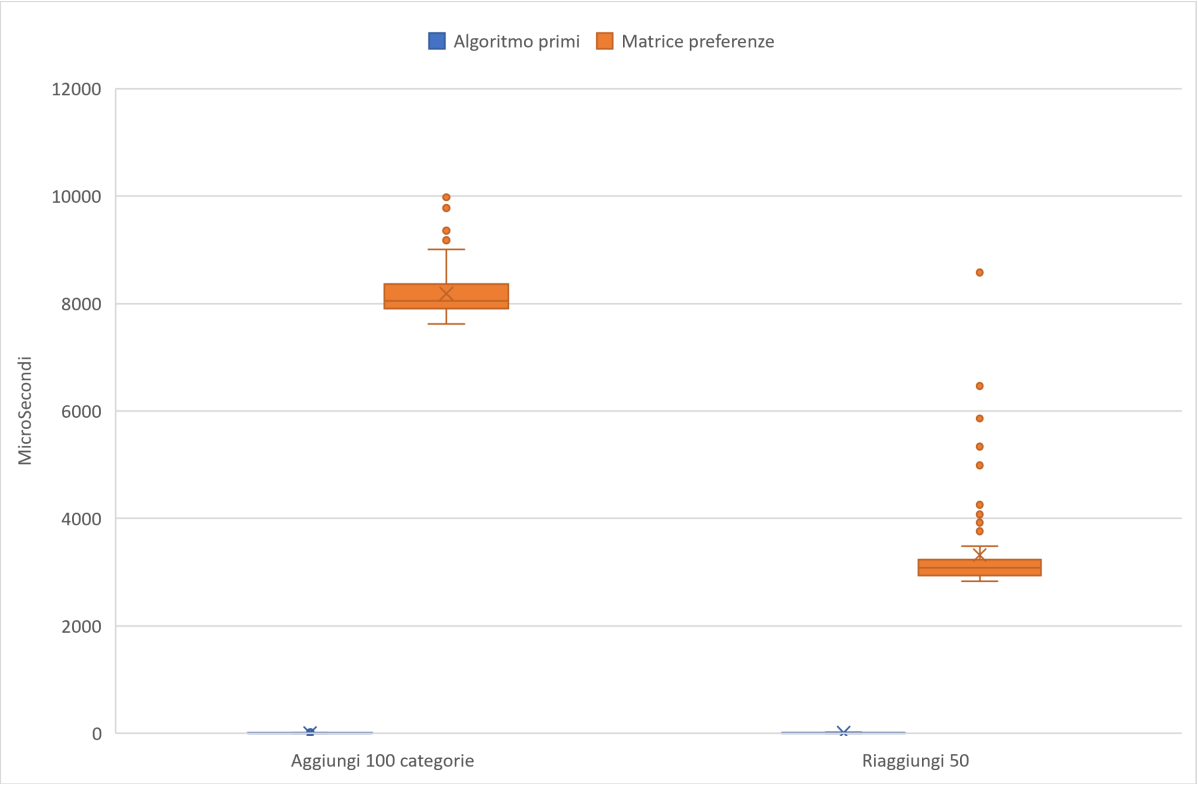


Figura 4.9: Aggiunta e riaggiunta in caso pessimo per entrambi

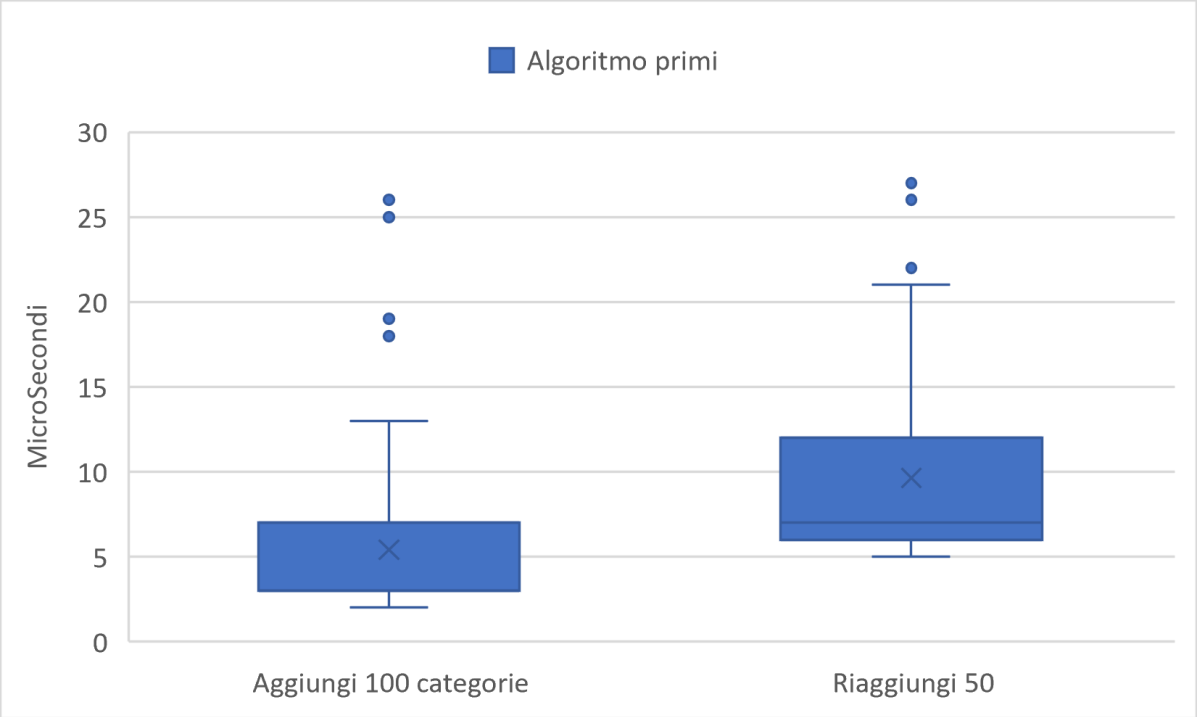


Figura 4.10: Vista in dettaglio delle performance dell'algoritmo Primi

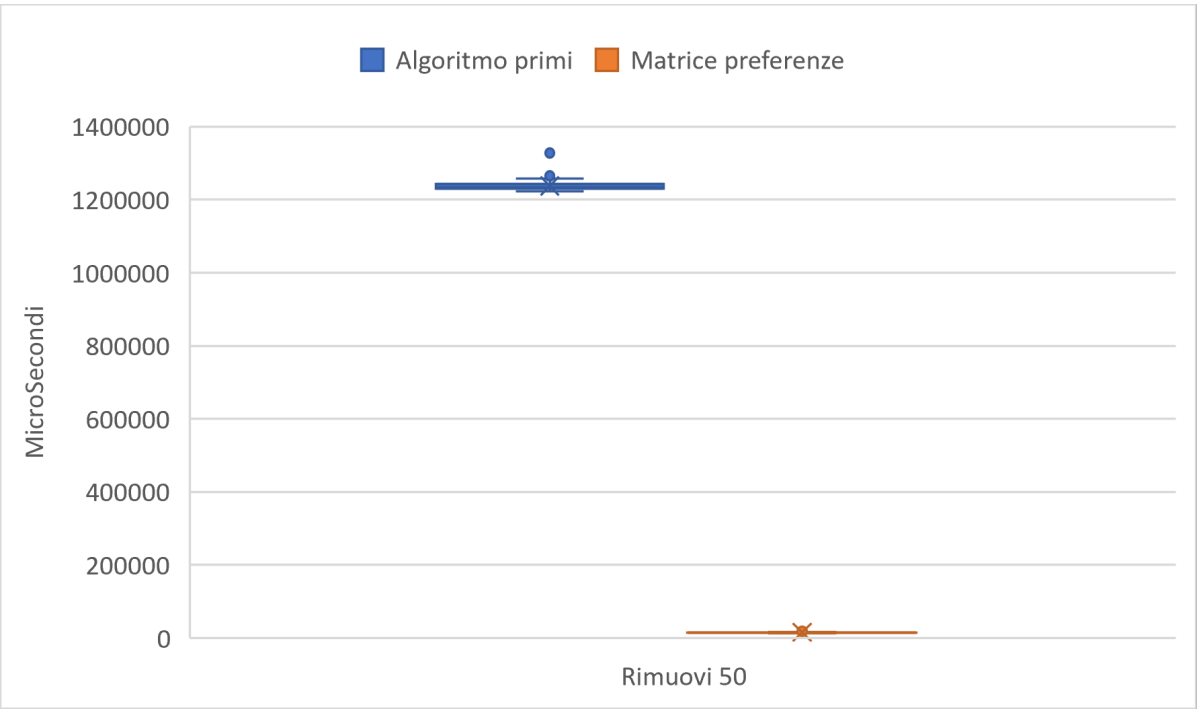


Figura 4.11: Rimozione in caso pessimo per entrambi

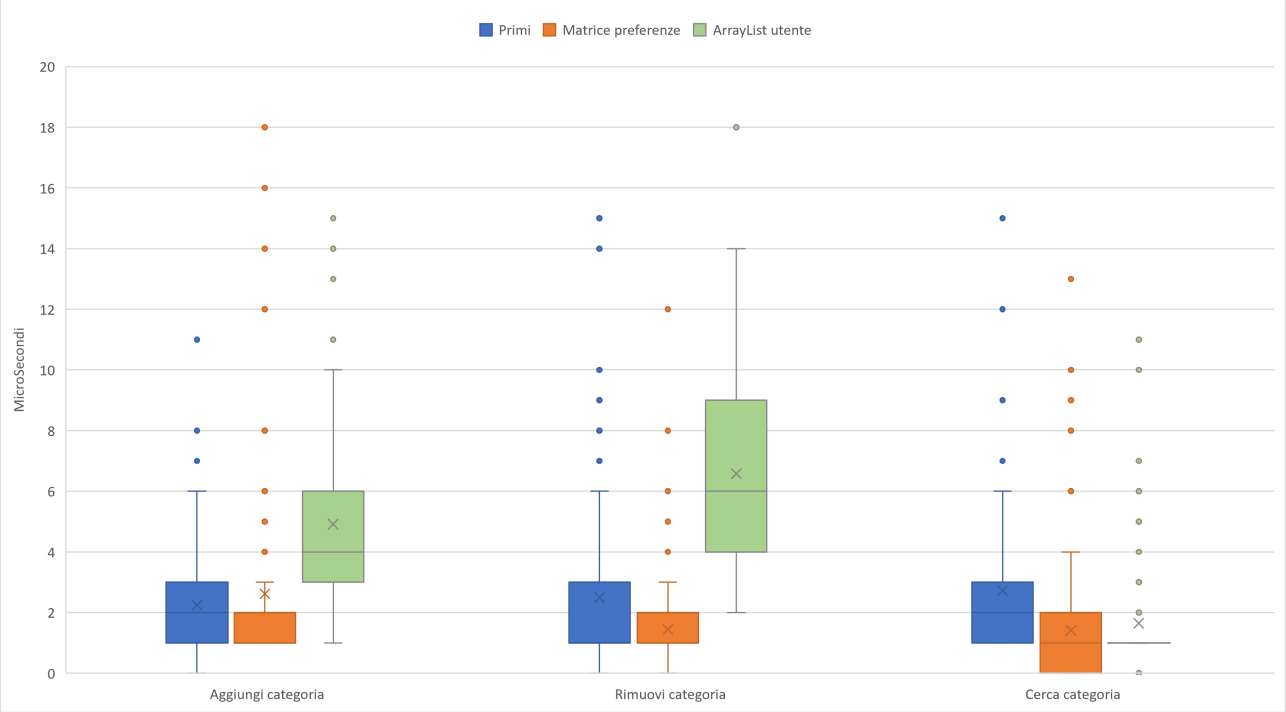


Figura 4.12: Esempio grafico di come vengono salvati i dati

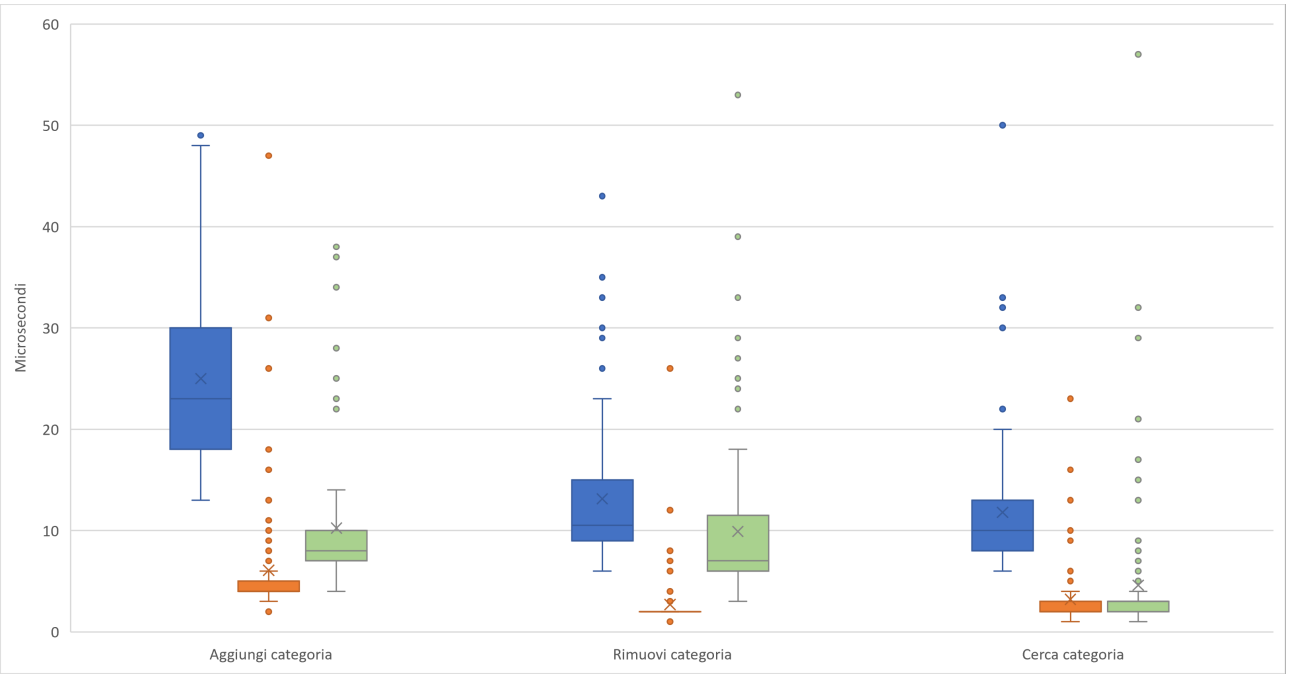


Figura 4.13: Esempio grafico di come vengono salvati i dati

Capitolo 5

Etica

Tutto questo ha un costo? Ovviamente sì! Ed è un costo che stiamo pagando noi. Questo perché, citando l'ex design ethicist di Google e cofondatore del Center for Human Technology, Tristan Harris, "Se non stai pagando per un prodotto, allora