UNIVERSITÀ DEGLI STUDI DI PALERMO

DIPARTIMENTO DI INGEGNERIA

CORSO DI LAUREA MAGISTRALE IN INGEGNERIA INFORMATICA

Progettazione e sviluppo di tecniche avversarie di attacco e difesa per sistemi di raccomandazione basati su LLM

TESI DI LAUREA DI
GIORGIO DI CRISTOFALO

RELATORE
Prof. ALESSANDRA DE PAOLA

CONTRORELATORE
Ch.mo Prof. ANTONIO CHELLA

CORRELATORE
Prof. MARCO MORANA

ANNO ACCADEMICO 2023 – 2024

UNIVERSITÀ DEGLI STUDI DI PALERMO
DIPARTIMENTO DI INGEGNERIA

LAUREA MAGISTRALE IN INGEGNERIA INFORMATICA

PROGETTAZIONE E SVILUPPO DI TECNICHE AVVERSARIE DI ATTACCO E DIFESA PER SISTEMI DI RACCOMANDAZIONE BASATI SU LLM

Tesi di Laurea di
Giorgio Di Cistofalo

Relatore:
Prof. Alessandra De Paola

Controrelatore:
Ch.mo Prof. Antonio Chella

Correlatore:
Prof. Marco Morana

Sommario

Questa tesi affronta il tema della sicurezza nei sistemi di raccomandazione basati su *Large Language Models* (LLMs), concentrandosi in particolare sulle vulnerabilità di questi sistemi dagli attacchi avversari. L'adozione sempre più diffusa di modelli avanzati come BERT, GPT e in particolare *BERT4Rec* ha rivoluzionato il campo delle raccomandazioni personalizzate, migliorando l'accuratezza e la capacità di adattamento ai bisogni degli utenti. Tuttavia, l'integrazione di questi modelli introduce anche nuove minacce, tra cui attacchi di *model extraction* e *data poisoning*, che possono compromettere la qualità delle raccomandazioni e mettere a rischio la sicurezza dei dati. L'ipotesi di ricerca di questa tesi è che sia possibile sviluppare un'architettura difensiva efficace, in grado di rilevare e contrastare gli attacchi avversari senza impattare negativamente sulle prestazioni del sistema.

Per verificare questa ipotesi, la ricerca è stata articolata in più fasi, a partire da un'analisi approfondita dello stato dell'arte sui sistemi di raccomandazione e sulle loro vulnerabilità. Successivamente, sono stati studiati gli attacchi più diffusi nell'ambito dell'*Adversarial Machine Learning* e analizzate le loro implicazioni nei contesti reali. In risposta a queste minacce, è stato progettato un modello difensivo denominato *BERT4Def*, basato su architettura *Transformer* in grado di distinguere tra utenti legittimi e attaccanti attraverso l'uso di modelli surrogati. L'efficacia della soluzione proposta è stata valutata sperimentalmente su dataset reali, tra cui ML-1M

e Beauty, misurando le prestazioni del sistema in termini di accuratezza delle raccomandazioni e capacità di resistere a tentativi di manipolazione.

I risultati dimostrano che gli attacchi avversari possono alterare significativamente il comportamento dei sistemi di raccomandazione, compromettendone l'affidabilità e l'efficacia. Tuttavia, l'integrazione di BERT4Def ha permesso di ridurre drasticamente l'impatto degli attacchi, migliorando la sicurezza del sistema senza sacrificare la qualità delle raccomandazioni. L'analisi critica ha inoltre evidenziato la necessità di un equilibrio tra protezione e prestazioni, sottolineando l'importanza di strategie difensive che non introducano costi computazionali eccessivi.

Questa ricerca rappresenta un passo avanti nella protezione dei sistemi di raccomandazione basati su LLM, proponendo una soluzione innovativa che combina tecniche di difesa avanzate con un'analisi dettagliata delle minacce emergenti. Le conclusioni ottenute aprono la strada a future indagini su meccanismi di sicurezza ancora più sofisticati, contribuendo al miglioramento della robustezza dei sistemi di intelligenza artificiale.

Indice

Capitolo 1

Introduzione

I sistemi di raccomandazione rappresentano una delle soluzioni tecnologiche più avanzate per la gestione della crescente quantità di informazioni nell'era digitale [7]. Il loro scopo principale è fornire suggerimenti personalizzati agli utenti, ottimizzando l'interazione con piattaforme di e-commerce, servizi di streaming e social network. Questi sistemi sono progettati per migliorare l'esperienza utente e aumentare la fidelizzazione, offrendo contenuti pertinenti basati su modelli predittivi sofisticati [13]. Inizialmente, tali sistemi si basavano su tecniche di *Collaborative Filtering* e *Content-Based Filtering*. Il Collaborative Filtering si basa sull'analisi del comportamento e le preferenze degli utenti per generare raccomandazioni, in particolare questo approccio identifica somiglianze tra utenti o elementi, suggerendo contenuti che hanno ottenuto valutazioni positive da utenti con gusti simili [9]. Di contro il *Content-Based Filtering* si concentra sulle caratteristiche intrinseche degli elementi da raccomandare, confrontandole con i profili degli utenti. Quindi ogni utente riceve suggerimenti basati sulle caratteristiche dei contenuti che ha precedentemente apprezzato [37].

L'avvento dei Large Language Models (LLMs) ha determinato un cambiamento radicale nel paradigma della raccomandazione [60]. Grazie alla loro architettura basata sul Multi-Head Self-Attention [54], riescono a comprendere il contesto delle interazioni con grande successo, rivoluzionando i sistemi di raccomandazione e superando limiti storici come il problema del cold start e la scarsità di dati. Modelli avanzati come BERT, GPT e in particolare BERT4Rec consentono una personalizzazione più precisa, arricchendo l'analisi semantica dei contenuti e migliorando l'efficacia dei suggerimenti proposti agli utenti. BERT4Rec impiega un approccio avanzato di modellazione bidirezionale delle sequenze di interazioni, sfruttando un'architettura

derivata da BERT per prevedere con maggiore accuratezza gli elementi futuri che un utente potrebbe apprezzare. Questo meccanismo consente di ottimizzare la qualità delle raccomandazioni in contesti sequenziali, migliorando la capacità del sistema di adattarsi dinamicamente alle preferenze degli utenti [52]. Tuttavia, l'adozione di questi modelli introduce anche nuove vulnerabilità, in particolare in termini di sicurezza e robustezza. Il campo dell'Adversarial Machine Learning (AML) [4] ha dimostrato che anche i sistemi più avanzati possono essere soggetti a manipolazioni malevole. Tra gli attacchi più significativi figurano la *model extraction*, che consiste nella capacità di replicare il modello del sistema di raccomandazione, e il *data poisoning*, che prevede l'inserimento di utenti malevoli all'interno del dataset di riaddestramento al fine di compromettere l'integrità del sistema e alterare i risultati delle raccomandazioni. Questi attacchi possono compromettere l'affidabilità delle raccomandazioni, con conseguenze significative sia per gli utenti che per i provider del servizio, la possibilità che un attaccante manipoli il sistema per influenzare i suggerimenti rappresenta una minaccia concreta che richiede soluzioni innovative di difesa.

Questa ricerca affronta il problema della sicurezza nei sistemi di raccomandazione sequenziali basati su Large Language Models (LLM), come BERT4Rec, proponendo un approccio difensivo mirato a mitigare gli attacchi avversari di data poisoning senza comprometterne l'efficacia. Poiché il sistema di raccomandazione espone i propri risultati, si assume che chiunque possa effettuare un'operazione di model extraction, replicando il modello e potenzialmente sfruttandolo per scopi malevoli. La metodologia proposta si basa sull'architettura Transformer per distinguere tra utenti legittimi e attaccanti, sviluppando un modello in grado di rilevare e contrastare le manipolazioni senza alterare le prestazioni del sistema. Questo approccio ha portato alla progettazione di BERT4Def, un framework difensivo che consente ai sistemi di raccomandazione di operare in ambienti ostili riducendo il rischio di compromissione e garantendo al contempo un'elevata qualità delle raccomandazioni.

Inoltre, si analizzano criticamente i rischi associati all'implementazione di strategie di difesa. Il bilanciamento tra sicurezza e prestazioni è una sfida complessa, che impone di valutare attentamente il compromesso tra accuratezza delle raccomandazioni e capacità di rilevare minacce avversarie. Pertanto, lo studio non si limita a proporre un modello di difesa, ma ne esamina anche i limiti e le potenziali implicazioni, considerando diversi scenari di attacco e valutando l'efficacia della soluzione in ambienti simulati. A tal fine viene proposto un attacco in grado di superare marginalmente le difese, cosi da mettere in luce i possibili rischi in cui si incorre all'utilizzo *BERT4Def*.

La tesi è strutturata in tre capitoli principali. Il secondo capitolo presenta lo stato dell'arte, con un'analisi approfondita sui sistemi di raccomandazione, l'evoluzione introdotta dai LLMs e le loro vulnerabilità dagli attacchi avversari. Verranno discussi i principi teorici che guidano il funzionamento di questi sistemi, le strategie di personalizzazione e i principali approcci utilizzati per generare suggerimenti personalizzati. Inoltre, verranno illustrate le vulnerabilità più comuni e le strategie note per sfruttarle. Nel terzo capitolo viene illustrata la metodologia adottata, descrivendo il modello di difesa sviluppato per *BERT4Def*, le strategie implementate per contrastare le minacce e una possibile modalità d'attacco proprio contro *BERT4Def*. Infine, il quarto capitolo presenta gli esperimenti condotti e i risultati ottenuti, fornendo un'analisi critica dell'efficacia della soluzione proposta. Saranno illustrati i dataset utilizzati, le metriche di valutazione e i risultati sperimentali, con particolare attenzione all'impatto delle misure di sicurezza sulla qualità delle raccomandazioni. Verranno, inoltre, discusse le implicazioni pratiche risultanti e le possibili evoluzioni future della ricerca in questo ambito.

Capitolo 2

Stato dell'arte

2.1 Sistemi di Raccomandazione

2.1.1 Definizione e Finalità

I sistemi di raccomandazione nascono come risposta a una delle sfide più rilevanti dell'era digitale: la gestione dell'enorme quantità di informazioni a disposizione degli utenti. Con la diffusione di internet, il volume di contenuti online è aumentato in maniera vertiginosa, spaziando da prodotti su piattaforme di e-commerce, a film e serie su servizi di streaming, fino ad articoli di notizie e post sui social media. Questa esplosione di informazioni ha portato al fenomeno noto come *information overload* [7], ovvero una situazione in cui gli individui sono sopraffatti dalla quantità di opzioni disponibili, con la conseguenza di rendere difficile individuare contenuti realmente pertinenti o di interesse personale. L' *information overload* non solo causa disorientamento, ma aumenta anche l'incapacità di scelta, poiché gli utenti devono costantemente scegliere tra un numero spropositato di alternative. Per risolvere questo problema, i sistemi di raccomandazione sono stati sviluppati con l'obiettivo di semplificare la navigazione attraverso l'ingente quantità di contenuti disponibili [30, 13].

Questi sistemi sono progettati per filtrare e personalizzare le informazioni presentate agli utenti, riducendo così il sovraccarico informativo. Grazie a un sofisticato processo di analisi dei dati, i sistemi di raccomandazione sono in grado di elaborare informazioni su diversi aspetti del comportamento degli utenti, come la cronologia delle visualizzazioni o degli acquisti, i dati demografici, le recensioni e le valutazioni fornite. Tutto ciò viene utilizzato per formulare previsioni su ciò che potrebbe piacere o essere utile a un determinato utente in futuro. Il risultato

è un'esperienza personalizzata e su misura, che non solo rende più agevole la fruizione dei contenuti, ma migliora anche l'interazione con le piattaforme.

Il valore dei sistemi di raccomandazione non si limita all'aspetto della soddisfazione dell'utente: essi rappresentano un vantaggio strategico per le aziende, in particolare per quanto riguarda l'aumento dell'engagement e delle vendite, più un sistema è in grado di fornire suggerimenti mirati, maggiore sarà il tempo che un utente trascorrerà su una piattaforma, esplorando prodotti o contenuti che altrimenti non avrebbe considerato [29]. Questo si traduce in un miglioramento della *retention*, ossia la capacità di trattenere gli utenti, e di conseguenza in un potenziale incremento delle vendite o della monetizzazione.

L'e-commerce, ad esempio, utilizza questi sistemi per suggerire prodotti basati sugli acquisti precedenti, sul feedback delle raccomandazioni, sulla correlazione degli utenti, aumentando così la probabilità che un utente continui a fare acquisti. Allo stesso modo, le piattaforme di streaming come Netflix e Spotify utilizzano raccomandazioni per mantenere alto l'interesse degli utenti, offrendo contenuti che si allineano ai loro gusti, riducendo così il rischio che smettano di utilizzare il servizio [49]. Nel resto del capitolo descriveremo e discuteremo quali sono i tipi (2.1.2) di sistemi di raccomandazione più in voga al momento e ne analizzeremo sfide e limiti (2.1.3).

2.1.2 Tassonomia

Di seguito viene illustrata una panoramica di alcune tipologie di sistemi di raccomandazione presenti in letteratura (si veda figura 2.1).

Collaborative Filtering

Un sistema di raccomandazione basato su *Collaborative Filtering (CF)* prevede le preferenze di un utente in base ai comportamenti collettivi degli utenti che usufruiscono del servizio. Il principio di base è che gli utenti con preferenze simili in passato saranno probabilmente d'accordo in futuro [48].

Il CF non necessita di attributi specifici per gli articoli, il che lo rende particolarmente utile in domini in cui tali metadati sono scarsi o difficili da quantificare, come i film, la musica o le recensioni generate dagli utenti. In generale prevede 4 fasi di lavoro descritte di seguito

- **Collezione dei dati** In questa fase si acquisiscono per vie dirette (ratings) o indirette (click, cronologia, ecc) le interazioni dell'utente con i prodotti.

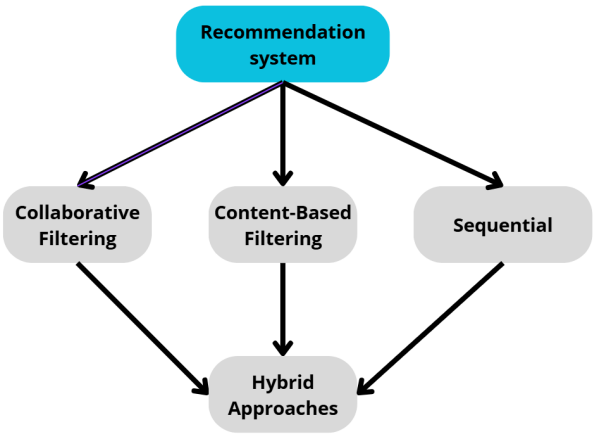

Figura 2.1: Tassonomia dei RS analizzati

- **Rappresentazione** Si costruisce una matrice dove le righe rappresentano gli utenti e le colonne i prodotti. I valori all'interno di questa matrice rappresentano i dati raccolti e quindi le interazioni utente-prodotto. La cella vuota, invece, indica l'assenza dell'interazione.
- **Tecniche di analisi** Lo scopo di questa fase è trovare le similarità. Le tecniche si suddividono in due macro-approcci.
  - **Memory-Based Approaches:** calcolale similarità tra utenti o tra prodotti sulla base dei dati raccolti all'interno della matrice.
  - **Model-Based Approaches:** usa modelli predittivi come Matrix Factorization o modelli di Machine Learning per estrarre caratteristiche latenti dai dati.
- **Generazione raccomandazioni** Il sistema prevede le valutazioni mancanti o classifica gli articoli in base alla rilevanza per consigliarli agli utenti.

Content-Based Filtering

I *Recommender Systems* basati su *Content-Based Filtering (CBF)* cercano di abbinare gli utenti ad articoli simili a quelli che sono piaciuti in passato. Questa somiglianza non si basa necessariamente sulle correlazioni di valutazione tra gli utenti, ma sulla base degli attributi degli oggetti apprezzati dall'utente. A differenza dei sistemi collaborativi, che sfruttano esplicitamente le valutazioni degli altri utenti oltre a quelle dell'utente target, i sistemi basati sui contenuti si

concentrano principalmente sulle valutazioni dell'utente target e sugli attributi degli oggetti graditi dall'utente [1]. Il CBF prevede un flusso di lavoro composto da 3 fasi

- **Preprocessing ed estrazione caratteristiche** In questa prima fase vengono estratti dai protoddi tutte le caratteristiche necessarie su cui poi effettuare il calcolo di similarità. Per dati strutturati potrebbero essere gli aggettivi dell'oggetto in questione. Per dati non strutturati come film e libri si estraggono informazioni dalle descrizioni, come keyword o attraverso tecniche di NLP si estraggono i topic principali, i nomi delle entità chiave, ecc.
- **Creazione Profilo Utente** In questa fase si strutturano le relazioni User-Item, questo come nel CF, puo essere sbolto tramite Feedback diretti e indiretti.
- **Generazioni Raccomandazioni** Il sistema in questa fase calcola la similarità tra i prodotti di cui l'utente ha condiviso un feedback e i prodotti di cui l'utente non ha feedback. I più simili vengono proposti.

Sequential

Sequential Recommender Systems (SRS) sono modelli di raccomandazione avanzati progettati per catturare le preferenze in evoluzione degli utenti analizzando l'ordine sequenziale delle loro interazioni. A differenza dei sistemi tradizionali che considerano le preferenze statiche degli utenti, gli SRS incorporano dinamiche temporali, riconoscendo che gli interessi degli utenti cambiano nel tempo [6].

Gli aspetti principali che i SRS tengono in considerazione sono:

- **Interazioni User–Item:** memorizzare le interazioni tra utenti e item.
- **Dipendenze Temporali:** memorizzare l'ordine e quindi la dipendenza temorale delle interazioni.
- **Preferenze a lungo termine vs breve termine:** differenziare le sessioni a lungo termine da quelle a breve termine.

I SRS generalmente hanno il seguente flusso di lavoro

- **Rappresentazione dei Dati:** Le interazioni degli utenti sono strutturate come sequenze (ad esempio, clic su articoli o acquisti in ordine cronologico). Possono essere integrati metadati, timestamp o informazioni contestuali.

- **Modellazione delle Sequenze:** I modelli predicono il prossimo elemento con cui l'utente è probabile che interagisca, basandosi sulle sequenze passate, le dinamiche temporali vengono catturate per identificare tendenze e preferenze.
- **Tecniche Utilizzate:**
  - **Metodi Tradizionali:** Includono Catene di Markov o la Fattorizzazione della Matrice [39].
  - **Approcci Avanzati:** Sfruttano reti neurali come RNN, meccanismi basati sull'attenzione o strutture basate su grafi.
- **Predizione e Raccomandazione:** Il sistema predice gli articoli più probabilmente interessanti per l'utente basandosi sulla sua storia sequenziale. Genera raccomandazioni personalizzate.

Per una tassonomia sui tipi esistenti di sistemi di raccomandazione fare riferimento alla Fig.2.2

I sistemi di raccomandazione sequenziali possono essere classificati in diverse tipologie basate sugli approcci di modellazione, di seguito viene riportata una breve descrizione tratta da [6]:

- **Approcci Tradizionali:**
  - **Catene di Markov:**
    - Predicono il prossimo elemento basandosi sulle interazioni più recenti dell'utente.
    - Efficaci per catturare dipendenze a breve termine, ma limitate nell'affrontare preferenze a lungo termine.
  - **Fattorizzazione della Matrice con Modellazione delle Sequenze:**
    - Combina modelli di fattori latenti utente-oggetto con dinamiche temporali [45].
    - Esempio: Metodi come *Fossil* integrano la fattorizzazione della matrice e le catene di Markov.
- **Approcci Basati su Reti Neurali:**

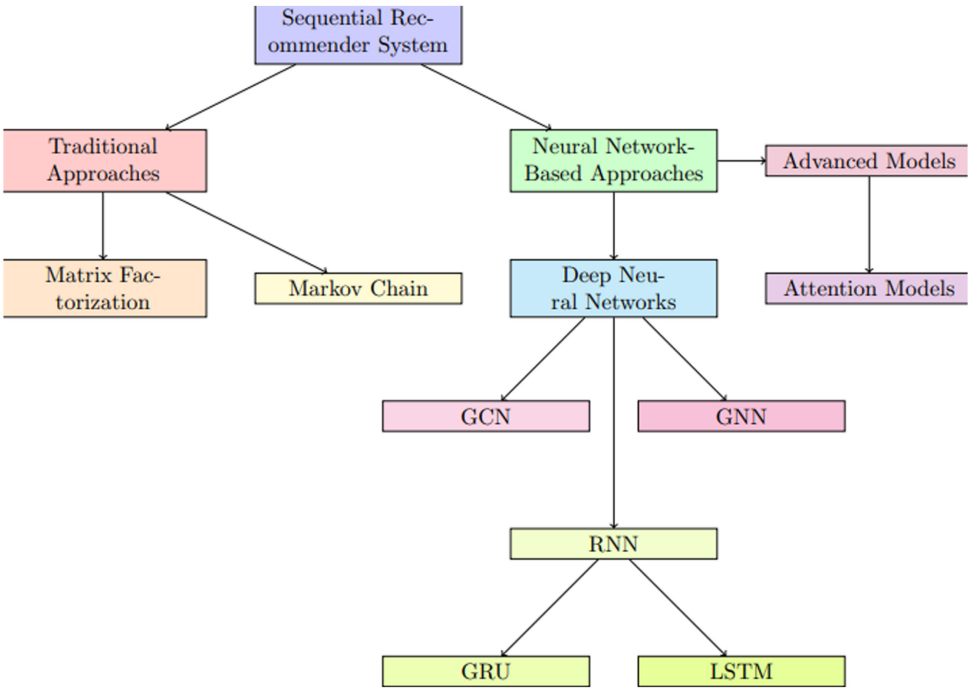

Figura 2.2: Tassonomia dei tipi di Sequential RS esistenti [6].

– **Reti Neurali Ricorrenti (RNN):**
  * Efficienti nella modellazione di dati sequenziali grazie alla capacità di catturare schemi temporali.
  * Varianti come *Long Short-Term Memory* (LSTM) [34] e *Gated Recurrent Units* (GRU) migliorano la gestione delle dipendenze a lungo termine.

– **Reti Neurali Convoluzionali (CNN):**
  * Trattano le sequenze utente-oggetto come "immagini" e utilizzano filtri per identificare schemi nelle sequenze di interazione.
  * Maggiormente adatte a catturare schemi globali piuttosto che relazioni strettamente ordinate.

– **Modelli Basati sull'Attenzione:**
  * Utilizzano meccanismi come i *Transformers* per valutare la rilevanza di diversi elementi in una sequenza.
  * Esempio: *BERT4Rec* modella sequenze bidirezionali per raccomandazioni più sfumate.

