Università degli Studi di Salerno
Dipartimento di Informatica

Corso di Laurea Magistrale in Informatica

Mappatura Automatica dei Log Honeypot su MITRE CAPEC mediante un Approccio Ibrido di Information Retrieval guidato da Interpretazione LLM Upstream

Tesi in Sicurezza Informatica

Relatore
Prof. Arcangelo Castiglione

Candidato
Daniele Gregori
Matricola: 0522501595

Anno Accademico 2024/2025

Abstract

L'analisi automatizzata delle minacce informatiche rappresenta una sfida centrale per i *Security Operations Center* (SOC), chiamati a gestire volumi crescenti e dati di monitoraggio sempre più eterogenei. In particolare, l'interpretazione dei log generati da architetture *multi-honeypot* richiede di colmare il *gap semantico* che separa l'evento tecnico grezzo dalla conoscenza strutturata fornita da standard quali MITRE CAPEC.

Le metodologie basate su firme statiche risultano inadeguate rispetto all'evoluzione delle tecniche di attacco, mentre l'impiego diretto di *large language model* (LLM) come classificatori *end-to-end* introduce problematiche rilevanti in termini di affidabilità, costi computazionali e trasparenza del processo decisionale. La presente tesi propone un'architettura ibrida e modulare che affronta tali limitazioni attraverso un uso controllato e interpretabile dell'intelligenza artificiale.

L'approccio sviluppato assegna a un modello *open weight* locale (`Mistral-7B`) la funzione di **interprete semantico *upstream***, guidato da una strategia di *prompt engineering* adattivo che incorpora vincoli decisionali strutturati (*threat hierarchy*) e un processo di ragionamento esplicito (*chain-of-thought*). Il modello produce descrizioni formali degli intenti dell'attaccante a partire da log aggregati e non uniformi.

Tali rappresentazioni alimentano un **motore di *matching* ibrido** basato su due indici complementari della *knowledge base* CAPEC: un indice vettoriale (alimentato da `ATT&CK-BERT`) per la similarità semantica e un indice lessicale (`TF-IDF`) per la rilevanza terminologica. La combinazione dei *ranking* mediante ***Reciprocal Rank Fusion*** (RRF) consente di mitigare le carenze dei singoli approcci e di consolidare il risultato finale.

La validazione sperimentale, condotta su un dataset eterogeneo di sessioni reali (`Cowrie`, `Honeytrap`, `Dionaea`, `SentryPeer`, `CiscoASA`), evidenzia prestazioni superiori rispetto a una *baseline state-of-the-art* (`Gemini 2.5 Pro`), con una `Precision@1` del **76.0%** (contro il 72.0% della *baseline*) e un `Mean Reciprocal Rank` (MRR) pari a **0.973**. L'analisi di ablazione conferma il contributo determinante dell'integrazione ibrida, evidenziando un miglioramento significativo rispetto ai singoli componenti.

In conclusione, il lavoro dimostra che la separazione funzionale tra interpretazione e recupero consente un'identificazione più accurata dell'*apex threat* e offre vantaggi concreti in termini di trasparenza, sovranità dei dati e sostenibilità delle risorse computazionali.

Indice

Elenco delle figure

Elenco delle tabelle

Elenco dei listati

Capitolo 1

Introduzione

L'evoluzione del panorama delle minacce informatiche ha raggiunto livelli di complessità e scala senza precedenti. La progressiva automazione degli attacchi e la disponibilità di strumenti offensivi avanzati hanno reso la difesa delle infrastrutture digitali una sfida asimmetrica, in cui i difensori devono analizzare volumi crescenti di dati per identificare segnali di compromissione sempre più sfuggenti. In questo contesto, la *Cyber Threat Intelligence* (CTI) assume un ruolo centrale: comprendere non solo "cosa" sta accadendo, ma anche "come" e "perché", attraverso la mappatura delle azioni degli avversari su tassonomie standardizzate, è fondamentale per passare da una difesa reattiva a una proattiva.

1.1 Contesto e motivazione

Gli honeypot, sistemi esca progettati per attrarre e monitorare gli attaccanti, rappresentano una delle fonti più preziose di intelligence. A differenza dei sistemi di produzione, dove distinguere il traffico malevolo da quello legittimo è complesso, in un honeypot ogni interazione è, per definizione, sospetta. Piattaforme moderne come `T-Pot` orchestrano ecosistemi di honeypot eterogenei, catturando una vasta gamma di dati: dalle sessioni shell interattive alle scansioni di rete, fino ai tentativi di sfruttamento di vulnerabilità specifiche.

Tuttavia, questa ricchezza informativa porta con sé un problema critico: il cosiddetto *data deluge*. Il volume di log generati, la loro eterogeneità (formati diversi, livelli di astrazione diversi) e la presenza di rumore di fondo rendono l'analisi manuale umanamente insostenibile. I Security Operations Center sono spesso sopraffatti, incapaci di trasformare tempestivamente i dati grezzi in conoscenza strutturata.

Le tecniche tradizionali di analisi, basate su firme statiche o espressioni regolari (Regex), si dimostrano sempre più inefficaci contro attacchi che mutano rapidamente o utilizzano l'offuscamento. D'altro canto, l'avvento dei *large language model* (LLM) ha aperto nuove prospettive per l'interpretazione semantica dei dati, ma il loro utilizzo diretto come "oracoli"

introduce rischi di allucinazioni, costi computazionali elevati e difficoltà nell'aggiornamento della conoscenza.

1.2 Definizione del problema

Il problema centrale affrontato in questa tesi è il *gap semantico* tra il log grezzo e l'intelligence strutturata. Da un lato vi sono eventi tecnici di basso livello (es. una sequenza di comandi shell, un payload esadecimale); dall'altro esistono standard di alto livello come il MITRE CAPEC (*Common Attack Pattern Enumeration and Classification*), che descrive i meccanismi logici dell'attacco (es. "Command Injection", "SQL Injection").

Mappare automaticamente i primi sui secondi è complesso per tre motivi principali:

1. **Ambiguità:** Lo stesso comando può sottintendere intenti diversi a seconda del contesto.
2. **Specificità vs. astrazione:** I log contengono dettagli tecnici effimeri (IP, hash, stringhe casuali), mentre i pattern CAPEC sono descrizioni astratte.
3. **Limiti degli strumenti attuali:** La ricerca per keyword fallisce sui sinonimi, mentre la ricerca semantica pura (*embedding*) rischia di perdere precisione sui termini tecnici specifici.

1.3 Obiettivi della ricerca

L'obiettivo di questo lavoro è progettare, implementare e validare una pipeline automatizzata per l'analisi dei log di honeypot e la loro mappatura sullo standard MITRE CAPEC. La soluzione mira a superare i limiti degli approcci esistenti attraverso:

- **Automazione intelligente:** Utilizzare l'intelligenza artificiale non per sostituire l'analista, ma per pre-elaborare e interpretare su vasta scala i dati.
- **Robustezza e interpretabilità:** Garantire che il sistema sia resistente al rumore e fornisca spiegazioni chiare per le sue classificazioni.
- **Efficienza operativa:** Progettare un'architettura sostenibile, che non dipenda esclusivamente da costose API cloud e che possa operare in ambienti con risorse limitate.

1.4 Contributi innovativi

Per raggiungere tali obiettivi, questa tesi propone una soluzione architetturale innovativa che integra LLM e information retrieval in un approccio ibrido. I principali contributi originali sono:

1. **Pre-elaborazione olistica *actor-centric*:** A differenza dei sistemi che analizzano log atomici, è stato sviluppato un modulo che aggrega gli eventi per attore, ricostruendo la sequenza temporale e il contesto completo dell'attacco prima dell'analisi.

2. **L'LLM come interprete *upstream*:** Si propone un cambio di paradigma nell'uso degli LLM. Il modello (basato su `Mistral-7B`) non è utilizzato per classificare direttamente (evitando così le allucinazioni), bensì per "tradurre" il log tecnico in una descrizione semantica strutturata. Questo processo è guidato da un *prompt engineering* adattivo che inietta dinamicamente regole di contesto e gerarchie di minacce.

3. **Motore di matching ibrido con Rank Fusion:** È stato implementato un sistema di recupero a doppio indice che combina la ricerca vettoriale (basata su `ATT&CK-BERT`) per la similarità semantica e la ricerca lessicale basata su Term Frequency-Inverse Document Frequency (TF-IDF) per la precisione terminologica. I risultati vengono fusi tramite l'algoritmo *Reciprocal Rank Fusion*, una tecnica che si è dimostrata superiore ai singoli componenti.

4. **Valutazione rigorosa e confronto con lo stato dell'arte (SOTA):** La soluzione è stata validata su un dataset reale eterogeneo e confrontata con una baseline allo stato dell'arte (`Gemini 2.5 Pro`). I risultati dimostrano che l'architettura proposta supera il modello SOTA nelle metriche di prioritizzazione (*strict metrics*), offrendo al contempo maggiore trasparenza e minori costi operativi.

1.5 Struttura della tesi

Il resto dell'elaborato è organizzato come segue:

- Il **Capitolo 2 (Background)** fornisce i fondamenti teorici necessari, analizzando i sistemi honeypot, gli LLM e lo standard MITRE CAPEC, oltre alle basi dell'information retrieval.
- Il **Capitolo 3 (Stato dell'arte)** esamina la letteratura scientifica esistente, evidenziando i limiti degli approcci attuali e posizionando la presente ricerca rispetto ai lavori correlati.
- Il **Capitolo 4 (Soluzione proposta)** descrive l'architettura concettuale del sistema, dettagliando il flusso dei dati, la logica di aggregazione olistica e il design del motore di matching ibrido.
- Il **Capitolo 5 (Implementazione della soluzione proposta)** illustra le scelte tecnologiche, lo stack software utilizzato e i dettagli ingegneristici.
- Il **Capitolo 6 (Valutazione sperimentale della soluzione proposta)** presenta i risultati dei test condotti, inclusa l'analisi di ablazione e il confronto con la baseline SOTA, corredati da un'analisi qualitativa di casi studio reali.

- Il **Capitolo 7 (Conclusioni e sviluppi futuri)** sintetizza i risultati conseguiti, discute i limiti dello studio e propone direzioni per sviluppi futuri.

Capitolo 2

Background

Questo capitolo fornisce i fondamenti tecnologici e concettuali necessari per comprendere il contesto, la metodologia e il contributo innovativo di questo lavoro di tesi. La discussione è organizzata in quattro aree tematiche principali, presentate in una sequenza logica che procede dal problema alla soluzione.

La trattazione si aprirà con una disamina del *contesto del problema*, analizzando i sistemi honeypot e la piattaforma `T-Pot`, che costituiscono la fonte dei dati grezzi e definiscono le sfide analitiche da affrontare. Successivamente, verranno introdotti i *large language model* (LLM), la tecnologia chiave impiegata per l'interpretazione semantica dei dati, illustrandone i fondamenti architetturali, il processo di addestramento e le capacità emergenti. Verrà poi presentato lo standard di riferimento, MITRE CAPEC, che rappresenta la *knowledge base* di destinazione per la classificazione e la mappatura degli attacchi osservati. Infine, verranno analizzate diverse tecniche di *information retrieval* (IR), le quali forniscono gli strumenti metodologici per implementare il *matching* tra l'interpretazione dei log e i pattern di attacco.

Complessivamente, questo capitolo mira a dotare il lettore di tutte le nozioni preliminari indispensabili per comprendere appieno la soluzione proposta nei capitoli successivi.

2.1 Contesto del problema: analisi dei log da sistemi honeypot

L'evoluzione delle minacce informatiche ha spinto la ricerca a esplorare approcci difensivi proattivi. Tra questi, gli honeypot sono emersi come strumenti strategici di *deception technology*, progettati non solo per difendere, ma anche per attrarre e studiare gli avversari [1], [2]. Un honeypot è una risorsa IT (un sistema, un servizio, una rete) creata appositamente come esca, la cui principale caratteristica è l'assenza di traffico legittimo. Di conseguenza, qualsiasi interazione con esso è, per definizione, un indicatore di attività sospetta o malevola, eliminando il problema del "falso positivo" che affligge molti sistemi di sicurezza tradizionali [3], [4].

2.1.1 Principi strategici e classificazione degli honeypot

Il valore strategico di un honeypot si manifesta attraverso tre funzioni principali: la deviazione degli attaccanti dagli asset critici, il rilevamento precoce di attività ostili grazie ad avvisi ad alta fedeltà, e, in particolare, la raccolta di *threat intelligence* [3]. Quest'ultima funzione è cruciale, poiché l'osservazione del comportamento degli avversari in un ambiente controllato consente l'acquisizione di una conoscenza approfondita delle loro tattiche, tecniche e procedure (TTP), inclusi strumenti, comandi e campioni di malware [4], [5].

Gli honeypot sono comunemente classificati in base al livello di interazione che consentono, un fattore che influenza direttamente la ricchezza dei dati raccolti e il livello di rischio associato [1], [3]:

- **Honeypot a bassa interazione (*low-interaction*):** Emulano solo i servizi essenziali per registrare attività automatizzate come scansioni di rete e tentativi di connessione. Sono semplici da gestire e a basso rischio, ma forniscono dati superficiali e sono facilmente identificabili da avversari esperti [2].
- **Honeypot ad alta interazione (*high-interaction*):** Offrono un ambiente completo, come un sistema operativo reale, permettendo di catturare TTP complesse e malware. La profondità dei dati raccolti è controbilanciata da un'elevata complessità di gestione e da significativi rischi di sicurezza, che richiedono sofisticate misure di isolamento [2].
- **Honeypot a media interazione (*medium-interaction*):** Rappresentano un compromesso ottimale, emulando funzionalità di servizio più complesse (es. una shell) senza esporre un intero sistema operativo. Un esempio emblematico è `Cowrie`, un honeypot SSH/Telnet che registra in dettaglio i comandi impartiti, offrendo un eccellente equilibrio tra qualità dei dati e gestibilità [6], [7], [8].

La Tabella 2.1 riassume le caratteristiche distintive di queste tre categorie.

Tabella 2.1: Classificazione degli honeypot per livello di interazione

Livello	**Descrizione**	**Vantaggi**	**Svantaggi**
Bassa Interazione	Simula solo i servizi essenziali (es. porte aperte, banner).	Semplicità, basso rischio, efficienza.	Dati limitati, facili da rilevare per esperti.
Media Interazione	Emula più funzionalità di un servizio (es. un ambiente shell).	Equilibrio tra facilità e ricchezza di dati.	Più complessi di quelli a bassa interazione, rischio intermedio.
Alta Interazione	Fornisce un sistema operativo o un'applicazione completa.	Raccolta di dati dettagliati, TTP complesse, malware.	Complessità di gestione, rischio elevato se non isolati.

2.1.2 T-Pot: un ecosistema multi-honeypot per la raccolta dati

Per superare i limiti di un singolo honeypot, piattaforme integrate come `T-Pot` orchestrano un intero ecosistema di sensori. `T-Pot` è una piattaforma *open source*, basata su Docker, che dispiega e gestisce simultaneamente una vasta gamma di honeypot (30 nella versione standard), fornendo una visione olistica delle minacce attraverso molteplici vettori di attacco [9]. La sua architettura integra honeypot specializzati, come il già citato `Cowrie`, `Honeytrap` (per le scansioni di rete [10]) e `Dionaea` (per la cattura di malware su servizi come SMB/FTP [11]), affiancati da sensori specifici come `SentryPeer` (per il traffico VoIP) e `CiscoASA` (per la simulazione di dispositivi web), oltre a strumenti di *network security monitoring* (NSM) come `Suricata` [12]. Tutti i dati generati da questi componenti eterogenei vengono convogliati, normalizzati e archiviati tramite l'`Elastic Stack` (ELK), rendendo `T-Pot` una potente, ma complessa, fonte di dati per la *threat intelligence*. La Tabella 2.2 illustra alcuni esempi significativi dei sensori inclusi.

Tabella 2.2: Esempi di honeypot integrati in `T-Pot` e log generati

Honeypot	**Interazione**	**Vettore Attacco**	**Log Generati (Esempi)**
`Cowrie`	Media	SSH, Telnet	JSON (comandi eseguiti, credenziali, file scaricati)
`Honeytrap`	Bassa	Scansioni porte TCP/UDP	Testo, JSON (indirizzo IP origine, porta destinazione)
`Dionaea`	Bassa/Media	Servizi vulnerabili (SMB, FTP)	Log di exploit, cattura di file binari di malware
`SentryPeer`	Bassa	VoIP (SIP)	Metodi SIP (REGISTER, INVITE), User-Agent malevoli
`CiscoASA`	Bassa	Web (Simulazione Cisco)	Richieste HTTP verso *path* vulnerabili (es. /+CSCOE+/)

L'output aggregato di questi sensori viene centralizzato e visualizzato attraverso dashboard interattive basate su `Elastic Stack`, come quella riportata in Figura 2.1. Questa interfaccia offre una panoramica immediata dello stato di sicurezza, evidenziando metriche quantitative impressionanti, come il volume totale degli attacchi (nell'esempio, oltre 466.000 eventi in una sola settimana) e la loro distribuzione tra i vari sensori.

Tuttavia, l'immagine mette in luce anche l'estrema eterogeneità delle fonti: i dati fluiscono simultaneamente da sensori con logiche diverse (`Honeytrap`, `Cowrie`, `CiscoASA`), creando un flusso informativo complesso. Sebbene utile per il monitoraggio di alto livello, questa rappresentazione visiva illustra concretamente il problema del *data deluge*: la mole di dati è tale che per un analista umano diviene impossibile scendere nel dettaglio dei singoli eventi per estrarne un significato semantico profondo senza l'ausilio di strumenti di automazione avanzata.

Figura 2.1: Panoramica della dashboard operativa di `T-Pot` basata su `Elastic Stack`

2.1.3 Sfide analitiche nell'interpretazione dei log honeypot

L'efficacia di un sistema come `T-Pot` dipende dalla capacità di interpretare i dati raccolti, un processo ostacolato da sfide intrinseche che rendono l'analisi manuale insostenibile [13]. Questa fase rappresenta una delle maggiori sfide aperte nel campo: meta-analisi della letteratura scientifica hanno evidenziato come l'analisi e la visualizzazione dei dati degli honeypot siano aree di ricerca con contributi limitati, identificandole come un'opportunità critica per future investigazioni [14].

- **Volume e velocità:** Un honeypot esposto su Internet genera un volume enorme di log, spesso decine di migliaia di eventi al giorno, a causa di scansioni e attacchi automatizzati costanti. Questo "diluvio" di dati sovraccarica gli analisti e crea sfide di archiviazione ed elaborazione [15], [16].

- **Eterogeneità e complessità:** La forza di `T-Pot`, ovvero la sua natura multi-honeypot, è anche la sua principale debolezza analitica. I log provengono da fonti diverse, con formati (JSON, testo libero), livelli di dettaglio e semantiche differenti, richiedendo complesse pipeline di normalizzazione.

- **Distinzione tra "segnale" e "rumore":** Sebbene tutto il traffico sia malevolo, la maggior parte è "rumore" a basso impatto (scansioni automatiche). Il vero "segnale" – interazioni manuali complesse o TTP innovative – è spesso nascosto in questo rumore, portando al fenomeno dell'*alarm fatigue* tra gli analisti. Studi specifici sui log di `Cowrie` confermano che l'analisi manuale è impraticabile a causa dell'enorme volume di dati, dominato da attacchi automatizzati e a bassa complessità che costituiscono un "rumore" di fondo costante [15]. Per superare questa sfida, la ricerca si è concentrata sullo sviluppo di modelli per identificare pattern specifici, come l'analisi probabilistica delle sequenze di comandi per prevedere le azioni future dell'attaccante [6].

- **Veridicità dei dati e contro-intelligence:** Attaccanti esperti possono identificare un honeypot e tentare di "avvelenare" la *threat intelligence* raccolta, inondandolo di dati falsi o fuorvianti. L'analisi deve quindi valutare non solo la minaccia, ma anche la potenziale manipolazione dei dati stessi [2].

2.1.4 Verso un'analisi intelligente: il ruolo dei large language model

Le sfide analitiche appena descritte evidenziano l'insufficienza degli approcci tradizionali basati su regole fisse, i quali faticano a cogliere le sfumature e il contesto di attacchi complessi registrati in log eterogenei. Un'analisi efficace richiede una comprensione semantica del comportamento dell'attaccante, ponendo le basi per l'impiego di soluzioni basate sull'intelligenza artificiale. Gli LLM offrono un potenziale rivoluzionario in questo ambito. La loro capacità di comprendere il linguaggio naturale e il contesto li rende candidati ideali per interpretare dati non strutturati o semi-strutturati come i log degli honeypot [17], [18]. Gli LLM possono:

- **Gestire l'eterogeneità:** Analizzare log di diversi formati come un unico "racconto" coerente, estraendo significato anche da testo libero e non standardizzato [19].
- **Sintetizzare e comprendere il contesto:** Riassumere lunghe sessioni di attacco in descrizioni concise e leggibili, aiutando a distinguere il "segnale" dal "rumore". Un esempio di questa capacità è l'honeypot `DECEIVE` di Splunk, che utilizza l'IA per analizzare e sintetizzare il comportamento dell'attaccante [20].

Il presente lavoro di tesi si fonda su questo presupposto: se gli honeypot sono strumenti insostituibili per la raccolta dati, la loro efficacia è vincolata a un'analisi intelligente. Rispondendo direttamente alla lacuna identificata nella letteratura scientifica riguardo all'analisi dei dati honeypot [14], l'applicazione degli LLM si propone come la soluzione avanzata per superare le sfide di volume, eterogeneità e complessità, giustificando la metodologia che verrà presentata nei capitoli successivi, la quale si allinea con le direzioni di ricerca più recenti che vedono negli agenti basati su LLM la chiave per l'analisi autonoma della *threat intelligence* [15].

2.1.5 Conclusioni su honeypot e sfide analitiche

In conclusione, gli honeypot, e in particolare gli ecosistemi multi-sensore come `T-Pot`, si confermano strumenti insostituibili per la raccolta di dati di *threat intelligence* ad alta fedeltà. Fornendo un ambiente controllato per osservare le azioni dirette degli avversari, essi permettono di superare i limiti dei sistemi di sicurezza tradizionali, offrendo una visione non contaminata delle TTP impiegate nel mondo reale.