il prodotto sei tu". Sempre più servizi sono offerti gratuitamente, ma in realtà non è così: le aziende mirano a comprare la nostra attenzione senza pagarla. Noi siamo il prodotto, i nostri dati, le nostre informazioni sono il prodotto, questo è un mercato nuovo da miliardi di dollari. La raccolta dei dati degli utenti e il loro utilizzo da parte di aziende e organizzazioni sono diventati un argomento di grande preoccupazione etica in questi ultimi anni. Molte aziende vendono i dati degli utenti a terze parti per scopi pubblicitari o di marketing. Questo può comportare il rischio di abusi della privacy, come la profilazione senza il consenso dell'utente, la diffusione di informazioni personali sensibili e l'utilizzo improprio dei dati.
Siti Internet, app, video vengono progettati per catturare l'attenzione e farla durare nel tempo. Questa attenzione è di fondamentale importanza per chiunque debba vendere qualcosa. Se persone o brand hanno tanta attenzione (basti pensare agli influencer per esempio) non hanno problemi a ottenere denaro in grandi quantità. E questo perché chi li segue è disposto a fare molto per accontentare le loro richieste e per sostenere il loro progetto. Goldhaber [1], nel suo "The attention and the Net", sostiene che se vogliamo pagare una persona per darci attenzione questo non funziona. Ricompensare con del denaro una persona per leggere un articolo non vuol dire avere un'attenzione continua. Tutto il web design, la user experience (ossia lo studio dell'esperienza dell'utente e delle interazioni che ha con il prodotto digitale), la stessa SEO (ottimizzazione di siti, testi e pagine per essere trovata dai motori di ricerca) hanno a che fare con questo. Nel marketing e per il marketing l'attenzione è tutto.
E questo è solo la punta dell'iceberg, il caso Cambridge Analytica nel 2018 ha portato alla ribalta queste preoccupazioni etiche. La società di analisi dei dati aveva raccolto informazioni su milioni di utenti di Facebook senza il loro consenso e le aveva utilizzate per influire sulle elezioni politiche. Questo ha sollevato domande sulle pratiche di raccolta

dei dati e sul modo in cui vengono utilizzati da parte delle aziende.
È importante che le aziende e le organizzazioni si impegnino a proteggere la privacy degli utenti e ad utilizzare i loro dati in modo etico e trasparente.