- **Approcci Basati su Grafi:**
  - Rappresentano le interazioni utente-oggetto come grafi, sfruttando le reti neurali basate su grafi (*Graph Neural Networks*, GNN) [58].
  - Catturano relazioni complesse tra interazioni, permettendo raccomandazioni spiegabili.
- **Approcci Ibridi:**
  - Combinano diverse tecniche, come l'integrazione di modelli neurali con metodi tradizionali.
  - Esempio: Meccanismi di attenzione integrati con RNN o GNN per una modellazione sequenziale migliorata.
- **Approcci Basati sull'Apprendimento Contrastivo:**
  - Utilizzano l'apprendimento contrastivo per migliorare l'accuratezza delle raccomandazioni identificando differenze distinte nelle preferenze degli utenti.

Hybrid Approaches

I *recommender system* che si basano su approcci ibridi sono una struttura avanzata progettata per ottimizzare il processo di raccomandazione unendo i punti di forza di diversi algoritmi e mitigando i loro limiti individuali. Questi sistemi operano combinando strategicamente più tecniche di raccomandazione per migliorare le loro prestazioni complessive, in particolare nell'affrontare sfide come i sparse data, il problema del cold-start e le dynamic user needs, di cui parleremo nel prossimo paragrafo [2].

Per comprendere la funzionalità dei sistemi ibridi, è essenziale approfondire i meccanismi delle loro metodologie di integrazione. I sistemi ibridi si basano sul principio della combinazione di diverse tecniche di raccomandazione, come il CF, il CBF e i SRS, per creare un sistema più robusto e flessibile. Questa integrazione può assumere varie forme, ciascuna adattata a un contesto o a un insieme di dati specifici.

Un approccio promettente è il metodo dell'*ensemble*, in cui i risultati di più recommender indipendenti vengono combinati in un'unica previsione unificata. Questo processo può comportare una semplice media o una media ponderata delle previsioni di diversi algoritmi. Ad

esempio, le tecniche *Content Based* e quelle basate su *Collaborative Filtering* possono produrre raccomandazioni separate, che vengono poi unite utilizzando una media ponderata per tenere conto dei rispettivi livelli di accuratezza o di fiducia.

Al contrario, i sistemi *monolitic* comportano un'integrazione più profonda, in cui il processo di raccomandazione è implementato come un singolo algoritmo che attinge da più fonti di dati. In questo caso, i componenti del sistema non sono facilmente separabili e l'algoritmo opera come un'unità coesa. Ad esempio, un sistema monolitico potrebbe analizzare contemporaneamente le preferenze dell'utente e i metadati degli articoli per generare raccomandazioni, senza distinguere tra gli aspetti collaborativi e quelli basati sui contenuti del suo funzionamento.

I sistemi di raccomandazione *ibridi* combinando diverse tecniche di raccomandazione, migliorano significativamente l'accuratezza e la robustezza, ottenendo un bilanciamento delle distorsioni e le varianze insite nei singoli modelli, garantendo che le raccomandazioni finali siano affidabili e contestualmente rilevanti. Inoltre, eccellono nell'affrontare i problemi del cold start, in quanto possono incorporare metodi come i CBF per fornire raccomandazioni iniziali in assenza di dati sufficienti sull'interazione con l'utente.

2.1.3 Sfide e Limiti

In questa sezione si descrivono in modo generale i tre problemi generali che affliggono i sistemi di raccomandazione facendo riferimento a [1, 2, 46].

Il problema del **cold start** è una sfida pervasiva per i sistemi di raccomandazione, particolarmente evidente quando si introducono nuovi utenti o articoli nella piattaforma. Questo problema sorge perché molti algoritmi di raccomandazione, in particolare il CF, si basano molto sui dati storici, come le interazioni utente-oggetto (ad esempio, valutazioni, acquisti o click). Quando un nuovo utente si iscrive, le informazioni sulle sue preferenze sono scarse o nulle. Allo stesso modo, quando un nuovo elemento viene aggiunto all'inventario, mancano le interazioni storiche necessarie al sistema per valutarne la rilevanza per gli utenti esistenti. I recommender basati sulla conoscenza e sul contenuto attenuano parzialmente questo problema, basandosi sulle preferenze o sugli attributi degli articoli forniti dagli utenti. Tuttavia, anche questi approcci hanno dei limiti, in quanto richiedono input espliciti da parte degli utenti, che non sempre sono disponibili, o metadati ricchi sugli articoli, che non sempre sono completi o sufficientemente descrittivi. Questo problema ha un impatto significativo sul coinvolgimento iniziale e sull'efficacia del sistema di

raccomandazione, in particolare in ambienti in cui la soddisfazione dell'utente è fondamentale fin dall'inizio.

La **Data sparsity** aumenta le sfide che i sistemi di raccomandazione devono affrontare per fornire suggerimenti accurati e significativi. Questo problema è particolarmente pronunciato nei sistemi con dati di interazione scarsi, in cui la matrice utente-elemento è per lo più vuota, con pochi utenti che hanno interagito con o valutato una piccola frazione degli articoli disponibili. La scarsità dei dati crea difficoltà agli algoritmi, in particolare a quelli che si basano sul filtraggio collaborativo, poiché la mancanza di modelli di co-occorrenza tra utenti e articoli compromette la capacità dell'algoritmo di stabilire somiglianze o correlazioni affidabili. Questa scarsità è spesso esacerbata in domini di nicchia o in sistemi che si rivolgono a basi di utenti altamente personalizzate e frammentate, dove le interazioni sono distribuite in modo non uniforme tra gli articoli. In questi scenari, anche tecniche avanzate come i modelli a fattori latenti o la fattorizzazione delle matrici possono faticare a generalizzare in modo efficace, a causa delle informazioni insufficienti disponibili per l'apprendimento di modelli robusti. Inoltre, la scarsità di dati spesso porta a un ciclo di feedback in cui gli articoli meno conosciuti o che interagiscono di rado diventano sempre più difficili da raccomandare, perpetuando la loro sottorappresentazione nelle interazioni degli utenti.

Un altro aspetto critico è la natura dinamica delle preferenze degli utenti, che si evolve nel tempo e rappresenta una sfida significativa per l'adattabilità dei sistemi di raccomandazione. Le preferenze degli utenti sono raramente statiche; possono cambiare a causa di cambiamenti nelle circostanze personali, tendenze stagionali o influenze sociali più ampie. Un utente che guarda spesso film d'azione, per esempio, potrebbe temporaneamente preferire contenuti adatti alle famiglie durante le vacanze o quando li guarda con i bambini. I sistemi di raccomandazione che si basano molto su dati statici o storici non riescono a tenere conto di queste variazioni temporali, portando a raccomandazioni obsolete o non pertinenti. Per affrontare le preferenze dinamiche è necessario che i sistemi incorporino modelli temporali e caratteristiche consapevoli del contesto. Approcci come i raccomandatori basati sulle sessioni, che analizzano l'attività recente dell'utente, o le funzioni di decadimento temporale, che assegnano un peso maggiore alle interazioni recenti, sono esempi di sforzi per catturare queste dinamiche. Tuttavia, anche questi metodi devono affrontare un compromesso tra reattività ed efficienza computazionale, poiché l'incorporazione di aggiornamenti in tempo reale o di fattori contestuali può aumentare significativamente la complessità del sistema.

2.2 Large Language Models per Sistemi di Raccomandazione

In questa sezione ci occupiamo di descrivere in modo generale cosa sono i *Large Language Models*, qual è l'architettura alla base di questi, come è sfruttata da BERT e come è possibile sfruttarli nei sistemi di raccomandazione.

2.2.1 Panoramica LLMs

I *Large Language Models* (LLMs) [40] sono un tipo specifico di architettura di rete neurale, tipicamente costruita sul modello Transformer, contenente centinaia di miliardi di parametri. Questi parametri sono i pesi del modello che vengono regolati durante l'addestramento per catturare le complessità del linguaggio. I modelli LLM come GPT-3, BERT e GPT-4 sono addestrati su enormi insiemi di dati, tra cui libri, articoli, codici e altri archivi di testi digitali. Il core principale degli LLMs è predire la parola successiva all'interno di una sequenza di parole; questo task è riconosciuto come *language modeling*, ovvero, data una sequenza di parole, predire la successiva [60] . Gli LLMs si basano sui seguenti meccanismi:

- **Transformer Architecture** [54] il Transformer si basa su meccanismi di self-attention per analizzare le relazioni tra le parole di una frase, indipendentemente dalla loro distanza l'una dall'altra, permettendo al modello di comprendere profondamente il contesto.
- **Pre-training and Fine-tuning** Gli LLMs sono dapprima allenati in una grande quantità di dati per apprendere i pattern linguistici e i fatti (pre-train). In seguito vengono sottoposti al fine-tuned, ovvero allenati su dataset specifici, al fine di specializzare i compiti del LLMs.

La potenza degli LLMs deriva dalle loro dimensioni, dall'architettura e dalla diversità dei dati di addestramento. Possono generalizzare tra i vari compiti senza una programmazione esplicita per ciascuno di essi. Ad esempio, GPT-3 ha dimostrato di poter apprendere il contesto, eseguendo compiti come la traduzione o l'aritmetica senza riaddestrarsi, semplicemente fornendo esempi in ingresso.

I tipi di LLMs possono essere categorizzati in due grandi famiglie [57], in modelli discriminativi e modelli generativi; ovviamente, l'utilizzo di certi modelli piuttosto che altri dipende fortemente dagli obiettivi che si vogliono raggiungere e dai dati che si hanno a disposizione.

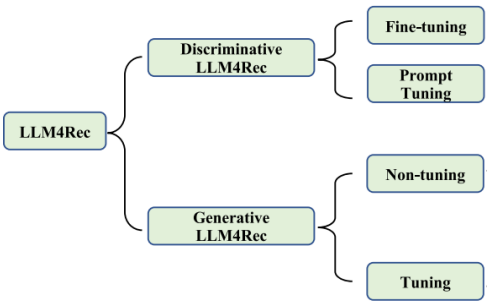

Figura 2.3: Una tassonomia della ricerca sui modelli linguistici di grandi dimensioni nei sistemi di raccomandazione [57].

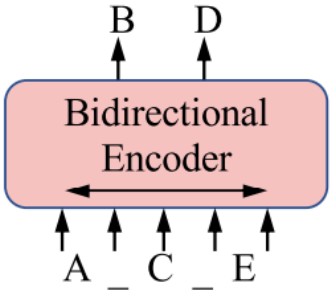

Figura 2.4: LLM discriminativo (es. BERT) [57].

Di seguito vengono descritti gli approcci dei due modelli e viene riportata una figura che illustra la tassonomia dei tipi Fig 2.3

Modelli Discriminativi

I **LLMs Discriminativi** sono progettati per affrontare compiti in cui è fondamentale l'allineamento tra i dati di input e gli obiettivi specifici. Utilizzati nella comprensione del linguaggio naturale (NLU), questi modelli eccellono nelle previsioni strutturate, nei compiti di classificazione e nella comprensione del contesto, rendendoli indispensabili per i sistemi di raccomandazione. Il loro sviluppo nasce dall'esigenza di gestire vasti insiemi di dati testuali e di fornire previsioni specifiche per il dominio, accurate e interpretabili.

I LLMs discrimitivi sono costruiti su *Transformer-based architectures* come BERT [11]. In generale il loro funzionamento ricade nel mascherare alcune parti dell'input e addestrare il modello cercando di predire in output i token delle parti dell'input mascherate, vedi Fig.2.4. Sono preaddestrati su grandi quantità di documenti utilizzando tecniche di *self-supervised learning*, che consentono loro di sviluppare rappresentazioni linguistiche ricche e generiche. La

seconda fase del loro ciclo di vita prevede il *fine tune* su insiemi di dati specifici per il dominio, consentendo al modello di specializzarsi in compiti di nicchia.

Nel contesto dei sistemi di raccomandazione, questo processo di addestramento in due fasi garantisce che il modello catturi gli schemi complessi del comportamento degli utenti e delle descrizioni degli articoli, offrendo raccomandazioni precise e contestualmente rilevanti. In particolare, nella fase di *Fine-tuning* si assicura che le rappresentazioni linguistiche siano associate al dominio di raccomandazione. Per esempio, BERT è stato sottoposto alla tecnica del Fine-Tuning per applicazioni di rappresentazione di gruppi e utenti [56]. Modelli come BERT4Rec [52] sono progettati specificamente per la raccomandazione sequenziale, consentendo loro di prevedere le preferenze dell'utente in base alle azioni precedenti.

Modelli Generativi

I **LLMs generativi** sono progettati per la generazione di linguaggio naturale, producono output testuali coerenti e ricchi di contesto. Questi modelli considerano le attività di raccomandazione come problemi di generazione del linguaggio, consentendo loro di creare suggerimenti personalizzati, spiegazioni o persino contenuti inediti in risposta alle richieste degli utenti. L'emergere dei modelli generativi è dovuto alla crescente domanda di sistemi di raccomandazione interattivi, in grado non solo di raccomandare, ma anche di giustificare e contestualizzare i suggerimenti. Le loro capacità sono particolarmente preziose in scenari che richiedono creatività, coinvolgimento dinamico dell'utente e adattabilità a dati scarsi o non strutturati. Il loro funzionamento ricade nel generare dati che sono contestualmente simili a quelli visti in precedenza, vedere Fig. 2.5.

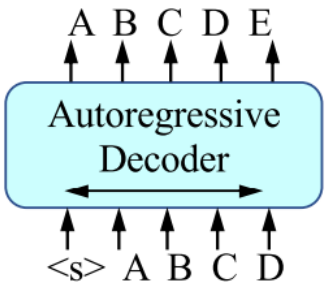

Figura 2.5: LLM Generativo (es. GTP) [57].

2.2.2 Architettura Transformer

BERT

BERT acronimo di ***B**idirectional **E**ncoder **R**epresentations from **T**ransformers* [11] è un modello di rappresentazione del linguaggio sviluppato da Google, per compiti generali nel *natural language processing* (NLP). L'innovazione principale che rende BERT un LLM di tutto rispetto è la sua abilità nel comprendere il contesto delle parole in modo bidirezionale, cioè considerando ciò che viene prima di una parola e ciò che viene dopo, diversamente da come fanno i modelli basati su GPT [19]. Di base BERT è un LLMs *general purpose*, ciò significa che può svolgere diversi compiti. Prevede due passi fondamentali *pre-training* e *fine tune*. Durante il *pre-training* il modello viene addestrato con tecniche particolari su una grande mole di dati per apprendere le relazioni tra parole. Durante la fase di *fine-tuning*, invece, il modello precedentemente addestrato viene modificato e sottoposto a *retrain* per un compito specifico come il *question answering* o la *sentiment analysis*. Questo approccio salva tempo e risorse perché l'operazione di *fine-tuning* costa generalmente meno che addestrare tutto il modello dall'inizio. Fare riferimento alla Fig.2.6 per una descrizione visiva delle due operazioni.

Architettura e Principi di Funzionamento BERT è un LLM basato su uno stack di *Transformer Layer* ovvero delle particolari reti neurali che permettono di trovare relazioni tra le parole attraverso il meccanismo di attenzione, ovvero si tiene conto di tutte le parole allo stesso momento durante l'elaborazione [54]. Di seguito vengono descritte le caratteristiche principali che rendono BERT unico nel suo genere.

Transformer Layer L'architettura Transformer[1] , descritta in "Attention is All You Need" [54], rivoluziona l'elaborazione del linguaggio naturale consentendo ai modelli di catturare le dipendenze complesse all'interno delle sequenze di input senza ricorrere a strati ricorrenti o convoluzionali. In BERT questa architettura è adattata per creare rappresentazioni profonde bidirezionali considerando i contesti destro e sinistro simultaneamente su tutti gli strati. Per una visualizzazione di massima fare riferimento alla figura Fig.2.7 [2]

[1]Toward Data Science: Keeping up with the BERTs. Available at: `https://towardsdatascience.com/keeping-up-with-the-berts-5b7beb92766`. Accessed on December 28, 2024.

[2]Transformers. Available at: `https://towardsdatascience.com/transformers-89034557de14`. Accessed on December 28, 2024.

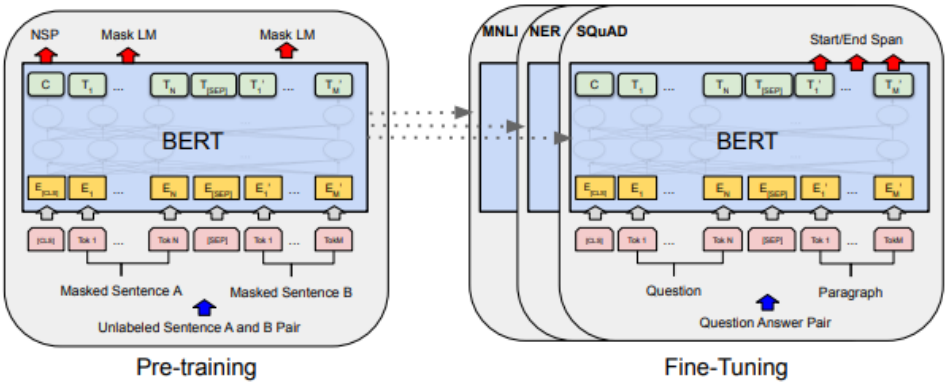

Figura 2.6: Procedure di *pre-training* e *fine-tuning* messa a punto per BERT. A parte gli strati di uscita, le stesse architetture sono utilizzate sia per il *pre-training* che per *fine-tuning*. Gli stessi parametri del modello pre-addestrato vengono usati per inizializzare modelli per i diversi compiti down-stream [11].

Nell'architettura oltre a *token embeddings* e *Segment embeddings* di cui parleremo brevemente nel paragrafo 2.2.2, è present il ***Positional encoding*** o *Positional embedding*, fondamentale per tenere traccia dell'ordine dei token nelle sequenze di ingresso, fattore critico per la comprensione della sintattica e della semantica del linguaggio, poiché si sfrutta il meccanismo di attenzione, che di base non possiede un meccanismo di memorizzazione dell'ordine, perché elabora le sequenze come set aggregato e non come informazioni sequenziali. L'encoding è creato in modo deterministico sfruttando le funzioni seno e coseno.

$$PE(pos, 2i) = \sin\left(\frac{pos}{10000^{\frac{2i}{d}}}\right)$$

$$PE(pos, 2i+1) = \cos\left(\frac{pos}{10000^{\frac{2i}{d}}}\right)$$

d rappresenta la dimensionalità dello spazio di embedding.

Ogni *trasformer layer* comprende un blocco *multi-head self-attention mechanism* e un blocco *position-wise fully connected feed-forward networks*.

I blocchi *multi-head self-attention mechanism* consentono ai modelli di catturare relazioni intricate all'interno di sequenze di input, prestando attenzione a diverse parti della sequenza simultaneamente. A differenza dei *single-head self-attention mechanism*, che si concentrano su un singolo contesto, i *multi-head self-attention mechanism* imparano a enfatizzare vari aspetti dell'input. Questo approccio permette al modello di catturare una serie di dipendenze, da quelle

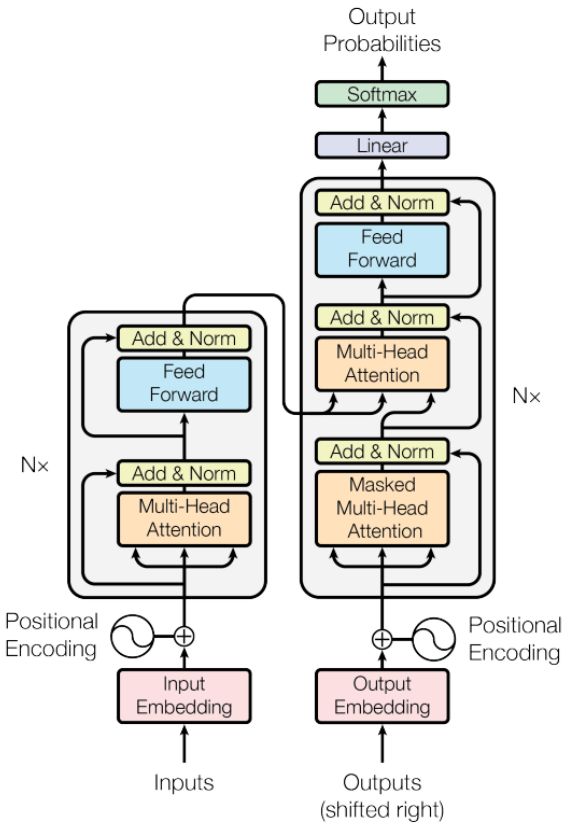

Figura 2.7: Architettura dei modelli Trasformer [54].

a breve termine a quelle a lungo termine, migliorando la sua capacità di comprendere modelli linguistici complessi.

Ogni *Attention head* elabora i dati in modo indipendente, calcolando un'attenzione scalare a prodotto di punti proiettando l'input in vettori di query, chiave e valore attraverso trasformazioni lineari che dipendono dai parametri del modello. Queste proiezioni permettono al modello di valutare la rilevanza di ciascun token rispetto agli altri, identificando in modo efficace quali parti dell'input sono più pertinenti a un determinato token. Grazie all'impiego del *multi-head* il modello è in grado di considerare le informazioni provenienti da diversi sottospazi di rappresentazione in varie posizioni, facilitando una comprensione più ricca e sfumata della sequenza di input. In output tutti i risultati vengono concatenati e trasformati linearmente, e passati al *position-wise fully connected feed-forward networks*. Ogni token della sequenza viene trasformato in modo indipendente dallo stesso FFN, che consiste in due strati lineari con una funzione di attivazione ReLU nel mezzo. Questa struttura permette al modello di catturare

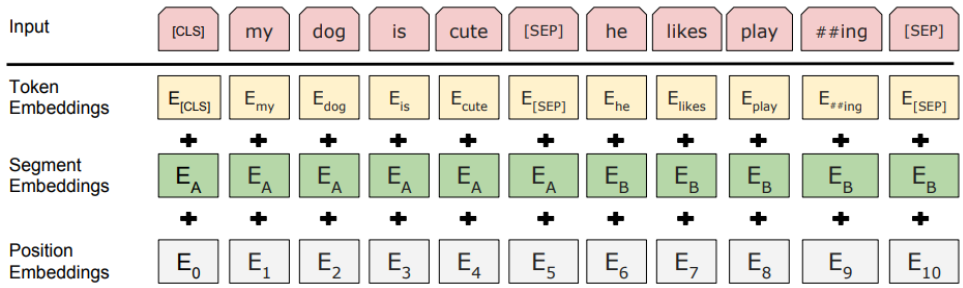

Figura 2.8: Rappresentazione dell'input di BERT. Gli input Embeddings sono la somma dei token embeddings, dei segmentation embeddings e dei position embeddings [11].

modelli e relazioni complesse all'interno dei dati. Il termine "position-wise" indica che la FFN viene applicata a ciascuna posizione della sequenza separatamente, consentendo al modello di apprendere caratteristiche specifiche della posizione, migliorando la capacità del modello di comprendere e generare il linguaggio, catturando efficacemente le informazioni contestuali per ciascun token.