Tuttavia, la loro stessa efficacia nella raccolta dati introduce sfide analitiche significative: l'enorme volume, l'eterogeneità dei formati e la necessità di distinguere il "segnale" (attacchi complessi e innovativi) dal "rumore" (scansioni automatizzate) rendono l'analisi manuale

impraticabile. Questa lacuna, ben documentata in letteratura, evidenzia la necessità di passare a paradigmi di analisi più intelligenti e automatizzati. Come argomentato, gli LLM si posizionano come la tecnologia chiave per affrontare queste sfide, offrendo il potenziale per interpretare il contesto e la semantica dei log eterogenei. Il contesto problematico delineato in questa sezione costituisce, pertanto, il fondamento e la motivazione principale per la soluzione metodologica proposta in questa tesi.

2.2 Large language model come strumento di analisi

Gli *LLM* costituiscono una classe avanzata di algoritmi di *deep learning* che hanno profondamente trasformato il campo dell'elaborazione del linguaggio naturale (NLP). Basati sull'architettura Transformer, questi modelli devono l'aggettivo "large" alla vasta scala dei dati testuali su cui vengono addestrati, il che conferisce loro notevoli capacità di riconoscimento, traduzione, previsione e generazione di testo e altri contenuti [21]. Gli LLM sono, infatti, progettati per comprendere e produrre linguaggio in modo analogo a quello umano, estendendo le loro funzionalità anche alla gestione di diverse forme di contenuto grazie all'ampiezza del loro *training* [22]. Tale capacità di elaborare e generare testo è stata fondamentale nel presente studio per l'interpretazione automatizzata dei log degli honeypot.

L'influenza degli LLM sull'elaborazione del linguaggio naturale è stata trasformativa, consentendo alle macchine di elaborare e generare linguaggio con una sofisticazione e su una scala precedentemente irraggiungibili. Tali sistemi rientrano nella categoria dei *foundation model*: modelli addestrati su dataset di vasta scala che acquisiscono capacità generalizzate, applicabili a una molteplicità di *task* e domini applicativi specifici [22]. In questo contesto, mentre l'NLP tradizionalmente si focalizza sull'interazione macchina-linguaggio umano per la comprensione e generazione testuale [23], gli LLM hanno elevato esponenzialmente queste capacità, superando molte limitazioni delle tecniche NLP convenzionali nell'emulare la fluidità e la pertinenza contestuale del linguaggio umano [24].

2.2.1 Fondamenti: architettura e processo di addestramento

L'efficacia dei moderni LLM è intrinsecamente legata alla loro architettura e al modo in cui vengono addestrati.

Architettura Transformer

Il successo dei moderni LLM è in gran parte attribuibile all'architettura Transformer, introdotta nel 2017 con l'articolo "Attention Is All You Need" [25]. Questa architettura ha segnato un punto di svolta, superando i limiti dei precedenti modelli sequenziali come le reti neurali ricorrenti (RNN). Il concetto cardine del Transformer è il meccanismo di *self-attention*, che permette al modello di pesare dinamicamente l'importanza di ciascuna parola in una sequenza di input

rispetto a tutte le altre, catturando così il contesto e le dipendenze a lungo raggio in modo molto efficace [21], [24]. L'efficacia di questo meccanismo è una delle ragioni per cui modelli basati su Transformer come `Mistral-7B` sono stati scelti per l'analisi di sequenze di log potenzialmente complesse. A differenza dei modelli sequenziali, il Transformer può elaborare tutti gli elementi di una sequenza in parallelo, un fattore cruciale per l'efficienza nell'addestramento su vasti corpus di dati [26]. Per i *task* generativi, come quelli affrontati dagli LLM per la produzione di testo, sono comuni le architetture *decoder-only* (come quelle di `GPT` e del modello `Mistral` utilizzato in questo lavoro), che si concentrano sulla predizione del token successivo in una sequenza. Per garantire che il modello comprenda l'ordine delle parole, data l'elaborazione parallela, viene impiegato il *positional encoding*, che fornisce informazioni sulla posizione di ciascun token all'interno della sequenza [21], [25]. Questi elementi costitutivi permettono agli LLM di raggiungere un elevato grado di comprensione e generazione del linguaggio naturale.

Processo di addestramento

L'addestramento degli LLM si articola tipicamente in due fasi principali: il *pre-training* e il *fine-tuning*.

Pre-training: Apprendimento fondamentale del linguaggio La fase di *pre-training* ha l'obiettivo primario di dotare il modello di una comprensione generale e profonda del linguaggio, includendo la grammatica, la semantica, i pattern linguistici e una vasta base di conoscenza del mondo. Questo risultato è ottenuto addestrando il modello su corpus testuali di dimensioni imponenti [21], [22]. Per i modelli LLM generativi, il *task* di addestramento predominante durante il *pre-training* è il *causal language modeling* (CLM), comunemente noto anche come *next token prediction*. Attraverso questo approccio di apprendimento auto-supervisionato, il modello impara a prevedere il token (parola o sottoparola) successivo in una sequenza, basandosi sui token che lo precedono. Nonostante la sua apparente semplicità, l'esecuzione di questo *task* su scala massiva e su dati testuali eterogenei e non etichettati permette al modello di interiorizzare le strutture sintattiche, le relazioni semantiche e una notevole quantità di informazioni fattuali [22].

Fine-tuning: Specializzazione e adattamento Il *fine-tuning* rappresenta una fase cruciale successiva al *pre-training*, finalizzata ad adattare un LLM pre-addestrato a compiti specifici o ad allinearne il comportamento a requisiti particolari. Questo processo implica un ulteriore addestramento del modello su un dataset più ristretto e mirato, che gli consente di specializzarsi in determinati domini o attività. Ad esempio, come evidenziato da `Elastic` [27], il *fine-tuning* su dataset specifici per un dominio permette al modello di apprendere e generare linguaggio pertinente a tale contesto, migliorando significativamente la pertinenza e l'accuratezza delle sue risposte. Una metodologia di *fine-tuning* particolarmente rilevante è l'*instruction tuning*. Questa tecnica consiste nell'ottimizzare ulteriormente il modello LLM utilizzando un dataset composto da coppie di (prompt di istruzione, risposta desiderata). Tale approccio addestra

esplicitamente il modello a interpretare e seguire istruzioni formulate in linguaggio naturale, potenziando la sua capacità di eseguire compiti specifici, inclusa la generazione di output conformi a strutture predefinite (ad esempio, in formato JSON) [28], [29]. Modelli come `Mistral-7B-Instruct`, impiegato in questo studio, esemplificano l'efficacia dell'*instruction tuning*. Questi modelli vengono addestrati su una varietà di dataset conversazionali e istruzioni pubblicamente disponibili, affinando la loro abilità nel comprendere e rispondere in modo appropriato a prompt specifici forniti dall'utente [29]. In conclusione, se il *pre-training* fornisce agli LLM una vasta comprensione generale del linguaggio, il *fine-tuning* – e in particolare l'*instruction tuning* – li specializza e li adatta a compiti mirati. Questo li rende strumenti significativamente più efficaci e versatili per applicazioni pratiche che necessitano di output strutturati o di risposte precise a istruzioni dettagliate.

2.2.2 Capacità chiave e interazione tramite *prompt engineering*

Le fondamenta architetturali e di addestramento conferiscono agli LLM capacità emergenti che possono essere sfruttate e guidate tramite un'attenta interazione.

Comprensione contestuale avanzata

Gli LLM eccellono nell'inferire significati complessi dal contesto circostante, il che si traduce nella generazione di risposte coerenti e altamente pertinenti [22]. Questa abilità è intrinsecamente legata al meccanismo di *self-attention*, un componente cardine dell'architettura Transformer [25], [30]. Il *self-attention* permette al modello di valutare dinamicamente l'importanza relativa di ciascuna parola o token all'interno della sequenza di input, identificando e focalizzandosi sulle porzioni di testo più salienti per l'interpretazione corrente. In tal modo, gli LLM riescono a cogliere dipendenze a lungo raggio e relazioni semantiche sottili, indipendentemente dalla distanza tra i termini nel testo, garantendo così la produzione di output accurati e contestualmente appropriati [21]. Questa comprensione del contesto è stata sfruttata per permettere all'LLM di discernere l'intento dietro le attività registrate nei log, un passaggio cruciale per una corretta classificazione.

Generazione di output strutturato

Una capacità distintiva degli LLM è la loro abilità nel generare testo conforme a formati predefiniti, quali JSON, XML o altri schemi di dati strutturati. Tale funzionalità è resa possibile attraverso tecniche mirate come il *prompt engineering* accurato, il *fine-tuning* specifico del modello, o l'impiego di strategie di decodifica vincolata. La produzione di output strutturato da parte degli LLM si rivela particolarmente vantaggiosa in contesti applicativi, poiché semplifica notevolmente l'analisi automatizzata dei dati e l'integrazione fluida con altri sistemi software e API, rendendoli strumenti efficaci per numerose applicazioni aziendali. Nel nostro sistema, la capacità di generare

output JSON è stata indispensabile per strutturare l'analisi dell'LLM in modo che potesse essere facilmente elaborata dai moduli successivi di mappatura su CAPEC.

Capacità di seguire istruzioni (*instruction following*)

Una delle evoluzioni più significative negli LLM è la loro abilità nel seguire istruzioni complesse fornite dall'utente. Mentre i modelli LLM pre-addestrati sono primariamente ottimizzati per la predizione del token o della parola successivi, l'*instruction tuning* (o *fine-tuning* supervisionato su istruzioni) rappresenta un processo di addestramento supplementare. Questo processo adatta ulteriormente i modelli pre-addestrati esponendoli a vasti dataset composti da coppie di (istruzione, output desiderato). Di conseguenza, il modello impara a generalizzare la comprensione delle istruzioni e a generare risposte appropriate, permettendogli di eseguire compiti eterogenei come l'analisi del *sentiment*, la generazione di codice o la produzione di testo secondo specifiche direttive [21], [28]. L'*instruction tuning* migliora quindi notevolmente la capacità del modello di interpretare i prompt dell'utente, potenzia le sue prestazioni in scenari *zero-shot* (dove non ha visto esempi specifici del *task* durante il *fine-tuning*) e favorisce un maggiore allineamento con le aspettative umane, incrementandone versatilità e usabilità [28]. Modelli come `Mistral-7B-Instruct`, utilizzato in questo lavoro, sono esempi emblematici di LLM specificamente ottimizzati attraverso l'*instruction tuning* per eccellere in questa capacità [31], la quale è stata direttamente impiegata per guidare il modello a produrre i campi specifici necessari per la nostra analisi.

Interazione tramite prompt engineering e tecniche base

Il *prompt engineering* si configura come una disciplina essenziale, focalizzata sullo sviluppo e l'ottimizzazione dei prompt – le istruzioni fornite in input – al fine di utilizzare efficacemente gli LLM in una vasta gamma di applicazioni e contesti di ricerca. Data l'elevata sensibilità di questi modelli all'input ricevuto, la formulazione di un prompt incide in modo determinante sulla qualità, pertinenza e accuratezza della risposta da essi generata. Le competenze in questo ambito trascendono la mera progettazione di prompt, estendendosi a un insieme più ampio di tecniche e abilità cruciali per interagire, sviluppare e comprendere appieno sia le capacità sia i limiti intrinseci degli LLM. I ricercatori, ad esempio, impiegano il *prompt engineering* per potenziare le prestazioni degli LLM su *task* complessi come il *question answering* o il ragionamento logico-matematico. Parallelamente, gli sviluppatori lo utilizzano per costruire interfacce di *prompting* robuste ed efficienti, capaci di integrare gli LLM con altri strumenti e sistemi software. Di conseguenza, la padronanza dei principi di *prompt engineering* è fondamentale non solo per massimizzare l'efficacia operativa degli LLM, ma anche per esplorare nuove frontiere applicative. Queste includono il miglioramento della loro sicurezza e l'incremento delle loro capacità attraverso l'integrazione con basi di conoscenza dominio-specifiche e strumenti esterni [32]. Nel

contesto del presente studio, un attento processo di *prompt engineering* è stato applicato per elicitare dall'LLM analisi strutturate e semanticamente ricche dei log provenienti dagli honeypot.

Diverse tecniche fondamentali di *prompt engineering* consentono un'interazione efficace con gli LLM. Tra le più comuni si annoverano:

- **Zero-shot prompting:** Questa tecnica prevede di istruire l'LLM a eseguire un compito senza fornire alcun esempio dimostrativo all'interno del prompt. Il modello si affida interamente alla sua vasta conoscenza pre-addestrata per comprendere ed eseguire l'istruzione ricevuta [32].

- **Few-shot prompting:** In contrasto con l'approccio *zero-shot*, il *few-shot prompting* consiste nel fornire all'LLM un piccolo numero di esempi (o "*shot*") del *task* desiderato direttamente nel prompt. Come dimostrato da Brown et al. [33], l'inclusione di tali dimostrazioni può migliorare significativamente le prestazioni del modello, specialmente per *task* complessi, e il numero di esempi può essere modulato in base alla difficoltà. Questa tecnica si rivela particolarmente utile, ad esempio, per guidare l'LLM a generare output strutturati in formati specifici, come JSON, fornendo la struttura desiderata come parte degli esempi [32].

- **Chain-of-Thought (CoT) prompting:** Introdotta da Wei et al. [34], questa tecnica mira a migliorare le capacità di ragionamento degli LLM. Consiste nell'includere nel prompt una sequenza di passaggi di ragionamento intermedi che conducono alla soluzione di un problema. Fornendo questi "pensieri concatenati", specialmente in combinazione con il *few-shot prompting*, si guida l'LLM attraverso un processo deliberativo, potenziando notevolmente le sue prestazioni su *task* che richiedono un'analisi complessa prima di formulare una risposta.

- **Role prompting:** Questa strategia consiste nell'assegnare esplicitamente un ruolo, una personalità o un contesto specifico che l'LLM deve adottare durante la generazione della risposta. Ad esempio, si può istruire il modello ad agire come "un esperto di cybersecurity" o "un analista di dati". Ciò contribuisce a personalizzare l'output, rendendolo più pertinente a una determinata prospettiva o esigenza comunicativa [32].

In sintesi, le tecniche di *prompting* di base come lo *zero-shot*, il *few-shot*, il *Chain-of-Thought* e il *role prompting* offrono un repertorio di strategie fondamentali per guidare efficacemente il comportamento degli LLM. Lo *zero-shot prompting* si dimostra adeguato per *task* in cui la conoscenza intrinseca del modello è sufficiente, mentre il *role prompting* permette di affinare il tono e la prospettiva dell'output. Altre tecniche, come il *few-shot prompting*, sono particolarmente utili per istruire il modello su formati di output specifici o per migliorare le prestazioni su *task* più complessi, garantendo che il testo generato soddisfi requisiti strutturali che ne facilitano l'elaborazione e l'integrazione [32]. Nel presente lavoro, l'approccio di *prompting* adottato per l'analisi dei log honeypot da parte dell'LLM si basa principalmente sui principi dello *zero-shot*

prompting, arricchito da elementi di *role prompting* (istruendo l'LLM ad agire come esperto di cybersecurity) e dalla richiesta esplicita di un output strutturato in formato JSON, che può essere vista come una forma implicita di *few-shot* (o *format-as-shot*) *prompting*.

2.2.3 Applicazioni nel dominio della cybersecurity

Gli LLM stanno progressivamente trovando applicazione nel dominio della cybersecurity, offrendo nuove prospettive per il potenziamento delle strategie difensive e per l'analisi avanzata delle minacce informatiche [35]. La loro intrinseca capacità di comprendere ed elaborare sia il linguaggio naturale sia il codice sorgente li qualifica come strumenti di notevole valore per una pluralità di impieghi:

- **Rilevamento e risposta avanzati alle minacce (*threat detection and response*):** Gli LLM possono analizzare ingenti volumi di dati, quali log di sicurezza e flussi di traffico di rete, per identificare pattern anomali, comportamenti sospetti e potenziali minacce in tempo reale. Ciò si traduce in un miglioramento della velocità e dell'accuratezza dei sistemi di rilevamento [35]. Analogamente, nel nostro studio, l'LLM analizza i log honeypot per identificare e descrivere i pattern di attacco osservati.

- **Analisi e identificazione di vulnerabilità nel codice sorgente:** Attraverso la scansione e l'interpretazione del codice sorgente, gli LLM sono in grado di individuare debolezze strutturali e lacune di sicurezza, suggerendo altresì correzioni mirate per incrementare la resilienza delle applicazioni contro gli attacchi [35].

- **Automazione di processi di sicurezza routinari:** Numerosi compiti di cybersecurity caratterizzati da ripetitività, come la scansione preliminare di vulnerabilità o la gestione di *patch* per vulnerabilità note e semplici, possono essere automatizzati mediante LLM. Questo permette ai team di sicurezza di focalizzare le proprie risorse su problematiche di maggiore complessità e priorità strategica [35].

- **Supporto e potenziamento delle attività di *penetration testing*:** Nello svolgimento di attività di *penetration testing*, gli LLM possono assistere nell'identificazione proattiva delle vulnerabilità dei sistemi e nella formulazione di contromisure efficaci, contribuendo a rafforzare le difese prima che le debolezze vengano sfruttate [35].

- **Individuazione di tentativi di *phishing*:** Gli LLM dimostrano efficacia nell'identificare email e siti web fraudolenti riconducibili ad attacchi di *phishing*, attraverso l'analisi semantica del contenuto e il riconoscimento di pattern caratteristici [36].

- **Accelerazione dell'analisi per la risposta agli incidenti (*incident response*):** In scenari di *incident response*, gli LLM possono processare rapidamente grandi moli di dati eterogenei e informazioni di *threat intelligence*. Questo facilita una più celere e accurata identificazione delle TTP degli attaccanti, fornendo un supporto analitico cruciale [37].

- **Generazione di codice per funzionalità di sicurezza:** Gli LLM possono essere impiegati per generare automaticamente frammenti di codice o script destinati a specifici compiti di sicurezza, come la creazione di regole di rilevamento o l'automazione di controlli [37].

È tuttavia fondamentale riconoscere la natura "*dual-use*" di tali tecnologie: nonostante il loro considerevole potenziale difensivo, gli LLM possono essere altresì sfruttati da attori malevoli per orchestrare minacce più sofisticate. Esempi includono la generazione di email di *phishing* altamente personalizzate e convincenti o la creazione di varianti di codice malware difficilmente rilevabili [35].

2.2.4 Limiti e considerazioni nell'uso per l'analisi di log

Nonostante il loro significativo potenziale, l'impiego degli LLM per l'interpretazione dei log di cybersecurity, come avviene nel presente studio, richiede un'attenta considerazione di alcune limitazioni intrinseche. Tali limitazioni possono influenzare l'affidabilità e l'accuratezza dei risultati generati, rendendo cruciale una valutazione critica del loro output [24], [38]. Tra le principali si annoverano:

- **Allucinazioni e accuratezza fattuale limitata:** Gli LLM possono generare interpretazioni dei log o descrizioni di pattern di attacco che, sebbene superficialmente plausibili, risultano fattualmente inaccurate o non aderenti alla realtà osservata – un fenomeno noto come "allucinazioni". La loro operatività, intrinsecamente basata sul riconoscimento di pattern statistici nei dati di addestramento, non garantisce una comprensione concettuale profonda, potendo condurre a errate classificazioni degli eventi di sicurezza o a descrizioni fuorvianti degli stessi [24], [38].

- **Conoscenza statica e difficoltà con minacce emergenti:** La base di conoscenza di un LLM è "congelata" allo stato dei dati al momento del suo ultimo addestramento. Di conseguenza, il modello potrebbe mostrare incapacità nel riconoscere o interpretare correttamente tecniche di attacco inedite, *exploit zero-day* o varianti di malware recenti non presenti nel suo dataset di *training*. Questa staticità può limitare la sua efficacia nel mappare accuratamente attività particolarmente innovative o recenti ai pattern CAPEC esistenti [38]. Questa limitazione è rilevante per il nostro approccio, poiché la mappatura su CAPEC dipende dalla capacità dell'LLM di generalizzare dai suoi dati di *training*.

- **Difficoltà nel ragionamento complesso su log offuscati o ambigui:** L'analisi dei log di sicurezza, specialmente quelli derivanti da honeypot che possono registrare attività complesse, intenzionalmente offuscate o frammentate, esige sovente capacità di ragionamento *multi-step* e di inferenza avanzata. Gli LLM, pur in continua evoluzione, possono ancora incontrare ostacoli nell'interpretare correttamente l'intento dell'attaccante o la sequenza logica delle azioni in scenari particolarmente intricati, portando a una comprensione superficiale o a conclusioni errate [38].

- **Bias ereditati dai dati di addestramento:** Qualora i dati su cui l'LLM è stato addestrato contengano *bias* relativi a specifici tipi di attacchi, vettori, o modalità di descrizione delle minacce, il modello è suscettibile di perpetuare tali distorsioni nelle sue analisi. Ciò potrebbe influenzare, ad esempio, la priorità implicitamente attribuita a certi eventi o la natura delle descrizioni generate ai fini della mappatura su CAPEC [24], [38].

- **Opacità del processo decisionale (sfide di interpretabilità):** Comprendere appieno il percorso logico attraverso cui un LLM perviene a una specifica interpretazione di un log o a una determinata mappatura CAPEC rimane una sfida considerevole, data la natura intrinsecamente complessa e spesso assimilabile a una "scatola nera" di tali modelli. Questa carenza di trasparenza può ostacolare la validazione della correttezza del ragionamento del modello e l'identificazione delle cause sottostanti a eventuali errori di analisi [38].

Le suddette limitazioni sottolineano l'importanza cruciale della supervisione umana e dell'adozione di un approccio critico nell'utilizzo degli LLM per l'analisi dei log di sicurezza. Sebbene questi modelli possano fornire un potente supporto analitico, i risultati da essi prodotti necessitano inderogabilmente di un'attenta verifica e validazione da parte di analisti esperti, specialmente quando si tratta di assumere decisioni operative o strategiche basate su tali interpretazioni.