5.1 The social dilemma

In un mondo sempre più digitale, i social media sono diventati una parte integrante della vita di molte persone. Da semplici piattaforme per la condivisione di foto e pensieri, sono diventati mezzi potenti di comunicazione e di influenza sulla società. Ma questo potere ha anche un lato oscuro: l'*addiction*, ossia la dipendenza dai social. Gli utenti trascorrono ore e ore sui social, controllando costantemente le loro notifiche, le foto dei loro amici e le ultime tendenze. Questo atteggiamento compulsivo è stato largamente condannato dalla comunità scientifica, che sottolinea come l'uso eccessivo dei social media possa avere effetti negativi sulla salute mentale e sulla qualità della vita delle persone. I social media competono per la nostra attenzione, il modello imprenditoriale di queste società è tenere le persone il più possibile attaccate allo schermo, per farlo devono suggerire il miglior contenuto possibile, provando a predire il comportamento umano e il continuo rilascio di dopamina. La dopamina è un neurotrasmettitore che è coinvolto in molte funzioni cerebrali, tra cui la motivazione, il piacere e la ricompensa. Quando una persona utilizza i social media e il cervello ottiene risposte positive, come mi piace o lo *scroll* infinito, viene rilasciata dopamina, creando un'associazione positiva tra l'utilizzo dei social media e il piacere. Questo processo può portare a un'escalation nell'utilizzo dei social media e a una dipendenza, poiché il cervello desidera continuare a ottenere queste scariche di dopamina. La struttura dei social media in sé, è basata per dare un rilascio continuo di dopamina in modo facile e veloce, come i like, lo scroll infinito oppure il sistema di notifica. Il bisogno di dopamina ha portato i social media ad evolversi, passando da Facebook dove i post erano per lo più testo a oggi dove tutti i social si stanno spostando ad un modello simile a TikTok, ovvero contenuti video, brevi che devono catturare subito l'attenzione. Questo ha anche causato di incredibile abbassamento dell'attenzione, passando nel 2010 a un tempo massimo di concentrazione di 3 minuti a oggi che è di 40 secondi. Questo *overload* di informazione era stato predetto da Herbert Simon[8], premio Nobel per l'Economia nonché psicologo, già nel secolo scorso quando ipotizzò che l'attenzione fosse il collo di bottiglia dell'essere umano: "L'informazione consuma attenzione. Quindi l'abbondanza di informazione genera una povertà di attenzione e induce il bisogno di allocare quell'attenzione efficientemente tra le molte fonti di informazione che la possono consumare".

5.2 Caso Cambridge analityca

I social media possono influenzare il comportamento e le emozioni nel mondo reale dell'utente, il tutto senza mai innescare la consapevolezza dell'utente, lo dimostra il caso Facebook-Cambridge Analytica, uno dei più grandi scandali di privacy degli ultimi anni. Nel 2018 fu rivelato che Cambridge Analytica aveva raccolto i dati personali di 87 milioni di account Facebook senza il loro consenso e li aveva usati per scopi di propaganda politica, in particolare la campagna elettorale di Donald Trump e per la Brexit, questo fatto ha messo in luce i pericoli della raccolta e dell'utilizzo dei dati personali da parte delle aziende.
La società di analisi dei dati, fondata nel 2013, aveva acquisito informazioni su milioni di utenti di Facebook senza il loro consenso. Questi dati erano stati utilizzati per creare profili dettagliati sulla personalità e le preferenze politiche degli utenti, che a loro volta erano stati utilizzati per influire sulle elezioni politiche, venivano mostrati annunci a categorie segmentate di utenti, in particolare nel caso della campagna elettorale di Trump del 2016 mostrava contenuti propagandistici agli individui che fossero sostenitori di Trump o potenziale voti oscillanti, questo ha sollevato domande sulle pratiche di marketing politico e sulla necessità di regolamentare questo settore.
Dopo lo scandalo, Cambridge Analytica è fallita e molti utenti di Facebook hanno cominciato a prendere maggiormente coscienza della loro privacy e del modo in cui i loro dati vengono utilizzati. Inoltre, ci sono stati sforzi a livello globale per regolamentare la raccolta e l'utilizzo dei dati personali da parte delle aziende e per garantire che gli utenti abbiano il controllo sui loro dati. Le aziende devono essere responsabili e trasparenti nella raccolta e nell'utilizzo dei dati, e gli utenti devono essere informati e in grado di controllare i loro dati personali.