Inputs and Outputs L'**input** consiste in una sequenza tokenizzata che può rappresentare una singola frase o due frasi connesse, separate da un token speciale [SEP]. Oltre ai *token embeddings*, vengono aggiunti altri embeddings per affrontare le sfide principali. *Segment embeddings* distinguono i token appartenenti alla prima frase (segmento A) da quelli della seconda (segmento B), mentre i *position embeddings* codificano l'ordine dei token per preservare la struttura della sequenza. Per una versione figurativa che aiuta a comprendere come vengono trattate le parole fare riferimento alla Figura 2.8

L'**output** dipende dal compito. Per compiti di classificazione come la Next Sentence Prediction (NSP), lo stato finale del token speciale [CLS] viene inserito in una rete neurale feedforward (FFNN) con uno strato Softmax per prevedere se la seconda frase è l'effettiva continuazione della prima. Per compiti come la *masked word prediction*, gli stati finali dei token [MASK] che vengono passati attraverso un altro strato FFNN+Softmax per prevedere i token mascherati dal vocabolario. Questi meccanismi permettono a BERT di gestire più compiti contemporaneamente, sfruttando in modo flessibile i suoi stati finali.

Masked Model Language è una tecnica non supervisionata usata durante la fase di *fine-tuning*, ed è la responsabile della comprensione bi-direzionale. Questa tecnica funziona masche-

rando parzialmente i token in ingresso in una sequenza e addestrando il modello a prevedere i token originali, concentrandosi solo sulle parole mascherate anziché sull'intera sequenza, MLM incoraggia il modello ad apprendere le dipendenze contestuali tra le parole precedenti e quelle successive. Durante la fase di *pre-train* circa il 15% dei token nella sequenza viene sostituito con il token [MASK] e di questi:

- Il 10% delle volte, il token selezionato viene sostituito con una parola casuale del vocabolario.
- Il 10% delle volte, il token selezionato rimane invariato.
- L'80% delle volte, il token selezionato viene sostituito con il token speciale [MASK].

Il modello elabora la sequenza utilizzando la sua architettura di trasformatori, catturando le relazioni contestuali tra tutti i token (sia mascherati che non), per ogni token mascherato, il modello produce un vettore di predizione corrispondente alla dimensione del vocabolario. Questo vettore viene passato attraverso la funzione softmax per produrre probabilità per ogni parola del vocabolario. La parola con la probabilità più alta è considerata la previsione del modello.

Next Sequence Prediction è una componente fondamentale nel preaddestramento di alcuni modelli linguistici, in particolare BERT, per aiutare il modello a comprendere le relazioni tra le frasi. Sebbene i modelli linguistici siano efficaci nel predire le sequenze di parole e nel comprendere le sfumature contestuali all'interno di una singola frase, non colgono intrinsecamente le relazioni logiche o semantiche tra frasi consecutive. Questa comprensione è importante per compiti *Question Answering (QA)* o *Natural Language Inference (NLI)*. NSP introduce un obiettivo di addestramento binarizzato per colmare questa lacuna. Durante il preallenamento, il modello viene esposto a coppie di frasi, etichettate come "IsNext" o "NotNext". Nella categoria "IsNext", la seconda frase (B) è la vera continuazione della prima frase (A) dello stesso documento o contesto. Al contrario, nella categoria "NotNext", la frase B viene selezionata casualmente dal corpus, assicurando che non abbia alcun collegamento logico o sequenziale con la frase A. L'obiettivo del modello è prevedere questa relazione binaria imparando le sottigliezze del flusso di testo coerente. Deve capire se B è una naturale continuazione di A, il che implica la comprensione della coerenza, dell'argomento e della progressione logica delle idee. L'equilibrio 50-50 nel set di dati è cruciale per l'addestramento, in quanto garantisce che il modello non sia prevenuto nel prevedere la continuazione o la non continuazione. Questo equilibrio costringe il

modello a valutare attentamente le relazioni semantiche e contestuali, piuttosto che affidarsi a schemi di frequenza o a scorciatoie.

Applicazioni di BERT BERT ha fatto progredire in modo significativo il campo dell'elaborazione del linguaggio naturale (NLP), raggiungendo prestazioni all'avanguardia in una vasta gamma di applicazioni. La sua versatilità deriva dalla capacità di generare rappresentazioni profonde e bidirezionali, che consentono una comprensione del contesto linguistico. Una principale applicazione di BERT è nei sistemi di risposta alle domande (QA) [35], dove eccelle nella comprensione delle domande e nel recupero di risposte precise da dati testuali estesi. Questa capacità è stata dimostrata dalle sue prestazioni superiori su benchmark come lo Stanford Question Answering Dataset (SQuAD). L'impatto di BERT si estende al Named Entity Recognition (NER) [41], dove identifica e classifica efficacemente entità come nomi, date e luoghi all'interno del testo. La sua comprensione del contesto migliora l'accuratezza del riconoscimento delle entità, a vantaggio delle attività di estrazione delle informazioni [36]. Nella Sentiment Analysis[55], la profonda comprensione contestuale del BERT permette di catturare il sentiment espresso nel testo con un'elevata precisione, rendendolo prezioso per le applicazioni che richiedono la comprensione di informazioni soggettive[3]. Inoltre, BERT è stato applicato alla sintesi di testi, aiutando a generare riassunti concisi e coerenti attraverso la comprensione del significato centrale di documenti più lunghi. La sua architettura facilita anche i compiti di classificazione del testo, dove BERT è stato messo a punto per categorizzare il testo in etichette predefinite, migliorando le prestazioni in varie sfide di classificazione.

2.2.3 LLMs per i sistemi di raccomandazione

I Large Language Models nei sistemi di raccomandazione colmano quelli che sono gli svantaggi principali dei sistemi di raccomandazione tradizionali quali, data sparsity, cold start, la personalizzazione, rendendoli altamente versatili. La scarsità di dati, una sfida frequente nei sistemi di raccomandazione, deriva dalla limitatezza dei dati di interazione utente-oggetto. I LLM contrastano questa limitazione grazie al loro ampio preallenamento, che li dota della capacità di dedurre relazioni significative anche in scenari scarsi. Lo *Zero-shot learning* consente ai LLM di fornire raccomandazioni senza richiedere dati specifici per l'attività, basandosi sulla comprensione contestuale derivata dalla loro conoscenza preaddestrata [57]. Few-shot learning migliora ulteriormente questa adattabilità, consentendo ai LLM di affinare la loro comprensione

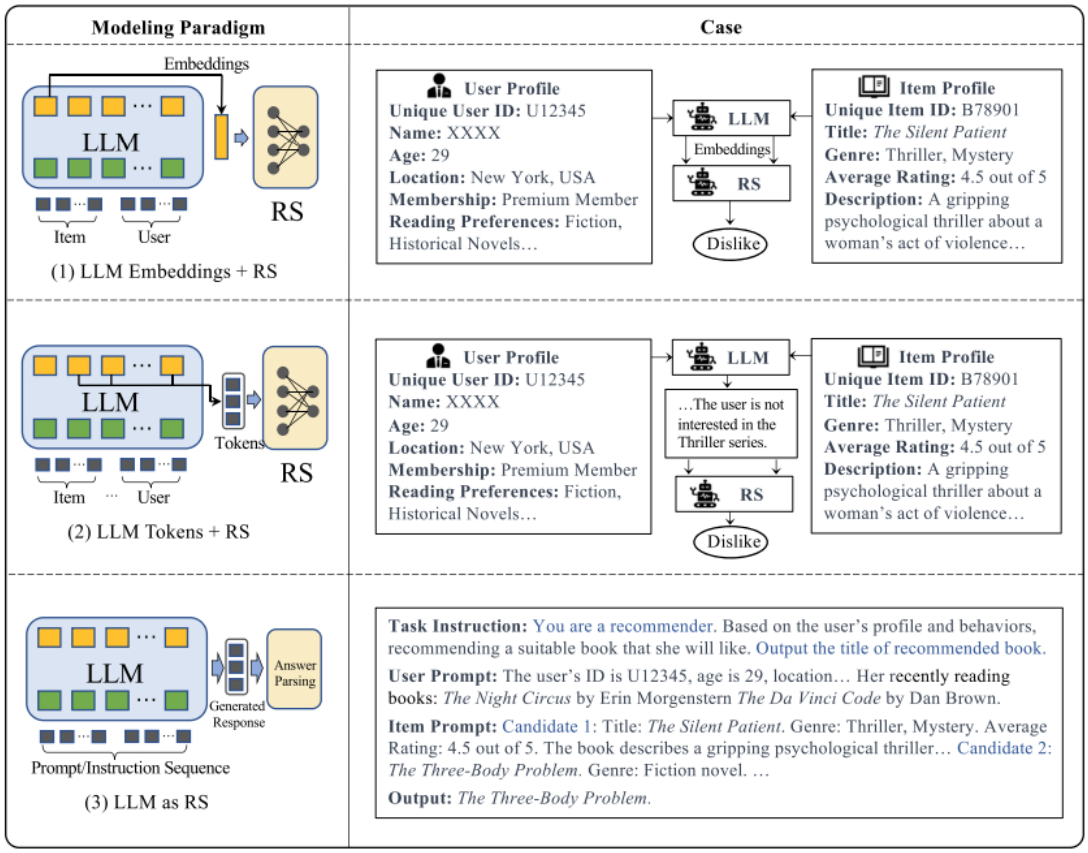

Figura 2.9: Tre paradigmi di modellazione rappresentativi della ricerca di modelli linguistici di grandi dimensioni sui sistemi di raccomandazione [57].

di un dominio con esempi minimi di attività specifiche, migliorando l'accuratezza delle raccomandazioni [25, 50]. I LLM sono abili nell'elaborare input testuali come descrizioni di prodotti, recensioni di utenti e profili, generando embeddings ad alta dimensionalità che incapsulano ricche informazioni semantiche. Questi embeddings contribuiscono a un'accurata rappresentazione degli utenti, analizzando i contenuti generati dagli utenti per dedurre le preferenze. Inoltre, migliorano la rappresentazione degli articoli codificando attributi, categorie e recensioni in uno spazio vettoriale strutturato in cui gli articoli simili si raggruppano naturalmente. Ad esempio, nelle applicazioni di e-commerce, gli LLM estraggono caratteristiche di sentiment e di attributi dalle recensioni e dai metadati dei prodotti, consentendo di classificare e abbinare con precisione gli articoli alle preferenze degli utenti[17]. Esistono tre tipologie di paradigmi per integrare gli LLM nei sistemi di raccomandazione, ognuno dei quali mette in rilevanza alcuni aspetti piuttosto che altr, per una descrizione visiva fare riferimento alla Fig. 2.9. Di seguito una descrizione dei

paradigmi:

1. **LLM Embeddings + RS** I LMMs sono usati come estrattori di caratteristiche, le descrizione testuali degli oggetti e degli utenti sono dati in input agli LLMs per generare gli Embedding, questi sono poi processati dagli RS tradizionali per attività come ranking o la classificazione. Questo approccio sfrutta gli embeddings codificati dalla conoscenza per arricchire i dati di interazione utente-oggetto.

2. **LLM Tokens + RS** Questo paradigma prevede la generazione di token (parole o frasi) a partire dalle caratteristiche dell'utente e dell'articolo utilizzando gli LLM. I token riflettono le preferenze latenti dell'utente o gli attributi dell'articolo. Vengono inseriti nei componenti decisionali a valle della RS. L'estrazione semantica dei token aiuta a catturare preferenze sfumate e a migliorare la precisione delle raccomandazioni.

3. **LLM as RS** in questo caso gli LLM agiscono direttamente come motore di raccomandazione, elaborando sequenze di input che comprendono profili di utenti, richieste di comportamento e istruzioni per i compiti, producendo elementi raccomandati. Questo paradigma enfatizza lo sfruttamento della capacità dei LLM preaddestrati di comprendere e generare testo per fornire raccomandazioni, spesso senza effettuare il retrain.

Gli Embeddings dei LLM eccellono nel catturare le relazioni semantiche implicite, modellando efficacemente le relazioni e le dipendenze all'interno dei dati; questo porta a una comprensione del contesto di azione, facendo sì che le raccomandazioni dipendano dalle interazioni passate.

2.3 Attacchi Avversari

2.3.1 Definizione e Obiettivi

L'*Adversarial Machine Learning* (AML) [4] è un campo in evoluzione che esplora le vulnerabilità dei sistemi di apprendimento automatico (ML), sfruttando input manipolati, noti come esempi avversari, gli attaccanti mirano a manipolare i risultati di un modello, spesso per raggiungere obiettivi che contrastano l'uso previsto del sistema, per pilotare il comportamento o per invalidarne il risultato stesso. La prevalenza del machine learning in applicazioni come i sistemi di raccomandazione [23], la sicurezza informatica, i veicoli autonomi e l'assistenza sanitaria, amplifica i rischi associati agli attacchi avversari. Questi attacchi non sono solo preoccupazioni teoriche, ma dimostrano la suscettibilità dei modelli alle interruzioni, sollevando importanti questioni sulla robustezza e sulla sicurezza degli algoritmi di ML.

L'integrazione dei *large language models* (LLMs) nei sistemi di raccomandazione aumenta questi rischi. I sistemi basati su LLM interpretano gli input degli utenti e generano output personalizzati sfruttando un'architettura complessa ed ad alta dimensionalità, questo porta ad avere una superficie d'attacco maggiore che favorisce e apre le porte ad utenti il cui scopo è attaccare il modello. Gli attacchi a questi sistemi possono avere conseguenze significative, come raccomandazioni fuorvianti, esperienze pessime per gli utenti o perdite economiche per i venditori e chi fornisce il servizio. Per rafforzare i sistemi di raccomandazione basati su LLM contro gli attacchi avversari, è indispensabile comprendere le basi dell'apprendimento automatico avversario, la tassonomia di questi attacchi e i meccanismi sfruttati dagli avversari.

Comprendere gli attacchi avversari

Un *Adversarial Attack* o *attacco avversario*, è un particolare attacco che cerca di manipolare il comportamento di un modello ML introducendo input specificamente progettati per sfruttare le sue vulnerabilità, che a differenza del rumore naturale o delle perturbazioni casuali, gli esempi avversari sono perturbazioni calcolate che spesso rimangono impercettibili all'uomo, pur causando un significativo comportamento scorretto del modello. Il fenomeno è stato identificato per la prima volta nel campo del riconoscimento delle immagini, dove piccole modifiche apparentemente insignificanti ai pixel hanno indotto i classificatori a sbagliare l'identificazione degli oggetti con un'elevata affidabilità. Come si può vedere in figura 2.10, dapprima il modello riesce a riconoscere un animale ben specifico nell'immagine, dopo l'aggiunta di una perturba-

zione scelta con dei criteri stabiliti, il modello riconosce un animale completamente nonostante all'occhio umano l'animale in figura non sia cambiato. Gli attacchi avversari sfruttano il modo in cui i modelli di ML generalizzano dai dati di addestramento, prendendo di mira i confini decisionali appresi durante il processo di addestramento. Il successo di questi attacchi deriva spesso dai compromessi intrinseci dei sistemi di ML tra accuratezza, efficienza e complessità. Inoltre, i sistemi di ML, a causa della loro dipendenza da enormi insiemi di dati e da architetture complesse, sono particolarmente inclini alle manipolazioni avversarie [43]. Detto ciò, i motivi per sferrare un attacco di tipo avversario può avere diverse finalità, le quali dipendono dall'attaccante e il successo dalla tecnica utilizzata. Solitamente gli obiettivi di un attaccante rientrano in due macro categorie, *target* e *untarget*.

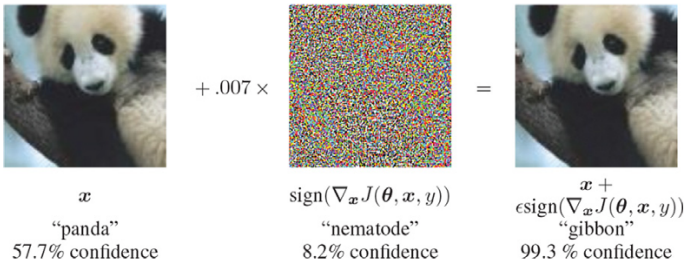

Figura 2.10: Attacco avversario che mostra come le immagini vengono misclassificate dopo l'aggiunta di una perturbazione pre-calcolata [43].

Untarget Attack L'obiettivo principale di un attacco non mirato è quello di degradare le prestazioni complessive del modello. Ad esempio, in un sistema di raccomandazione basato su LLM, un aggressore può introdurre input che portano a raccomandazioni irrilevanti o prive di senso, minando la fiducia degli utenti e l'utilità del sistema.

Target Attack Gli attacchi mirati mirano a raggiungere un obiettivo avversario specifico, come la promozione o la retrocessione di un particolare articolo o categoria in un sistema di raccomandazione. Questo è particolarmente comune in ambito commerciale, dove gli avversari potrebbero cercare di gonfiare artificialmente le classifiche dei prodotti o diminuire la visibilità delle offerte dei concorrenti. [42, 5, 43]

La conoscenza che l'attaccante ha dei sistemi usati dalle vittime è sicuramente un vantaggio e gioca una ruolo fondamentale nella scelta e nel successo dell'attacco. [5] nella loro survey

forniscono una panoramica completa dell'evoluzione degli attacchi avversari e mettono in evidenza le distinzioni critiche tra i diversi livelli di conoscenza dell'attaccante: attacchi *white-box, gray-box* e *black-box*.

white-box gli attacchi basati su *white-box* rappresentano lo scenario in cui l'attaccante ha una conoscenza completa del modello bersaglio, compresa la sua architettura, i parametri e i dati di addestramento. Questo livello di accesso consente all'attaccante di creare esempi avversari molto efficaci sfruttando direttamente i gradienti del modello. L'attaccante può eseguire calcoli precisi per determinare le perturbazioni ottimali necessarie per ingannare il modello. Questo scenario è spesso considerato il più impegnativo dal punto di vista della difesa, perché l'attaccante può sfruttare ogni aspetto delle vulnerabilità del modello.

gray-box in questo caso si presuppone che l'attaccante abbia una conoscenza parziale del modello bersaglio. Ciò potrebbe includere la conoscenza dell'architettura del modello, ma non dei suoi parametri specifici o dei dati di addestramento. Gli attaccanti gray-box spesso si basano sulla trasferibilità, una proprietà per cui gli esempi avversari generati per un modello possono ingannare anche altri modelli addestrati su dati simili. Addestrando un modello surrogato che approssima il comportamento del modello target, l'attaccante può generare esempi avversari che probabilmente saranno efficaci contro il modello target. Questo approccio sfrutta le somiglianze tra i modelli per aggirare la mancanza di accesso diretto ai parametri del modello target. Gli attacchi gray-box evidenziano l'importanza di comprendere la trasferibilità degli esempi avversari e la necessità di difese robuste in grado di generalizzarsi a diversi modelli.

black-box Gli attacchi basati su black-box presentano lo scenario più restrittivo per l'attaccante, che non ha accesso diretto all'architettura, ai parametri o ai dati di addestramento del modello target. Al contrario, l'attaccante può solo interrogare il modello e osservare i suoi risultati. Nonostante queste limitazioni, gli attacchi black-box possono essere molto efficaci grazie a tecniche come gli attacchi basati su query e l'uso di modelli surrogati. Gli attacchi basati su query prevedono l'affinamento iterativo degli esempi avversari interrogando il modello target e utilizzando le uscite osservate per approssimare i gradienti del modello. Il successo di questi attacchi sottolinea la necessità di difese che non si basino esclusivamente sull'oscurità, ma si concentrino invece sul potenziamento della robustezza intrinseca del modello.

La distinzione tra attacchi white-box, gray-box e black-box è fondamentale per comprendere il panorama dell'apprendimento automatico avverso. Gli attacchi white-box, con il loro accesso completo al modello, rappresentano la minaccia più grave e richiedono difese sofisticate in grado

di resistere allo sfruttamento diretto delle vulnerabilità del modello. Gli attacchi gray-box, che fanno leva sulla trasferibilità, sottolineano la necessità di difese in grado di generalizzarsi tra modelli e architetture diverse. Gli attacchi black-box, nonostante l'accesso limitato, dimostrano l'ingegnosità degli aggressori nello sfruttare metodi basati su query e modelli surrogati per raggiungere i loro obiettivi. Poiché l'apprendimento automatico avversario continua a evolversi, la comprensione di questi diversi livelli di conoscenza degli attaccanti è essenziale per sviluppare sistemi di apprendimento automatico robusti e resistenti.

2.3.2 Tipologie di Attacchi Avversari

Gli attacchi possono essere classificati in diverse categorie in base a vari criteri, quali la conoscenza dell'attaccante, la fase della pipeline di addestramento del modello e gli obiettivi dell'attacco. [18] presenta una panoramica sugli attacchi possibile durante la varie fasi di vita degli LLMs, il tutto riassunto in figura 2.11

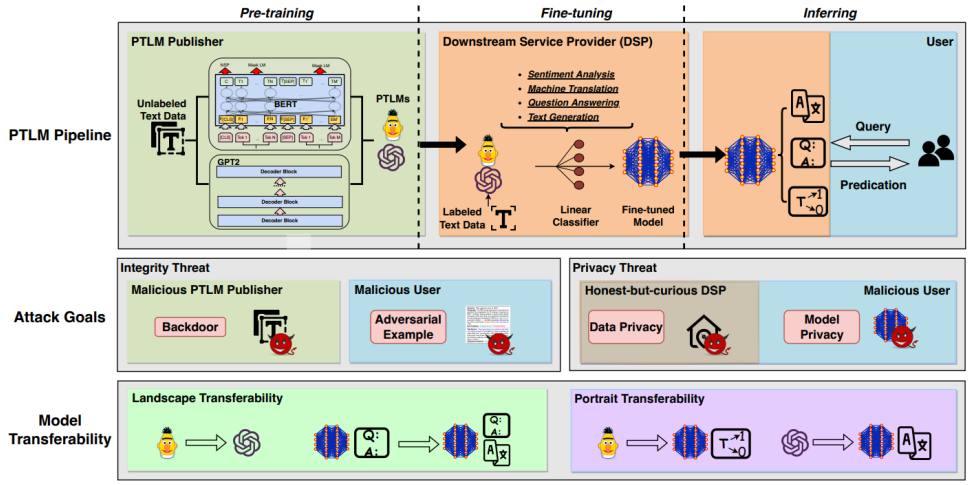

Figura 2.11: Pipeline dei LLM con gli obiettivi degli attacchi e transferibilità [18].

Evasion Attacks Gli attacchi di evasione si verificano durante la fase di inferenza, in cui l'attaccante crea esempi avversari progettati per ingannare il modello e indurlo a fare previsioni errate. Nel contesto dei sistemi di raccomandazione basati su LLM, questi attacchi possono essere particolarmente efficaci a causa dell'elevata complessità e non linearità dei modelli. L'attaccante genera input che sembrano legittimi ma che vengono sottilmente alterati per ingannare

il modello. Ad esempio, in un sistema di raccomandazione cinematografica, un aggressore potrebbe modificare le recensioni o le valutazioni degli utenti in modo da indurre il sistema a raccomandare o declassare determinati film. L'obiettivo principale è quello di aggirare le difese del modello mantenendo l'apparenza di input normali [27, 33].