2.2.5 Conclusioni sui large language model

In conclusione, gli LLM si attestano come una pietra miliare nell'evoluzione dell'intelligenza artificiale e, in particolare, dell'elaborazione del linguaggio naturale. La loro sofisticata architettura, tipicamente fondata sul modello Transformer, unitamente a processi di addestramento articolati in fasi di *pre-training* e *fine-tuning* (inclusa la specializzazione mediante *instruction tuning*), conferisce a tali sistemi capacità senza precedenti. L'interazione guidata dal *prompt engineering* ne amplifica ulteriormente la versatilità, rendendoli strumenti estremamente potenti per un ampio spettro di applicazioni, che spaziano dalla cybersecurity avanzata alla generazione automatizzata di contenuti strutturati e codice. Ciononostante, un impiego consapevole ed efficace degli LLM non può prescindere dalla piena comprensione dei loro limiti intrinseci. Fenomeni quali le "allucinazioni" (generazione di informazioni non veritiere), la potenziale perpetuazione di *bias* ereditati dai dati di addestramento e le sfide legate all'interpretabilità dei loro processi decisionali, impongono un approccio cauto. Sottolineano, infatti, l'imprescindibile necessità della validazione umana e di una supervisione critica, specialmente in contesti applicativi sensibili. L'evoluzione continua dei modelli, che mirano a ottimizzare il compromesso tra elevate prestazioni, efficienza computazionale e capacità di seguire istruzioni complesse, testimonia il dinamismo della ricerca in questo settore. Tale progresso apre costantemente nuove frontiere per l'integrazione responsabile e l'applicazione innovativa degli LLM in scenari del mondo reale, prefigurando ulteriori trasformazioni nel modo in cui interagiamo con l'informazione e la tecnologia.

2.3 Lo standard di riferimento: CAPEC

Il panorama della sicurezza informatica è caratterizzato da una costante evoluzione delle minacce, rendendo imperativo per i professionisti del settore comprendere a fondo le metodologie utilizzate dagli aggressori. Un elemento fondamentale in questa comprensione è il concetto di *attack pattern*. Gli *attack pattern* possono essere definiti come descrizioni di metodi comuni per sfruttare le debolezze del software, fornendo una prospettiva dal punto di vista dell'attaccante. Come descritto da MITRE, questi schemi derivano dal concetto di *design pattern*, ma sono applicati in un contesto distruttivo anziché costruttivo, basandosi sull'analisi approfondita di esempi reali di *exploit*; essi definiscono inoltre le sfide che un avversario deve affrontare e le modalità con cui le supera [39]. Ciò sottolinea la necessità di adottare la mentalità dell'attaccante per una difesa efficace [40].

Il CAPEC si inserisce proprio per rispondere a questa esigenza di comprensione e sistematizzazione delle metodologie di attacco, rappresentando una risorsa chiave nel campo della sicurezza informatica. La MITRE Corporation, che ne cura la manutenzione, lo definisce come un catalogo pubblicamente disponibile di *attack pattern* comuni, progettato per aiutare gli utenti a comprendere come gli aggressori sfruttano le vulnerabilità nelle applicazioni e in altre capacità abilitate dal cyberspazio. Questa risorsa svolge un ruolo cruciale nell'avanzare la comprensione della comunità e nel migliorare le difese per analisti, sviluppatori, *tester* ed educatori [39].

Per comprendere appieno la sua rilevanza attuale, è utile ripercorrere brevemente l'evoluzione: la storia di CAPEC, come documentato dalla MITRE Corporation, risale al 2007, quando fu inizialmente rilasciato dal Dipartimento della Sicurezza Nazionale degli Stati Uniti (U.S. Department of Homeland Security). Da allora, ha continuato a evolversi grazie alla partecipazione e ai contributi della comunità, con l'obiettivo primario di costituire un meccanismo standard per l'identificazione, la raccolta, il perfezionamento e la condivisione di *attack pattern*. Questa evoluzione guidata dalla comunità, unitamente al sostegno iniziale del Dipartimento della Sicurezza Nazionale, dimostra la natura dinamica e adattiva dello standard, riflettendo un panorama delle minacce in continua trasformazione e sottolineando l'importanza strategica attribuita alla comprensione e alla classificazione degli schemi di attacco per migliorare la postura di sicurezza informatica a livello nazionale [39]. Questo aspetto collaborativo ne accresce la completezza e la rilevanza nel tempo.

2.3.1 Scopo, obiettivi e struttura

Lo scopo primario di CAPEC è la creazione di un catalogo pubblicamente accessibile di *attack pattern* comuni, classificati in modo intuitivo, unitamente alla fornitura di uno schema completo per la descrizione degli attacchi correlati e per la condivisione di informazioni su di essi [41].

Per realizzare questo scopo primario, CAPEC si pone una serie di obiettivi specifici. Molti di questi sono delineati da MITRE [39], tra cui:

- **Avanzare la comprensione della comunità:** Assistere analisti, sviluppatori, *tester* ed educatori nella comprensione degli attacchi noti.
- **Migliorare le difese:** Fornire conoscenze che possano essere utilizzate per potenziare le misure di sicurezza e mitigare i rischi.
- **Fornire contesto per l'analisi del rischio:** Offrire un quadro per l'analisi del rischio architetturale attraverso la comprensione dei potenziali vettori di attacco.
- **Assegnare priorità alle attività di revisione:** Guidare le revisioni dell'implementazione concentrandosi sugli *attack pattern* comuni.
- **Guidare il *penetration testing*:** Informare lo sviluppo di strategie di *penetration testing* efficaci basate su metodi di attacco noti.
- **Comprendere tendenze e attacchi per il monitoraggio:** Aiutare nel monitoraggio dei sistemi riconoscendo gli *attack pattern* comuni durante la fase di rilascio.
- **Sfruttare le lezioni apprese per la prevenzione:** Incorporare la conoscenza degli attacchi passati in una guida preventiva durante la risposta agli incidenti.

Dall'analisi di questi obiettivi specifici emerge chiaramente che il duplice obiettivo di CAPEC, focalizzato sia sulla comprensione che sul miglioramento delle difese, indica una posizione proattiva nell'ambito della sicurezza informatica. Non si tratta semplicemente di documentare gli attacchi, ma anche di fornire alla comunità gli strumenti per prevenirli. Inoltre, fungendo da linguaggio comune e da metro di paragone per gli strumenti di sicurezza, CAPEC mira a standardizzare il panorama della sicurezza informatica, facilitando una migliore comunicazione, comparazione e, in ultima analisi, pratiche di sicurezza più efficaci.

Struttura gerarchica e livelli di astrazione

CAPEC si presenta come un dizionario completo e una tassonomia di classificazione degli attacchi noti, organizzati attraverso visualizzazioni a grafo che ne forniscono rappresentazioni gerarchiche [41].

Le *entry* di CAPEC sono presentate attraverso diverse visualizzazioni (*view*), che costituiscono organizzazioni predefinite dell'intera collezione di pattern. Tra queste, due visualizzazioni di rilievo sono "Meccanismi di attacco" (*Mechanisms of Attack*), che permette di focalizzarsi sui pattern relativi a specifici ambiti della sicurezza informatica, e "Domini di attacco" (*Domains of Attack*), che raggruppa metodi di attacco simili. Tali visualizzazioni sono spesso gerarchiche e rappresentano il livello più alto di una struttura che può essere espansa per rivelare livelli sottostanti [42].

Questa struttura espandibile si fonda su una precisa gerarchia: CAPEC si articola infatti su quattro livelli di astrazione: Categoria, Meta, Standard e Dettagliato, come illustrato anche sul

sito ufficiale. Questi livelli, pur organizzando i pattern in una struttura profonda, seguono un principio meno formale, cercando l'adattamento migliore per la classificazione piuttosto che una rigida aderenza al concetto di un pattern figlio come mero affinamento del genitore. Le definizioni dei livelli, basate principalmente sul glossario ufficiale [43], sono:

- **Livello categoria (*category level*):** Una collezione di *attack pattern* raggruppati per una caratteristica comune, specificamente un'aggregazione basata sull'effetto o sull'intento. Questo tipo di aggregazione non costituisce un attacco direttamente azionabile e si distingue dai *meta attack pattern*, che aggregano per azioni o meccanismi.
- **Livello meta (*meta level*):** Una caratterizzazione astratta di una metodologia o tecnica di attacco, spesso priva di riferimenti a tecnologie specifiche. È utile per la comprensione di approcci di alto livello e per l'analisi del rischio architetturale.
- **Livello standard (*standard level*):** Si focalizza su una specifica metodologia o tecnica usata in un attacco, spesso vista come un singolo componente di un attacco completo. Fornisce dettagli sufficienti per comprendere la tecnica e il suo obiettivo, rappresentando una specificazione di un più astratto *meta attack pattern*.
- **Livello dettagliato (*detailed level*):** Offre il massimo livello di dettaglio tecnico, tipicamente sfruttando una tecnica specifica e mirando a una tecnologia particolare. Esprime un flusso di esecuzione completo, spesso combinando diversi *standard attack pattern*, e richiede meccanismi di protezione specifici per la mitigazione.

Oltre a questa organizzazione gerarchica per livelli di astrazione, ogni singola *entry* di *attack pattern* in CAPEC possiede una struttura informativa standardizzata con elementi identificativi di base come un ID numerico, un titolo e una descrizione sintetica dello schema, come delineato in [42]. A questi si aggiungono numerosi dettagli specifici, come evidenziato in analisi quali quella di [44], che comprendono: un "flusso di esecuzione" (articolato in fasi come Esplorazione, Sperimentazione e Sfruttamento); informazioni sulle "Debolezze correlate" (CWE); valutazioni di probabilità e gravità; elenchi di prerequisiti, competenze e risorse necessarie; descrizione delle conseguenze; strategie di mitigazione; ed esempi di istanze.

La Figura 2.2 mostra un esempio concreto di questa struttura, prendendo come riferimento il pattern `CAPEC-88: OS Command Injection`. Si noti come la scheda fornisca una descrizione testuale dettagliata ("Description") e un flusso di esecuzione strutturato ("Execution Flow"), elementi che, come vedremo nei capitoli successivi, sono fondamentali per l'addestramento e l'interrogazione del nostro sistema basato su LLM e ricerca semantica.

Questa ricca struttura informativa, organizzata gerarchicamente, consente agli utenti di navigare nel vasto catalogo in base alle loro esigenze specifiche, sia per analisi strategiche di alto livello (meta) che per implementazioni tecniche dettagliate (dettagliato). L'inclusione del "flusso di esecuzione" in ogni *entry* fornisce una comprensione pratica e passo-passo di come viene condotto un attacco, colmando il divario tra la conoscenza teorica delle vulnerabilità e le

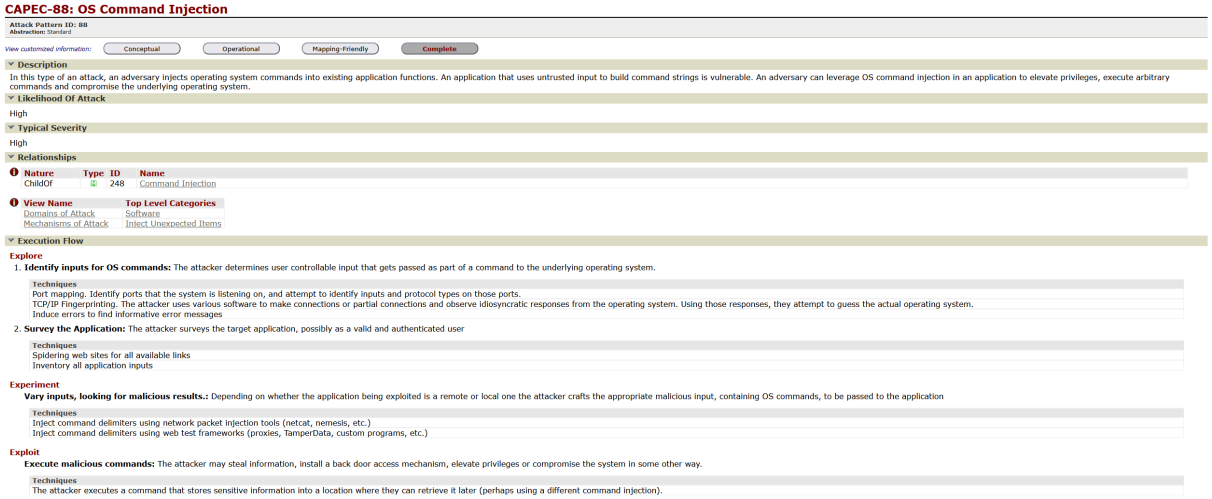

Figura 2.2: Struttura informativa di un pattern di attacco (`CAPEC-88: OS Command Injection`) sul sito ufficiale MITRE.

tecniche di sfruttamento nel mondo reale. Questo è particolarmente prezioso per la formazione e il *penetration testing*.

Per fornire un quadro riassuntivo dei livelli di astrazione, le cui definizioni sono state precedentemente dettagliate basandosi su [43], e per illustrarne i principali casi d'uso derivati da [45], si riporta la Tabella 2.3.

Tabella 2.3: Livelli di astrazione CAPEC

Livello	**Descrizione [43]**	**Casi d'uso [45]**
Categoria	Raccolta di *attack pattern* basati su caratteristiche comuni, con aggregazione fondata su effetti o intenti.	Organizzazione degli attacchi per analisi ad alto livello.
Meta	Rappresentazione astratta di una tecnica o metodologia di attacco, spesso priva di riferimenti a tecnologie o implementazioni specifiche.	Modellazione delle minacce a livello di architettura o design.
Standard	Definizione di una tecnica specifica usata in un attacco, solitamente come singolo passo all'interno di una sequenza più ampia.	Comprensione dettagliata della tecnica e del suo obiettivo.
Dettagliato	Descrizione di un attacco con livello di dettaglio molto elevato, mirato a una tecnologia specifica e con un flusso di esecuzione completo.	Richiede meccanismi di protezione specifici; utile per affrontare scenari concreti.

2.3.2 CAPEC nel contesto degli standard di sicurezza

Per comprendere appieno il ruolo e il valore di CAPEC, è essenziale analizzare le sue interconnessioni con altri standard fondamentali nel panorama della sicurezza informatica. Lo sviluppo di CAPEC, ad esempio, ha seguito metodologie affini a quelle di CVE (*Common Vulnerabilities and Exposures*), il noto sistema di identificatori per vulnerabilità software [46]. Data questa affinità, gli *attack pattern* di CAPEC possono arricchire le informazioni delle *entry* CVE. Infatti, vi sono sforzi continui per mappare gli ID CAPEC agli ID CVE correlati attraverso misure di similarità, sebbene tale collegamento non sia sempre esplicito o di semplice realizzazione [47].

Allo stesso modo, CAPEC è strettamente associato a CWE (*Common Weakness Enumeration*), un elenco sviluppato dalla comunità che cataloga le debolezze comuni nella sicurezza del software. Le *entry* CAPEC, infatti, specificano le debolezze CWE correlate che l'*attack pattern* intende sfruttare, sebbene la mappatura tra i due standard non sia necessariamente biunivoca (uno-a-uno) [42]. Inoltre, CWE può fungere da tramite per tracciare corrispondenze tra gli ID CAPEC e gli ID CVE [47].

Passando a un altro framework di MITRE, la relazione tra CAPEC e il MITRE ATT&CK Framework è complementare, sebbene i due abbiano focus distinti. ATT&CK si concentra primariamente sulla difesa della rete e sulle fasi operative degli avversari, mentre CAPEC è orientato alla sicurezza delle applicazioni e agli *exploit* contro sistemi vulnerabili. Nonostante queste differenze, molti *attack pattern* di CAPEC trovano impiego attraverso le tecniche specifiche descritte da ATT&CK, e i due framework sono incrociati ove pertinente [48]. Un esempio di integrazione pratica si osserva in piattaforme di *penetration testing* come `AttackForge`, dove CAPEC può essere utilizzato per mappare le cosiddette *attack chain*, le quali possono poi essere collegate al framework ATT&CK [49], [50]. A ulteriore testimonianza di questa sinergia, la versione 3.4 di CAPEC ha introdotto una visualizzazione specifica denominata "Pattern correlati ad ATT&CK" [51], [52].

Oltre a queste specifiche interazioni con CVE, CWE e ATT&CK, il ruolo di CAPEC va inquadrato nel contesto più ampio della standardizzazione della sicurezza, mirata a renderla misurabile e gestibile. Ne è un esempio l'iniziativa "Making Security Measurable", illustrata da Martin [46]. Tale iniziativa pone le enumerazioni standardizzate come uno dei quattro pilastri essenziali per l'automazione e la misurazione della sicurezza. All'interno di questo schema, CAPEC assume un ruolo fondamentale: come indicato da Martin nella Tabella 1 del suo studio, è l'enumerazione chiave specificamente designata per fornire identificatori standardizzati per gli attacchi, complementare a CVE per le vulnerabilità e a CWE per le debolezze. Questo ecosistema si completa con riferimenti ad altri standard, tra cui il CVSS (*Common Vulnerability Scoring System*), anch'esso menzionato da Martin.

Complessivamente, quindi, l'interconnessione di CAPEC con standard quali CVE, CWE e MITRE ATT&CK sottolinea la necessità di un approccio olistico alla comprensione della sicurezza informatica. In questo ecosistema, CAPEC chiarisce il "come" vengono perpetrati

gli attacchi, spesso in relazione al "cosa" (le vulnerabilità specifiche identificate da CVE) e al "perché" (le debolezze sottostanti catalogate da CWE). Il tutto può essere ulteriormente contestualizzato attraverso il framework ATT&CK, che descrive il comportamento più ampio degli avversari. La presenza di un comitato congiunto CWE/CAPEC [53] testimonia un impegno formale e continuo verso l'allineamento di questi due standard fondamentali. Tale collaborazione mira a perfezionare l'accuratezza e l'utilità delle mappature tra debolezze e pattern di attacco, un aspetto cruciale per un'efficace analisi del rischio e per la mitigazione delle minacce.

Le interconnessioni di CAPEC con altri standard chiave della sicurezza informatica, discusse in precedenza, sono riassunte nella Tabella 2.4 per una visione d'insieme.

Tabella 2.4: Relazioni di CAPEC con altri standard di sicurezza informatica

Standard/Framework	**Relazione con CAPEC**	**Focus distintivo/Sinergia**
CVE (Common Vulnerabilities and Exposures)	Sviluppato con metodologie affini [41]. I pattern CAPEC arricchiscono le *entry* CVE con il contesto dell'attacco. Sforzi per mappare ID CAPEC a ID CVE, sebbene il collegamento non sia sempre esplicito [47].	**CVE:** Elenco di identificatori per vulnerabilità note. **CAPEC:** Descrive *come* le vulnerabilità vengono sfruttate.
CWE (Common Weakness Enumeration)	Strettamente associato; le *entry* CAPEC specificano le debolezze CWE sfruttate. Mappatura non necessariamente biunivoca [42]. CWE può fungere da tramite per tracciare corrispondenze CAPEC-CVE [47]. Esistenza di un comitato congiunto CWE/CAPEC [53].	**CWE:** Catalogo di debolezze comuni nel software. **CAPEC:** Fornisce i pattern di attacco che sfruttano tali debolezze.
MITRE ATT&CK Framework	Molti pattern CAPEC trovano impiego attraverso tecniche ATT&CK. Framework incrociati ove pertinente [48]. CAPEC può mappare *attack chain* collegabili ad ATT&CK (es. in `AttackForge` [49], [50]). Visualizzazione CAPEC "Pattern correlati ad ATT&CK" [51], [52].	**ATT&CK:** Difesa della rete, tattiche e tecniche degli avversari (fasi operative). **CAPEC:** Sicurezza delle applicazioni, *exploit* contro sistemi vulnerabili.

2.3.3 Applicazioni e rilevanza nella ricerca

Applicazioni pratiche e casi d'uso

CAPEC offre una vasta gamma di applicazioni pratiche e casi d'uso in diversi ambiti della sicurezza informatica, dimostrando la sua versatilità.

Nel *threat modeling*, piattaforme come `IriusRisk` sfruttano CAPEC per generare modelli di minaccia dinamici [54]. Questo approccio aiuta a identificare potenziali vulnerabilità, a comprendere come gli aggressori potrebbero sfruttarle, e supporta la progettazione proattiva di difese con la relativa prioritizzazione dei controlli [55]. Il suo impiego si estende alla modellazione delle minacce a livello di applicazione [48], fornendo un contesto cruciale per l'analisi del rischio architetturale [39].

Nel campo del *penetration testing*, gli *attack pattern* di CAPEC costituiscono una risorsa per definire template di test per i *red team* [56] e per guidare l'esecuzione di test appropriati [39]. Inoltre, come sottolineato da Martin [46], CAPEC serve anche a definire e documentare gli scenari di attacco specifici che i team di sviluppo hanno considerato e contro cui si sono difesi durante le fasi di sviluppo e *penetration testing*. Piattaforme di gestione dedicate, come `AttackForge`, lo utilizzano per standardizzare il linguaggio delle vulnerabilità, ottimizzando così i tempi di registrazione e reportistica per i *penetration tester* [50].

Per quanto riguarda il *security testing* e l'analisi, CAPEC offre un supporto multiforme, come evidenziato da MITRE [56]. Assiste i *tester* nella costruzione di scenari di attacco sistematici e realistici e nell'identificazione di casi di test di sicurezza. Viene impiegato per valutare la copertura degli *attack pattern* da parte di strumenti di analisi statica e dinamica e per correlare i risultati tra queste diverse metodologie. Inoltre, CAPEC supporta la risposta agli incidenti e l'analisi delle minacce, allineando le strategie di mitigazione, e può essere utilizzato per l'analisi del comportamento dei *malware* tramite l'etichettatura degli attacchi.

Nell'ambito dell'educazione e della formazione, CAPEC è impiegato dagli educatori per migliorare la comprensione comunitaria degli attacchi [39]. Si rivela inoltre una risorsa eccellente per la formazione sulla consapevolezza della sicurezza, in quanto comunica efficacemente la prospettiva dell'attaccante e aiuta a formare diverse figure professionali – sviluppatori, *tester*, acquirenti e manager – sulle debolezze sfruttabili e sui metodi di attacco [56].