5.3 Fake news

Un altro grande problema dei post che consigliati sono le fake news, in particolare su teoria complottiste come i vaccini, scie chimiche e 5G. Le fake news, ovvero notizie false o distorte, sono un esempio di come i social media possano influenzare le opinioni delle persone. Queste notizie possono essere create intenzionalmente per manipolare l'opinione pubblica o diffondere informazioni false, ma possono anche essere il risultato di una semplice disinformazione o di un errore umano. Le fake news sono particolarmente pericolose perché possono essere condivise e diffuse molto rapidamente sui social media, raggiungendo un pubblico enorme in pochissimo tempo, uno studio del MIT dice che una fake news si diffonde sei volte più velocemente di una notizia vera[12]. Questo può portare a una diffusione incontrollata di informazioni false, che può a sua volta influire sulle opinioni e sulle decisioni delle persone. Ad esempio, la quantità di informazioni negative che vediamo sui social media può portare a una percezione distorta della realtà,

facendoci credere che il mondo sia più pericoloso o negativo di quanto non sia in realtà. Per questo motivo, è importante essere consapevoli delle informazioni che vediamo sui social media e verificare sempre la loro veridicità prima di condividerle o di basare le nostre opinioni su di esse. Inoltre, è importante che i social media stessi prendano provvedimenti per combattere la diffusione delle fake news e per garantire che le informazioni che circolano sulla loro piattaforma siano accurate e affidabili.
In conclusione, i social media sono diventati una parte inestricabile della vita quotidiana di molte persone, ma c'è anche un prezzo da pagare per questo potere. Se non utilizzati in modo responsabile, possono avere conseguenze negative per la società e per l'individuo. È importante essere consapevoli di questo potere e utilizzare i social media in modo responsabile e informato.

5.4 Raccolta, salvataggio e privacy dei dati dell'utente: GDPR

Il Regolamento Generale sulla Protezione dei Dati (General Data Protection Regulation, GDPR)[10] è una legge europea che è entrata in vigore il 25 maggio 2018. Il GDPR stabilisce le regole per la protezione dei dati personali dei cittadini dell'UE e garantisce che le informazioni personali siano trattate in modo responsabile e protette da eventuali perdite o abusi. La legge si applica a tutte le aziende che operano nell'UE, indipendentemente da dove i dati siano elaborati o conservati. Il GDPR stabilisce il diritto all'informazione, al controllo, alla cancellazione, alla portabilità e all'oblio dei dati personali degli individui, oltre a stabilire sanzioni severe per le aziende che non rispettano le sue disposizioni.
Il GDPR definisce il "dato personale" come qualsiasi informazione che riguarda un individuo identificato o identificabile, come nome, indirizzo, indirizzo email, numero di telefono e dati biometrici. Il GDPR richiede che le aziende siano trasparenti riguardo alla raccolta e all'utilizzo di questi dati e che ottengano il consenso esplicito degli individui per farlo.
Inoltre, il GDPR stabilisce il diritto degli individui di sapere quali informazioni sono state raccolte su di loro e come vengono utilizzate. Gli individui hanno il diritto di richiedere la correzione o la cancellazione di questi dati e hanno il diritto di trasferire i loro dati a un altro fornitore di servizi.
Le aziende che non rispettano il GDPR possono essere soggette a sanzioni fino a 20 milioni di euro o al 4% del fatturato mondiale annuo, a seconda di quale importo sia maggiore. Il GDPR richiede inoltre alle aziende di nominare un Responsabile della Protezione dei Dati (Data Protection Officer, DPO) per garantire il rispetto delle sue disposizioni.
In sintesi, il GDPR mira a proteggere i diritti fondamentali della privacy degli individui e a garantire che le aziende trattino i dati personali in modo responsabile e sicuro.