Poisoning Attacks Gli attacchi di avvelenamento prendono di mira la fase di train o re-train nella pipeline di apprendimento automatico. L'attaccante inietta dati dannosi nel set di addestramento, corrompendo il processo di apprendimento e ottenendo un modello compromesso. Nei sistemi di raccomandazione basati su LLM, gli attacchi di poisoning possono essere particolarmente dannosi perché possono manipolare il comportamento del modello su un'ampia gamma di input. Ad esempio, un aggressore potrebbe iniettare nei dati di addestramento profili e valutazioni di utenti falsi per promuovere o declassare articoli specifici. Questo può portare a raccomandazioni distorte che favoriscono gli interessi dell'attaccante. Gli attacchi di avvelenamento possono essere classificati in attacchi mirati e non mirati. Gli attacchi mirati mirano a influenzare il comportamento del modello in relazione a input o output specifici, mentre gli attacchi non mirati mirano a degradare le prestazioni complessive del modello. In figura 2.12 è presente una tassonomia che descrive come possono avvenire gli attacchi di data poisoning [14, 10].

Figura 2.12: Tassonomia degli attacchi avversari [42].

Model Inversion Attacks Gli attacchi di inversione del modello mirano a estrarre informazioni sensibili sui dati di addestramento interrogando il modello. Nel contesto dei sistemi di raccomandazione basati su LLM, questi attacchi possono essere particolarmente preoccupanti perché possono compromettere la privacy degli utenti. L'attaccante sfrutta le previsioni del modello per dedurre le proprietà dei dati di addestramento, ricostruendo potenzialmente informazioni sensibili come le preferenze dell'utente o i suoi dati personali. Ad esempio, in un sistema di raccomandazione musicale, un aggressore potrebbe utilizzare tecniche di inversione del modello per dedurre la cronologia di ascolto o le preferenze di un utente. Questi attacchi sfruttano il fatto che i modelli di apprendimento automatico spesso conservano nei loro parametri informazioni sui dati di addestramento, che possono essere estratte attraverso query accuratamente elaborate [15].

Membership Inference Attacks Gli attacchi di *Membership Inference* determinano se uno specifico set di dati faceva parte del set di addestramento del modello. Questo può essere particolarmente dannoso in scenari in cui la presenza di determinati punti di dati nell'insieme di addestramento è un'informazione sensibile. Nei sistemi di raccomandazione basati su LLM, questi attacchi possono rivelare se i dati di un utente sono stati utilizzati per addestrare il modello, compromettendo potenzialmente la sua privacy. Ad esempio, in un sistema di raccomandazione di libri, sapere che la cronologia di lettura di un particolare utente è stata utilizzata per addestrare il modello potrebbe rivelare informazioni sensibili sulle sue preferenze. Gli attacchi all'inferenza di appartenenza sfruttano le differenze di comportamento del modello quando viene interrogato con punti di dati che facevano parte dell'insieme di addestramento rispetto a quelli che non ne facevano parte [26].

Gli attacchi avversari ai sistemi di raccomandazione basati su LLM pongono sfide significative alla sicurezza e all'affidabilità di questi sistemi. Comprendere i diversi tipi di attacchi avversari è fondamentale per sviluppare difese robuste per proteggersi da essi. Ogni tipo di attacco sfrutta aspetti diversi della pipeline di apprendimento automatico, dalla manipolazione dei dati di addestramento all'evasione del tempo di inferenza, e richiede meccanismi di difesa personalizzati per mitigarne l'impatto. Con la continua evoluzione dell'apprendimento automatico avversario, la ricerca continua e i progressi nelle strategie di difesa sono essenziali per salvaguardare l'integrità e la privacy dei sistemi di raccomandazione basati su LLM.
Gli attacchi avversari ai sistemi di raccomandazione basati su LLM pongono sfide significative alla sicurezza e all'affidabilità di questi sistemi. Comprendere i diversi tipi di attacchi avversari

è fondamentale per sviluppare difese robuste per proteggersi da essi. Ogni tipo di attacco sfrutta aspetti diversi della pipeline di apprendimento automatico, dalla manipolazione dei dati di addestramento all'evasione del tempo di inferenza, e richiede meccanismi di difesa personalizzati per mitigarne l'impatto. Con la continua evoluzione dell'apprendimento automatico avversario, la ricerca continua e i progressi nelle strategie di difesa sono essenziali per salvaguardare l'integrità e la privacy dei sistemi di raccomandazione basati su LLM.

2.4 Attacchi Avversari nei Sistemi di Raccomandazione Sequenziali

Negli ultimi anni, i sistemi di raccomandazione sono parte fondamentale di molte piattaforme online, offrendo suggerimenti personalizzati che migliorano l'esperienza degli utenti e guidano la crescita del business. I sistemi di raccomandazione sequenziali (SRS) [57], in particolare, si distinguono per la loro capacità di catturare le preferenze in evoluzione degli utenti e i modelli di comportamento sequenziali. Sfruttando architetture avanzate come le reti neurali ricorrenti (RNN) [47], i trasformatori e modelli come BERT4Rec, questi sistemi prevedono il prossimo articolo che un utente probabilmente utilizzerà o acquisterà in base alla sua storia di interazione. Tuttavia, man mano che questi modelli vengono sempre più integrati nei processi decisionali, si trovano ad affrontare crescenti minacce da parte di utenti malevoli. Tra i più preoccupanti vi sono gli attacchi di estrazione del modello e gli attacchi avversari, in particolare in contesti black-box 2.3 in cui l'attaccante non ha accesso diretto ai pesi del modello o ai dati di addestramento. Questi attacchi sfruttano le vulnerabilità dell'SRS replicando le sue funzionalità (estrazione del modello) o manipolando le sue raccomandazioni attraverso input accuratamente manipolati (attacchi avversari). La domanda centrale della ricerca affrontata dagli autori in [59] è: Gli attacchi avversari possono compromettere l'integrità e la robustezza dei sistemi di raccomandazione sequenziali in contesti black-box e privi di dati? L'articolo verrà spiegato in modo dettagliato, perché è la base per costruire una difesa solida dagli attacchi di tipo black-box; inoltre, verrà introdotta l'architettura di BERT4Rec, perché parte integrante dell'articolo descritto e parte integrante nella metodologia usata.

2.4.1 Contesto

Il paper [59] affronta il problema degli attacchi di tipo black-box nei sistemi di raccomandazione sequenziale; a tal fine utilizza le più recenti architetture nello stato dell'arte per i SRS utilizzati, **BERT4Rec** che verrà approfonditamente discusso in 2.4.2, Neural Attentive Recommendation Machine **NARM** [31], Self-Attentive Sequential Recommendation **SASR** [28]. Il nostro contributo verterà maggiormente nell'utilizzo di ***BERT4Rec*** poiché basato ampiamente sui livelli transformer, Fig.2.7. Di contro ***SASR*** e ***NARM*** sono fondamentali per studiare i risultati ottenuti.

Neural Attentive Recommendation Machine è un modello progettato per migliorare l'accuratezza delle raccomandazioni concentrandosi sulle parti più rilevanti delle sessioni degli utenti basato sulle reti neurali ricorrenti (RNN) [31]. Inoltre, integra un meccanismo di attenzione, creando un potente strumento per catturare le dipendenze sia globali che locali nelle interazioni degli utenti. Il primo layer dell'architettura è uno strato di embedding che trasforma gli elementi discreti in rappresentazioni continue, fase cruciale per dare una rappresentazione spaziale degli elementi. Successivamente, entra in gioco la Gated Recurrent Unit (GRU); questa gestisce le informazioni sequenziali, catturando efficacemente le dipendenze a lungo e a breve termine all'interno delle sessioni degli utenti. Questa doppia capacità assicura che il modello possa comprendere gli schemi generali del comportamento degli utenti, prestando al contempo attenzione alle interazioni immediate e recenti. I layer di attenzione poi riescono a individuare ed enfatizzare i comportamenti degli utenti così da creare i più pertinenti ed in linea con il comportamento medio dell'utente. Infine, il livello di output utilizza la somiglianza tra il vettore di contesto e gli embeddings degli elementi per generare un elenco di raccomandazioni. Questo processo garantisce che le raccomandazioni siano strettamente allineate con gli interessi e i comportamenti attuali dell'utente. NARM eccelle in particolare negli scenari che prevedono raccomandazioni brevi basate su sessioni, in cui l'ordine delle interazioni è cruciale.Tuttavia, deve affrontare delle difficoltà con sequenze più lunghe, a causa dei limiti intrinseci delle RNN nel gestire dipendenze estese.

Self-Attentive Sequential Recommendation a differenza di NARM, SASR è molto più simile a bert4rec poiché adotta un'architettura basata su Transformer, particolarmente efficace grazie ai suoi meccanismi di *self-attention* ampiamente approfonditi in 2.4.2 [28] . Anche qui, troviamo all'inizio un layer di embeddings seguito da layer di *multi-head self-attention* e *layer di feed-forward* che identificano e catturano le dipendenze da elemento a elemento nell'intera sequenza. Questo approccio consente al modello di considerare un contesto più ampio quando fa previsioni, invece di limitarsi alle sole interazioni recenti. Durante l'addestramento, SASRec impiega una strategia di predizione *masked model language*, la stessa utilizzata in Bert 2.2.2, che aiuta il modello a comprendere contesto e relazione tra elementi. Uno dei vantaggi significativi di SASRec è la sua capacità di parallelizzare i calcoli, che lo rende più efficiente delle RNN, soprattutto quando si tratta di gestire dipendenze a lungo termine. A differenza di BERT4Rec, l'architettura di SASR è basata su transformer unidirezional, invece, durante il training utilizza un obiettivo autoregressivo basato sul *Next sequence item prediction* che bert4rec rimuove per

enfatizzare il contesto bidirezionale.

2.4.2 BERT4rec

BERT4Rec[52] è un sistema di raccomandazione sequenziale progettato per affrontare le sfide inerenti alla caratterizzazione e alla previsione del comportamento degli utenti in contesti dinamici e in evoluzione. A differenza dei sistemi di raccomandazione tradizionali, che spesso si basano su preferenze dell'utente statiche o rigidamente ordinate, BERT4Rec è pensato per scenari in cui gli interessi dell'utente sono modellati da interazioni storiche che non sempre aderiscono a una sequenza fissa. Ciò è particolarmente evidente nei domini in cui i fattori esterni o la natura delle scelte degli utenti interrompono il rigido ordine cronologico.

La necessità di BERT4Rec nasce dalle limitazioni osservate nelle architetture precedenti, come i *sequential recommender* basati su RNN [8]. Sebbene le RNN e altri modelli unidirezionali [24, 12, 22, 32] abbiano dimostrato un notevole successo nei compiti di raccomandazione sequenziale, essi codificano il comportamento dell'utente in modo sequenziale da sinistra a destra. Questa natura intrinsecamente unidirezionale limita la loro capacità di sfruttare le informazioni contestuali in modo completo. Ogni rappresentazione di un elemento in un modello unidirezionale è derivata esclusivamente dagli elementi precedenti, il che spesso non riesce a catturare le interdipendenze più ampie all'interno delle sequenze di comportamento dell'utente. Inoltre, i modelli unidirezionali sono spesso sviluppati per insiemi di dati con un ordine sequenziale naturale, come i dati delle serie temporali o i testi. Nei sistemi di raccomandazione, tuttavia, i comportamenti degli utenti sono raramente regolati da un ordine così rigido, rendendo questi modelli non ottimali per molte applicazioni reali.

Il componente principale di BERT4Rec è il suo *self-attention mechanism*, ispirato all'architettura BERT [11], consentendo al modello di condizionare congiuntamente i contesti destro e sinistro delle sequenze di comportamento dell'utente, superando le limitazioni intrinseche dei paradigmi unidirezionali. Questo approccio consente alla rappresentazione di ogni elemento di incorporare informazioni dall'intero contesto circostante, portando a incorporazioni più robuste e olistiche delle preferenze dell'utente. L'adozione del *Cloze Task* o *Masked Model Language* per l'addestramento è un'altra innovazione fondamentale. Invece di prevedere l'elemento successivo in una sequenza, il *Cloze Task* prevede il mascheramento casuale degli elementi e l'utilizzo del modello per prevedere la loro identità in base al contesto circostante non mascherato. In questo modo si evita la perdita di informazioni, un problema comune quando si addestrano

modelli bidirezionali per la predizione sequenziale, e si garantisce che il modello non banalizzi il compito "vedendo" indirettamente l'elemento target.

Architettura e Principi di Funzionamento di BERT

Di seguito si riporta il problema che il paper [52] vuole risolvere. Nella raccomandazione sequenziale, sia $U = \{u_1, u_2, \ldots, u_{|U|}\}$ un insieme di utenti, $V = \{v_1, v_2, \ldots, v_{|V|}\}$ un insieme di oggetti, e la lista $S_u = [v_1^{(u)}, \ldots, v_t^{(u)}, \ldots, v_{n_u}^{(u)}]$ rappresenti la sequenza di interazioni in ordine cronologico per l'utente $u \in U$, dove $v_t^{(u)} \in V$ è l'oggetto con cui u ha interagito al tempo t e n_u è la lunghezza della sequenza di interazione per l'utente u.

Data la cronologia delle interazioni S_u, la raccomandazione sequenziale mira a prevedere l'oggetto con cui l'utente u interagirà al tempo $n_u + 1$. Questo può essere formalizzato modellando la probabilità su tutti gli oggetti possibili per l'utente u al tempo $n_u + 1$:

$$p\left(v_{n_u+1}^{(u)} = v \mid S_u\right)$$

BERT4Rec acronimo di *Bidirectional Encoder Representations from Transformers to a new task, Sequential Recommendation*, come illustrato in Fig.2.13, si basa su uno stack di livelli transformer, che consente di catturare le dipendenze complesse nelle sequenze di comportamento degli utenti in tutte le posizioni.

Embedding Layer Lo strato di embedding in BERT4Rec è essenziale per rappresentare gli oggetti di input in uno spazio vettoriale denso. Ogni oggetto nella sequenza riceve un embedding vettoriale di dimensione d, e agli embedding degli oggetti vengono aggiunti gli embedding posizionali per incorporare l'ordine della sequenza. Poiché il Transformer non è intrinsecamente consapevole dell'ordine delle sequenze, gli embedding posizionali consentono al modello di distinguere la posizione di ciascun oggetto. La rappresentazione di input per una posizione i è calcolata come:

$$h_i^0 = v_i + p_i$$

dove v_i è l'embedding d-dimensionale dell'oggetto i e p_i è l'embedding posizionale d-dimensionale. Questa combinazione garantisce che il modello possa utilizzare sia l'identità dell'oggetto sia la sua posizione nella sequenza.

Transformer Layer Il livello Transformer è il nucleo di BERT4Rec. Come da Fig.2.14 Ogni livello Transformer calcola iterativamente rappresentazioni nascoste su tutte le posizioni,

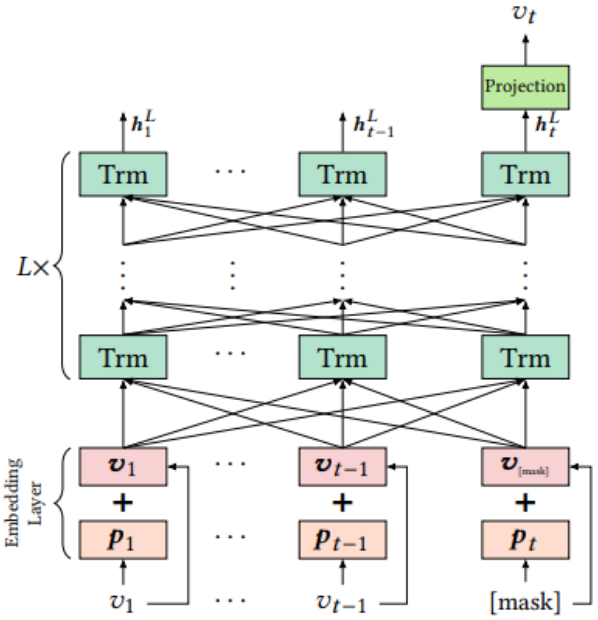

Figura 2.13: Architettura BERT4Rec [52].

affinando queste ultime scambiando informazioni utilizzando due sottocomponenti principali: *Multi-Head Self-Attention* e il *Position-wise Feed-Forward Network.*

Multi-Head Self-Attention Il meccanismo di Multi-Head Self-Attention consente di catturare le dipendenze tra oggetti in posizioni diverse della sequenza, indipendentemente dalla loro distanza. Questo è formalizzato come:

$$\text{Attention}(Q,K,V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d/h}}\right)V$$

Qui Q, K e V sono le matrici di query, key e value, proiettate dagli embedding di input usando parametri appresi. Il fattore di scala

$$\sqrt{d/h}$$

previene gradienti troppo piccoli, garantendo una stabilità durante l'addestramento. Questo processo viene applicato ad h subspazi paralleli, i cui risultati vengono concatenati e proiettati linearmente per formare l'output finale. Questo approccio permette al modello di focalizzarsi su aspetti diversi della sequenza contemporanea.

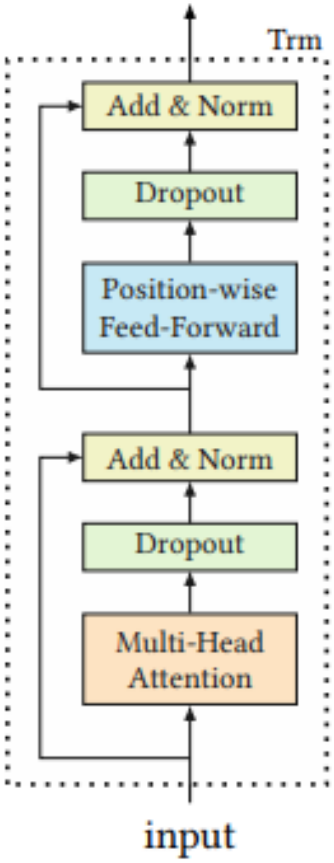

Figura 2.14: Architettura Transformet in BERT4Rec [52].

Position-wise Feed-Forward Network Dopo il self-attention, il Position-wise Feed-Forward Network introduce non linearità e cattura interazioni tra le dimensioni. Questo network applica due trasformazioni affini con una funzione di attivazione GELU intermedia:

$$\text{FFN}(x) = \text{GELU}(xW^{(1)} + b^{(1)})W^{(2)} + b^{(2)}$$

Qui $W^{(1)}, W^{(2)}, b^{(1)}, b^{(2)}$ sono parametri appresi. L'attivazione GELU aggiunge transizioni più morbide e migliora la capacità del modello di catturare pattern complessi. La feed-forward network viene applicata indipendentemente in ogni posizione, consentendo una raffinazione localizzata delle rappresentazioni.

Per stabilizzare l'addestramento e migliorare le prestazioni, vengono utilizzate connessioni residue e una normalizzazione a livello. Inoltre, si applica il dropout per mitigare l'overfitting.

Output Layer Dopo il passaggio attraverso L livelli *Transformer*, il modello restituisce una rappresentazione nascosta finale per ciascuna posizione nella sequenza. Per la raccomandazione sequenziale, l'oggetto di interesse (ad esempio un oggetto mascherato) viene previsto utilizzando questo stato nascosto finale. In particolare, la rappresentazione nascosta della posizione mascherata viene passata attraverso una rete feed-forward a due livelli con attivazione GELU per produrre una distribuzione di probabilità sull'intero set di oggetti:

$$P(v) = \text{softmax}\left(\text{GELU}(h_t^L W^P + b^P)E^\top + b^O\right)$$

Qui W^P, b^P, b^O sono parametri di proiezione e bias appresi, mentre E è la matrice di embedding condivisa per il set di oggetti. Condividere gli embedding tra lo strato di input e di output aiuta a ridurre l'overfitting e la dimensione del modello.

BERT4Rec vs tradizionali

BERT4Rec rappresenta un significativo progresso rispetto ai tradizionali sistemi di raccomandazione sequenziali, in particolare per la sua capacità di catturare dipendenze complesse e non lineari all'interno delle sequenze di interazione degli utenti. A differenza dei metodi tradizionali, come le catene di Markov o le reti neurali ricorrenti (RNN), che spesso si basano su rigidi presupposti di modellazione sequenziale, BERT4Rec impiega trasformatori bidirezionali che considerano simultaneamente le interazioni passate e future in una sequenza. Questa natura bidirezionale consente di superare la distorsione unidirezionale insita in modelli come le RNN o anche gli approcci autoregressivi, che elaborano le sequenze solo in una direzione, portando spesso a rappresentazioni subottimali dell'intento dell'utente. Inoltre, il *self-attention mechanism* consente al modello di assegnare dinamicamente livelli diversi di importanza alle diverse interazioni all'interno di una sequenza, indipendentemente dalla loro posizione, catturando così le dipendenze a lungo raggio e il contesto in modo più efficace rispetto ai modelli tradizionali che di solito faticano a gestire le dipendenze a distanza a causa della loro dipendenza dalla propagazione sequenziale delle informazioni. I sistemi di raccomandazione sequenziali tradizionali dipendono spesso da embeddings poco raffinati che non hanno la potenza espressiva necessaria per codificare le preferenze degli utenti, soprattutto quando si tratta di dati scarsi o rumorosi. Al contrario, BERT4Rec sfrutta embeddings profondi e contestualizzati, addestrati su dati di larga scala, che gli consentono di generare rappresentazioni robuste sia degli utenti che degli articoli. Questa capacità è particolarmente cruciale in scenari con interazioni scarse tra utenti e oggetti,

dove i metodi tradizionali spesso non riescono a generalizzare. Inoltre, il *pre-train* e *fine-tuning* di BERT4Rec consentono di trasferire efficacemente le rappresentazioni apprese in diversi domini, una caratteristica largamente assente nei metodi tradizionali, che sono tipicamente specifici per le attività e non hanno questa adattabilità. La scalabilità e l'efficienza dell'inferenza evidenziano ulteriormente la divergenza tra BERT4Rec e le sue controparti tradizionali. Sebbene i metodi tradizionali siano spesso leggeri ed efficienti dal punto di vista computazionale, sono inclini a sottoadattare i modelli di dati complessi. BERT4Rec, nonostante il costo computazionale più elevato, beneficia delle moderne accelerazioni hardware e delle tecniche di ottimizzazione, che gli consentono di gestire in modo efficiente insiemi di dati di grandi dimensioni e di fornire prestazioni all'avanguardia. Questo compromesso tra complessità computazionale e accuratezza predittiva sottolinea il cambiamento nella progettazione dei sistemi di raccomandazione verso modelli che privilegiano l'espressività e l'adattabilità alla complessità del mondo reale, qualità che BERT4Rec incarna in modo più efficace rispetto agli approcci tradizionali. Per questi motivi, BERT4Rec non solo fa progredire il campo della raccomandazione sequenziale, ma sfida anche i limiti e i presupposti dei suoi predecessori tradizionali, stabilendo un nuovo punto di riferimento per le prestazioni e la versatilità del settore.