In termini di *compliance* e *governance*, lo standard supporta la misurazione della conformità agli standard e alle linee guida di settore [56]. Si allinea inoltre con i framework GRC (*Governance, Risk, and Compliance*) fornendo dati azionabili sugli *attack pattern*, il che facilita la prioritizzazione dei rischi e l'implementazione di controlli di sicurezza pertinenti [55].

La vasta gamma di applicazioni pratiche, che spaziano attraverso l'intero ciclo di vita dello sviluppo del software e vari domini della sicurezza informatica, sottolinea la versatilità e il valore intrinseco di CAPEC. Esso funge da risorsa fondamentale che informa una molteplicità di attività di sicurezza, dalla progettazione iniziale fino alla risposta agli incidenti. L'adozione di CAPEC da parte di strumenti e piattaforme commerciali (ad es. `IriusRisk`, `AttackForge`,

Rapid7) ne testimonia il riconoscimento nel settore e la sua comprovata utilità pratica. Tale adozione contribuisce ulteriormente alla standardizzazione delle pratiche di sicurezza e facilita una migliore integrazione della conoscenza specialistica nelle applicazioni del mondo reale.

CAPEC nella ricerca accademica

La ricerca accademica ha esplorato diversi approcci metodologici per analizzare, correlare ed estendere le informazioni contenute in CAPEC.

Utilizzo dell'elaborazione del linguaggio naturale Un'area di ricerca particolarmente attiva riguarda l'impiego di tecniche NLP per collegare gli schemi di attacco CAPEC con le vulnerabilità documentate in CVE (*Common Vulnerabilities and Exposures*) [47], [57], [58]. Questa correlazione è fondamentale poiché permette di connettere le vulnerabilità tecniche a scenari di attacco concreti, migliorando così l'*intelligence* sulle minacce, la gestione delle vulnerabilità e la valutazione del rischio. Stabilire un collegamento chiaro tra CVE e CAPEC consente, infatti, un'analisi delle minacce più informata e una più efficace assegnazione di priorità agli sforzi di mitigazione.

Diverse metodologie NLP sono state esplorate per automatizzare questa correlazione, spesso basandosi su misure di similarità semantica e analisi di parole chiave [57]. Tra queste, alcuni studi hanno valutato l'efficacia di TF-IDF nel misurare la similarità tra le descrizioni testuali di CVE e CAPEC [47]. Altri ricercatori hanno invece impiegato modelli di *embedding* più avanzati, quali Doc2Vec, USE (*Universal Sentence Encoder*) e SBERT (*Sentence-BERT*), al fine di catturare relazioni semantiche più profonde [57]. È interessante notare come alcune analisi comparative suggeriscano che approcci più semplici come TF-IDF possano, in certi contesti specifici, fornire prestazioni superiori rispetto a modelli di *embedding* più complessi [47]. Per migliorare ulteriormente l'accuratezza, sono state proposte metodologie ibride che combinano la similarità semantica con l'analisi delle parole chiave, riuscendo così a catturare sia la comprensione contestuale sia i termini specifici del dominio [57]. Infine, l'ottimizzazione di modelli linguistici pre-addestrati per il dominio della sicurezza informatica, come nel caso di ATT&CK-BERT, ha mostrato potenziale nel migliorare le misurazioni della similarità semantica [57], [59].

Nonostante i promettenti progressi, l'applicazione dell'NLP ai dati CAPEC presenta diverse sfide significative. Una delle principali difficoltà risiede nella necessità di correlare fonti di dati eterogenee (CVE e CAPEC) che spesso mancano di collegamenti diretti ed espliciti [47], [57]. La correlazione manuale, d'altra parte, richiede un notevole sforzo da parte degli esperti di sicurezza e introduce inevitabili ritardi nel processo di analisi [57]. Sebbene esistano collegamenti indiretti tramite CWE (*Common Weakness Enumeration*), questi possono rivelarsi troppo generici o incompleti per determinare con precisione quali schemi di attacco siano rilevanti per una specifica vulnerabilità [47]. A ciò si aggiunge l'elevato volume di nuove vulnerabilità pubblicate quotidianamente, che rende la gestione manuale delle correlazioni intrinsecamente problematica e poco scalabile.

Un'ulteriore sfida è rappresentata dalla difficoltà nel catturare adeguatamente il contesto e i dettagli tecnici specifici del dominio della sicurezza informatica (acronimi, parole chiave specialistiche) utilizzando approcci NLP basati unicamente sulla semantica generale. Come accennato, gli approcci ibridi che integrano l'analisi delle parole chiave con la similarità semantica cercano di affrontare tale limitazione [57].

Impiego di tecniche di visualizzazione Riconoscendo la complessità e la vastità del database CAPEC, un altro filone della ricerca accademica si concentra sull'applicazione di tecniche di visualizzazione per migliorarne l'usabilità e il valore analitico. Diversi studi, come il lavoro di Vanamala et al. [60], propongono lo sviluppo di *dashboard* di visualizzazione interattivi. Questi strumenti impiegano frequentemente rappresentazioni quali *tree map* e grafici a rete (*network graph*) per elucidare sia la tassonomia gerarchica interna di CAPEC (relazioni padre-figlio) sia le sue mappature esterne verso altre basi di conoscenza sulla sicurezza. L'obiettivo primario di tali approcci è rendere i dati intricati di CAPEC più accessibili e immediatamente comprensibili, sia per i ricercatori che per i professionisti del settore. Le rappresentazioni visive efficaci, infatti, possono rivelare pattern e relazioni latenti difficilmente discernibili dalla sola analisi testuale. Permettendo un'esplorazione più intuitiva ed efficiente del database, questi *dashboard* interattivi facilitano l'analisi e possono potenzialmente condurre a nuove intuizioni di ricerca.

Generazione automatica di grafi di attacco basata su CAPEC Oltre all'analisi testuale e alla correlazione tramite NLP, la ricerca si è concentrata anche sull'uso strutturale delle definizioni CAPEC per automatizzare la generazione di grafi di attacco. Un esempio significativo è il framework "Attack Dynamics" proposto da Hankin et al. [61]. Questo lavoro si distingue per essere uno dei primi tentativi di utilizzare le definizioni CAPEC, in particolare i *meta attack pattern* stabili, non solo come riferimento esterno o per fini di categorizzazione, ma come elemento centrale per l'identificazione dei percorsi di attacco. L'approccio di Hankin et al. incapsula le descrizioni e le condizioni dei pattern CAPEC in metadati interni che vengono poi elaborati da motori di ragionamento (basati su logica e `Prolog`) per derivare i grafi di attacco specifici per una data topologia di sistema, vulnerabilità note (CVE) e debolezze (CWE). Il framework mira a superare i limiti dei precedenti generatori di grafi di attacco, offrendo una rappresentazione più ricca delle dinamiche di attacco basata sui pattern standardizzati di CAPEC e producendo visualizzazioni che legano direttamente i percorsi di attacco alla topologia originale del sistema. Questo approccio sottolinea il potenziale di CAPEC come base di conoscenza strutturata per la modellazione automatizzata delle minacce e l'analisi del rischio.

2.3.4 Conclusioni sullo standard CAPEC

Lo standard CAPEC rappresenta una risorsa fondamentale per la comunità della sicurezza informatica, offrendo un catalogo completo e strutturato di *attack pattern*. La sua definizione, i

suoi obiettivi mirati al miglioramento della comprensione e delle difese, la sua organizzazione gerarchica e la sua stretta relazione con altri standard chiave come CVE, CWE e MITRE ATT&CK lo rendono uno strumento indispensabile per professionisti, educatori e ricercatori. Il suo valore risiede non solo nel catalogo stesso, ma anche nel suo ruolo all'interno delle iniziative più ampie volte a rendere la sicurezza misurabile e gestibile, come l'iniziativa "Making Security Measurable" descritta da Martin [46]. Contribuendo alla standardizzazione del linguaggio degli attacchi, CAPEC facilita l'automazione, l'interoperabilità tra strumenti e processi, e una gestione più efficace e basata sui dati della postura di sicurezza. Le sue applicazioni pratiche spaziano dalla modellazione delle minacce al *penetration testing*, dall'analisi della sicurezza alla formazione, dimostrando la sua versatilità e il suo valore in tutto il ciclo di vita dello sviluppo del software e in vari domini della sicurezza informatica. Inoltre, la sua crescente rilevanza nella ricerca accademica sottolinea il suo ruolo come base di conoscenza autorevole e in continua evoluzione, con prospettive future che includono l'esplorazione di ontologie semantiche per una fruizione ancora più avanzata e automatizzata.

2.4 Tecniche di *information retrieval* per il *matching* di documenti

Per realizzare il *matching* tra l'interpretazione di un log e i pattern di attacco in una *knowledge base*, è necessario impiegare tecniche di *information retrieval* (IR). L'IR è la disciplina che applica metodi computazionali, molti dei quali provenienti dal *natural language processing* (NLP), per trovare informazioni pertinenti all'interno di grandi collezioni di documenti [62], [63]. Mentre l'NLP si occupa della comprensione e generazione del linguaggio in senso ampio, l'IR si focalizza sul compito specifico della ricerca e del *ranking*. L'evoluzione dell'IR ha segnato il passaggio da modelli basati su corrispondenze esatte di parole chiave, come la ricerca booleana, a paradigmi più sofisticati come il modello a spazio vettoriale (*vector space model*), introdotto da Salton et al. [62], [64], che costituisce il fondamento delle metodologie moderne.

La discussione seguente delineerà le due principali strategie per il *matching* di documenti: la ricerca semantica, basata su rappresentazioni vettoriali per comprendere il significato, e la ricerca lessicale, che si affida a modelli statistici per la precisione sui termini. Verrà infine analizzato il razionale che giustifica la combinazione di questi approcci in un sistema ibrido, una sinergia che offre soluzioni più robuste per le complesse esigenze di ricerca attuali.

2.4.1 Ricerca semantica: comprendere il significato tramite *embedding*

La ricerca semantica mira a comprendere l'intento dell'utente e il significato contestuale, superando i limiti della semplice corrispondenza di parole chiave. Il suo fondamento risiede nelle rappresentazioni vettoriali del testo, note come *embedding* [65].

Embedding: dalla parola al vettore di significato

Gli *embedding* sono rappresentazioni numeriche del testo (parole, frasi o documenti) sotto forma di vettori densi in uno spazio multidimensionale. Il principio fondamentale è che la distanza geometrica tra i vettori in questo spazio riflette la similarità semantica dei concetti che rappresentano [66]. L'evoluzione dei modelli di *embedding* ha segnato un passaggio cruciale da rappresentazioni statiche a quelle dinamiche e contestuali. I modelli iniziali, come `Word2Vec`, associavano a ogni parola un singolo vettore fisso, un approccio che non risolve il problema della polisemia (una parola con più significati) [67].

La rivoluzione è avvenuta con l'avvento di modelli basati su architetture Transformer, come `BERT` (*Bidirectional Encoder Representations from Transformers*) [65], [68]. Questi modelli generano *embedding* dinamici, in cui la rappresentazione vettoriale di una parola cambia in base all'intero contesto della frase. Per compiti di ricerca di similarità tra intere frasi, un'evoluzione chiave è `Sentence-BERT` (`SBERT`), un'architettura che produce *embedding* a livello di frase in modo computazionalmente efficiente [69].

Un aspetto cruciale per l'efficacia degli *embedding* è la pertinenza del corpus su cui sono stati addestrati. Modelli generici possono non cogliere le sfumature di un dominio altamente specializzato come la cybersecurity. La ricerca ha infatti dimostrato che, attraverso il *fine-tuning*, modelli come `BERT` possono essere specializzati su corpus di dominio, ottenendo prestazioni nettamente superiori rispetto a tecniche basate su *embedding* generici [65]. In linea con questa evidenza, nel presente studio è stato impiegato il modello `ATT&CK-BERT`, un *embedding model* specificamente addestrato sul framework MITRE ATT&CK, in grado di rappresentare il linguaggio tecnico del settore in modo significativamente più accurato di un modello generico [59].

Metriche e strumenti per la ricerca vettoriale

Una volta che il testo è convertito in vettori, è necessaria una metrica per misurarne la somiglianza. La *similarità coseno* è la metrica più comune per questo scopo, in quanto calcola il coseno dell'angolo tra due vettori. Un valore vicino a 1 indica un'alta similarità di direzione (e quindi di significato), ignorando la magnitudine dei vettori, che è spesso irrilevante per i dati testuali [62]. Tuttavia, è importante notare che, sebbene sia uno standard, la similarità coseno non è una metrica perfetta. La ricerca ha evidenziato una discrepanza tra i suoi punteggi e i giudizi umani di similarità, in parte perché la metrica geometrica non sempre riesce a catturare il concetto più ampio di "correlazione semantica" (*semantic relatedness*) che gli esseri umani associano alle parole [67].

Per gestire e interrogare efficientemente milioni di vettori, si utilizzano i *database vettoriali*. Questi strumenti sono ottimizzati per la ricerca di similarità ad alta velocità, impiegando algoritmi di *Approximate Nearest Neighbor* (ANN) come HNSW (*Hierarchical Navigable Small World*). Tali algoritmi sacrificano una minima parte di precisione per ottenere enormi guadagni in velocità, rendendo la ricerca semantica su larga scala computazionalmente fattibile [70]. Nel nostro

sistema, tale funzionalità è stata implementata utilizzando la libreria *open source* `ChromaDB`, un esempio pratico di database vettoriale leggero ed efficiente [71].

2.4.2 Ricerca lessicale: precisione sui termini con il modello TF-IDF

Nonostante l'ascesa delle tecniche semantiche, la ricerca lessicale basata su *keyword* rimane fondamentale per la sua precisione su termini esatti. Il modello più classico e robusto in questo ambito è il `TF-IDF` (*Term Frequency-Inverse Document Frequency*). Il punteggio `TF-IDF` valuta l'importanza di un termine in un documento rispetto a un'intera collezione, basandosi su due componenti [62], [63]:

- ***Term frequency* (TF):** Misura la frequenza di un termine all'interno di un documento. Un termine più frequente è potenzialmente più rilevante per quel documento.
- ***Inverse document frequency* (IDF):** Misura la rarità di un termine nell'intera collezione. I termini rari (spesso tecnici o specifici) sono considerati più informativi e ricevono un peso maggiore.

Tuttavia, il `TF-IDF` presenta limiti evidenti. Essendo un modello basato su un approccio *bag-of-words*, non cattura le relazioni semantiche (non riconosce i sinonimi) e non considera l'ordine delle parole, limitando la sua efficacia con query complesse basate sul linguaggio naturale [63], [72].

2.4.3 L'approccio ibrido: sinergia tra semantica e lessico

La ricerca semantica e quella lessicale non sono alternative, ma forze complementari. Un sistema di ricerca robusto deve unire la comprensione del significato con la precisione terminologica, e l'approccio ibrido nasce proprio da questa necessità. L'efficacia di questa sinergia è stata confermata sperimentalmente: in domini complessi come quello normativo, sistemi ibridi che combinano BM25 (una variante di `TF-IDF`) e modelli Transformer sottoposti a *fine-tuning* hanno dimostrato di superare in modo significativo le prestazioni dei singoli approcci lessicali o semantici [72]. Mentre la ricerca semantica, basata sugli *embedding*, eccelle nel comprendere l'intento dell'utente e nel gestire sinonimi e parafrasi, la ricerca lessicale, con modelli come `TF-IDF`, rimane insuperabile nella precisione su termini tecnici, acronimi e parole chiave rare. La Tabella 2.5 illustra i punti di forza e di debolezza di ciascun approccio, evidenziando la logica che giustifica la loro sinergia.

La combinazione di questi due metodi, come implementata nel presente studio, permette di sfruttare il meglio di entrambi i mondi. Un sistema ibrido è in grado di recuperare documenti che sono sia semanticamente pertinenti all'intento della query, sia precisamente allineati con i termini tecnici specifici in essa contenuti, offrendo così risultati di ricerca significativamente più accurati e robusti.

Tabella 2.5: Confronto tra ricerca lessicale e semantica

Caratteristica	**Ricerca lessicale (es. `TF-IDF`)**	**Ricerca semantica (*embedding*)**
Punto di forza	Precisione su termini esatti e rari.	Comprensione del significato, dei sinonimi e delle parafrasi.
Limiti	Non gestisce sinonimi e linguaggio sfumato (approccio *bag-of-words*).	Può non cogliere le sfumature di un lessico tecnico se il modello è generico.
Approccio	Basato sulla frequenza e rarità statistica dei termini.	Basato sulla similarità di significato in uno spazio vettoriale.
Utilità in domini specifici	Cruciale per la precisione su termini chiave esatti.	Essenziale per la gestione del linguaggio naturale e dell'intento dell'utente.

2.4.4 Conclusioni sulle tecniche di *information retrieval*

In sintesi, le moderne tecniche di *information retrieval* per il *matching* di documenti in domini specializzati si sono evolute verso paradigmi ibridi. La ricerca ha dimostrato che né l'approccio puramente lessicale né quello puramente semantico sono sufficienti da soli. La soluzione più efficace, adottata in questo lavoro, risiede nella combinazione sinergica della comprensione contestuale profonda fornita da modelli di *embedding* dominio-specifici con la precisione terminologica dei metodi statistici classici come il `TF-IDF`.

Capitolo 3

Stato dell'arte

3.1 Introduzione: contesto e prospettiva della ricerca

Il panorama della sicurezza informatica è in rapida trasformazione: attori delle minacce sempre più professionalizzati adottano pratiche automatizzate e strumenti avanzati che rendono gli attacchi più efficaci e difficili da rilevare. Rapporti recenti quali l'IBM X-Force Threat Intelligence Index [73] e il CrowdStrike Global Threat Report [74] evidenziano questa transizione verso modelli operativi più strutturati e industriali, nei quali l'intelligenza artificiale viene progressivamente integrata per automatizzare la generazione di codice malevolo, campagne di *phishing* e tecniche di ingegneria sociale. In questo contesto, l'analisi sistematica dei log di sicurezza rimane un'attività centrale per il *threat hunting* e le analisi forensi; tuttavia, il crescente volume, la molteplicità di formati (shell, HTTP, DNS, payload binari) e la natura spesso non strutturata dei dati rendono i metodi manuali o basati su regole insufficienti, generando un vero e proprio *data deluge* per i team di sicurezza [75].

Gli honeypot si confermano sensori privilegiati per lo studio del comportamento degli avversari: progettati per attrarre attività malevole, essi forniscono dati meno contaminati dal traffico legittimo e informazioni dirette sulle tattiche, tecniche e procedure (TTP) degli attaccanti [76]. Piattaforme come `T-Pot` centralizzano log eterogenei provenienti da diversi sensori honeypot, offrendo una risorsa preziosa per la ricerca ma imponendo al contempo importanti sfide di normalizzazione, aggregazione contestuale e interpretazione semantica. Parallelamente, l'avanzamento degli LLM basati su architetture Transformer ha aperto nuove possibilità: questi modelli sono in grado di processare grandi quantità di testo non strutturato e di fornire rappresentazioni semantiche profonde, rendendoli candidati naturali per attività come il *parsing* avanzato dei log, la ricostruzione delle intenzioni dell'attaccante e la generazione automatica di *intelligence* [13], [77].

Il capitolo che segue fornisce una rassegna critica della letteratura pertinente a queste tematiche, muovendo dai metodi più elementari (approcci lessicali e vettoriali tradizionali) fino alle tecniche più recenti che coinvolgono LLM, *retrieval* avanzato e strategie ibride di *matching*.

L'obiettivo è identificare i progressi empirici e metodologici, mettendo in luce i limiti che il presente lavoro di tesi intende colmare: in particolare l'elaborazione *end-to-end* di log eterogenei e il *mapping* automatico verso la *knowledge base* CAPEC mediante un motore di ricerca ibrido supportato da un interprete LLM *upstream*. Una sintesi comparativa dei principali studi analizzati è presentata nella **Tabella 3.1 a pagina 37**.

3.2 Revisione della letteratura: dalla ricerca lessicale alle tecniche ibride e LLM

L'evoluzione metodologica per l'analisi dei testi di sicurezza riflette un percorso ben definito: dai metodi basati sulla frequenza dei termini (*bag-of-words*, `TF-IDF`, `BM25`) alle tecniche di *embedding* distribuito (`Word2Vec`, `Universal Sentence Encoder`, `SBERT`) fino all'impiego di LLM generativi e a pipeline che combinano *retrieval* semantico e generazione controllata (RAG). Qui di seguito vengono esaminati, in ordine progressivo di complessità metodologica, i contributi più significativi e la loro rilevanza per l'obiettivo di mappare log eterogenei ai pattern CAPEC.

3.2.1 Approcci lessicali e primi confronti con embedding

Gli studi iniziali che affrontano il problema del *mapping* tra testi tecnici e tassonomie di attacco utilizzano spesso metodi lessicali come `TF-IDF` o `BM25`, notando la loro efficacia nel catturare termini tecnici rari e acronimi critici. Kanakogi et al. [58] conducono un confronto empirico fra `TF-IDF`, `Universal Sentence Encoder` (USE) e `SBERT` per associare descrizioni CVE ai pattern CAPEC: il risultato più significativo è che `TF-IDF`, nonostante la sua semplicità, si dimostra competitivo, a volte superiore agli *embedding* generici in questo dominio altamente tecnico. Similmente, Andrew et al. [78] mostrano che, quando l'unità di analisi è un comando shell isolato, i metodi lessicali possono fornire prestazioni migliori di modelli di *embedding* non specializzati. Tali risultati evidenziano una prima lezione: la precisione sui termini tecnici può risultare più determinante della comprensione semantica generale in contesti fortemente dominati da nomenclature specifiche.