Capitolo 6

Conclusioni

In questa tesi si è discusso sulle reti sociali, ovvero una rete di persone che si identifica un gruppo di individui connessi, in questo caso utilizzando i social media ovvero strumenti che permettono collegamenti sociali e comunicazioni interattive tra gli utenti scambiandosi varie tipologie di contenuti: testuali, audio e video. Alla base del funzionamento dei social media ci sono gli algoritmi di raccomandazione. Strumenti usati suggerire contenuti pertinenti agli utenti in base alle loro preferenze, ai loro interessi e al loro comportamento di navigazione.

La profilazione utente è un aspetto importante dei social media e degli algoritmi di raccomandazione. Per profilare gli utenti, vengono raccolti e analizzati i loro dati personali, le loro interazioni sociali, i loro interessi e le loro preferenze di navigazione. La profilazione degli utenti consente ai social media di fornire contenuti altamente personalizzati e rilevanti, ma solleva anche preoccupazioni per la privacy e la sicurezza dei dati.

Gli algoritmi di raccomandazione utilizzati dai social media sono di diversi tipi, tra cui filtri basati sul contenuto, filtri collaborativi e metodi ibridi. Il filtro basato sul contenuto suggerisce contenuti simili a quelli che l'utente ha già visualizzato, mentre il filtro collaborativo suggerisce contenuti in base alle scelte degli utenti con gusti simili. Il metodo ibrido combina i due approcci per offrire suggerimenti ancora più personalizzati.

Successivamente si è parlato di un nuovo algoritmo di clustering per il salvataggio dei dati basato sui numeri primi. Questo elaborato ha dimostrato che questa tecnica ha delle potenzialità, ma non è possibile giungere ad un giudizio definitivo in quanto per valutare un algoritmo servono molti più dati e molta più capacità di calcolo. Spero che questa tesi possa essere il punto d'inizio di uno studio più approfondito in quest'ambito.

In conclusione si è parlato della parte oscura dei social media sollevando anche preoccupazioni etiche riguardo alla loro trasparenza, equità e manipolazione. Vengono trattati i vari casi etici che hanno colpito i social media come il caso Cambridge Analityca, alla dipendenza che puntano ad avere e la propagazione delle fake news. Infine, si parla come la raccolta, il salvataggio e la privacy dei dati dell'utente sono regolati da leggi come il

GDPR (General Data Protection Regulation) in Europa. Le piattaforme di social media devono rispettare le leggi sulla privacy e garantire che i dati degli utenti siano trattati in modo etico e sicuro.

Bibliografia

[1] Michael H. Goldhaber. "The attention economy and the Net". In: *First Monday* 2 (1997).

[2] Google. *Sistemi di suggerimento*. URL: https://developers.google.com/machine-learning/recommendation?hl=it.

[3] Chris Meyer Ian MacKenzie e Steve Noble. *How retailers can keep up with consumers*. Rapp. tecn. Accenture Interactive, 2013.

[4] Scott Tieman Jeriad Zoghby e Javier Pérez Moiño. *Making it personal*. Pulse Check. Accenture Interactive, 2018.

[5] Sumitkumar Kanoje, Sheetal Girase e Debajyoti Mukhopadhyay. "User profiling trends, techniques and applications". In: *arXiv* (2015).

[6] James McInerney et al. "Explore, exploit, and explain: personalizing explainable recommendations with bandits". In: *Proceedings of the 12th ACM conference on recommender systems*. 2018, pp. 31–39.

[7] Microsoft. *Creare un sistema di raccomandazione basato sul contenuto*. URL: https://learn.microsoft.com/it-it/azure/architecture/solution-ideas/articles/build-content-based-recommendation-system-using-recommender.

[8] Herbert A. Simon. *Il labirinto dell'attenzione. Progettare organizzazioni per un mondo ricco di informazioni*. Luca Sossella Editore, 2019.

[9] Jiliang Tang, Yi Chang e Huan Liu. "Mining social media with social theories: a survey". In: *ACM Sigkdd Explorations Newsletter* 15.2 (2014), pp. 20–29.

[10] UE. *GDPR*. URL: https://www.garanteprivacy.it/regolamentoue.

[11] Mads Vestergaard Vincent F. Hendricks. *Reality Lost*. Springer Cham, 2019.

[12] Soroush Vosoughi, Deb Roy e Sinan Aral. "The spread of true and false news online". In: *science* 359.6380 (2018), pp. 1146–1151.

[13] Sheng Yu e Subhash Kak. "A survey of prediction using social media". In: *arXiv* (2012).