2.4.3 Descrizione del processo di attacco

Quando si attaccano modelli di tipo black-box, l'approccio che si usa di base è quello mostrato in figura 2.15, si premette che in questo caso l'attaccante non conosce il dataset di training del modello ne i pesi del modello. L'architettura è in parte conosciuta, l'attaccante sa che è basata su transformer (Bert4Rec, Narm o Sasr) ma effettivamente non sa quale sta attaccando tra le tre. Inoltre, il risultato delle query è limitato, ovvero il sistema di raccomandazione black-box restituisce i primi 100 elementi.

La ***Model Extraction*** [38, 53] è il primo passo fondamentale per effettuare attacchi di tipo avversari. L'obiettivo di questa fase è quello di costruire un modello surrogato white-box che rispecchi fedelmente il comportamento del sistema black-box, consentendo ulteriori analisi o attacchi; questo perché, come descritto nella sezione 2.3.2, black-box significa che non conosciamo né l'architettura né i pesi del modello che vogliamo attaccare. Per trasformare un sistema black-box in un surrogato white-box, l'attaccante sintetizza i dati di addestramento e li utilizza per imitare il comportamento in uscita del modello black-box. Ciò comporta la generazione di sequenze di input, la loro presentazione all'API esposte dal sistema di raccomandazione e la

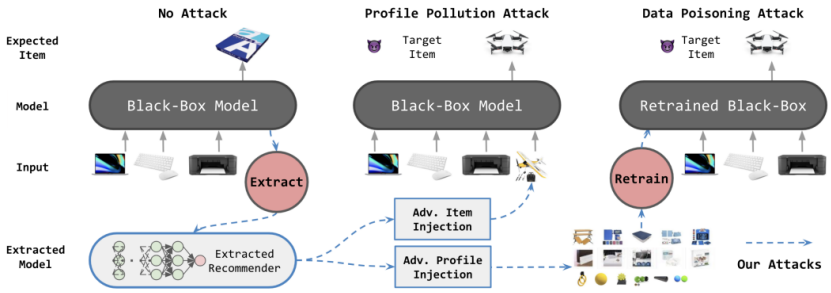

Figura 2.15: Processo di attacco contro sistemi blackbox [59].

registrazione delle raccomandazioni classificate risultanti. Il modello surrogato viene quindi addestrato utilizzando questi dati sintetici in un processo chiamato distillazione della conoscenza. L'obiettivo dell'addestramento è minimizzare la distanza tra l'output del modello black-box e le previsioni del modello surrogato, assicurando che quest'ultimo replichi il primo nel modo più accurato possibile. Una sfida significativa nell'estrazione del modello è rappresentata dal numero limitato di interrogazioni che un aggressore può effettuare al sistema black-box. Le API spesso impongono limiti alle query per mitigare gli abusi, il che limita la quantità di dati disponibili per l'addestramento del modello surrogato. Per ovviare a questo problema, gli aggressori danno priorità a strategie che massimizzano l'informativa di ciascuna query, sfruttando dati autoregressivi. La generazione di dati sintetici è una pietra miliare dell'estrazione di modelli in assenza di accesso a set di dati di addestramento reali. I dati generati servono come base per l'addestramento del modello surrogato white-box. Il documento esplora due metodi principali per la generazione di dati sintetici:

Random Data Generation in questo caso gli elementi vengono campionati in modo randomico dallo spazio di input per costruire sequenze. Queste sequenze vengono poi inviate all'API del modello black-box e la lista degli elementi vengono utilizzate per addestrare il surrogato. Sebbene questo metodo sia semplice e richieda presupposti minimi, soffre di limitazioni significative. Le sequenze casuali non riescono a catturare le dipendenze e i modelli inerenti al comportamento reale degli utenti. Di conseguenza, i dati mancano di rappresentatività e il modello surrogato addestrato presenta prestazioni subottime, in particolare quando si cerca di replicare i processi decisionali del modello black-box.

Autoregressive Data Generation Per superare i limiti del campionamento casuale, la generazione auto regressiva dei dati sfrutta la natura sequenziale dei sistemi di raccomandazione. Il processo inizia con la selezione casuale di un elemento iniziale che serve come punto di partenza di una sequenza. Questo elemento viene inviato all'API black-box, che restituisce un elenco classificato di raccomandazioni. Da questo elenco, l'elemento successivo della sequenza viene campionato e aggiunto alla sequenza esistente. La sequenza estesa viene quindi ripresentata all'API e il processo si ripete fino al raggiungimento della lunghezza desiderata della sequenza. La natura iterativa e guidata dal feedback di questo approccio garantisce che le sequenze generate riflettano le dipendenze temporali e contestuali tipiche delle interazioni reali degli utenti. Utilizzando il modello black-box stesso per guidare il processo di generazione, l'attaccante garantisce che i dati sintetici si allineino strettamente ai modelli appresi dal sistema. In 2.16 è presente una dimostrazione figurativa di come i dati siano costruiti. Rispetto al campionamento randomico, la generazione auto regressiva produce dati di qualità superiore che portano a modelli surrogati più accurati; questo perché, sfruttando le raccomandazioni della black-box, le sequenze finali hanno la stessa distribuzione che hanno quelle originali create da utenti veri [21].

Di seguito viene descritto l'attacco principale descritto nel paper, questo verrà preso in esame per lo sviluppo delle tecniche di difesa e i relativi miglioramenti dell'attacco.

Data Poisoning Attack Questa tipologia di attacchi si basa sull'idea di iniettare dati malevoli all'interno dei dati di addestramento del modello per influenzare le sue previsioni. Questo è possibile farlo perché il modello è periodicamente riaddestrato sui dati acquisiti dagli utenti per aggiornare i pesi del modello in modo tale che, il sistema di raccomandazione rifletta e apprenda le variazioni degli interessi degli utenti nel tempo. Nel contesto dei SRS, l'obiettivo è spesso quello di promuovere o declassare specifici articoli nelle raccomandazioni introducendo profili

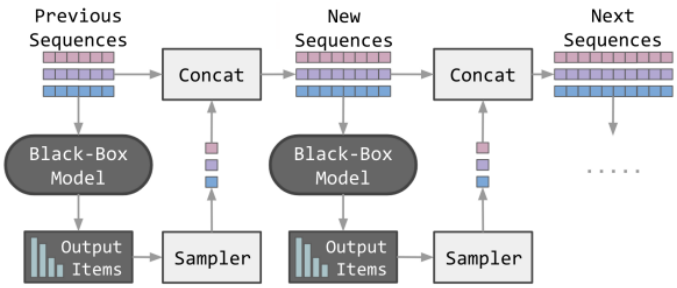

Figura 2.16: Processo di generazione autoregressiva dei dati per la creazione del surrogato [59].

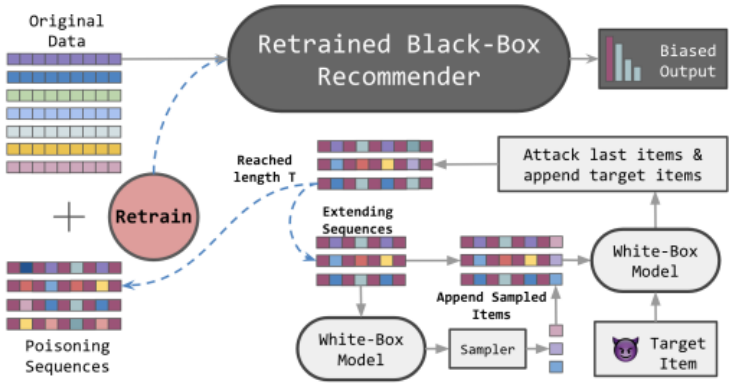

Figura 2.17: Processo di attacco di Data Poisoning [59].

utente falsi che manipolano i pesi appresi dal modello. Questi attacchi sfruttano il fatto che i sistemi di raccomandazione si basano sulle interazioni storiche degli utenti, iniettando profili malevoli, una volta incorporati nel processo di addestramento, fanno virare il comportamento del sistema verso gli obiettivi dell'attaccante. La generazione delle sequenze malevole avviene nel seguente modo, si inizia con lo scegliere l'insieme degli elementi da promuovere, insieme target. Ogni sequenza malevola corrisponde a un nuovo profilo e questa inizia con un elemento scelto dall'insieme target. L'attaccante può iniziare creando una sequenza di interazioni alternate tra l'elemento bersaglio e elementi apparentemente non correlati [51]; queste interazioni alternate simulano le preferenze degli utenti che associano fortemente l'elemento target ad altri elementi, ingannando di fatto il sistema di raccomandazione per amplificare la sua importanza. Il modello *white-box* estratto fornisce all'attaccante informazioni sul gradiente che guidano la creazione di sequenze che massimizzeranno l'esposizione dell'elemento target nel modello *black-box*, in Figura 2.17 è presente una visione grafica dell'attacco. La creazione delle sequenze avviene in modo bilanciato poiché modelli che seguono pattern rigidi sono facili da scovare con algoritmi di rilevamento delle anomalie. Per evitare questo problema, le sequenze vengono costruite con strategie per aumentare il realismo dei profili falsi, preservando al contempo il loro intento malevolo, ad esempio con l'introduzione di rumore o variabilità nelle interazioni, includendo elementi contestualmente rilevanti ma non correlati all'obiettivo. In questo modo si diversificano i profili e si riduce il rischio di rilevamento. I risultati sperimentali indicano che è particolarmente efficace per gli articoli meno popolari, l'iniezione di profili mirati può aumentare sostanzialmente la loro visibilità nelle raccomandazioni del sistema black-box. Ad esempio, gli articoli di coda, che inizialmente sono poco popolari, dopo l'avvelenamento registrano un aumento significativo.

Capitolo 3

Metodo Proposto

3.1 Motivazioni

Il sistema proposto in questo lavoro parte dall'analisi delle limitazioni degli attacchi di data poisoning descritti in [59], proponendone una revisione e un potenziamento. BERT4Rec, un modello noto per la sua efficacia nell'elaborazione sequenziale dei dati e nelle raccomandazioni personalizzate, è stato impiegato come black-box di riferimento per analizzare e confrontare il comportamento del sistema di raccomandazione sotto l'influenza di sequenze malevole.

La ricerca si concentra sulla progettazione di un meccanismo di difesa robusto contro attacchi avversari, basato su BERT4Rec e ottimizzato per la discriminazione delle sequenze malevole. Per ottenere questo risultato, il modello è stato modificato attraverso una revisione architetturale e un fine-tuning specifico per la classificazione, da qui nasce BERT4Def. La potenzialità di quest'ultimo deriva dalla sua configurazione: esso condivide alla base lo stesso modello black-box (BERT4Rec) utilizzato per effettuare raccomandazioni, garantendo così una base di conoscenza solida fondamentale per l'identificazione di sequenze che differiscono da quelle non malevole. In questo modo, il sistema, a regime, è in grado di rilevare utenti fraudolenti che tentano di alterare il sistema e rimuoverli, evitando così che contaminino il dataset su cui il sistema di raccomandazione effettua il riaddestramento.

Per validare l'efficacia della nostra difesa, abbiamo sviluppato sei modelli surrogati 2.4.3, progettati per emulare il comportamento della black-box originale. Questi modelli sono stati utilizzati per generare dati avversari con diverse strategie, consentendoci di sottoporre il sistema a situazioni ostili e di valutarne la resilienza. Attraverso un approccio iterativo, il nostro modello è

stato addestrato e testato in condizioni normali e in presenza di attacchi avversari, con l'obiettivo di esplorare le vulnerabilità del sistema e proporre soluzioni per migliorarne la robustezza e l'affidabilità.

Inoltre, si propone una revisione dell'attacco esistente, con l'obiettivo di ridurre parzialmente le prestazioni di BERT4Def. Questo approccio consente di individuare nuove vulnerabilità e migliorare ulteriormente le strategie di difesa.

Nel corso di questo capitolo si discutono le criticità da cui nasce l'idea di BERT4Def 3.2, si approfondisce la struttura di BERT4Rec e se ne giustificano le scelte architetturali 3.3, infine si propone un attacco in grado di superare l'ulteriore difesa 4.5.

3.2 Analisi criticità attacchi di Data Poisoning

Gli attacchi di *data poisoning* rappresentano una grave minaccia per i sistemi di raccomandazione sequenziali. Questi attacchi comportano l'iniezione di dati manipolati nel set di addestramento per alterare il processo di apprendimento del modello, portando a raccomandazioni distorte o vantaggiose per l'attaccante.

L'articolo analizzato [59] introduce un framework per eseguire tali attacchi su sistemi di raccomandazione sequenziali di cui non si conosce parzialmente l'architettura, evidenziando le vulnerabilità di questi sistemi in condizioni avverse con query limitate. Questa analisi esamina la metodologia dell'attacco di *data poisoning*, identificandone la fattibilità, i componenti critici, la rilevabilità e le debolezze algoritmiche sottostanti. L'impatto dell'attacco va oltre il semplice pregiudizio nelle raccomandazioni, poiché può degradare sistematicamente l'integrità del sistema, portando a esperienze utente subottimali. Comprendere la meccanica dell'attacco e i meccanismi di difesa è cruciale per sviluppare contromisure robuste che garantiscano la resilienza dei sistemi di raccomandazione.

Fattibilità ed Esecuzione degli Attacchi di Avvelenamento dei Dati Un attacco di *data poisoning* è fattibile quando l'attaccante può generare profili utente sintetici che influenzano la distribuzione dei dati di addestramento senza dover accedere al dataset originale. Gli autori di [59] propongono un approccio senza dati che sfrutta l'estrazione del modello per approssimare il comportamento del sistema target.

Il successo dell'attacco dipende dai seguenti fattori:

- Generazione di sequenze utente avversariali utilizzando modelli autoregressivi che imitano le interazioni reali.
- Creazione di profili di poisoning con sequenze basate sugli oggetti target.
- Sfruttamento delle funzioni di perdita basate sul ranking per perfezionare la generazione delle sequenze avversariali.
- Affinamento iterativo dei profili avversariali per aumentarne l'efficacia a lungo termine.

Addestrando un modello surrogato utilizzando dati sintetici recuperati dall'API del sistema di raccomandazione, gli attaccanti possono approssimare la funzione di ranking del modello target. I profili avvelenati, una volta iniettati nel set di addestramento, amplificano l'esposizione dell'elemento target, modificando efficacemente il comportamento del sistema dopo il riaddestramento. La difficoltà nel distinguere i dati manipolati dalle interazioni autentiche è una vulnerabilità chiave sfruttata dagli attaccanti, rendendo necessarie strategie di difesa proattive.

3.2.1 Vulnerabilità Algoritmiche Critiche

Il successo dell'attacco dipende dall'affidamento del sistema di raccomandazione sequenziale sulla modellazione autoregressiva, in cui ogni previsione è condizionata agli stati precedenti. La natura autoregressiva consente all'attaccante di costruire sequenze statisticamente simili ai comportamenti autentici degli utenti, rendendo difficile distinguere le interazioni avversariali da quelle legittime. Gli aspetti algoritmici più critici che consentono il *data poisoning* efficace includono:

- **Ottimizzazione Basata sul Ranking:** A differenza dei modelli di classificazione, i sistemi di raccomandazione operano su output ordinati. L'attaccante sfrutta il ranking della *white-box* per garantire che le sequenze avvelenate elevino il rango dell'elemento target rispetto agli altri, sfruttando pattern di *co-visitation*.
- **Selezione Avversariale Guidata dal Gradiente:** Utilizzando modelli *white-box*, gli attaccanti calcolano i gradienti per identificare le perturbazioni ottimali degli elementi, migliorando l'efficacia del profilo avversariale nella promozione dell'elemento target.

- **Sfruttamento delle Caratteristiche Latenti:** Gli attaccanti possono manipolare gli embedding delle caratteristiche latenti, facendo sì che i profili avversariali si integrino perfettamente con i dati utente reali.
- **Manipolazione dei Pattern Temporali:** Le sequenze avvelenate possono essere progettate per introdurre correlazioni temporali che rafforzano ulteriormente l'esposizione dell'elemento target.

La complessità di questi attacchi è mitigata dall'incapacità del modello di differenziare tra sequenze avversariali costruite artificialmente e sequenze genuine, dato che le distribuzioni dei dati di addestramento vengono apprese in modo euristico piuttosto che verificate rispetto a una verità di base. Ciò apre la porta a tecniche avanzate di apprendimento avversariale che affinano continuamente le strategie di attacco in risposta all'evoluzione degli algoritmi di raccomandazione.

Rilevabilità e Meccanismi di Difesa Nonostante l'efficacia dell'attacco, il rilevamento dell'avvelenamento dei dati rimane possibile grazie alle irregolarità statistiche introdotte dall'attacco. Gli aspetti chiave che rendono possibile la rilevazione includono:

- **Pattern di Co-Visita Innaturali:** I profili iniettati contengono co-occorrenze artificialmente elevate di elementi target con elementi irrilevanti.
- **Test di Invarianza del Modello:** Il confronto delle variazioni nelle raccomandazioni prima e dopo il riaddestramento con dati avvelenati evidenzia fluttuazioni insolite nei ranking degli elementi.
- **Strategie di Difesa Ibride:** La combinazione di addestramento avversariale e rilevamento delle anomalie può fornire una resistenza robusta contro metodologie di attacco in evoluzione.
- **Monitoraggio in Tempo Reale:** I sistemi possono implementare tecniche di sorveglianza online per identificare pattern di raccomandazione anomali prima che diventino dirompenti.

In particolare, la tecnica di difesa che presenteremo si basa sulle assunzioni che poiché i pattern di *covisitation* introducono all'interno delle sequenze, seppur soggette a rumore, degli

elementi con scarsa correlazione, la conoscenza pregressa del sistema di raccomandazione sulle sequenze malevole, unita alla potenzialità del *multi-head self-attention* possa scovare facilmente le sequeze che seguono questo pattern.

3.3 Architettura di difesa: BERT4Def

La tecnica di difesa che intendiamo presentare si basa su un'analisi approfondita del comportamento dei sistemi di raccomandazione in presenza di sequenze malevole, sfruttando le proprietà dei pattern di *covisitation*. Questi pattern emergono quando più utenti visitano gli stessi elementi in un breve arco di tempo, generando correlazioni all'interno delle sequenze. Tuttavia, quando tali pattern vengono introdotti artificialmente per manipolare il sistema, le sequenze risultanti tendono a contenere elementi con una correlazione intrinsecamente più debole a causa del rumore introdotto dal comportamento anomalo e all'alternarsi con elementi che solitamente sono insoliti essere nelle vicinanze.

L'idea centrale della nostra tecnica di difesa è che la conoscenza pregressa del sistema di raccomandazione sulle sequenze benevole può essere combinata con la capacità del multi-head self-attention per individuare con maggiore precisione le sequenze sospette, ovvero le sequenze che differiscono da quelle viste durante l'addestramento. In particolare, il meccanismo di self-attention consente di modellare le dipendenze tra gli elementi delle sequenze e di identificare quelle in cui i pattern di *covisitation* sono stati generati in modo artificiale. Grazie alla capacità di focalizzarsi su diverse parti della sequenza contemporaneamente (multi-head), il modello è in grado di evidenziare anomalie che sarebbero difficilmente rilevabili con approcci più tradizionali.

In sintesi, il nostro metodo sfrutta la capacità del *multi-head self-attention* di individuare correlazioni deboli e pattern anomali, consentendo una rilevazione più efficace degli attacchi che si basano sulla manipolazione dei pattern di *covisitation*.

Come illustrato in Figura 3.1, l'architettura proposta si basa su quattro componenti principali. La prima è la *black box*, che rappresenta il nostro sistema di raccomandazione, del quale l'attaccante non possiede alcuna conoscenza, ad eccezione del fatto che si tratti di un sistema di raccomandazione sequenziale. Il funzionamento e le caratteristiche di questo sistema sono stati ampiamente descritti nel Capitolo 2.4.2. Per il sistema di raccomandazione è stato scelto BERT4Rec [52], poiché utilizza tecniche avanzate condivise con i più noti *Large Language Models*, come i *Transformer Layer* e il *Masked Model Language*, discusse nel Capitolo 2.2.

Queste tecniche dimostrano prestazioni superiori rispetto a sistemi di raccomandazione simili, introducendo un grado di complessità maggiore sia per le strategie di attacco che di difesa. In particolare, gli attaccanti devono costruire sequenze che siano sempre più simili a quelle reali, mantenendo al contempo proprietà malevole. Le componenti *white-box* vengono estratte come descritto nel Capitolo 2.4, discuteremo nel dettaglio di queste in seguito. Queste componenti sono particolarmente fondamentali per la generazione dei dati necessari al classificatore. Infine, il modulo BERT4Def, basato sulla black-box, è progettato per discriminare tra sequenze malevole e non malevole.