3.2.2 Transizione verso embedding e modelli dominio-specifici

Per superare le limitazioni intrinseche delle rappresentazioni lessicali, la ricerca si è orientata verso tecniche capaci di una profonda comprensione contestuale, come i *sentence-embedding* e i modelli Transformer. In questo ambito, il lavoro di Abdeen et al. [59] su `SMET` introduce una metodologia particolarmente innovativa. La loro pipeline, invece di analizzare l'intero testo, impiega il *Semantic Role Labeling* (SRL) per estrarre selettivamente le frasi che descrivono azioni di attacco ("*attack vectors*") dalle descrizioni CVE. La similarità tra questi vettori viene poi misurata da `ATT&CK-BERT`, un modello da loro sviluppato tramite *fine-tuning* sul framework

ATT&CK, ottenendo miglioramenti significativi rispetto a *baseline* come SBERT e TF-IDF. Lo studio di SMET fornisce due intuizioni fondamentali per il presente lavoro. Primo, convalida l'uso di rappresentazioni specializzate per la *cybersecurity*; è proprio per questa ragione che il modello ATT&CK-BERT è stato adottato come componente centrale della nostra soluzione. Secondo, dimostra il valore dell'estrazione di sottostrutture informative per un *mapping* più accurato. Ciononostante, il loro approccio viene validato unicamente su testi puliti come le descrizioni CVE, evidenziando la difficoltà e l'originalità della sfida affrontata in questa tesi: trasferire una strategia simile ai dati grezzi e rumorosi generati dagli honeypot.

3.2.3 LLM per *parsing* e normalizzazione dei log

Un passo fondamentale verso l'analisi di log eterogenei è il *parsing*: la trasformazione di messaggi grezzi in *template* o strutture normalizzate. Ma et al. [79] propongono LLMParser, che affronta il *parsing* come un problema di traduzione testo-a-testo utilizzando modelli generativi (Flan-T5, LLaMA) in regime *few-shot*; i risultati su 16 dataset *open source* mostrano prestazioni superiori rispetto ai *parser* tradizionali, con accuratezze medie elevate. Questo lavoro supporta l'idea che gli LLM possono sostituire o integrare i tradizionali moduli di *parsing*, semplificando la pipeline di normalizzazione per log con formati eterogenei.

Parimenti, Setianto et al. [80] illustrano come modelli generativi (GPT-2C) adattati al dominio possano estrarre informazioni utili da log Cowrie, ottenendo buoni risultati nella classificazione di comandi ed utilità eseguite dagli attaccanti. Tali contributi indicano come il *matching* tra *parsing* e successiva interpretazione semantica sia un'area feconda per l'impiego di LLM. Karlsen et al. [81] forniscono un contributo metodologico di natura empirica essenziale per la progettazione di sistemi basati su LLM per l'analisi dei log. Il loro lavoro, denominato LLM4Sec, consiste in una valutazione comparativa su larga scala di molteplici architetture (tra cui varianti di BERT/RoBERTa e modelli *decoder-like*) applicate a *task* classici sui log – *parsing*, classificazione di eventi e rilevamento anomalie – mediante esperimenti condotti su sei dataset pubblici rappresentativi di log applicativi e di sistema. La lezione empirica più rilevante è che il *fine-tuning* specifico al dominio migliora in modo sostanziale le prestazioni: i modelli adattati ottengono incrementi marcati su metriche classiche (*precision*, *recall*, F1) rispetto alle medesime architetture usate *out-of-the-box*, fino a valori molto elevati di F1 quando il dominio è ben rappresentato dai dati di addestramento.

3.2.4 LLM come interpreti e *generative agents* per l'analisi di sessioni honeypot

Spostandosi dall'estrazione sintattica all'interpretazione semantica, emergono studi che trattano l'LLM non solo come strumento di rappresentazione, ma come vero e proprio "interprete" degli eventi. Ozkok et al. [82] valutano quantitativamente l'uso di ChatGPT (GPT-4) in modalità

zero-shot sull'analisi di log da honeypot (`Elasticsearch` e SSH): il modello si dimostra in grado di spiegare le conseguenze degli attacchi e di mappare azioni su MITRE ATT&CK con elevati tassi di accuratezza (superiore al 96% nella spiegazione delle conseguenze degli attacchi e fino al 98.84% nel mappare correttamente i log SSH alle tecniche ATT&CK), offrendo prove empiriche sulla praticabilità degli LLM come supporto per l'analista. Parallelamente, Boffa et al. [83] presentano `LogPrécis`, una pipeline che segmenta sessioni shell e produce "*fingerprint*" tattici – sequenze di tattiche ATT&CK – su grandi raccolte di sessioni (circa 400k), dimostrando come l'astrazione tattica possa ridurre significativamente il carico di analisi. Questi lavori evidenziano un pattern comune: gli LLM sono efficaci nel generare rappresentazioni di alto livello, ma spesso il *mapping* realizzato è orientato verso ATT&CK e non verso CAPEC, e i metodi tendono a essere applicati a tipologie di log relativamente omogenee (es. shell). Rafiey e Namadchian [84] esplorano una direzione complementare a quella del *fine-tuning*: l'uso pragmatico di LLM generativi in modalità *prompting* e *few-shot* per il *mapping* automatico tra descrizioni tecniche (CVE) e tecniche del framework MITRE ATT&CK. La loro metodologia dimostra che, mediante *prompt engineering* e l'esposizione di alcuni esempi corretti, modelli della famiglia GPT (`GPT-3.5/GPT-4`) possono inferire *mapping* con prestazioni competitive, fornendo un'alternativa rapida e scalabile al *training* supervisionato tradizionale. Gli autori riportano che le prestazioni variano in funzione della qualità e della rappresentatività degli esempi nel prompt, della complessità sintattica dell'input e della presenza di terminologia altamente specialistica; pertanto, pur essendo utile per la prototipazione rapida e per scenari in cui non sono disponibili grandi set annotati, questo approccio beneficia in modo evidente dall'integrazione con segnali lessicali e da modelli *domain-specific* quando è richiesta precisione sui termini tecnici.

3.2.5 Approcci ibridi e adattivi per il *mapping* verso knowledge base di attacco

La letteratura più recente converge inequivocabilmente su soluzioni ibride che uniscono *retrieval* lessicale (`TF-IDF` / `BM25`, *keyword matching*) e *retrieval* semantico (*sentence-embedding*, modelli Transformer specializzati). In questo filone, Bonomi et al. [57] propongono un metodo esplicitamente ibrido per collegare descrizioni CVE ai pattern CAPEC: il loro sistema calcola vettorialmente la similarità tramite *sentence-embedding* (`SBERT` e un modello dominio-specifico `ATT&CK-BERT`) e, in parallelo, esegue un *matching* lessicale che evidenzia la presenza di token tecnici e acronimi (ad es. "XSS", "SQLi", nomi di exploit/tool). I due segnali di rilevanza vengono poi fusi in una funzione di punteggio combinata che bilancia la capacità degli *embedding* di catturare la somiglianza contestuale con la precisione dei *match* lessicali sui termini tecnici. Bonomi et al. riportano miglioramenti consistenti rispetto ai singoli approcci su più metriche di *retrieval*, mostrando in particolare come il segnale lessicale riduca i falsi positivi generati da *retrieval* puramente denso mentre gli *embedding* correggono i casi in cui sinonimia e parafrasi renderebbero inefficace una ricerca basata solo su *keyword*. Sauze-Kadar e Loubier [85]

estendono questa idea introducendo un'ulteriore dimensione di adattività: invece di applicare sempre la stessa combinazione fissa di metodi, il loro framework analizza preliminarmente le caratteristiche del documento (lunghezza, densità di termini tecnici, presenza di sigle, struttura sintattica) e seleziona dinamicamente, da un pool di strategie (varianti `TF-IDF`, diversi modelli di *embedding*, regole *keyword*), la modalità di *matching* più adatta al caso specifico. Su un ampio dataset costruito ad hoc di coppie CVE–CAPEC, il sistema adattivo ottiene guadagni di accuratezza rispetto a modelli monolitici e rispetto a una semplice fusione statica, in particolare quando il testo presenta caratteristiche atipiche (descrizioni molto brevi o, al contrario, lunghe e ricche di contesto). Questo approccio riduce la dipendenza da un'unica strategia di recupero e migliora la copertura su sottogruppi eterogenei di documenti. Complessivamente, questi contributi forniscono una solida validazione empirica al principio metodologico che guida la presente tesi: (i) la precisione lessicale è essenziale per catturare termini tecnici e ridurre falsi positivi; (ii) la comprensione semantica apportata da *embedding* dominio-specifici è cruciale per cogliere sinonimia e parafrasi; (iii) la fusione (o la selezione adattiva) dei due segnali aumenta la robustezza del *mapping*.

3.2.6 *Retrieval-Augmented Generation* (RAG) e filtraggio degli *artifact* di retrieval

Un'ulteriore evoluzione metodologica è rappresentata dall'impiego di architetture RAG: in questi sistemi un modulo di *retrieval* propone candidati da una *knowledge base* e un LLM generativo effettua un raffinamento e una verifica contestuale. Webb et al. [86] applicano RAG per mappare CAPEC e ATT&CK in contesti ICS (*Industrial Control Systems*): la fase generativa agisce da filtro per ridurre i falsi positivi derivanti da una ricerca basata puramente su vettori densi. Questo approccio è interessante perché combina le forze della ricerca semantica con la capacità dell'LLM di valutare e contestualizzare risultati, ma va notato che in molti casi RAG viene sperimentato su *mapping* tra conoscenze testuali già strutturate piuttosto che su log grezzi che richiedono un'interpretazione preliminare.

3.2.7 Applicazioni *end-to-end* e limiti empirici

Alcuni studi recenti provano a chiudere la catena *end-to-end*: Tejero-Fernández e Sánchez-Macián [87] propongono una pipeline LLM per log IoT che include *mapping* su CAPEC e generazione di raccomandazioni di mitigazione, dimostrando la fattibilità del *mapping* CAPEC in scenari con dataset strutturati (Edge-IIoTset). Pur rappresentando un passo importante, questi sistemi si basano su dati già normalizzati e su compiti di classificazione diretta, mentre la sfida aperta rimane: come applicare e validare una metodologia ibrida e interpretabile su log grezzi e rumorosi provenienti da un ecosistema eterogeneo di honeypot (come `T-Pot`)?

3.3 Sintesi critica e gap della letteratura

Dalla rassegna della letteratura, i cui punti salienti sono riassunti nella Tabella 3.1, emerge un quadro metodologico maturo ma ancora parziale. I metodi lessicali tradizionali (`TF-IDF`, `BM25`) conservano un ruolo cruciale in domini altamente tecnici, dove la presenza di termini rari, acronimi e *token* specialistici costituisce spesso il fattore discriminante per un corretto abbinamento testuale; studi comparativi mostrano infatti che gli *embedding* generici non sempre soppiantano le soluzioni basate su frequenza di termine in questi contesti [58], [78]. Parallelamente, l'adozione di *embedding* e modelli dominio-specifici (ad es. `ATT&CK-BERT`) accresce la capacità di ragionamento semantico, in particolare se la rappresentazione vettoriale è accompagnata dall'estrazione di sottostrutture informative che riducono il rumore testuale e focalizzano il *matching* su elementi rilevanti [59]. Inoltre, gli LLM generativi si dimostrano efficaci sia per il *parsing* avanzato dei log sia come interpreti *upstream* in grado di normalizzare, sintetizzare e arricchire l'input destinato a fasi successive di *retrieval* o classificazione [79], [82], [83]. Infine, le architetture ibride e i sistemi RAG rappresentano oggi lo stato dell'arte per bilanciare *recall* semantico e precisione lessicale e per mitigare i falsi positivi tipici del *retrieval* denso [57], [85], [86].

Nonostante questi progressi, la letteratura presenta limiti metodologici e *gap* empirici che motivano chiaramente il contributo di questa tesi. Si possono sintetizzare tre *gap* principali:

- **Validazione su dati puliti vs. dati rumorosi**: la maggior parte delle valutazioni viene condotta su testi relativamente strutturati (CVE, *report*) o su log omogenei (es. sessioni shell), mentre sono scarsi gli studi che validino le stesse tecniche su log eterogenei, frammentati e talvolta offuscati tipici di piattaforme multi-honeypot come `T-Pot`.
- **Robustezza degli approcci ibridi su input degradato**: benché le soluzioni ibride dimostrino vantaggi su dataset puliti, manca una valutazione esaustiva della loro resistenza a rumore, *truncation*, *encoding/obfuscation* dei payload e variabilità di formato presente nei log reali.
- **Ruolo dell'LLM: *upstream* vs. *downstream***: numerosi lavori utilizzano LLM come moduli di verifica o raffinamento a valle del processo di *retrieval*; molto meno comune è l'impiego sistematico dell'LLM a monte come interprete che genera *query* strutturate e arricchite per un motore di ricerca ibrido – una scelta metodologica che, come si argomenta nella tesi, può migliorare la qualità del *retrieval* e la spiegabilità del *mapping*.

In sintesi, la letteratura fornisce una base consolidata che giustifica l'adozione di schemi ibridi e di modelli specializzati; tuttavia, la reale innovazione richiede la loro rigorosa validazione su log eterogenei e degradati e l'esplorazione del ruolo strategico dell'LLM come interprete *upstream*. La proposta di questa tesi si inserisce precisamente in questo spazio di ricerca, offrendo

una risposta metodologicamente strutturata e sperimentalmente verificabile a problemi aperti che risultano di grande rilievo per la comunità della *cybersecurity*.

Tabella 3.1: Tabella riassuntiva dello stato dell'arte

Riferimento	Fonte dati	Task principale	Metodologia chiave	Gap / Limite
Kanakogi et al. [58]	Descrizioni CVE	*Mapping* CVE → CAPEC	Confronto lessicale (`TF-IDF`) vs. semantico (`SBERT`)	Non esplora approcci ibridi; limitato a dati puliti (CVE).
Andrew et al. [78]	Comandi Shell (isolati)	*Mapping* comando → ATT&CK	Confronto lessicale vs. `Word2Vec`	Analisi non contestuale; usa modelli semantici datati.
Abdeen et al. (SMET) [59]	Descrizioni CVE	*Mapping* CVE → ATT&CK	Estrazione "*attack vectors*" (SRL) + `ATT&CK-BERT`	Validato solo su dati puliti; non è approccio ibrido.
Ma et al. (LLM-Parser) [79]	Log eterogenei	*Log parsing*	LLM generativo (`Flan-T5`, `LLaMA`) in *few-shot*	Si ferma al *parsing* sintattico, no interpretazione semantica.
Setianto et al. (GPT-2C) [80]	Log Honeypot (`Cowrie`)	*Parsing* comandi	`GPT-2` *fine-tuned* (*task* di QA)	Limitato a un singolo tipo di log (`Cowrie`).
Karlsen et al. (LLM4Sec) [81]	Log generici	Classificazione (Anomalo/Normale)	*Fine-tuning* di modelli Transformer	*Task* di classificazione binaria, non *mapping* complesso.
Ozkok et al. [82]	Log Honeypot (`Cowrie`, `ES`)	Interpretazione + *mapping* log → ATT&CK	ChatGPT (`GPT-4`) in *zero-shot*	*Mapping* solo ad ATT&CK; non usa approccio ibrido.
Boffa et al. (LogPrécis) [83]	Log Honeypot (Shell)	Classificazione tattiche ATT&CK	LLM *fine-tuned* per creare "*fingerprint*"	Limitato a log shell; *mapping* solo ad ATT&CK.
Rafiey e Namadchian [84]	Descrizioni CVE	*Mapping* CVE → ATT&CK	LLM generico (`GPT-4`) in *few-shot*	Validato solo su dati puliti; non è approccio ibrido.
Bonomi et al. [57]	Descrizioni CVE	*Mapping* CVE → CAPEC	Approccio ibrido (`ATT&CK-BERT` + *keyword*)	Validato su dati puliti; no LLM interprete *upstream*.

Tabella 3.1: Tabella riassuntiva dello stato dell'arte (continua)

Riferimento	**Fonte dati**	**Task principale**	**Metodologia chiave**	**Gap / Limite**
Sauze-Kadar e Loubier [85]	Descrizioni CVE	*Mapping* CVE → CAPEC	Approccio ibrido adattivo (multi-modello)	Validato su dati puliti; no LLM interprete *upstream*.
Webb et al. (RAG) [86]	CAPEC / ATT&CK	*Mapping* CAPEC ↔ ATT&CK	RAG (*retrieval* semantico + LLM generativo)	*Mapping* tra standard, non da log grezzi; *retrieval* non ibrido.
Tejero-Fernández e Sánchez-Macián [87]	Log IoT strutturati	Classificazione + *mapping* log → CAPEC	LLM *fine-tuned* + generazione mitigazioni	Dati già strutturati; *mapping* diretto, non *retrieval*.

Capitolo 4

Soluzione proposta

4.1 Introduzione e obiettivi della soluzione

Come evidenziato nel capitolo precedente, la letteratura scientifica sull'analisi automatizzata delle minacce presenta una chiara frammentazione. Da un lato, la ricerca sull'analisi dei log ha iniziato a impiegare con successo gli LLM, ma spesso per compiti preliminari (come il *parsing* o la classificazione binaria) o affidandosi a un'inferenza puramente semantica per il *mapping* verso standard di alto livello. Dall'altro lato, la ricerca sul *matching* tra *knowledge base* ha sviluppato metodologie molto più sofisticate (ibride, adattive), ma le ha applicate quasi esclusivamente a testi puliti e strutturati come le descrizioni CVE, la cui robustezza su dati grezzi e rumorosi rimane non verificata.

Questa frammentazione ha un'implicazione pratica significativa: le tecniche di analisi dei log più avanzate spesso producono output che mancano della granularità necessaria per una risposta agli incidenti dettagliata, mentre le sofisticate metodologie di *matching* non sono state validate su dati reali e "sporchi", lasciando incerta la loro efficacia in scenari operativi. Manca quindi una *pipeline integrata* che si proponga come un ponte metodologico tra questi due domini, applicando le tecniche di *matching* più avanzate al problema più complesso dell'analisi dei log di sicurezza eterogenei.

Per colmare queste lacune, il presente capitolo descrive l'architettura di una soluzione innovativa, progettata per mappare automaticamente i log provenienti da sistemi honeypot eterogenei ai pattern di attacco del framework MITRE CAPEC. L'obiettivo è creare una pipeline automatizzata che traduca dati grezzi e non strutturati in conoscenza strutturata, contestualizzata e azionabile, fornendo agli analisti di sicurezza uno strumento per comprendere rapidamente la natura degli attacchi osservati.

Per affrontare queste sfide, la soluzione proposta introduce diversi contributi metodologici che ne guidano la progettazione e ne definiscono l'innovatività.

1. **Aggregazione olistica del comportamento dell'attaccante:** Il primo contributo consiste nell'adozione di un approccio di aggregazione olistica del comportamento dell'attaccante.

A differenza di metodologie che analizzano eventi di log isolati, il sistema aggrega tutti i dati provenienti da un singolo attore (identificato tramite indirizzo IP o ID di sessione) in un unico profilo di comportamento. Questo passaggio è cruciale perché permette di ricostruire il contesto completo e la sequenza cronologica delle azioni, trasformando una serie di eventi atomici e frammentati in una narrazione coerente dell'attacco, consentendo un'analisi incentrata sull'attore anziché sul singolo evento.

2. **Interpretazione semantica *upstream* tramite *prompt engineering* avanzato:** Il secondo contributo concerne il posizionamento strategico dell'LLM come interprete semantico *upstream*, guidato da una strategia di *prompt engineering* avanzato. Invece di agire come classificatore finale, il ruolo del modello è quello di "tradurre" il comportamento aggregato dell'attaccante in un'analisi strutturata. Ciò viene realizzato tramite un *prompt template* sofisticato che orchestra molteplici strategie: il *role prompting* assegna al modello il ruolo di esperto di cybersecurity; il *chain-of-thought* lo guida in un processo di ragionamento interno per favorire l'astrazione; e il *few-shot learning* fornisce esempi specifici (esemplari) per ogni tipo di honeypot, contestualizzando l'inferenza. L'output è un'analisi JSON arricchita che funge da query di alta qualità per la fase successiva, disaccoppiando efficacemente la comprensione del linguaggio naturale dal *task* di *information retrieval*.

3. **Implementazione di un motore di *matching* ibrido basato su *rank fusion*:** Il terzo e principale contributo risiede nell'implementazione di un motore di *matching* ibrido basato su *rank fusion*. Per garantire la massima robustezza, il sistema interroga due indici paralleli della *knowledge base* CAPEC: uno vettoriale, basato su *embedding* dominio-specifici (`ATT&CK-BERT`) per la comprensione semantica, e uno lessicale, basato su `TF-IDF`, per la precisione sui termini tecnici. La fusione dei risultati non avviene tramite una semplice combinazione di punteggi grezzi – che possono essere su scale eterogenee e quindi non direttamente confrontabili – ma attraverso la *Reciprocal Rank Fusion*. Questa tecnica, metodologicamente più robusta e allineata con lo stato dell'arte dell'*information retrieval*, combina le classifiche (*rank*) prodotte dalle due ricerche, premiando i documenti che appaiono in alto in entrambe le liste e rendendo il sistema meno sensibile alle specifiche metriche di punteggio di ciascun motore di ricerca.

Nelle sezioni seguenti, verrà illustrata in dettaglio l'architettura generale del sistema, seguita da una descrizione approfondita di ciascun modulo logico che la compone: la contestualizzazione olistica, l'interpretazione basata su LLM, la costruzione della *knowledge base* e, infine, il *matching* ibrido con *rank fusion*.

4.2 Architettura generale del sistema

L'architettura della soluzione proposta è concepita come una pipeline modulare che elabora i dati in modo sequenziale, trasformando i log grezzi in una lista classificata di pattern di attacco CAPEC.

Ogni modulo svolge un compito logico distinto, garantendo la separabilità delle funzioni e una chiara tracciabilità del flusso di dati. L'adozione di un'architettura a pipeline sequenziale è stata preferita rispetto a modelli *end-to-end* monolitici per due ragioni fondamentali. In primo luogo, garantisce l'interpretabilità del processo: ogni modulo produce un output intermedio (i profili di comportamento aggregati, la descrizione testuale, l'analisi JSON) che può essere ispezionato e validato, un requisito cruciale in contesti di sicurezza. In secondo luogo, offre modularità ed estensibilità: ogni componente può essere migliorato o sostituito indipendentemente dagli altri, ad esempio integrando un nuovo tipo di honeypot nel modulo di *preprocessing* o sperimentando con una diversa tecnica di fusione nel modulo di *matching*.