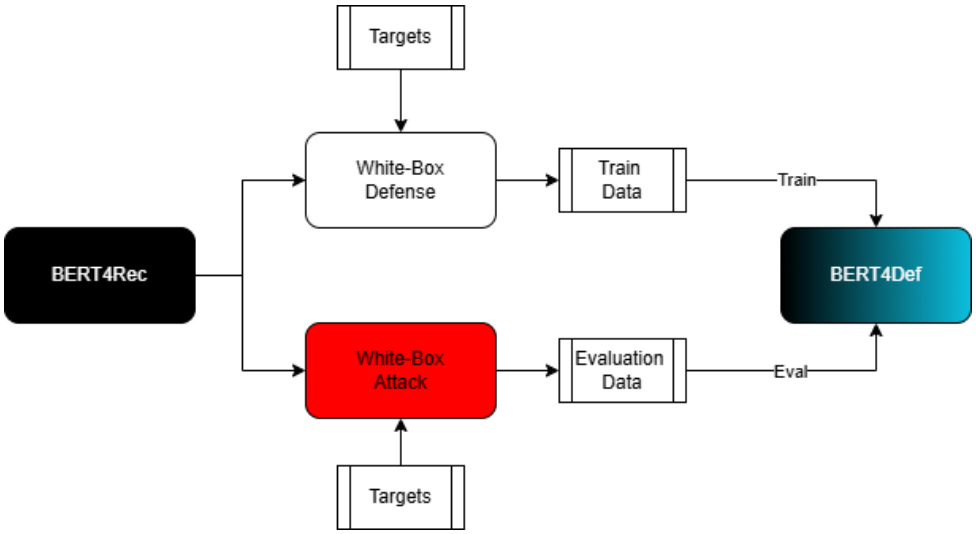

Figura 3.1: Architettura tecnica di difesa proposta

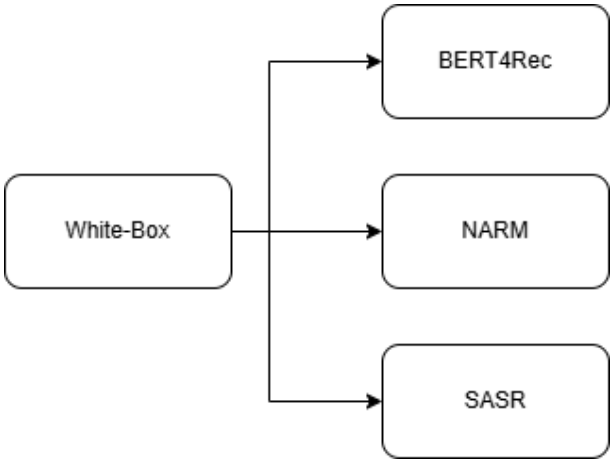

Figura 3.2: Tipologie di *white-box* utilizzate

Capitolo 4

Esperimenti e Risultati

In questo capitolo è presente un resoconto completo degli esperimenti condotti per valutare le prestazioni e la robustezza del nostro sistema di discriminazione delle sequenze *BERT4Def*. L'obiettivo principale di questi esperimenti è quello di valutare l'efficacia della tecnica di difesa sviluppata per proteggere il sistema di raccomandazione *BERT4Rec* e trovare la migliore configurazione per creare un sistema in grado di discriminare le sequenze nella maggior parte dei casi. Data la crescente importanza delle tecniche dei LLMs su larga scala e la loro applicazione nei sistemi di raccomandazione, diventa cruciale capire come potersi difendere e fin quando è possibile difendersi. Esploriamo l'ipotesi che l'integrazione di *BERT4Def*, addestrato a rilevare schemi anomali e incongruenze nelle raccomandazioni, possa migliorare significativamente la robustezza complessiva del sistema, in questo contesto, analizziamo i compromessi tra accuratezza e robustezza. Un'ulteriore dimensione della nostra indagine riguarda le sfide di fondo poste dalla natura sequenziale delle interazioni degli utenti in dataset come ML-1M e Beauty. Ipotizziamo che le dinamiche temporali catturate da questi dataset non solo forniscano un terreno fertile per scenari di raccomandazione realistici, ma amplifichino anche l'impatto delle perturbazioni avversarie sulle previsioni del modello. Infine presenteremo come è possibile aggirare, in parte, queste difese con sequenze malevole in grado in parte di scavalcare le ulteriori difese.

Il capitolo è organizzato in diverse sezioni interconnesse che dettagliano metodicamente la progettazione, l'implementazione e i risultati. Si inizia con una descrizione dettagliata dei set di dati utilizzati nel nostro studio, illustrando le motivazioni alla base della scelta dei set di dati ML-1M e Beauty. Questi dataset sono particolarmente adatti a compiti di raccomandazione se-

quenziale grazie alla struttura temporale intrinseca delle interazioni degli utenti, che consente di modellare efficacemente le preferenze dinamiche degli utenti. Dopo la panoramica sul set di dati, si spiega brevemente la metrica di valutazione usata, in questo studio è inclusa anche una spiegazione approfondita dei concetti di falsi positivi e falsi negativi, dato il loro ruolo fondamentale nella comprensione delle prestazioni e dell'affidabilità degli approcci proposti. Successivamente, il capitolo presenta l'impostazione sperimentale e le procedure di addestramento dei modelli black-box. Questa sezione illustra le scelte architetturali, i parametri di addestramento e le strategie di ottimizzazione impiegate per ottenere raccomandazioni sequenziali ad alte prestazioni. In seguito, viene fornita un'esposizione simile per i modelli white-box, progettati per offrire maggiore interpretabilità e trasparenza.

Infine è presente una sezione dedicata agli esperimenti condotti per la valutazione di *BERT4Def*. Questo è fondamentale per capire in che misura gli attacchi avversari possono essere mitigati attraverso meccanismi di rilevamento, fornendo un'analisi completa delle prestazioni del classificatore, compreso un resoconto dettagliato dei suoi iperparametri, del regime di addestramento e dei risultati della valutazione in base alle metriche descritte. I risultati di questi esperimenti sono sintetizzati per offrire spunti di riflessione sui meriti relativi dell'incorporazione di un livello di rilevamento all'interno della pipeline di raccomandazione. Infine si descriverà come è possibile aggirare in modo marginale le misure di sicurezza adottate.

4.1 Datasets

In questa sezione, forniamo un'esposizione dettagliata dei dataset utilizzati nel nostro framework sperimentale, sottolineando la loro idoneità a compiti di raccomandazione sequenziale e la loro capacità di esprimere le dinamiche tra utenti e articoli nel corso del tempo. I dataset principali utilizzati per addestrare la *black-box* nel nostro studio sono MovieLens 1M (ML-1M) [20] e Amazon Beauty. Ognuno di questi dataset presenta caratteristiche uniche che li rendono particolarmente adatti a studiare i modelli di interazione temporale e l'evoluzione delle preferenze degli utenti.

- **ML-1M** derivato dall'archivio MovieLens, comprende una ricca collezione di valutazioni di film fornite da un'ampia base di utenti. Con un milione di valutazioni che coprono un'ampia gamma di film, questo dataset rappresenta un ambiente interessante per lo studio di raccomandazione sequenziale. La struttura intrinseca dei dati, caratterizzata

da un ordinamento temporale delle interazioni degli utenti, facilita la modellazione del comportamento e delle preferenze dinamiche degli utenti. Il dataset ML-1M permette ai ricercatori di tracciare l'evoluzione delle preferenze di un utente nel tempo, consentendo così lo sviluppo e la validazione di modelli in grado di cogliere sia le tendenze a breve termine sia gli interessi a lungo termine. La natura sequenziale del consumo di film, in cui le interazioni passate informano le scelte di visione future, fornisce un banco di prova ideale per valutare modelli che incorporano dipendenze temporali e informazioni contestuali.

- **Beauty** offre una grande quantità di informazioni sulle interazioni dei consumatori nell'ambito dei prodotti di bellezza. Questo set di dati comprende le recensioni degli utenti, le valutazioni e i metadati associati a un'ampia gamma di prodotti di bellezza, presentando così una visione del comportamento degli utenti in un contesto di e-commerce. Le interazioni di questi con i prodotti di bellezza, dall'esplorazione iniziale alle decisioni di acquisto, sono intrinsecamente sequenziali e riflettono l'evoluzione delle preferenze influenzate da tendenze, fattori stagionali ed esigenze personali. Il set di dati Beauty, quindi, ci permette di esplorare le sfumature della raccomandazione sequenziale, in quanto rispecchia la complessa interazione tra le traiettorie dei singoli utenti e l'evoluzione temporale della popolarità e della rilevanza dei prodotti in un ambiente commerciale reale.

I dataset ML-1M eBeauty sono completi per la valutazione dei modelli di raccomandazione sequenziali, grazie all'ordinamento temporale intrinseco delle interazioni degli utenti che essi rappresentano. Questi datasets sono particolarmente adatti alla modellizzazione delle preferenze dinamiche degli utenti nel tempo, permettendo così una valutazione rigorosa dei sistemi di raccomandazione in ambienti che rispecchiano fedelmente scenari reali. Il *preprocessing* dei dati viene eseguito in conformità con l'algoritmo descritto in [59], garantendo coerenza e riproducibilità nella gestione dei dati temporali. Nella tabella 4.1 sono riportate le statistiche più importanti dei dati, queste sono utili alla costruzione dei modelli

.

Datasets	**Users**	**Items**	**Avg.len**	**Max.len**
ML-1M	6040	3416	166	2277
Beauty	40226	54542	9	293

Tabella 4.1: Dataset statistics: number of users, average sequence length (Avg.len), and maximum sequence length (Max.len).

Nel contesto dell'addestramento e della valutazione del framework BERT4Def, il processo inizia con l'estrazione di molteplici modelli *white-box* ognuno di questi genera successivamente un set di dati malevoli, seguendo la procedura autoregressiva descritta nel capitolo 2.4. È importante sottolineare che, data la diversità dei modelli *white-box* – ognuno dei quali progettato con un'architettura di base diversa – i dataset di sequenze malevoli risultanti sono disgiunti. Tale disgiunzione garantisce che le anomalie introdotte da ciascun modello white-box siano uniche e non sovrapposte, migliorando così la robustezza del processo di valutazione. Successivamente, le sequenze malevoli vengono aggregate con quelle che rispecchiano il comportamento normale e benigno degli utenti. Questa combinazione porta alla creazione di un dataset composito, opportunamente etichettato per distinguere in modo chiaro tra interazioni malevole e benigno. Tale strategia di etichettatura è fondamentale, poiché consente di separare nettamente il comportamento utente normale dalle modifiche avversarie, elemento essenziale per l'analisi successiva e per la valutazione del modello.

Per ciascun dataset in esame, vengono impiegati due tipi distinti di modelli white-box, come descritto in Figura 3.1, un tipo è dedicato alla generazione dei dati per il training, mentre l'altro è specificamente incaricato di produrre i dati per la valutazione. Ciascuno di questi modelli white-box sfrutta un modello di base differente, come illustrato in Figura 3.2, e di conseguenza è responsabile della generazione di una variante unica del dataset malevolo. Per una rappresentazione più dettagliata e accurata di questo processo di generazione dei dati, si rimanda alla Figura 4.1. L'impiego di una molteplicità di modelli white-box di tipo differente risulta determinante per consentire una cross-validation completa. Tale procedura attraversa tutte le possibili combinazioni dei dataset nella fase di valutazione, assicurando così che la robustezza e la resilienza del framework BERT4Def siano esaminate in maniera approfondita attraverso un'ampia gamma di scenari avversari. Questo approccio non solo incrementa l'affidabilità dei risultati sperimentali, ma fornisce anche una visione più profonda delle vulnerabilità e dei punti di forza dei modelli di raccomandazione sequenziali in condizioni di attacco avversario.

4.2 Metriche

Per quanto riguarda la valutazione di *BERT4Def* incaricato di rilevare le raccomandazioni malevole si adotta l'*F1-Score* che bilancia precisione e recall per offrire una visione più approfondita delle prestazioni, L'*accuracy* non viene presa in considerazione in quanto il problema

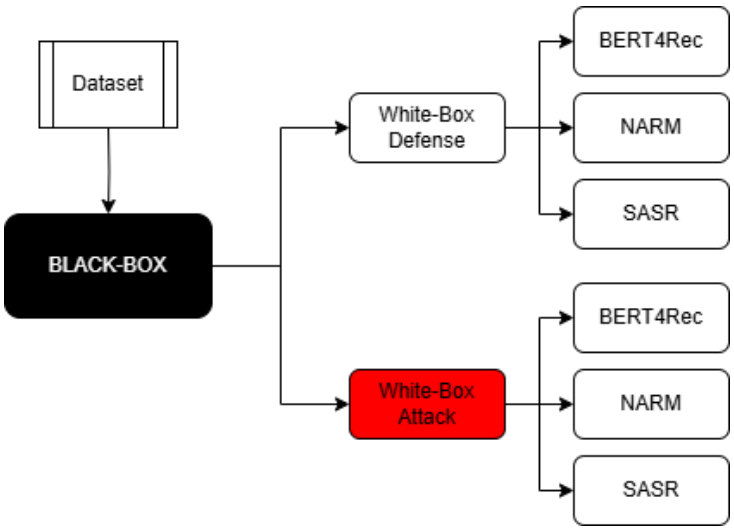

Figura 4.1: WorkFlow per la generazione dei dataset malevoli

che stiamo affrontando è sbilanciato, ovvero siamo in una posizione in cui i campioni di casi benigni sono in maggior numero rispetto i malevoli. L'F1-Score è calcolato come:

F1-Score

$$\text{F1-Score} = 2 \cdot \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{4.1}$$

dove la precisione e il recall sono definiti, rispettivamente, come:

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \tag{4.2}$$

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \tag{4.3}$$

L'F1-Score risulta particolarmente adatto allo studio condotto in quanto riesce a cogliere l'equilibrio tra il rischio di mancare istanze rilevanti (falsi negativi) e quello di classificare erroneamente come malevoli comportamenti benigni (falsi positivi). Un aspetto fondamentale nella valutazione degli attacchi avversari è la comprensione dei falsi positivi e dei falsi negativi. Un falso positivo si verifica quando il classificatore etichetta erroneamente un'interazione benigna come malevola, mentre un falso negativo si verifica quando un'interazione veramente malevola viene

erroneamente classificata come benigno. Le implicazioni di tali errori sono significative: un elevato tasso di falsi positivi può portare a interventi non necessari e a una riduzione della fiducia degli utenti, mentre un elevato tasso di falsi negativi può permettere il passaggio inosservato di azioni avversarie, compromettendo l'integrità complessiva del sistema di raccomandazione.

4.3 Addestramento Modelli

In questa sezione descriveremo come abbiamo addestrato il sistema di raccomandazione e i surrogati per la generazione dei dati malevoli per il *train* e l'*evaluation*.

4.3.1 Sistema di raccomandazione - BlackBox

Come sistema di raccomandazione con la quale effettuare tutti gli esperimenti si è scelto *BERT4Rec*, il cui funzionamento è stato ampiamente discusso in 2.4.2. L'addestramento di BERT4Rec richiede un'attenta configurazione dell'architettura del modello e dei parametri di ottimizzazione, al fine di garantire prestazioni elevate in differenti dataset. La scelta dei parametri dipende dalle caratteristiche del dataset, come la lunghezza media delle sequenze di interazioni e la diversità degli elementi raccomandati. I tre dataset considerati in questo studio—ML-1M e Beauty—presentano caratteristiche distinte, richiedendo una parametrizzazione adattata per ogni scenario. L'architettura di BERT4Rec si basa su un modello Transformer bidirezionale, che apprende rappresentazioni latenti delle sequenze di interazioni degli utenti utilizzando meccanismi di autoattenzione. Un elemento cruciale nella configurazione del modello è la dimensione dello spazio latente, che è fissata a 64 per tutti i dataset. Tuttavia, altri parametri variano per adattarsi alle peculiarità dei dati. Uno degli aspetti fondamentali dell'addestramento è la lunghezza massima della sequenza di interazioni considerate durante la fase di apprendimento. Nel dataset ML-1M, caratterizzato da sequenze di interazioni più lunghe, viene impostato a 200, mentre il dataset Beauty, che presenta sequenze più brevi, è ridotto a 50 per migliorare l'efficienza computazionale. Un altro parametro essenziale è la probabilità di mascheramento, che determina la frazione di elementi nelle sequenze di interazioni che vengono nascosti durante la fase di addestramento, in modo che il modello impari a prevedere gli elementi mancanti. Nei dataset con interazioni più strutturate, come ML-1M, viene fissato a 0.2, mentre nei dataset più rumorosi, come Beauty, è aumentato a 0.6 per enfatizzare la capacità del modello di generalizzare da dati parzialmente osservati.

Parametro	ML-1M	Beauty
Hidden Units	64	64
Max Length	200	50
Mask Probability	0.2	0.6
Dropout	0.1	0.5
Attn Dropout	0.1	0.2
Max Predictions	40	30
Learning Rate	0.001	0.001
Weight Decay	0.01	0.01

Tabella 4.2: Parametri di addestramento di BERT4Rec per i dataset ML-1M e Beauty.

La fase di addestramento utilizza un ottimizzatore AdamW con *learning rate* pari a 0.001 e un *weight decay* pari a 0.01. Infine, la probabilità di dropout sul livello di embedding e sul livello di attenzione è mantenuta più bassa in ML-1M (0.1), mentre viene aumentata in Beauty (0.5) per adattarsi alla complessità delle interazioni utente-oggetto. Fare riferimento alla tabella 4.2 che sintetizza i parametri più importanti utilizzati per il training.

4.3.2 Surrogati - WhiteBox

Come ampiamente discusso nei paragrafi precedenti, i modelli surrogati sono generati secondo la metodologia descritta in [59]. Non si è ritenuto necessario apportare modifiche all'architettura o al processo di *model extraction*, in quanto l'obiettivo principale di questa tesi è sviluppare strategie di difesa contro gli attacchi condotti mediante le sequenze malevole generate da questa specifica tipologia di surrogato.

Nello specifico, vengono creati sei modelli white box: tre sono impiegati per generare i dati malevoli utilizzati nell'addestramento di BERT4Def, mentre gli altri tre sono utilizzati per generare i dati necessari alla simulazione degli attacchi e alla valutazione delle prestazioni del modello risultante. Questo approccio consente di condurre un'analisi incrociata al fine di determinare la configurazione ottimale e valutare in quali scenari BERT4Def dimostri un'elevata efficacia e, più significativamente, in quali contesti presenta delle limitazioni nella capacità di difesa.

Il numero di tre modelli per ciascun gruppo è determinato dall'architettura sottostante adottata nella configurazione white box, che prevede tre opzioni: BERT4Rec, NARM e SASR. Si

sottolinea che tutte le *white box* sono generate a partire dal processo di *model extraction* eseguito sul sistema di raccomandazione *BERT4Rec* precedentemente addestrato.

4.4 Prestazioni BERT4Def

In questa sezione si analizzano i risultati sperimentali di *BERT4Def*, si procede nella descrizione dei parametri dell'architettura, dei parametri di addestramento e dei risultati dei test. Per la descrizione dei risultati si procede schematicamente nel seguente modo. D'apprima si addestra *BERT4Def* con le sequenze malevole generate dai tre surrogati estratti per la difesa e successivamente vengono valutate le metriche sopra descritte sui dataset malevoli, uno per volta, generati dai surrogati per l'attacco, per ogni tipologia di architettura del surrogato.

4.4.1 Dataset: ML-1M

In questa sezione, si esaminano le prestazioni del modello *BERT4Def* con il dataset: MovieLens con un milione di interazioni *user-items*. Il modello è stato addestrato utilizzando un tasso di apprendimento di 0.001 e un fattore di regolarizzazione L2 di 0.001. È importante sottolineare come il modello dimostri una notevole capacità di apprendimento rapido nella distinzione tra sequenze malevole e benevole. In particolare, i dati di addestramento illustrati in Figura 4.2 evidenziano un processo di addestramento condotto su sole 5 epoche, 30 sequenze malevole e 30 benigne. Questo comportamento è attribuibile al fatto che *BERT4Def* incorpora la conoscenza del modello *blackbox BERT4Rec*, da cui vengono estratti i surrogati. Di conseguenza, le sequenze che si discostano da quelle precedentemente osservate sono facilmente identificabili come malevole.

Training Model	Eval: BERT4Rec	Eval: NARM	Eval: SASRec
BERT4Rec	0.97	0.97	0.97
NARM	0.89	0.97	0.98
SASRec	0.75	0.98	0.85

Tabella 4.3: Risultati F1-Score su dataset ML-1M al variare del modello con differenti dati di valutazione. La testa della tabella indica il tipo modello white-box con cui sono stati generati i dati di valutazione, la prima colonna a sinistra mostra con quale white-box sono stati generati i dati di addestramento di BERT4Def.

I risultati presentati nella Tabella 4.3 forniscono un'analisi delle prestazioni ottenute da diverse configurazioni di addestramento quando vengono valutate su tre modelli distinti: BERT4Rec, NARM e SASRec.

Il modello addestrato con dati provenienti da *BERT4Rec* mostra un F1-Score costantemente elevato (0.97) su tutte le valutazioni, indipendentemente dall'architettura utilizzata per l'*evaluation*. Questo suggerisce che quando BERT4Def viene addestrato con dati che provengono dal modello da cui è derivato, questo riesce a mantenere una forte capacità di generalizzazione. Il fatto che le prestazioni rimangano invariate anche nei test con NARM e SASRec indica che le rappresentazioni apprese dal modello sono robuste e trasferibili tra architetture differenti.

L'addestramento con con dati provenienti da *NARM* mostra invece una situazione diversa. In questo caso, l'F1-Score cala quando viene valutato su *BERT4Rec* (0.89), mentre rimane elevato per *NARM* (0.97) e *SASRec* (0.98). Questo risultato indica che, sebbene *NARM* sia in grado di catturare pattern sequenziali utili, il modello addestrato in questa configurazione non si adatta altrettanto bene alla valutazione su *BERT4Rec*. Tuttavia, il miglioramento delle prestazioni nel test su *SASRec* suggerisce che l'addestramento con *NARM* aiuti a catturare le dipendenze a lungo termine, il che si allinea bene con l'architettura di *SASRec*.

Un altro fenomeno osservato risulta quando l'addestramento viene effettuato con il modello addestrato con dati da *SASR*. In questo caso, si osserva un calo significativo dell'F1-Score quando il modello viene valutato su *BERT4Rec* (0.75). Questo suggerisce che l'addestramento con *SASRec* introduce delle differenze nel modo in cui il modello apprende le sequenze, rendendolo meno compatibile con il paradigma di *BERT4Rec*. D'altra parte, le prestazioni rimangono elevate nel test con *NARM* (0.98), il che suggerisce una maggiore compatibilità tra queste due architetture. Tuttavia, il punteggio *F1-score* si riduce anche nella valutazione con *SASRec* stesso (0.85), indicando che l'addestramento con *SASRec* potrebbe non essere efficace quanto lo è con *BERT4Rec*.

I risultati suggeriscono che l'addestramento con dati generati dalla *white box* basata su *BERT4Rec* è la strategia più efficace, come dimostrato dagli *F1-Score* uniformi (0.97). Al contrario, l'addestramento tra modelli diversi può introdurre delle discrepanze, con *NARM* che mostra una maggiore capacità di adattamento rispetto a *SASRec*. Questi risultati suggeriscono che, nella progettazione di strategie di addestramento per i sistemi di raccomandazione, l'uso di un approccio in cui la tecnica di difesa condivide la conoscenze del sistema di raccomandazione e questa è addestrata su dati provenienti dalla *model extraction* del sistema di raccomandazione stesso è sicuramente la scelta migliore rispetto le altre.

Al fine di provare la veridicità dei risultati, abbiamo testato il modello su 500 sequenze beninge estratte dal dataset principale di riferimento per capire se la comprensione delle sequenze benigne sia valida. Queste 500 sequenze benigne non sono mai state usate durante tutto il processo di addestramento, ne in fase di *fine tuning* di *BERT4Def* ne in fase di addestramento del sistema di raccomandazione *BERT4Rec* dalla quale vengono distillate le *white box*. *BERT4Def* le classifica correttamente con un'accuratezza del 95%. Questo dimostra che il modello presentato riesce a discriminare correttamente combinazioni di sequenze che non ha mai visto prima.