La Figura 4.1 illustra la visione d'insieme dell'architettura e le interazioni tra i suoi componenti principali.

Il flusso operativo del sistema, che traduce i dati grezzi in conoscenza azionabile, può essere descritto come una sequenza di trasformazioni progressive dei dati:

1. **Acquisizione e aggregazione dei dati:** Il punto di partenza è un insieme di log eterogenei. Invece di processare eventi singoli, il sistema applica immediatamente un'aggregazione contestuale, raggruppando tutti i log correlati a un singolo attore (identificato tramite `src_ip` o `session_id`). Questo trasforma una moltitudine di righe di log atomiche in una collezione di "profili di comportamento dell'attaccante", che costituiscono la vera unità di analisi per il resto della pipeline.

2. **Preprocessing e sintesi olistica:** Ciascun profilo di comportamento viene processato dal modulo di *preprocessing* e contestualizzazione. Questo componente non esegue una semplice pulizia, ma applica logiche di sintesi specifiche per il tipo di honeypot. Ordina gli eventi cronologicamente, decodifica i payload offuscati, e astrae l'intera sequenza di attività in un'unica descrizione testuale olistica. Questa descrizione è una narrazione in linguaggio naturale dell'intera campagna dell'attore, ottimizzata per l'interpretazione da parte dell'LLM.

3. **Interpretazione semantica e generazione della query:** La descrizione olistica viene passata al modulo di interpretazione semantica. Sfruttando un LLM in un contesto *few-shot*, questo modulo astrae ulteriormente, trasformando la narrazione dell'evento in un'analisi JSON strutturata. L'output JSON non è una semplice stringa di ricerca, ma una query arricchita e multi-sfaccettata che disaccoppia il segnale semantico (per la ricerca vettoriale) da quello lessicale (per la ricerca *keyword*).

4. **Matching ibrido e rank fusion:** L'analisi JSON generata funge da query per il modulo di *matching* ibrido. Questo modulo esegue due ricerche parallele sulla *knowledge base* CAPEC, una semantica e una lessicale, ottenendo due classifiche (*ranked list*) distinte di pattern candidati. Il nucleo di questo modulo è l'algoritmo di *Reciprocal Rank Fusion*, che

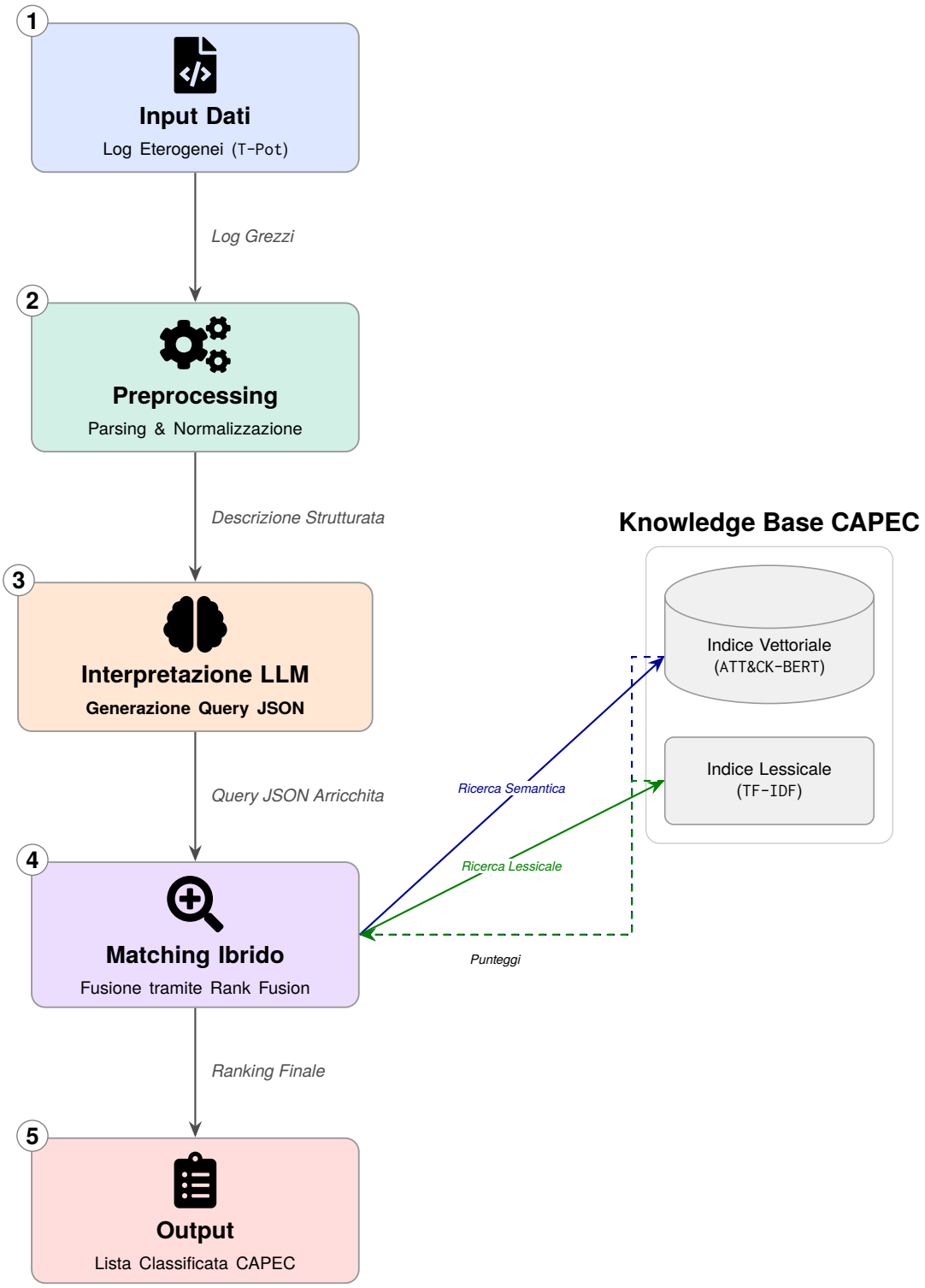

Figura 4.1: Diagramma dell'architettura generale della pipeline di analisi e matching

fonde le due classifiche in un'unica lista finale, più robusta e affidabile, basandosi sulla posizione relativa dei risultati anziché sui loro punteggi grezzi.

5. **Generazione dell'output:** L'output finale del sistema è una lista classificata dei pattern di attacco CAPEC più pertinenti per il profilo di comportamento analizzato, ordinati secondo il punteggio RRF decrescente. Questo fornisce all'analista una valutazione immediata e gerarchizzata della natura della minaccia osservata.

Questa architettura modulare permette di isolare la complessità di ogni fase. Il disaccoppiamento tra l'interpretazione semantica (gestita dall'LLM) e il *matching* (gestito dal motore di ricerca ibrido) costituisce il nucleo metodologico della soluzione. Tale scelta consente al sistema di beneficiare sia della comprensione contestuale profonda dei LLM per l'analisi di dati complessi e aggregati, sia della precisione e dell'efficienza delle tecniche di *information retrieval* più avanzate per il confronto con una vasta *knowledge base*. Le sezioni seguenti dettaglieranno la logica interna di ciascuno di questi moduli.

4.3 Modulo di *preprocessing* e contestualizzazione dei dati

Il primo passo operativo della pipeline è affidato al modulo di *preprocessing* e contestualizzazione, un componente cruciale progettato per trasformare i dati grezzi, rumorosi ed eterogenei in un formato pulito, contestualizzato e semanticamente ricco, idoneo per l'interpretazione da parte degli LLM. Come evidenziato nello stato dell'arte, la gestione dell'eterogeneità e del rumore dei log è una delle sfide principali nell'analisi automatizzata. Pertanto, lo scopo di questo modulo non è una semplice pulizia dei dati, ma una sofisticata ricostruzione del contesto dell'attività dell'attaccante.

Per garantire che l'architettura sia in grado di generalizzare su vettori di attacco radicalmente diversi, il design del modulo si focalizza su un sottoinsieme rappresentativo dei sensori disponibili nell'ecosistema `T-Pot`. Sebbene la piattaforma offra 30 tipologie di honeypot, la soluzione è stata progettata per gestire le cinque categorie semantiche più significative e diffuse:

- **Interazione shell (`Cowrie`):** Per analizzare comandi complessi e sessioni interattive.
- **Scansione di rete (`Honeytrap`):** Per interpretare pattern di ricognizione su protocolli eterogenei.
- **Malware e servizi (`Dionaea`):** Per catturare *exploit* su servizi vulnerabili come SMB e FTP.
- **Traffico VoIP (`SentryPeer`):** Per gestire segnali specifici di frode telefonica (SIP).
- **Interazione web (`CiscoASA`):** Per analizzare richieste HTTP mirate a vulnerabilità note.

Questa selezione garantisce che il sistema venga validato su log con strutture sintattiche e livelli di astrazione opposti, massimizzando la robustezza della pipeline.

La logica di funzionamento del modulo è illustrata nel flowchart in Figura 4.2 e si basa su un'elaborazione differenziata che tiene conto della natura specifica di ogni fonte dati.

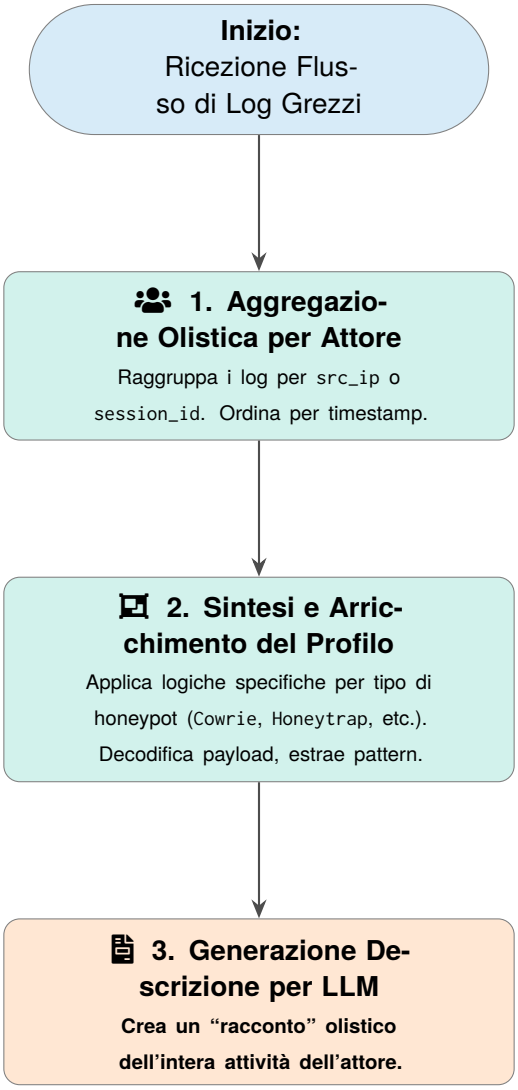

Figura 4.2: Flowchart del processo logico di aggregazione olistica e contestualizzazione dei log

Il processo si articola in una sequenza di passaggi logici, applicati dopo un'aggregazione preliminare degli eventi.

4.3.1 Aggregazione olistica del comportamento dell'attaccante

Il principio guida di questa fase è l'analisi incentrata sull'attore anziché sull'evento singolo. Tutti i log provenienti da un'unica fonte di attacco vengono raggruppati in un unico "profilo di comportamento". La chiave di aggregazione è scelta in modo adattivo:

- Per honeypot interattivi come `Cowrie`, che forniscono un identificativo di sessione univoco, l'aggregazione avviene per `session_id`.

- Per honeypot non interattivi o di servizio (`Honeytrap`, `Dionaea`, etc.), dove un ID di sessione non è disponibile, l'aggregazione avviene per `src_ip`. Questo permette di raggruppare tutte le attività (es. scansioni su porte diverse, tentativi di connessione a servizi diversi) provenienti da un singolo attore in un unico contesto di analisi.

Ciascun gruppo di log viene quindi ordinato cronologicamente per timestamp, ricostruendo la sequenza temporale delle azioni dell'attaccante. Questo passaggio trasforma una collezione di eventi atomici in una serie di narrazioni coerenti, una per ogni attore osservato.

4.3.2 Sintesi e arricchimento specifico per fonte dati

Una volta aggregato il profilo di comportamento, il modulo applica una logica di sintesi e arricchimento specifica per il tipo di honeypot, al fine di estrarre le caratteristiche più salienti di ogni interazione.

Analisi delle sessioni Cowrie (SSH/Telnet): Data la natura interattiva di `Cowrie`, l'analisi si concentra sulla ricostruzione della catena di azioni post-autenticazione. Vengono estratti e sintetizzati:

- **Stato dell'autenticazione:** Si determina se il login ha avuto successo e, in tal caso, con quale utente, distinguendo da semplici tentativi di *brute-force*.
- **Riepilogo dei comandi eseguiti:** Per gestire sessioni lunghe, viene applicata un'euristica di riepilogo. Invece di troncare la lista dei comandi, il sistema preserva i comandi iniziali e finali (spesso cruciali per *staging* e *cleanup*) e identifica i comandi "notevoli" nella parte centrale della sessione (es. comandi unici, complessi o che usano piping/redirezione).
- **Attività di rete e file system:** Vengono identificate e aggregate attività specifiche come il download/upload di file (estraendo URL e percorsi) e i tentativi di *tunneling*/*proxy* (estraendo le destinazioni), che sono forti indicatori di TTP avanzate.

Analisi dei log di rete (Honeytrap, SentryPeer): Per honeypot che monitorano connessioni di rete, l'analisi si focalizza sull'aggregazione di pattern di scansione e sull'ispezione dei payload.

- **Profilo di scansione:** Vengono aggregati tutti i protocolli e le porte uniche targetizzate da un singolo attore per delinearne il profilo di ricognizione.
- **Inferenza sul payload:** Per i payload di rete, viene applicata una logica di decodifica e un'analisi euristica dei primi byte per inferire il tipo di traffico (es. TLS Handshake, HTTP Request), arricchendo il log con un contesto che non sarebbe esplicito.
- **Pattern specifici (es. SentryPeer):** Per log specializzati come quelli SIP, vengono implementate euristiche per riconoscere pattern di attacco specifici del dominio, come

l'enumerazione sequenziale di interni ("dial plan scanning"), analizzando la distribuzione numerica delle chiamate.

Analisi dei log di servizio (Dionaea, CiscoASA): Per honeypot che emulano servizi specifici, l'analisi si concentra sulle interazioni a livello applicativo.

- **Interazioni applicative:** Vengono estratte sequenze di comandi specifici del protocollo (es. comandi FTP per `Dionaea`) o richieste HTTP (per `CiscoASA`), ricostruendo il dialogo tra l'attaccante e il servizio emulato.
- **Cattura di credenziali:** Viene data priorità all'identificazione e all'aggregazione di tentativi di autenticazione, estraendo coppie di username/password che rappresentano un forte segnale di attacchi di tipo "*credential access*".

4.3.3 Generazione della descrizione testuale per l'LLM

Infine, tutte le informazioni estratte, aggregate e arricchite vengono assemblate in una descrizione testuale semi-strutturata. Questa scelta è deliberata: invece di un formato rigido come un JSON, si produce un "racconto" in linguaggio naturale dell'evento. Tale formato è progettato per massimizzare le capacità di comprensione contestuale degli LLM, che sono pre-addestrati su vasti corpus di testo narrativo e descrittivo. L'output include frasi complete e campi chiave etichettati (es. "Specific observations:", "Command Summary:"), fornendo all'LLM un input che è al contempo ricco di dettagli e facile da interpretare semanticamente.

In sintesi, questo modulo agisce come un'interfaccia intelligente e specializzata, che non solo normalizza i dati ma li traduce in una rappresentazione semantica di alto livello, pronta per essere analizzata dal modulo successivo.

4.4 Modulo di interpretazione semantica basato su LLM

Una volta che il profilo di comportamento dell'attaccante è stato sintetizzato in una descrizione testuale olistica, interviene il modulo di interpretazione semantica. Questo componente costituisce il nucleo innovativo della pipeline, poiché sfrutta le capacità di ragionamento e comprensione del linguaggio naturale di un LLM per "tradurre" le informazioni fattuali e tecniche in un'analisi semantica di alto livello, strutturata e generalizzata.

4.4.1 Ruolo dell'LLM come interprete *upstream*

A differenza di approcci presenti in letteratura che utilizzano l'LLM come classificatore finale o come raffinatore "*downstream*" (ad esempio in architetture RAG), in questa architettura il modello agisce a monte del processo di *matching*. Il suo ruolo non è quello di fornire la

risposta finale, ma di creare una query arricchita e strutturata per il motore di ricerca ibrido. Questa scelta metodologica si basa sul principio di separazione dei compiti: l'LLM eccelle nella comprensione del linguaggio non strutturato e nell'inferenza contestuale, mentre i sistemi di *information retrieval* specializzati sono più efficienti e accurati nel *matching* su larga scala contro una *knowledge base*.

Posizionare l'LLM a monte del processo offre un vantaggio chiave: la separazione tra l'interpretazione del dato grezzo e il recupero dell'informazione. L'LLM si specializza nel compito in cui eccelle (comprensione del linguaggio naturale), mentre il motore di ricerca si occupa del compito in cui è più efficiente (*matching* su indici strutturati). Questo approccio evita di sovraccaricare l'LLM con il compito di "ricordare" l'intero catalogo CAPEC, un *task* per cui non è ottimizzato, riducendo il rischio di "allucinazioni" e aumentando la robustezza complessiva del sistema. L'LLM, quindi, funge da ponte tra la rappresentazione fattuale e spesso rumorosa del log e l'astrazione concettuale necessaria per interrogare efficacemente il catalogo CAPEC.

4.4.2 Progettazione del prompt: un approccio ibrido e adattivo

L'interazione con l'LLM è governata da un *prompt template* attentamente ingegnerizzato, che orchestra diverse strategie di *prompt engineering* per massimizzare la qualità e la consistenza dell'output. La logica di questo processo è illustrata in Figura 4.3.

Il *prompt template* è strutturato utilizzando intestazioni in formato Markdown (es. `### Your Role`, `### Critical Rules`) per segmentare chiaramente le istruzioni. Questo approccio coniuga una struttura statica con un contenuto *dinamicamente adattivo*: il contenuto del prompt viene modificato in base al tipo di honeypot che ha generato i dati. Rispetto agli approcci standard, la strategia di *prompting* è stata evoluta per includere vincoli decisionali rigorosi:

- ***Role prompting* specializzato ("Threat Intelligence Lexicographer"):** Al modello non viene assegnato il ruolo di un generico analista, ma quello specifico di "Lessicografo di Threat Intelligence". L'istruzione esplicita è di non narrare l'evento specifico, bensì di fornire una *definizione formale ed enciclopedica* della tecnica di attacco osservata. Questo disaccoppia la descrizione dai dettagli effimeri del log, favorendo il *matching* semantico.

- **Gerarchia delle minacce (*threat hierarchy*):** Per risolvere l'ambiguità in scenari complessi in cui un attaccante esegue molteplici azioni (es. scansione seguita da *exploit*), il prompt impone una gerarchia di priorità decisionale esplicita: *Execution* > *Credential Access* > *Reconnaissance*. Il modello è istruito a identificare sempre l'*apex threat*, ovvero l'azione con la priorità più alta nella gerarchia, ignorando le attività propedeutiche di livello inferiore per focalizzarsi sull'obiettivo primario dell'attacco.

- ***Chain-of-thought* (CoT) strutturato:** Viene imposto un processo di ragionamento interno obbligatorio articolato in fasi sequenziali: ingestione delle evidenze, verifica della validità degli input, inferenza dell'intento reale (es. distinguere un errore di login da un tentativo di

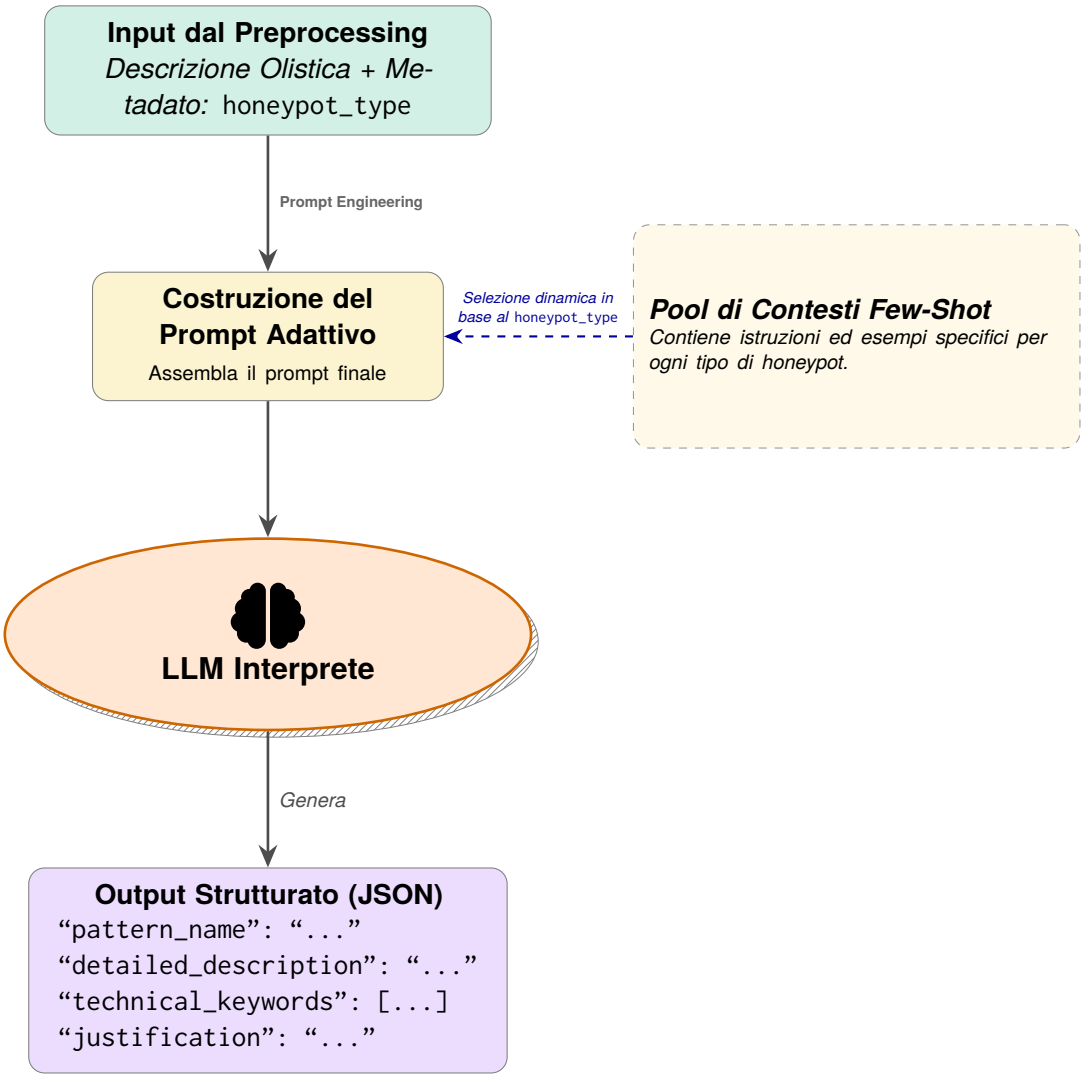

Figura 4.3: Modello concettuale dell'interazione con l'LLM

credential access) e controllo di coerenza contestuale. Solo al termine di questo processo deduttivo il modello è autorizzato a generare l'output JSON.