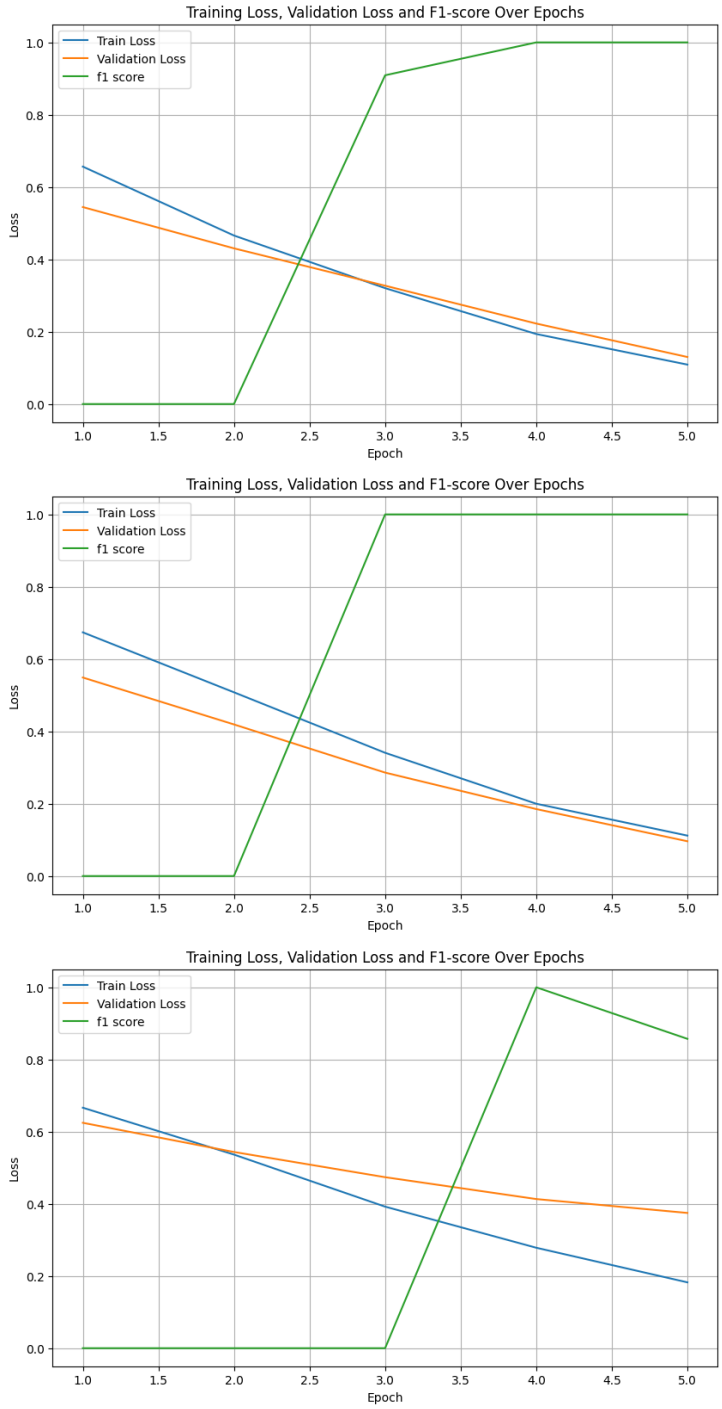

Figura 4.2: Il grafico rappresenta il processo di addestramento di BERT4Def con dati malevoli generati da white-box, basate rispettivamente su BERT4Rec, NARM e SASR (dall'alto verso il basso)

4.4.2 Dataset: Beauty

In questa sezione esaminiamo le prestazioni del modello *BERT4Def* dataset: Beauty, analizzando il suo comportamento rispetto ai modelli di riferimento. Come nei precedenti esperimenti, il modello è stato addestrato utilizzando un tasso di apprendimento di 0.001 e un fattore di regolarizzazione L2 di 0.01. Anche in questo caso, il modello mostra un'elevata capacità di discriminazione tra sequenze benigne e malevole. L'addestramento è stato condotto su un set di 250 sequenze malevole e 500 sequenze benigne per un totale di 10 epoch, i risultati d'addestramento sono illustrati in Figura 4.3. Tuttavia, osserviamo che per ottenere prestazioni comparabili a quelle ottenute su altri dataset, è necessario un numero maggiore di dati di addestramento e un numero superiore di epoche. Questa necessità è giustificata dalla maggiore complessità del dataset Beauty, che presenta un volume di dati significativamente superiore rispetto ai dataset precedenti, sia in termini di numero di utenti e item, sia in termini di interazioni complessive 4.1.

Un altro fattore rilevante è la dimensione delle sequenze: in questo dataset, la lunghezza massima delle sequenze è limitata a 50 elementi. Questa caratteristica implica che il modello debba essere esposto a un numero maggiore di sequenze per identificare con precisione i pattern comportamentali degli utenti. Di conseguenza, il processo di apprendimento richiede un dataset di addestramento più ampio e una fase di ottimizzazione più estesa per ottenere un modello efficace e stabile.

Nella Tabella 4.4 riportiamo i risultati delle valutazioni del modello al variare dei dati di addestramento e del modello utilizzato per generare i dati di valutazione. Come per gli esperimenti precedenti, emerge chiaramente che l'allenamento del modello su dati generati a partire da surrogati basati su *BERT4Rec* produce i risultati migliori in termini di F1-score. Analizzando i valori nella tabella, si nota che il modello addestrato con dati generati da *BERT4Rec* mantiene prestazioni elevate su tutte le valutazioni, con un F1-score di 0.92 quando valutato su *BERT4Rec*, 0.97 su *NARM* e 0.91 su *SASRec*.

In confronto, il modello addestrato con dati generati da *NARM* mostra un *F1-score* inferiore, con valori pari a 0.81 su *BERT4Rec*, 0.87 su *NARM* e 0.83 su *SASRec*. Questo indica una minore generalizzazione rispetto al modello basato su *BERT4Rec*, probabilmente a causa delle limitazioni di *NARM* nel catturare relazioni complesse tra gli elementi della sequenza. D'altro canto, il modello addestrato con dati generati da *SASRec* mostra prestazioni intermedie, con un F1-score di 0.89 su *BERT4Rec*, 0.97 su *NARM* e 0.95 su *SASRec*, suggerendo che, pur essendo

Training Model	Eval: BERT4Rec	Eval: NARM	Eval: SASRec
BERT4Rec	0.92	0.97	0.91
NARM	0.81	0.87	0.83
SASRec	0.89	0.97	0.95

Tabella 4.4: Risultati F1-Score su dataset Beauty al variare del modello con differenti dati di valutazione. La testa della tabella indica il tipo modello white-box con cui sono stati generati i dati di valutazione, la prima colonna a sinistra mostra con quale white-box sono stati generati i dati di addestramento di BERT4Def.

efficace, non raggiunge l'elevata generalizzazione del modello *BERT4Rec*.

Questi risultati confermano che l'utilizzo di *BERT4Rec* come modello surrogato per la generazione dei dati di addestramento di *BERT4Def* rappresenta la scelta ottimale per garantire prestazioni elevate e una maggiore capacità di generalizzazione. L'architettura bidirezionale e la capacità di modellare sequenze con alta precisione permettono infatti di apprendere rappresentazioni più robuste, rendendo il modello più efficace nel discriminare tra sequenze benigne e malevole.

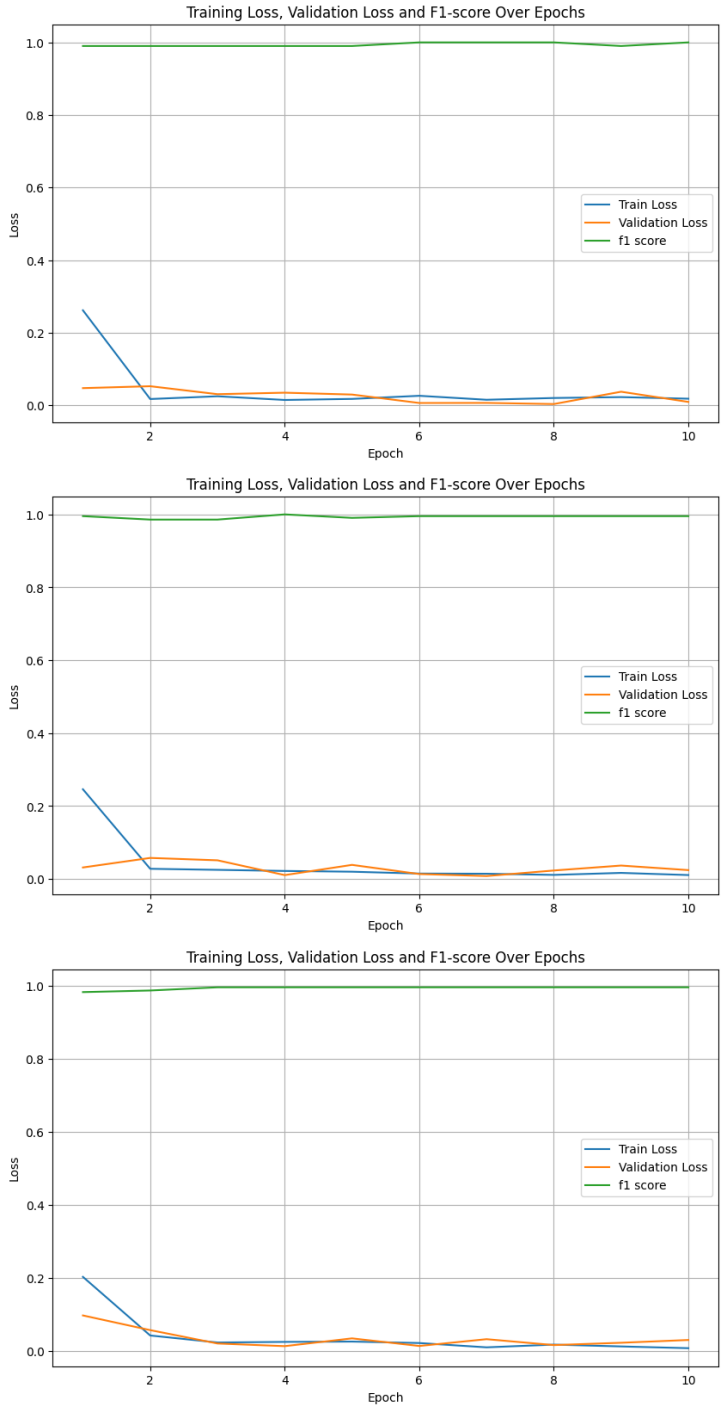

Figura 4.3: Il grafico rappresenta il processo di addestramento di BERT4Def con dati malevoli generati da una white-box, basata rispettivamente su BERT4Rec, NARM e SASR (dall'alto verso il basso)

4.5 Attaccare BERT4Def

L'attacco di *data poisoning* proposto si sviluppa a partire dall'analisi delle vulnerabilità su cui *BERT4Def* fonda la sua strategia difensiva nel contesto dei sistemi di raccomandazione sequenziali. In particolare, le sequenze malevole introdotte dagli attaccanti, oltre a conformarsi rigidamente ai pattern imposti dalla *covisitation*, tendono a essere di lunghezza massima, un comportamento poco realistico nelle interazioni naturali degli utenti. Nei sistemi di raccomandazione sequenziali, le sessioni utente possono occasionalmente presentare interazioni estese, ma un tale comportamento, se reiterato in modo sistematico, introduce anomalie facilmente identificabili.

Per aumentare l'efficacia e la furtività dell'attacco, la strategia prevede la generazione di sequenze malevole caratterizzate da una lunghezza inferiore rispetto al massimo consentito, variando tra il 40% e il 50% di tale valore. Questa riduzione introduce una maggiore somiglianza con le interazioni genuine, rendendo più difficile l'individuazione delle sequenze avversarie da parte di meccanismi di *detection*. Inoltre, si rivede il criterio di selezione degli elementi di intermezzo: invece di scegliere quelli più distanti dalle raccomandazioni generate dal modello *white-box*, si adottano elementi con una correlazione moderata rispetto ai target finali. Questo approccio contribuisce a confondere *BERT4Def*, riducendo la sua capacità di apprendere pattern discriminativi tra sequenze legittime e malevole.

L'efficacia dell'attacco si riflette in una significativa riduzione della metrica *F1-score* di *BERT4Def*, come dimostrato nelle sezioni sperimentali. Tuttavia, la diminuzione della lunghezza delle sequenze implica una ridotta enfatizzazione degli elementi target, il che rende necessaria un'iniezione di un numero maggiore di sequenze per ottenere risultati comparabili agli attacchi presentati in [59]. Questo aspetto rappresenta un potenziale svantaggio, in quanto una maggiore quantità di sequenze malevole incrementa il rischio di individuazione da parte di sistemi di difesa supplementari impiegati a supporto di *BERT4Def*.

Dal punto di vista del comportamento utente, studi sulla modellazione delle interazioni nei sistemi di raccomandazione sequenziali indicano che gli utenti tendono a interagire con un numero limitato di elementi per sessione, tipicamente distribuiti in modo non uniforme nel tempo [16, 44]. L'introduzione di sequenze di lunghezza variabile e con pattern di interazione più realistici è quindi coerente con le osservazioni empiriche sui comportamenti di navigazione. Inoltre, le tecniche di attacco che sfruttano la selezione strategica degli elementi di intermezzo si basano su principi noti di *adversarial machine learning*, secondo cui la manipolazione control-

lata degli input può alterare le decisioni di un modello mantenendo al contempo un certo grado di plausibilità nel dominio del problema.

L'approccio presentato costituisce quindi un avanzamento rispetto agli attacchi esistenti, in quanto migliora la furtività delle sequenze malevole riducendo la prevedibilità del pattern di iniezione.

4.5.1 Dataset: ML-1M

In questa sezione vengono presentati i grafici che illustrano l'andamento della *F1-score* in relazione alla lunghezza delle sequenze Figura 4.4. L'analisi dei dati mostra chiaramente come la riduzione della lunghezza delle sequenze comporti un progressivo deterioramento della *F1-score*, influenzando negativamente la capacità del modello di classificare correttamente gli input. Tale fenomeno è riscontrabile indipendentemente dall'architettura utilizzata, suggerendo che la quantità di informazioni contenute nelle sequenze sia un fattore critico per il mantenimento di prestazioni elevate.

Dai dati emerge che esiste un valore ottimale della lunghezza delle sequenze oltre il quale il sistema riesce a mantenere una capacità discriminativa accettabile. Tuttavia, poiché l'attaccante non ha conoscenza diretta del modello adottato dal difensore, si osserva che tale valore si colloca intorno a 140 elementi. Questo valore è identificato come una soglia critica, poiché oltre questo punto la *F1-score* tende a scendere al di sotto di 0.6, rendendo meno efficace la classificazione degli input avversari.

Analizzando i risultati riportati nei dataset di valutazione, si nota che i modelli *BERT4Def* addestrati con dati generati da *BERT4Rec* mantengono le prestazioni più elevate rispetto alle altre architetture. In particolare, nel caso della valutazione effettuata su dati generati da *BERT4Rec*, la *F1-score* inizia a calare in modo evidente sotto la soglia di 0.6 quando la lunghezza delle sequenze si riduce a meno di 140. Per esempio, nel modello addestrato con dati basati su *BERT4Rec* , la *F1-score* scende drasticamente a partire dalle sequenze di lunghezza inferiore a 120, con valori che arrivano fino a 0.168 e addirittura a 0.025 nelle sequenze più brevi.

Un trend simile si osserva anche nelle valutazioni condotte su modelli addestrati con dati generati da *NARM* e *SASRec*. Nel caso con dati basati su *NARM*, il calo delle prestazioni è evidente, con una *F1-score* che, per sequenze inferiori a 120, registra un rapido degrado fino a valori inferiori a 0.1 in alcuni casi. Analogamente, il modello basato su *SASR* presenta

un andamento simile, anche se con un leggero ritardo nel deterioramento delle prestazioni, suggerendo una maggiore resilienza rispetto agli altri modelli.

I dati confermano quindi che la lunghezza delle sequenze gioca un ruolo fondamentale nella robustezza del modello contro attacchi avversari. La riduzione della lunghezza delle sequenze comporta una perdita di informazioni contestuali, limitando la capacità del modello di rilevare pattern ricorrenti e di distinguere sequenze benigne da malevole. Questa osservazione è di particolare rilevanza dal punto di vista della sicurezza, poiché un attaccante potrebbe sfruttare tale vulnerabilità per ingannare il modello, somministrando sequenze di lunghezza ridotta al fine di eludere il meccanismo di classificazione. Si ricorda però, che tanto più è basso il numero di elementi all'interno della sequenza, tanto meno influiranno durante il riaddestramento

In conclusione, l'analisi dimostra che l'efficacia del modello *BERT4Def* è strettamente legata alla quantità di informazioni disponibili nelle sequenze di input. L'adozione di una soglia minima di lunghezza delle sequenze, fissata intorno a 140, potrebbe rappresentare una strategia efficace per mitigare l'impatto degli attacchi, mantenendo al contempo elevate capacità di discriminazione.

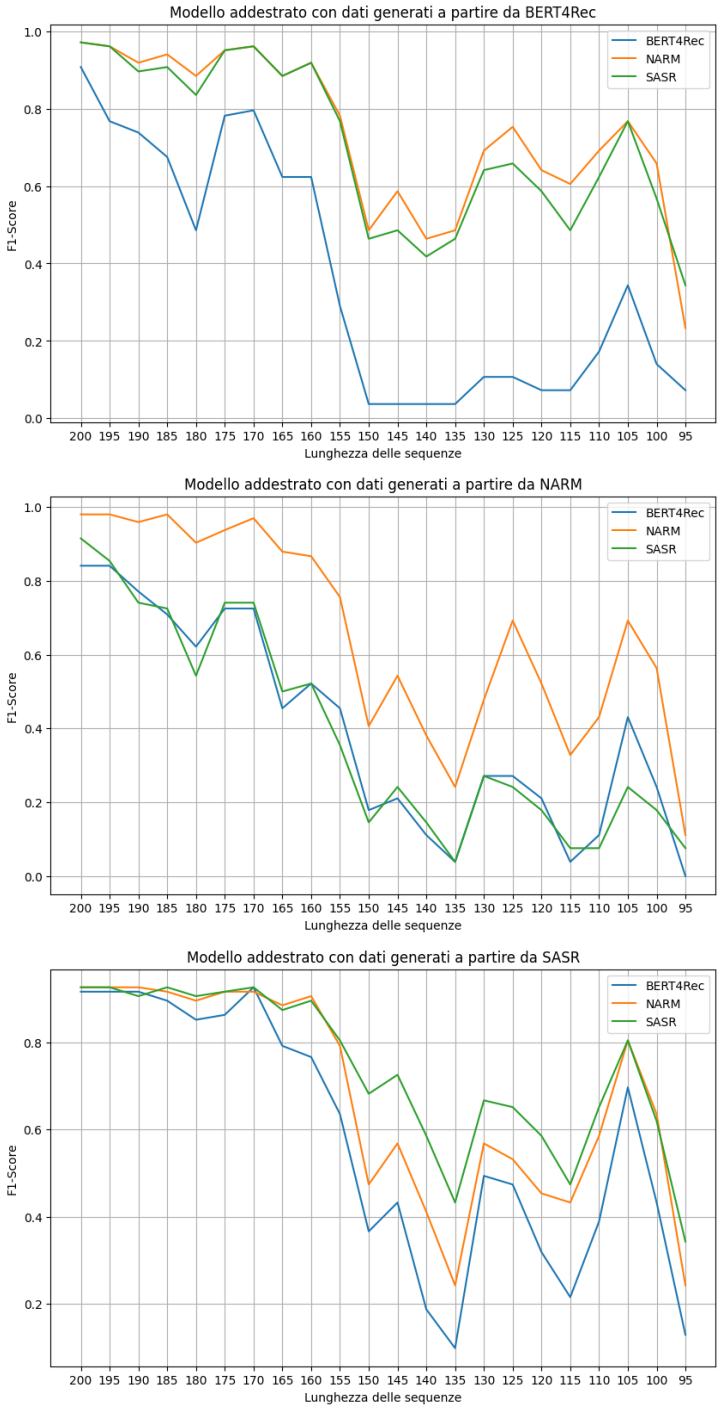

Figura 4.4: La figura illustra l'andamento della *F1-score* in funzione della riduzione della lunghezza delle sequenze, evidenziando l'impatto della loro dimensione sulle prestazioni del modello.

4.5.2 Dataset: Beauty

Gli stessi esperimenti vengono condotti sul dataset Beauty, il quale presenta un numero inferiore di sequenze rispetto al dataset precedente. Per questa ragione, viene effettuato uno studio dettagliato della *F1-score* al variare della lunghezza delle sequenze, limitando l'analisi a valori compresi tra 50 e 30 elementi. Ridurre ulteriormente la lunghezza delle sequenze non apporterebbe alcun beneficio significativo, poiché le sequenze troppo corte avrebbero un impatto trascurabile sul processo di riaddestramento del sistema di raccomandazione.

L'analisi dei risultati mostra che il valore ottimale per le lunghezze delle sequenze malevole si colloca intorno a 30 elementi, poiché consente di superare, anche se marginalmente, la difesa implementata dal sistema. Come evidenziato nella Figura 4.5, questo risultato è particolarmente significativo, in quanto conferma che gli attacchi avversari risultano più efficaci quando le sequenze malevole hanno una lunghezza sufficiente per modellare un pattern significativo senza risultare troppo ridondanti o facilmente identificabili dal sistema difensivo.

Dai dati riportati si osserva inoltre che *BERT4Def*, quando addestrato con dati generati da un surrogato basato su *BERT4Rec*, mostra una resistenza maggiore agli attacchi rispetto ai modelli basati su *NARM* e *SASRec*. Questo aspetto emerge chiaramente dai valori di *F1-score* ottenuti. Analizzando il modello *BERT4Def* addestrato con dati generati da *BERT4Rec*, si nota che, per lunghezze di sequenza di 50 e 45, le prestazioni sono significativamente elevate in tutte le valutazioni, con *F1-score* pari a 0.92 su *BERT4Rec*, 0.97 su *NARM* e 0.91 su *SASRec*. Tuttavia, all'aumentare della riduzione della lunghezza delle sequenze, le prestazioni calano progressivamente, registrando valori di *F1-score* pari a 0.32 su *BERT4Rec*, 0.57 su *NARM* e 0.37 su *SASRec* quando la lunghezza delle sequenze si riduce a 30 elementi.

Un trend simile si osserva per il modello addestrato con dati generati da *NARM*, sebbene con prestazioni inferiori. Per lunghezze di 50 e 45, il *F1-score* su *BERT4Rec* è pari a 0.81 e 0.81, rispettivamente, mentre si ottengono valori di 0.87 e 0.87 su *NARM* e 0.83 e 0.83 su *SASRec*. Con il diminuire della lunghezza delle sequenze, il degrado delle prestazioni è evidente, raggiungendo valori di *F1-score* pari a 0.32, 0.45 e 0.41 per sequenze di lunghezza 30 su *BERT4Rec*, *NARM* e *SASRec*, rispettivamente.