- **Regole critiche (*critical rules*):** Un set di vincoli negativi (es. "*Evidence-Bound Reasoning*", "*Strict Abstraction*") impedisce al modello di inferire azioni non supportate esplicitamente dai log (ad esempio, dedurre un attacco *brute-force* da un numero insufficiente di richieste) o di produrre semplici riassunti. Queste regole servono a mitigare le allucinazioni.

- ***Few-shot learning* adattivo:** Il prompt viene arricchito dinamicamente con un esempio *one-shot* specifico per il tipo di honeypot in esame (`Cowrie`, `Honeytrap`, etc.) e con *istruzioni di contesto specifiche*. Ad esempio, per i log di rete, viene esplicitato come interpretare i payload decodificati, mentre per le sessioni shell si forniscono euristiche sull'analisi delle sequenze di comandi.

- **Specificazione rigida del formato di output:** Viene richiesto esplicitamente un output in formato JSON con quattro campi predefiniti, garantendo che la risposta sia immediatamente parsabile dalla pipeline software.

4.4.3 Struttura dell'output JSON e ruolo dei campi

L'output JSON richiesto al modello è strutturato in quattro campi, ciascuno con uno scopo strategico per alimentare il motore di *matching* ibrido:

1. `pattern_name`: Un nome conciso e formalizzato per il TTP, strutturato come "Categoria Generale: Tecnica Specifica" (es. "Execution: Remote Payload Delivery and Execution"). Questo campo fornisce il titolo principale per la *ricerca semantica* e la *ricerca lessicale*.

2. `detailed_description`: Una definizione formale e astratta (stile manuale tecnico) della tecnica di attacco identificata come *apex threat*. A differenza di una semplice descrizione narrativa, questo campo è ottimizzato per massimizzare la similarità semantica con le descrizioni presenti nella *knowledge base* CAPEC, fungendo da ponte concettuale.

3. `technical_keywords`: Una lista di termini tecnici specifici (protocolli, comandi, tool, acronimi) estratti o inferiti dal log. Questo campo è fondamentale per alimentare la *ricerca lessicale*. Il suo scopo è garantire la precisione su termini che la ricerca semantica, per sua natura, potrebbe considerare meno importanti.

4. `justification`: Una singola frase che spiega quale specifica evidenza nel log è stata decisiva per l'identificazione dell'*apex threat*. Questo campo non viene usato per il *matching*, ma è cruciale per l'interpretabilità e la *Explainable AI* (XAI), permettendo all'analista di verificare rapidamente la correttezza del ragionamento del modello.

In questo modo, il modulo di interpretazione semantica produce una rappresentazione dell'evento che è al contempo rigorosa (grazie alla *threat hierarchy*) e ricca di sfumature (grazie al CoT), ottimizzata per il successivo recupero ibrido.

4.5 Costruzione della *knowledge base* CAPEC

Per poter mappare efficacemente l'analisi generata dall'LLM, è indispensabile disporre di una base di conoscenza (*knowledge base*) degli attacchi che sia completa, ben strutturata e ottimizzata per la ricerca. A tal fine, è stato progettato un modulo per la costruzione di una *knowledge base* a partire dal catalogo ufficiale MITRE CAPEC. La progettazione di questo modulo è guidata dal principio fondamentale della ricerca ibrida, che richiede la creazione di due indici paralleli e complementari, come illustrato in Figura 4.4.

Il punto di partenza per la costruzione della *knowledge base* è il corpus ufficiale dei pattern di attacco. Specificamente, è stato scaricato il catalogo completo di CAPEC in formato XML dal sito ufficiale di MITRE, utilizzando la *versione 3.9* dello standard. Questa versione costituisce la fonte dati autorevole e onnicomprensiva da cui vengono estratte tutte le informazioni necessarie.

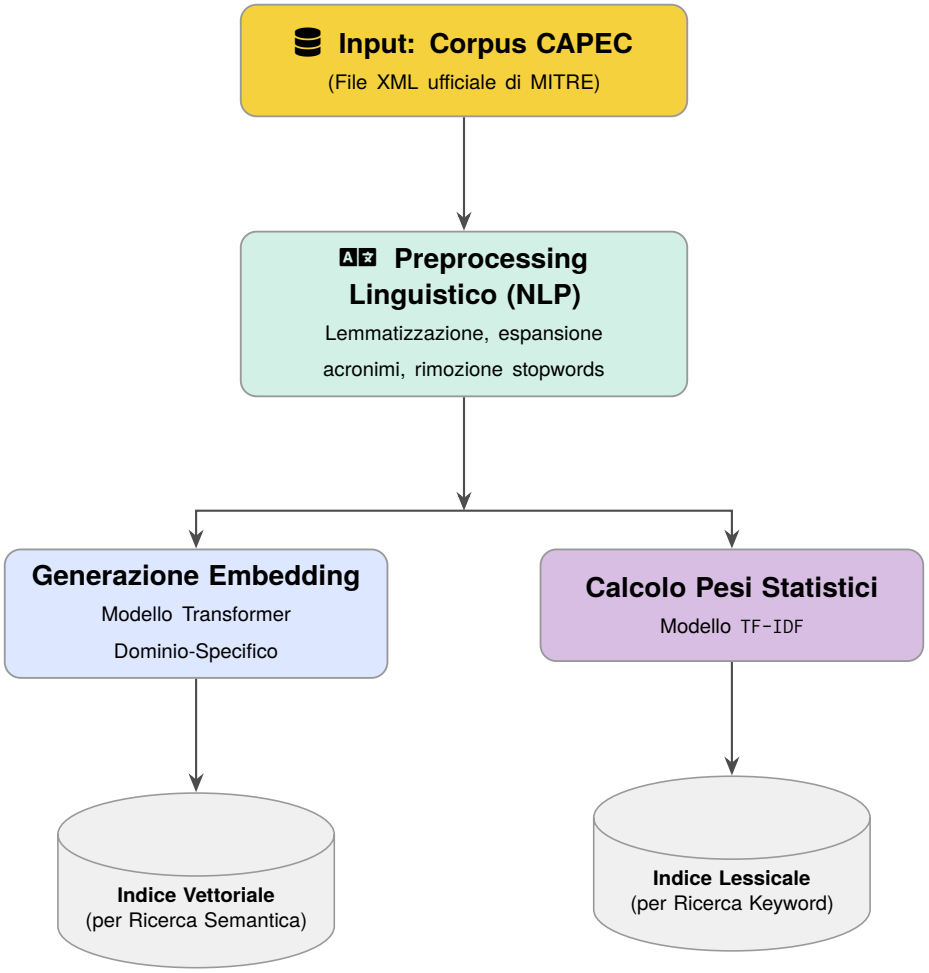

Figura 4.4: Architettura logica della knowledge base CAPEC a doppio indice

Il processo di costruzione si articola in due fasi principali, che trasformano questo corpus XML grezzo in una struttura di ricerca performante.

4.5.1 Selezione dei dati e *preprocessing* linguistico

Il primo passo consiste nell'estrarre e normalizzare il contenuto testuale dal file XML di CAPEC.

Selezione strategica dei contenuti. Non tutti i campi di un pattern CAPEC sono ugualmente informativi per il *task* di *matching*. Per massimizzare il segnale e ridurre il rumore, sono stati selezionati strategicamente solo i campi che descrivono la natura e l'esecuzione dell'attacco. Per ogni pattern vengono estratti:

- **Nome e descrizione:** Il nucleo testuale che definisce l'attacco.
- **Flusso di esecuzione (*execution flow*):** Fornisce una descrizione sequenziale e dettagliata delle azioni dell'attaccante, un'informazione strutturalmente ricca e molto pertinente.
- **Prerequisiti:** Descrivono le condizioni necessarie perché l'attacco abbia successo, aggiungendo contesto cruciale.
- Altri metadati come *Likelihood of Attack* e *Typical Severity* vengono inclusi per arricchire il contesto generale del pattern.

Campi come le mitigazioni o le conseguenze sono stati deliberatamente esclusi in questa fase per focalizzare il *matching* esclusivamente sulla descrizione dell'attacco, evitando di "confondere" il motore di ricerca con informazioni relative alla difesa o agli impatti. L'insieme di questi testi costituisce il "contesto semantico" di ogni pattern.

***Preprocessing* linguistico avanzato (NLP).** A differenza del *preprocessing* strutturale applicato ai log, il trattamento riservato a questo corpus testuale è di natura prettamente linguistica. L'obiettivo è normalizzare il testo per massimizzare la coerenza semantica e la rilevanza dei termini. Questo processo ha uno scopo duplice: preparare il testo sia per la ricerca semantica che per quella lessicale. Ogni passaggio è fondamentale:

- **Espansione di acronimi e sinonimi tecnici:** Il testo viene arricchito per rendere esplicite le relazioni concettuali (es. "SQLi" diventa "sql injection"). Questo è cruciale per l'indice semantico, che può così associare correttamente l'acronimo al suo significato completo.
- **Pulizia e normalizzazione testuale:** Vengono rimosse informazioni che costituiscono rumore (punteggiatura, IP di esempio) e il testo viene uniformato.
- **Rimozione di termini generici (*stopwords*):** Vengono eliminate non solo le *stopwords* comuni, ma anche termini generici del dominio della sicurezza (es. "attack", "pattern",

"adversary"). Questo passaggio è vitale per l'indice `TF-IDF`, poiché aumenta il peso relativo dei termini tecnici veramente discriminanti che altrimenti verrebbero penalizzati.

- **Analisi morfologica e lessicale:** Utilizzando tecniche NLP, le parole vengono ridotte alla loro forma base (lemma) e vengono estratti *token* significativi (nomi, verbi, aggettivi) e n-grammi. La generazione di bi-grammi e tri-grammi è particolarmente importante perché permette a entrambi gli indici di trattare espressioni composte (es. "cross site scripting") come un unico concetto, aumentando la precisione del *matching*.

4.5.2 Indicizzazione parallela: semantica e lessicale

Il testo normalizzato e arricchito di ogni pattern CAPEC viene quindi utilizzato per popolare due indici distinti, che verranno interrogati simultaneamente dal modulo di *matching*.

1. **Costruzione dell'indice vettoriale (semantico):** Per ogni pattern, viene generato un *embedding* a livello di documento utilizzando il modello Transformer dominio-specifico `ATT&CK-BERT`. Come discusso nello stato dell'arte, la scelta di un modello specializzato per la cybersecurity è fondamentale per garantire una rappresentazione vettoriale che catturi accuratamente la semantica del linguaggio tecnico. I vettori risultanti vengono memorizzati in un indice vettoriale persistente, implementato tramite `ChromaDB`, che è ottimizzato per la ricerca di similarità ad alta velocità basata sulla metrica della *similarità coseno*.

2. **Costruzione dell'indice lessicale (*keyword*):** Parallelamente, l'intero corpus di testi CAPEC preprocessati viene utilizzato per costruire un modello statistico `TF-IDF`. Questo indice viene configurato per utilizzare la stessa tokenizzazione avanzata (con lemmi e n-grammi) usata per il *preprocessing*, garantendo coerenza tra i due indici. L'indice `TF-IDF` calcola una matrice termine-documento in cui ogni termine ha un peso che riflette la sua importanza. Il suo scopo è garantire la massima precisione nell'identificare corrispondenze esatte su termini tecnici e parole chiave rare che potrebbero essere trascurati dalla pura analisi semantica.

La creazione di questa *knowledge base* a doppio indice è un prerequisito fondamentale per l'implementazione del motore di ricerca ibrido. La struttura a due indici permette al sistema di sfruttare la complementarietà tra la comprensione del significato contestuale e la precisione terminologica, producendo un *ranking* finale più robusto e accurato.

4.6 Modulo di *matching* ibrido con *rank fusion*

Il modulo di *matching* ibrido rappresenta la fase conclusiva e decisionale della pipeline. Il suo compito è utilizzare l'analisi semantica strutturata, generata dal modulo LLM, per orchestrare il

processo di ricerca, interrogare efficacemente la *knowledge base* a doppio indice e, infine, fondere i risultati in un'unica classifica di rilevanza. Questo modulo concretizza il principio ibrido della soluzione, combinando due segnali di rilevanza distinti per produrre un risultato più robusto e accurato.

La logica di funzionamento del modulo, illustrata nel diagramma di sequenza in Figura 4.5, si basa sul principio della ricerca parallela seguita dalla fusione delle classifiche tramite *Reciprocal Rank Fusion*.

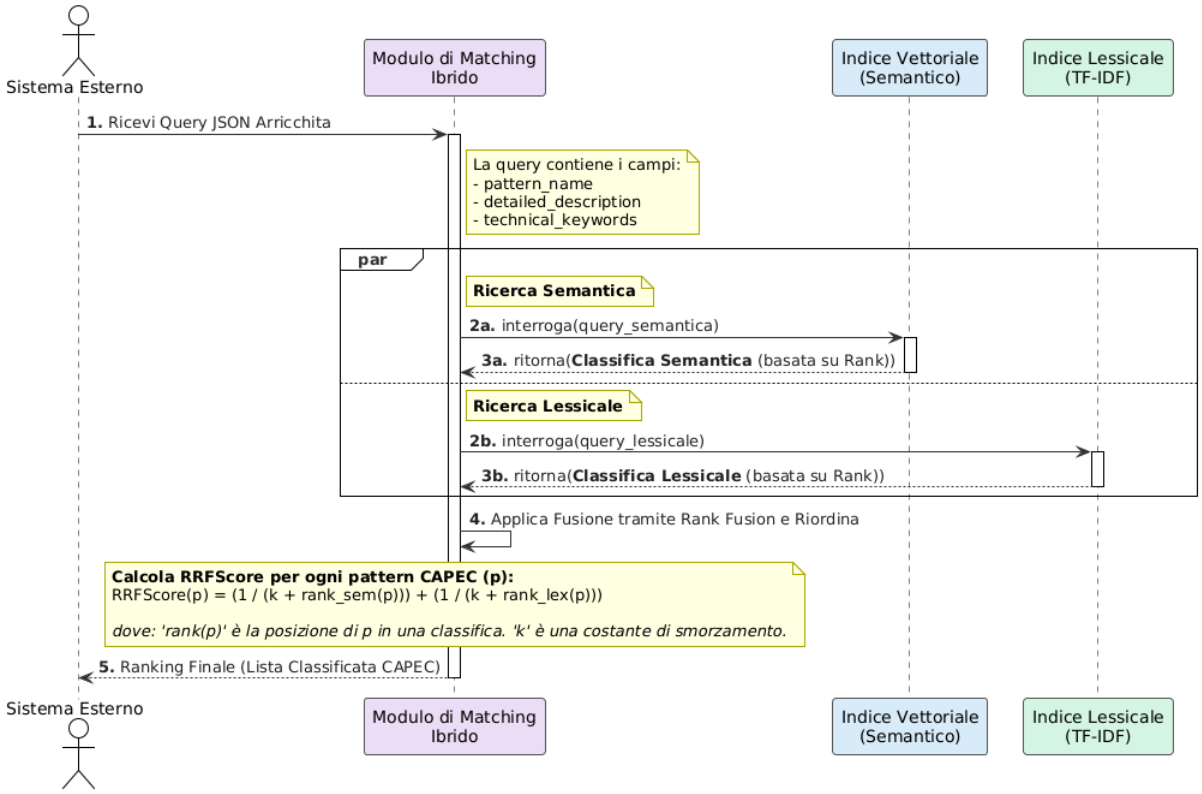

Figura 4.5: Diagramma di sequenza del processo logico di matching ibrido

Il processo si articola nei seguenti passaggi logici:

1. **Acquisizione e preparazione della query arricchita:** Il modulo riceve in input l'analisi in formato JSON prodotta dall'LLM. Questo JSON, con i suoi campi `pattern_name`, `detailed_description` e `technical_keywords`, costituisce una query multi-sfaccettata che viene preparata per alimentare entrambe le modalità di ricerca. Il contenuto testuale viene aggregato per la ricerca semantica, mentre le *keyword* vengono isolate per quella lessicale.

2. **Esecuzione della ricerca parallela e generazione delle classifiche:** Vengono eseguite simultaneamente due interrogazioni alla *knowledge base* CAPEC, ciascuna delle quali produce una lista di candidati ordinata (una classifica o *ranked list*):

   - **Ricerca semantica:** Il contenuto testuale della query viene trasformato in un vettore di *embedding*. Questo vettore viene utilizzato per interrogare l'indice vettoriale,

ottenendo una classifica semantica di N pattern CAPEC candidati, ordinati per similarità coseno decrescente.

- **Ricerca lessicale:** I termini della query vengono utilizzati per interrogare l'indice `TF-IDF`. Il sistema produce una seconda classifica lessicale di N pattern CAPEC candidati, ordinati per punteggio di rilevanza `TF-IDF` decrescente.

3. **Fusione delle classifiche tramite *Reciprocal Rank Fusion*:** A questo punto, il sistema dispone di due classifiche, ciascuna basata su una metrica di rilevanza diversa e non direttamente comparabile. Per superare questa eterogeneità, invece di tentare di combinare i punteggi grezzi, si adotta una tecnica di fusione più robusta: la *Reciprocal Rank Fusion*. L'RRF opera non sui punteggi, ma sulla posizione (*rank*) di ogni documento in ciascuna lista. L'intuizione fondamentale è che un documento che appare in alto in entrambe le classifiche è con maggiore probabilità rilevante.

   Per ogni pattern CAPEC p presente in almeno una delle due liste, viene calcolato un punteggio RRF secondo la seguente formula:

$$\text{RRFScore}(p) = \frac{1}{k + \text{rank}_{\text{sem}}(p)} + \frac{1}{k + \text{rank}_{\text{lex}}(p)} \tag{4.1}$$

   dove $\text{rank}_{\text{sem}}(p)$ e $\text{rank}_{\text{lex}}(p)$ sono le posizioni (*rank*) del pattern p rispettivamente nella classifica semantica e in quella lessicale. Se un pattern non appare in una delle liste, il suo contributo per quella lista è zero (o, più formalmente, il suo *rank* è considerato infinito). La costante k (nel nostro caso impostata a 60) è un parametro di smorzamento che riduce l'impatto dei documenti con un *rank* molto basso, stabilizzando la fusione.

4. **Generazione dell'output finale:** Tutti i pattern CAPEC candidati vengono riordinati in base al $\text{RRFScore}(p)$ decrescente. L'output finale del sistema è una lista classificata (*ranked list*) dei pattern di attacco più pertinenti, che rappresenta la mappatura più probabile per il profilo di comportamento analizzato.

Questo approccio, basato sulla fusione dei *rank*, costituisce il nucleo metodologico della fase di *matching*. Bilanciando implicitamente la comprensione del significato (premiata dalla classifica semantica) con la precisione terminologica (premiata dalla classifica lessicale), il sistema è in grado di produrre risultati più robusti e accurati, mitigando i punti di debolezza di ciascun approccio preso singolarmente, in linea con le pratiche più avanzate dell'*information retrieval*.

4.7 Sintesi dell'architettura proposta

In sintesi, il presente capitolo ha delineato l'architettura concettuale di una pipeline innovativa per l'analisi e il *mapping* di log di sicurezza eterogenei. La soluzione si fonda su una sequenza

di moduli logicamente disaccoppiati, ognuno progettato per affrontare una specifica sfida del problema.

Si è partiti da un modulo di *preprocessing* olistico, che aggrega gli eventi per attore per ricostruire il contesto delle minacce. Successivamente, è stato introdotto il ruolo strategico di un *LLM come interprete upstream*, il cui compito è tradurre i dati grezzi in una query semantica strutturata attraverso un processo di *prompt engineering* avanzato. È stata poi descritta la costruzione di una *knowledge base* CAPEC a doppio indice, ottimizzata per supportare una ricerca sia semantica che lessicale. Infine, è stato presentato il *motore di matching ibrido*, che sfrutta la robustezza della *Reciprocal Rank Fusion* per combinare i due segnali di ricerca in un'unica classifica di rilevanza.

Questa architettura, pur essendo descritta a un livello astratto, fornisce un *blueprint* metodologico completo e rigoroso per affrontare il problema. Avendo definito il "cosa" e il "perché" della soluzione, il capitolo successivo si concentrerà sul "come", descrivendo in dettaglio le scelte tecnologiche, le librerie e i dettagli ingegneristici adottati per realizzare concretamente questa pipeline.

Capitolo 5

Implementazione della soluzione proposta

5.1 Introduzione all'implementazione e alle scelte tecnologiche

Dopo aver delineato l'architettura concettuale della pipeline nel capitolo precedente, il presente capitolo si concentra sulla sua *realizzazione pratica*. L'obiettivo è tradurre i moduli logici e i flussi di dati astratti in un sistema software funzionante, descrivendo in dettaglio le scelte tecnologiche, le librerie e le metodologie ingegneristiche adottate. Questa sezione funge da raccordo tra la teoria e la pratica, illustrando il "come" è stata costruita la soluzione, dal trattamento dei dati grezzi fino alla generazione del *ranking* finale dei pattern CAPEC.