Il modello *BERT4Def* addestrato con dati generati da *SASRec* mostra un comportamento simile, ma con un degrado ancora più accentuato delle prestazioni man mano che la lunghezza delle sequenze si riduce. Sebbene per sequenze di 50 elementi si registrino valori elevati di *F1-score* (0.89 su *BERT4Rec*, 0.97 su NARM e 0.95 su *SASRec*), quando la lunghezza delle

sequenze viene ridotta a 30, le prestazioni calano drasticamente, raggiungendo rispettivamente 0.18, 0.48 e 0.31. Questo indica una minore capacità di generalizzazione del modello quando addestrato con dati generati da *SASRec*, suggerendo che tale approccio non sia il più efficace per la difesa del sistema.

Questi risultati forniscono ulteriori conferme del fatto che l'addestramento di *BERT4Def* con dati generati da surrogati basati su *BERT4Rec* rappresenta la strategia più efficace per la creazione di un modello di difesa resiliente. L'architettura bidirezionale di *BERT4Rec*, che cattura in modo più accurato le dipendenze a lungo termine nelle sequenze di interazioni, si traduce in una maggiore robustezza del modello difensivo. Inoltre, l'analisi dimostra che, pur esistendo una soglia ottimale per la lunghezza delle sequenze malevole, una loro eccessiva riduzione riduce significativamente la capacità dell'attacco di superare le barriere difensive del sistema, rendendo inefficace la strategia avversaria

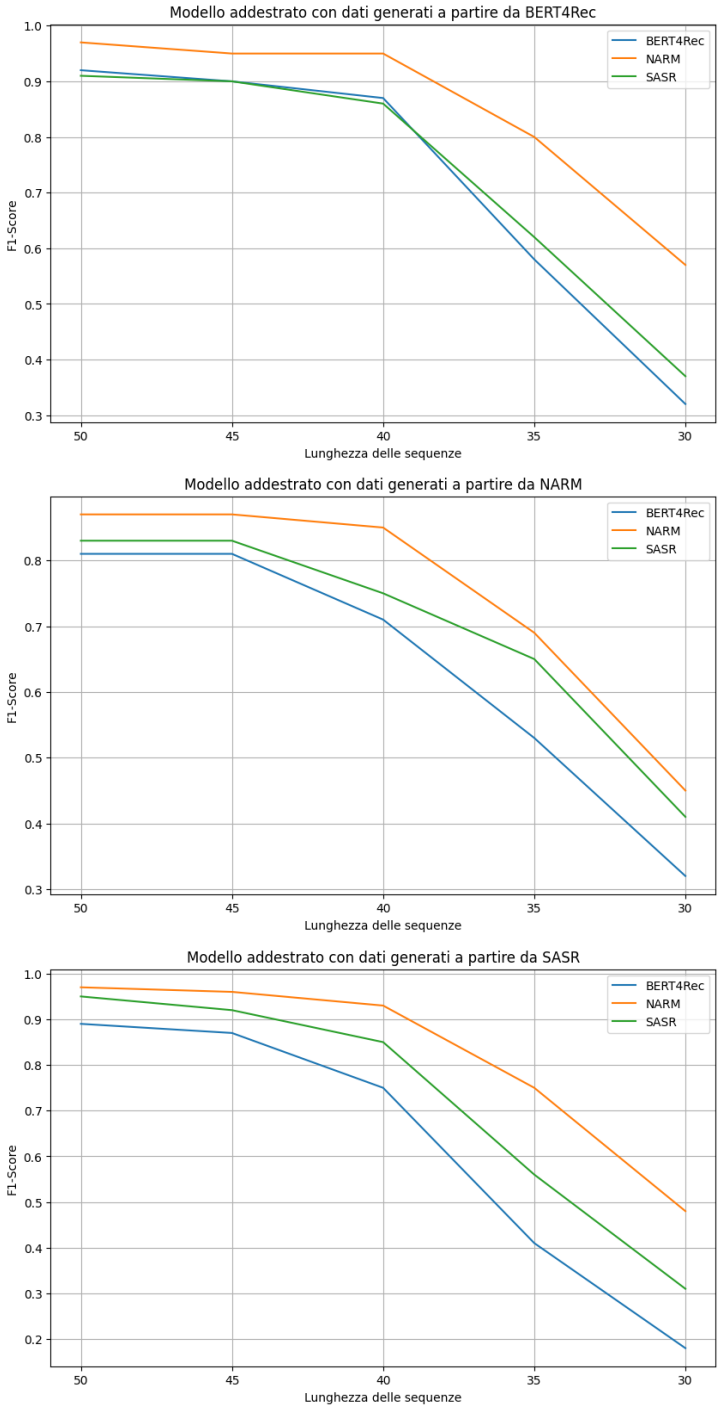

Figura 4.5: La figura illustra l'andamento della *F1-score* in funzione della riduzione della lunghezza delle sequenze, evidenziando l'impatto della loro dimensione sulle prestazioni del modello.

Capitolo 5

Conclusioni

In questa ricerca si sono esplorate le sfide e le soluzioni relative alla sicurezza dei sistemi di raccomandazione basati su *Large Language Models* (LLMs), concentrandosi in particolare sugli attacchi avversari e sulle strategie di difesa. Il principale contributo è lo sviluppo di *BERT4Def*, un meccanismo di difesa progettato per rilevare e mitigare le sequenze malevole nei sistemi di raccomandazione sequenziali. Costruito su un'architettura rivisitata di *BERT4Rec*, *BERT4Def* è in grado di distinguere efficacemente tra sequenze benigne e malevole, migliorando così la robustezza dei modelli di raccomandazione contro le manipolazioni avversarie.

Attraverso un'ampia sperimentazione, dimostra che *BERT4Def* rafforza significativamente la sicurezza dei sistemi di raccomandazione, riducendo l'impatto delle perturbazioni avversarie. Tuttavia, come per qualsiasi meccanismo di sicurezza, *BERT4Def* non è immune agli attacchi, evidenziando la continua evoluzione del confronto tra attaccanti e difensori nei sistemi di *machine learning*. L'analisi suggerisce che gli avversari possono ancora sviluppare nuove strategie d'attacco in grado di aggirare le difese esistenti, con costi computazionali sempre più elevati, sottolineando la necessità di un continuo aggiornamento e miglioramento delle tecniche difensive.

Un possibile sviluppo futuro riguarda il potenziamento della resilienza di *BERT4Def* attraverso l'addestramento avversario con dati malevoli aumentati, esponendo così il modello a una più ampia varietà di strategie d'attacco già in fase di training. Inoltre, l'integrazione di tecniche come il *contrastive learning* potrebbero migliorare le capacità di generalizzazione del classificatore, garantendo un rilevamento più efficace degli attacchi avversari in scenari diversi. Questi miglioramenti potrebbero portare a un sistema di raccomandazione più adattivo e resistente,

capace di mantenere la sua efficacia anche in ambienti ostili.

In definitiva, questo lavoro contribuisce al dibattito in corso sulla sicurezza dei sistemi di raccomandazione basati sull'intelligenza artificiale, dimostrando sia il potenziale che i limiti delle attuali strategie di difesa. Poiché i sistemi di raccomandazione fanno sempre più affidamento sugli LLMs, la continua interazione tra attacco e difesa rimarrà un'area di ricerca dinamica e cruciale, richiedendo innovazioni costanti per garantire l'integrità e l'affidabilità delle raccomandazioni personalizzate.

Elenco delle figure

Bibliografia

[1] C. C. Aggarwal. «Content-Based Recommender Systems». In: *Recommender Systems: The Textbook*. Cham: Springer International Publishing, 2016, pp. 139–166. URL: https://doi.org/10.1007/978-3-319-29659-3_4.

[2] C. C. Aggarwal. «Ensemble-Based and Hybrid Recommender Systems». In: *Recommender Systems: The Textbook*. Cham: Springer International Publishing, 2016, pp. 199–224. URL: https://doi.org/10.1007/978-3-319-29659-3_6.

[3] S. Alaparthi e M. Mishra. «BERT: a sentiment analysis odyssey». In: *Journal of Marketing Analytics* 9.2 (2021), pp. 118–126. URL: https://doi.org/10.1057/s41270-021-00109-8.

[4] H. Baniecki e P. Biecek. «Adversarial attacks and defenses in explainable artificial intelligence: A survey». In: *Information Fusion* 107 (2024), p. 102303. URL: https://www.sciencedirect.com/science/article/pii/S1566253524000812.

[5] B. Biggio e F. Roli. «Wild Patterns: Ten Years After the Rise of Adversarial Machine Learning». In: *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. CCS '18. Toronto, Canada: Association for Computing Machinery, 2018, pp. 2154–2156. URL: https://doi.org/10.1145/3243734.3264418.

[6] T. F. Boka, Z. Niu e R. B. Neupane. «A survey of sequential recommendation systems: Techniques, evaluation, and future directions». In: *Information Systems* 125 (2024), p. 102427. URL: https://www.sciencedirect.com/science/article/pii/S0306437924000851.

[7] Y.-C. Chen, R.-A. Shang e C.-Y. Kao. «The effects of information overload on consumers' subjective state towards buying decision in the internet shopping environment». In:

Electronic Commerce Research and Applications 8 (1 2009), pp. 48–58. URL: https://www.sciencedirect.com/science/article/pii/S1567422308000367.

[8] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk e Y. Bengio. «Learning phrase representations using RNN encoder-decoder for statistical machine translation». In: *arXiv preprint arXiv:1406.1078* (2014).

[9] Y. H. Cho e J. K. Kim. «Application of Web usage mining and product taxonomy to collaborative recommendations in e-commerce». In: *Expert systems with Applications* 26.2 (2004), pp. 233–246.

[10] A. E. Cinà, K. Grosse, A. Demontis, S. Vascon, W. Zellinger, B. A. Moser, A. Oprea, B. Biggio, M. Pelillo e F. Roli. «Wild Patterns Reloaded: A Survey of Machine Learning Security against Training Data Poisoning». In: *ACM Comput. Surv.* 55.13s (lug. 2023). URL: https://doi.org/10.1145/3585385.

[11] J. Devlin, M.-W. Chang, K. Lee e K. Toutanova. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. 2019. arXiv: 1810.04805 [cs.CL]. URL: https://arxiv.org/abs/1810.04805.

[12] T. Donkers, B. Loepp e J. Ziegler. «Sequential user-based recurrent neural network recommendations». In: *Proceedings of the eleventh ACM conference on recommender systems*. 2017, pp. 152–160.

[13] A. Edmunds e A. Morris. «The problem of information overload in business organisations: a review of the literature». In: *International Journal of Information Management* 20 (1 2000), pp. 17–28. URL: https://www.sciencedirect.com/science/article/pii/S0268401299000511.

[14] J. Fan, Q. Yan, M. Li, G. Qu e Y. Xiao. «A Survey on Data Poisoning Attacks and Defenses». In: *2022 7th IEEE International Conference on Data Science in Cyberspace (DSC)*. 2022, pp. 48–55.

[15] H. Fang, Y. Qiu, H. Yu, W. Yu, J. Kong, B. Chong, B. Chen, X. Wang, S.-T. Xia e K. Xu. *Privacy Leakage on DNNs: A Survey of Model Inversion Attacks and Defenses*. 2024. arXiv: 2402.04013 [cs.CV]. URL: https://arxiv.org/abs/2402.04013.

[16] M. Filipovic, B. Mitrevski, D. Antognini, E. L. Glaude, B. Faltings e C. Musat. *Modeling Online Behavior in Recommender Systems: The Importance of Temporal Context*. 2021. arXiv: 2009.08978 [cs.IR]. URL: https://arxiv.org/abs/2009.08978.

[17] P. S. Ghatora, S. E. Hosseini, S. Pervez, M. J. Iqbal e N. Shaukat. «Sentiment Analysis of Product Reviews Using Machine Learning and Pre-Trained LLM». In: *Big Data and Cognitive Computing* 8.12 (2024). URL: https://www.mdpi.com/2504-2289/8/12/199.

[18] S. Guo, C. Xie, J. Li, L. Lyu e T. Zhang. *Threats to Pre-trained Language Models: Survey and Taxonomy*. 2022. arXiv: 2202.06862 [cs.CR]. URL: https://arxiv.org/abs/2202.06862.

[19] X. Han, Z. Zhang, N. Ding, Y. Gu, X. Liu, Y. Huo, J. Qiu, Y. Yao, A. Zhang, L. Zhang et al. «AI Open». In: ().

[20] F. M. Harper e J. A. Konstan. «The MovieLens Datasets: History and Context». In: *ACM Trans. Interact. Intell. Syst.* 5.4 (dic. 2015). URL: https://doi.org/10.1145/2827872.

[21] B. Hidasi. «Session-based Recommendations with Recurrent Neural Networks». In: *arXiv preprint arXiv:1511.06939* (2015).

[22] B. Hidasi e A. Karatzoglou. «Recurrent neural networks with top-k gains for session-based recommendations». In: *Proceedings of the 27th ACM international conference on information and knowledge management*. 2018, pp. 843–852.

[23] Y. Himeur, S. S. Sohail, F. Bensaali, A. Amira e M. Alazab. «Latest trends of security and privacy in recommender systems: A comprehensive review and future perspectives». In: *Computers Security* 118 (2022), p. 102746. URL: https://www.sciencedirect.com/science/article/pii/S0167404822001419.

[24] S. Hochreiter. «Long Short-term Memory». In: *Neural Computation MIT-Press* (1997).

[25] Y. Hou, J. Zhang, Z. Lin, H. Lu, R. Xie, J. McAuley e W. X. Zhao. *Large Language Models are Zero-Shot Rankers for Recommender Systems*. 2024. arXiv: 2305.08845 [cs.IR]. URL: https://arxiv.org/abs/2305.08845.

[26] H. Hu, Z. Salcic, L. Sun, G. Dobbie, P. S. Yu e X. Zhang. «Membership Inference Attacks on Machine Learning: A Survey». In: *ACM Comput. Surv.* 54.11s (set. 2022). URL: https://doi.org/10.1145/3523273.

[27] D. Jin, Z. Jin, J. T. Zhou e P. Szolovits. *Is BERT Really Robust? A Strong Baseline for Natural Language Attack on Text Classification and Entailment*. 2020. arXiv: 1907.11932 [cs.CL]. URL: https://arxiv.org/abs/1907.11932.

[28] W.-C. Kang e J. McAuley. «Self-Attentive Sequential Recommendation». In: *2018 IEEE International Conference on Data Mining (ICDM)*. 2018, pp. 197–206.

[29] H. Ko, S. Lee, Y. Park e A. Choi. «A Survey of Recommendation Systems: Recommendation Models, Techniques, and Application Fields». In: *Electronics* 11 (1 2022). URL: https://www.mdpi.com/2079-9292/11/1/141.

[30] D. Lewis et al. «Dying for information». In: *London: Reuters Business Information* 2 (1996).

[31] J. Li, P. Ren, Z. Chen, Z. Ren, T. Lian e J. Ma. «Neural Attentive Session-based Recommendation». In: *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. CIKM '17. Singapore, Singapore: Association for Computing Machinery, 2017, pp. 1419–1428. URL: https://doi.org/10.1145/3132847.3132926.

[32] J. Li, P. Ren, Z. Chen, Z. Ren, T. Lian e J. Ma. «Neural attentive session-based recommendation». In: *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. 2017, pp. 1419–1428.

[33] J. Lin, J. Zou e N. Ding. *Using Adversarial Attacks to Reveal the Statistical Bias in Machine Reading Comprehension Models*. 2021. arXiv: 2105.11136 [cs.CL]. URL: https://arxiv.org/abs/2105.11136.

[34] B. Lindemann, T. Müller, H. Vietz, N. Jazdi e M. Weyrich. «A survey on long short-term memory networks for time series prediction». In: *Procedia CIRP* 99 (2021). 14th CIRP Conference on Intelligent Computation in Manufacturing Engineering, 15-17 July 2020, pp. 650–655. URL: https://www.sciencedirect.com/science/article/pii/S2212827121003796.

[35] A. Liu, Z. Huang, H. Lu, X. Wang e C. Yuan. «BB-KBQA: BERT-Based Knowledge Base Question Answering». In: *Chinese Computational Linguistics*. A cura di M. Sun, X. Huang, H. Ji, Z. Liu e Y. Liu. Cham: Springer International Publishing, 2019, pp. 81–92.

[36] N. Liu, Q. Hu, H. Xu, X. Xu e M. Chen. «Med-BERT: A Pretraining Framework for Medical Records Named Entity Recognition». In: *IEEE Transactions on Industrial Informatics* 18.8 (2022), pp. 5600–5608.

[37] P. Lops, M. De Gemmis e G. Semeraro. «Content-based recommender systems: State of the art and trends». In: *Recommender systems handbook* (2011), pp. 73–105.

[38] D. Lowd e C. Meek. «Adversarial learning». In: *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*. 2005, pp. 641–647.

[39] R. Mehta e K. Rana. «A review on matrix factorization techniques in recommender systems». In: *2017 2nd International Conference on Communication Systems, Computing and IT Applications (CSCITA)*. 2017, pp. 269–274.

[40] S. Minaee, T. Mikolov, N. Nikzad, M. Chenaghlu, R. Socher, X. Amatriain e J. Gao. *Large Language Models: A Survey*. 2024. arXiv: 2402.06196 [cs.CL]. URL: https://arxiv.org/abs/2402.06196.

[41] B. Mohit. «Named Entity Recognition». In: *Natural Language Processing of Semitic Languages*. A cura di I. Zitouni. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 221–245. URL: https://doi.org/10.1007/978-3-642-45358-8_7.

[42] T. T. Nguyen, N. Quoc Viet hung, T. T. Nguyen, T. T. Huynh, T. T. Nguyen, M. Weidlich e H. Yin. «Manipulating Recommender Systems: A Survey of Poisoning Attacks and Countermeasures». In: *ACM Comput. Surv.* 57.1 (ott. 2024). URL: https://doi.org/10.1145/3677328.

[43] Z. Qian, K. Huang, Q.-F. Wang e X.-Y. Zhang. «A survey of robust adversarial training in pattern recognition: Fundamental, theory, and methodologies». In: *Pattern Recognition* 131 (2022), p. 108889.

[44] M. Quadrana, P. Cremonesi e D. Jannach. *Sequence-Aware Recommender Systems*. 2018. arXiv: 1802.08452 [cs.IR]. URL: https://arxiv.org/abs/1802.08452.

[45] S. Rendle, W. Krichene, L. Zhang e Y. Koren. «Revisiting the performance of ials on item recommendation benchmarks». In: *Proceedings of the 16th ACM Conference on Recommender Systems*. 2022, pp. 427–435.

[46] F. Ricci, L. Rokach e B. Shapira. «Recommender Systems: Techniques, Applications, and Challenges». In: *Recommender Systems Handbook*. A cura di F. Ricci, L. Rokach e B. Shapira. New York, NY: Springer US, 2022, pp. 1–35. URL: https://doi.org/10.1007/978-1-0716-2197-4_1.

[47] H. Salehinejad, S. Sankar, J. Barfett, E. Colak e S. Valaee. *Recent Advances in Recurrent Neural Networks*. 2018. arXiv: 1801.01078 [cs.NE]. URL: https://arxiv.org/abs/1801.01078.

[48] J. B. Schafer, D. Frankowski, J. Herlocker e S. Sen. «Collaborative Filtering Recommender Systems». In: *The Adaptive Web: Methods and Strategies of Web Personalization*. A cura di P. Brusilovsky, A. Kobsa e W. Nejdl. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 291–324. URL: https://doi.org/10.1007/978-3-540-72079-9_9.

[49] S. Senecal e J. Nantel. «The influence of online product recommendations on consumers' online choices». In: *Journal of Retailing* 80 (2 gen. 2004), pp. 159–169.

[50] D. Sileo, W. Vossen e R. Raymaekers. «Zero-Shot Recommendation as Language Modeling». In: *Advances in Information Retrieval*. A cura di M. Hagen, S. Verberne, C. Macdonald, C. Seifert, K. Balog, K. Nørvåg e V. Setty. Cham: Springer International Publishing, 2022, pp. 223–230.

[51] J. Song, Z. Li, Z. Hu, Y. Wu, Z. Li, J. Li e J. Gao. «PoisonRec: An Adaptive Data Poisoning Framework for Attacking Black-box Recommender Systems». In: *2020 IEEE 36th International Conference on Data Engineering (ICDE)*. 2020, pp. 157–168.

[52] F. Sun, J. Liu, J. Wu, C. Pei, X. Lin, W. Ou e P. Jiang. «BERT4Rec: Sequential Recommendation with Bidirectional Encoder Representations from Transformer». In: *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. CIKM '19. Beijing, China: Association for Computing Machinery, 2019, pp. 1441–1450. URL: https://doi.org/10.1145/3357384.3357895.

[53] F. Tramèr, F. Zhang, A. Juels, M. K. Reiter e T. Ristenpart. «Stealing machine learning models via prediction {APIs}». In: *25th USENIX security symposium (USENIX Security 16)*. 2016, pp. 601–618.

[54] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser e I. Polosukhin. *Attention Is All You Need*. 2023. arXiv: 1706.03762 [cs.CL]. URL: https://arxiv.org/abs/1706.03762.

[55] M. Wankhade, A. C. S. Rao e C. Kulkarni. «A survey on sentiment analysis methods, applications, and challenges». In: *Artificial Intelligence Review* 55.7 (2022), pp. 5731–5780. URL: https://doi.org/10.1007/s10462-022-10144-1.

[56] C. Wu, F. Wu, Y. Yu, T. Qi, Y. Huang e X. Xie. *UserBERT: Contrastive User Model Pre-training*. 2021. arXiv: 2109.01274 [cs.IR]. URL: https://arxiv.org/abs/2109.01274.

[57] L. Wu, Z. Zheng, Z. Qiu, H. Wang, H. Gu, T. Shen, C. Qin, C. Zhu, H. Zhu, Q. Liu, H. Xiong e E. Chen. «A survey on large language models for recommendation». In: *World Wide Web* 27.5 (ago. 2024), p. 60. URL: https://doi.org/10.1007/s11280-024-01291-2.

[58] S. Wu, F. Sun, W. Zhang, X. Xie e B. Cui. «Graph Neural Networks in Recommender Systems: A Survey». In: *ACM Comput. Surv.* 55.5 (dic. 2022). URL: https://doi.org/10.1145/3535101.

[59] Z. Yue, Z. He, H. Zeng e J. McAuley. «Black-Box Attacks on Sequential Recommenders via Data-Free Model Extraction». In: *Proceedings of the 15th ACM Conference on Recommender Systems*. RecSys '21. Amsterdam, Netherlands: Association for Computing Machinery, 2021, pp. 44–54. URL: https://doi.org/10.1145/3460231.3474275.

[60] W. X. Zhao, K. Zhou, J. Li, T. Tang, X. Wang, Y. Hou, Y. Min, B. Zhang, J. Zhang, Z. Dong, Y. Du, C. Yang, Y. Chen, Z. Chen, J. Jiang, R. Ren, Y. Li, X. Tang, Z. Liu, P. Liu, J.-Y. Nie e J.-R. Wen. *A Survey of Large Language Models*. 2024. arXiv: 2303.18223 [cs.CL]. URL: https://arxiv.org/abs/2303.18223.