La filosofia implementativa ha seguito due principi guida. In primo luogo, si è privilegiato l'uso di tecnologie *open source*, mature e ampiamente validate dalla comunità scientifica e industriale. Questa scelta non solo garantisce la trasparenza e la *riproducibilità* della ricerca, un requisito fondamentale in ambito accademico, ma permette anche di basarsi su componenti robusti e performanti. In secondo luogo, è stata data priorità a soluzioni che offrissero il miglior compromesso tra *prestazioni computazionali ed efficienza delle risorse*. L'impiego di modelli linguistici con miliardi di parametri, infatti, pone sfide significative in termini di requisiti hardware; pertanto, l'adozione di tecniche come la quantizzazione è stata una scelta strategica per garantire l'esecuzione del sistema su hardware accessibile, senza sacrificare in modo significativo l'accuratezza.

La sfida implementativa principale è stata l'orchestrazione di una pipeline eterogenea, che integra componenti provenienti da diversi domini dell'informatica: dal *parsing* di dati semi-strutturati (log), all'inferenza con modelli di *deep learning* su larga scala, fino a tecniche di *information retrieval* sia dense (vettoriali) che sparse (lessicali).

Nelle sezioni seguenti, verrà fornita una disamina dello stack tecnologico e dell'ambiente di sviluppo che hanno reso possibile tale integrazione. Successivamente, si analizzerà l'implementazione di ciascun modulo logico, in corrispondenza con l'architettura descritta nella Sezione 4.2: la preparazione del dataset sperimentale, la costruzione della *knowledge base*, il *preprocessing* e la contestualizzazione dei log, l'integrazione del modulo LLM e, infine, l'implementazione del

motore di *matching* ibrido.

5.2 Stack tecnologico e ambiente di sviluppo

La realizzazione della pipeline proposta è stata interamente sviluppata in linguaggio `Python` (versione 3.10), scelto per la sua robustezza, la vasta disponibilità di librerie scientifiche e la sua posizione di standard de facto nel campo del *machine learning* e del *Natural Language Processing*. L'ecosistema di librerie *open source* ha permesso di integrare componenti eterogenei in un unico flusso di lavoro coerente. La gestione delle dipendenze è stata affidata al gestore di pacchetti `pip`, con le specifiche versioni delle librerie documentate per garantire la riproducibilità.

Lo stack tecnologico principale può essere suddiviso in quattro aree funzionali:

- **Elaborazione dei dati e preprocessing:**
  - **`Pandas`:** Questa libreria è stata fondamentale per l'intero ciclo di vita dei dati, dalla lettura del dataset CSV iniziale, alla sua pulizia e filtraggio, fino all'aggregazione olistica dei log per attore tramite le sue efficienti operazioni di `groupby`.
  - **Librerie standard (`re`, `json`, `xml`, `binascii`):** Sono state ampiamente utilizzate per compiti di basso livello. Le espressioni regolari (`re`) sono state impiegate per la pulizia dei testi e l'estrazione di pattern (es. nei payload). La libreria `json` è stata cruciale per il *parsing* dell'output strutturato dell'LLM. L'`xml.etree.ElementTree` è stato utilizzato per il *parsing* del file CAPEC, mentre `binascii` è stato indispensabile per la decodifica dei payload esadecimali.
- **Natural language processing:** Per il *preprocessing* linguistico avanzato applicato al corpus CAPEC sono state impiegate due librerie NLP di riferimento:
  - **`NLTK` (Natural Language Toolkit):** Utilizzato per compiti fondamentali come la tokenizzazione e la rimozione di *stopwords*.
  - **`SpaCy`:** Impiegato per analisi linguistiche più profonde, in particolare per la lemmatizzazione e il Part-of-Speech (POS) *tagging*, che hanno permesso di filtrare i token testuali in base alla loro funzione grammaticale.
- **Large language model e architetture Transformer:** Il cuore del modulo di interpretazione è stato costruito attorno all'ecosistema di `Hugging Face`.
  - **`Transformers`:** Questa libreria ha fornito l'accesso al modello `mistralai/Mistral-7B-Instruct-v0.2` e ha orchestrato l'intera pipeline di inferenza.
  - **`PyTorch`:** È stato utilizzato come framework di *deep learning* sottostante per l'esecuzione del modello.

  - **`Bitsandbytes`** e **`Accelerate`:** L'impiego di queste librerie è stato un requisito tecnico fondamentale. Hanno permesso di implementare la strategia di *quantizzazione a 4-bit*, riducendo drasticamente l'impronta di memoria del modello LLM e rendendone possibile l'esecuzione su una singola GPU con VRAM limitata.

- **Information retrieval e machine learning:** Il motore di ricerca ibrido è stato implementato combinando diverse librerie:

  - **`Sentence-Transformers`:** Utilizzata per caricare il modello di *embedding* dominio-specifico `basel/ATTACK-BERT` e per codificare i testi in vettori semantici.
  - **`ChromaDB`:** Scelto come database vettoriale per la sua leggerezza e facilità di integrazione. È stato utilizzato per indicizzare gli *embedding* di CAPEC e per eseguire ricerche di similarità basate su Approximate Nearest Neighbor (ANN) in modo efficiente.
  - **`Scikit-learn`:** Questa libreria fondamentale per il *machine learning* è stata impiegata per implementare la componente di ricerca lessicale, in particolare attraverso `TfidfVectorizer` per la creazione dell'indice e `cosine_similarity` per il calcolo dei punteggi.

Ambiente di sviluppo ed esecuzione. L'intero ciclo di vita del progetto, dallo sviluppo all'esecuzione degli esperimenti, è stato condotto all'interno dell'ambiente basato su cloud **Google Colaboratory**. Questa scelta è stata motivata dalla disponibilità di istanze con accelerazione hardware (GPU, specificamente NVIDIA T4), un requisito indispensabile per gestire il carico computazionale richiesto dall'inferenza dei modelli Transformer e dalla generazione degli *embedding* su larga scala. È importante sottolineare che, sebbene l'ambiente sperimentale utilizzato (Google Colaboratory) sia basato su cloud, l'esecuzione del modello è avvenuta interamente all'interno dell'istanza assegnata, senza alcuna chiamata ad API esterne per l'inferenza. Tale configurazione simula funzionalmente un deployment *on-premise*, validando la capacità dell'architettura di operare in ambienti isolati e garantendo la sovranità dei dati, in linea con gli obiettivi di progettazione.

5.3 Preparazione del dataset sperimentale

La validazione di una pipeline di analisi dei log richiede un dataset rappresentativo delle minacce reali. Come descritto nella Sezione 2.1.2, la piattaforma `T-Pot` offre una soluzione ideale, aggregando dati da un ricco ecosistema di sensori. Per questo studio, i dati sono stati raccolti da un'istanza *live* di tale piattaforma, esposta su Internet per un periodo definito. Questa sezione descrive il processo implementativo di estrazione, strutturazione e selezione dei dati che costituiscono il fondamento per l'addestramento e la valutazione del sistema proposto.

5.3.1 Estrazione unificata dei dati da T-Pot tramite Elasticsearch

L'architettura di `T-Pot` centralizza tutti i log generati dai suoi honeypot in un'istanza di `Elasticsearch`, che funge da *backend* di *storage* e indicizzazione. Questa scelta architetturale permette di interrogare e recuperare i dati in modo programmatico. Per creare il dataset, è stato sviluppato uno script `Python` che, utilizzando la libreria ufficiale `elasticsearch-py`, si connette all'endpoint `Elasticsearch` di `T-Pot` ed esegue un'unica query per estrarre su larga scala tutti i dati di interesse.

Logica di interrogazione e recupero su larga scala. Invece di eseguire interrogazioni separate per ogni tipo di sensore, è stata formulata una singola query generale nel linguaggio *query DSL* di `Elasticsearch` (Listato 1). Questa query è stata deliberatamente progettata con un unico vincolo principale: un *filtro sull'intervallo temporale*, per estrarre tutti i documenti, indipendentemente dal campo *type*, all'interno di una settimana di attività. Questo approccio garantisce la raccolta di un campione completo e temporalmente coerente di tutte le attività registrate dalla piattaforma, senza introdurre *bias* di selezione basati sul tipo di honeypot.

```
query = {
        "query": {
                "bool": {
                        "filter": [{
                                "range": {
                                        "@timestamp": {
                                                "gte": "2025-03-02T23:00:00.000Z",
                                                "lte": "2025-03-09T22:59:59.999Z"
                                        }
                                }
                        }]
                }
        },
        "size": 500,
        "_source": True
}
```

Listato 1: Esempio di query DSL per Elasticsearch utilizzata per l'estrazione massiva dei log

Poiché il numero di documenti da estrarre può ammontare a decine o centinaia di migliaia, superando il limite di una singola risposta, è stata utilizzata l'API *scroll* di `Elasticsearch`. Questa API è stata implementata tramite la funzione di utilità `helpers.scan`, che gestisce in modo astratto la paginazione dei risultati, iterando in modo efficiente su un grande volume di dati e garantendo che tutti i documenti corrispondenti alla query vengano recuperati senza perdite.

5.3.2 Strutturazione del dataset in formato tabellare (CSV)

I documenti recuperati da `Elasticsearch` sono in formato JSON e presentano una struttura gerarchica e nidificata, che varia a seconda dell'honeypot di origine. Per facilitare la manipolazione e l'analisi nelle fasi successive, questi dati sono stati trasformati e consolidati in un unico file in formato **CSV (Comma-Separated Values)**, utilizzando un punto e virgola (';') come delimitatore per garantire la compatibilità con software che potrebbero avere problemi con la virgola all'interno dei campi di testo.

Questa trasformazione è stata eseguita utilizzando la funzione `json_normalize` della libreria `Pandas`. Questa funzione è particolarmente adatta a questo compito, in quanto è in grado di linearizzare (*flatten*) una struttura JSON complessa. Durante questo processo, i campi nidificati (es. un oggetto JSON all'interno di un altro) vengono trasformati in colonne separate con nomi composti, utilizzando il punto come separatore (es. il campo JSON "`payload`": {"`data_hex`": "`...`"} diventa una colonna CSV denominata `payload.data_hex`).

Il risultato di questo processo è un unico file CSV, `All_Data.csv`, che funge da *data lake* grezzo per l'intera pipeline. Questa struttura tabellare è l'input diretto per il **Modulo di preprocessing e contestualizzazione dei dati**, il quale, in fase di caricamento, si occuperà di filtrare, pulire e infine raggruppare i dati. La scelta di creare un unico file CSV intermedio garantisce portabilità, facilità di ispezione e semplifica la pipeline, demandando la logica di filtraggio e aggregazione al modulo successivo, che opera in memoria.

5.3.3 Struttura, analisi e selezione dei dati del dataset

Il processo di estrazione e normalizzazione descritto in precedenza produce un unico file CSV che, pur essendo strutturato in formato tabellare, presenta un'elevata dimensionalità e complessità a causa dell'eterogeneità delle fonti dati. Un'analisi critica della sua struttura è un prerequisito fondamentale per una corretta implementazione della pipeline.

Analisi della struttura del dataset. Il file CSV risultante contiene diverse decine di colonne, generate dall'appiattimento dei documenti JSON di `Elasticsearch`. Queste colonne possono essere raggruppate in cinque categorie tematiche principali:

1. **Campi di identificazione e contesto di base:** Costituiscono il nucleo informativo comune a quasi tutti i log e sono essenziali per l'aggregazione e il filtraggio. Includono `@timestamp`, `type` (il tipo di honeypot), `src_ip`, `dest_ip`, `src_port`, e `dest_port`.

2. **Campi descrittivi dell'evento specifico:** Contengono i dati più ricchi sul comportamento dell'attaccante e variano significativamente in base all'honeypot. Esempi includono `eventid` (per `Cowrie`), `message`, `payload`, `payload_printable`, e campi specifici dei protocolli come `sip_method` (per `SentryPeer`) o `ftp.commands.command` (per `Dionaea`).

3. **Campi di arricchimento geografico e di rete:** Dati aggiunti da `T-Pot` durante l'ingestione dei log per fornire un contesto sull'origine dell'attacco. Questa categoria include `geoip.city_name`, `geoip.country_name`, `geoip.latitude`, `geoip.longitude`, `geoip.region_name`, e informazioni sull'AS (Autonomous System) come `geoip.as_org`.

4. **Metadati dell'infrastruttura di logging:** Un vasto insieme di campi aggiunti dall'`Elastic Stack` (Filebeat, Logstash) che descrivono l'ambiente di raccolta e non l'evento stesso. Esempi tipici sono `agent.*` (es. `agent.hostname`), `ecs.version`, `host.*`, `log.*` (es. `log.file.path`), e `tags`.

5. **Dettagli di basso livello del protocollo:** Campi estremamente granulari che descrivono aspetti specifici della connessione, come la suite di algoritmi crittografici utilizzati. Un esempio è la serie di campi `ssh.hassh.*` per le connessioni SSH.

Principio di selezione: focus sull'azione dell'attaccante. Non tutti questi campi sono utili per l'obiettivo di questa tesi, ovvero interpretare l'*intento* e il *comportamento* dell'attaccante per mapparlo a un pattern CAPEC. Un passo metodologico fondamentale, quindi, è la *selezione delle feature*. Il principio guida è stato quello di scartare i campi che descrivono l'infrastruttura di monitoraggio (l'osservatore) o dettagli irrilevanti per l'astrazione del comportamento, per focalizzarsi esclusivamente sulle colonne che contengono informazioni dirette sulle azioni e sugli obiettivi dell'attaccante.

In base a questo principio, il modulo di preprocessing e contestualizzazione è stato progettato per utilizzare selettivamente solo un sottoinsieme mirato di colonne, ignorando deliberatamente le altre.

Giustificazione dei dati esclusi. La decisione di escludere determinate categorie di dati è stata ponderata per ridurre il rumore e focalizzare l'analisi dell'LLM sulle informazioni più salienti.

- **Esclusione dei metadati di logging:** Tutti i campi aggiunti dall'`Elastic Stack` (es. `agent.*`, `host.*`, `log.file.path`) sono stati ignorati, in quanto descrivono l'infrastruttura di raccolta e non l'evento stesso. Includerli nel prompt avrebbe introdotto un rumore significativo, rischiando di distrarre il modello dall'analisi del comportamento.

- **Esclusione dei dati geografici e di rete:** Questa è una scelta metodologica chiave. Sebbene la provenienza geografica di un attacco (`geoip.*`) sia utile per la *threat intelligence* geopolitica, è irrilevante per la classificazione del pattern di attacco (TTP) in sé. Un attacco di "SQL Injection" rimane tale indipendentemente dalla sua origine. L'obiettivo è mappare il *comportamento tecnico*, non il profilo dell'attore. Includere tali informazioni avrebbe potuto introdurre *bias* nel modello e non aggiunge valore per la mappatura a CAPEC. Anche `src_ip` stesso, pur essendo usato per l'aggregazione, viene omesso dalla descrizione finale fornita all'LLM per favorire la generalizzazione.

- **Esclusione di campi ridondanti o a scarso contenuto informativo:** Campi come `dest_ip` (sempre l'IP dell'honeypot) e `src_port` (tipicamente effimera) sono stati scartati in quanto non forniscono informazioni variabili o significative per l'astrazione del comportamento.

Questa selezione mirata delle *feature*, implementata implicitamente nel modulo di preprocessing che legge selettivamente solo le colonne di interesse, è un passaggio fondamentale. Permette di focalizzare l'analisi sul "segnale" del comportamento dell'attaccante, riducendo il "rumore" dei metadati infrastrutturali, e costituisce la base per la successiva fase di aggregazione olistica.

5.4 Implementazione della *knowledge base* CAPEC

La costruzione della *knowledge base* è il primo passo fondamentale della pipeline implementativa, poiché crea la base di conoscenza strutturata contro cui verranno confrontate le analisi dei log. Questo processo traduce il corpus grezzo di MITRE CAPEC, fornito in formato XML, nei due indici paralleli (semantico e lessicale) descritti nell'architettura. L'implementazione è stata suddivisa in due fasi logiche: l'estrazione e la rappresentazione strutturata dei pattern di attacco, e il successivo preprocessing linguistico e indicizzazione.

5.4.1 Parsing del corpus XML e rappresentazione strutturata dei dati

Il punto di partenza è il file `CAPEC.xml`, specificamente la versione 3.9 dello standard, un documento complesso con una struttura gerarchica definita da *namespace* XML. Per gestire questa complessità, è stata implementata una classe `Python`, `CAPECPattern`, che agisce come un *parser* e un modello dati, trasformando ogni elemento `<Attack_Pattern>` del file XML in un oggetto strutturato e facilmente manipolabile.

Struttura del file XML. Per comprendere la sfida del *parsing*, è utile osservare la struttura di un'*entry* CAPEC. Come mostrato nel Listato 2, ogni pattern è un nodo XML complesso che contiene sia metadati (es. `ID`, `Name`), sia campi testuali semplici (es. `Likelihood_Of_Attack`), sia nodi strutturati che possono contenere *markup* XHTML (es. `<Description>`, `<Mitigations>`).

Implementazione del parser tramite la classe `CAPECPattern`. L'estrazione dei dati avviene tramite la libreria standard `xml.etree.ElementTree`. La classe `CAPECPattern` viene inizializzata con un elemento XML (`ET.Element`) che rappresenta un singolo pattern di attacco.

Un aspetto implementativo cruciale è la gestione dei *namespace* XML. Un dizionario `NS` viene definito a livello globale per mappare i prefissi ('capec', 'xhtml') ai loro URI. Questo dizionario viene poi passato a tutti i metodi `find` e `findall`, permettendo di eseguire query XPath corrette per navigare l'albero XML. Il frammento di codice riportato nel Listato 3 mostra l'inizializzazione della classe e l'estrazione degli attributi di base.

```
<Attack_Pattern ID="1" Name="Accessing Functionality Not Properly Constrained by ACLs" Abstraction="Standard"
↪ Status="Draft">
<Description>In applications, particularly web applications, access to functionality is mitigated by an
↪ authorization framework...</Description>
<Likelihood_Of_Attack>High</Likelihood_Of_Attack>
<Execution_Flow>
<Attack_Step>
<Step>1</Step>
<Phase>Explore</Phase>
<Description>[Survey] The attacker surveys the target application...</Description>
</Attack_Step>
...
</Execution_Flow>
<Mitigations>
<Mitigation>
<xhtml:p>In a J2EE setting, administrators can associate a role that is impossible for the authenticator to grant
↪ users...</xhtml:p>
</Mitigation>
</Mitigations>
...
</Attack_Pattern>
```

Listato 2: Estratto semplificato della struttura di un'entry `Attack_Pattern` nel file CAPEC.xml

```python
# Definizione dei namespace per le query XPath
NS = {'capec': 'http://capec.mitre.org/capec-3', 'xhtml': 'http://www.w3.org/1999/xhtml'}

class CAPECPattern:
def __init__(self, pattern_xml: ET.Element):
self.id = pattern_xml.get('ID')
self.name = pattern_xml.get('Name')
self.description = self._parse_description(pattern_xml)
self.execution_flow = self._parse_execution_flow(pattern_xml)
# ... estrazione di altri campi come Prerequisites, Mitigations, etc.
```

Listato 3: Inizializzazione della classe `CAPECPattern` e gestione dei namespace XML

Gestione di contenuti testuali complessi. I campi testuali più ricchi di CAPEC, come la descrizione, non sono semplici stringhe ma nodi che possono contenere *markup* XHTML (es. tag <p>). Per gestire questa eterogeneità, è stata sviluppata una funzione di utilità riutilizzabile, `_parse_text_from_complex_type`, che astrae la logica di estrazione del testo. Questa funzione è in grado di aggregare il contenuto testuale sia dal testo diretto di un nodo sia da tutti i sotto-elementi XHTML presenti, garantendo che nessuna informazione venga persa. È importante notare che l'output di questa funzione viene immediatamente passato alla pipeline di preprocessing linguistico `full_preprocess` (descritta nella prossima sottosezione), come mostra il Listato 4.

```python
def _parse_text_from_complex_type(self, element: Optional[ET.Element]) -> str:
texts = []
if element is not None:
if element.text:
texts.append(element.text.strip())
# Cerca ricorsivamente tutti i paragrafi XHTML al di sotto del nodo
for p in element.findall('.//xhtml:p', NS):
if p.text:
texts.append(p.text.strip())
# Il testo aggregato viene subito normalizzato dalla pipeline NLP
return full_preprocess(' '.join(filter(None, texts)))

# Esempio di utilizzo all'interno della classe
def _parse_execution_flow(self, pattern: ET.Element) -> List[str]:
steps = []
flow_elem = pattern.find('capec:Execution_Flow', NS)
if flow_elem is not None:
for step in flow_elem.findall('.//capec:Attack_Step', NS):
processed_text = self._parse_text_from_complex_type(step.find('capec:Description', NS))
if processed_text:
steps.append(processed_text)
return steps
```

Listato 4: Funzione di utilità per l'estrazione di testo da nodi XML complessi con markup XHTML

Creazione del contesto semantico unificato. Al termine del *parsing*, ogni oggetto `CAPEC Pattern` contiene i dati strutturati e pre-elaborati del pattern. Per preparare i dati per l'indicizzazione, la classe espone un metodo `semantic_context()`. Questo metodo aggrega tutti i campi testuali rilevanti (nome, descrizione, flusso di esecuzione, prerequisiti, ecc.) in un unico documento aggregato testuale. Sarà questo documento aggregato a rappresentare l'intero pattern di attacco e a servire come base per la successiva fase di generazione degli *embedding* e di calcolo dei pesi `TF-IDF`.

5.4.2 Preprocessing linguistico avanzato del corpus CAPEC

Come anticipato, ogni frammento di testo estratto dal corpus CAPEC viene immediatamente processato da una pipeline di preprocessing linguistico, implementata nella funzione `full_preprocess`. A differenza del preprocessing applicato ai log, questo processo è interamente basato su tecniche di natural language processing e mira a normalizzare e arricchire il testo per ottimizzare sia la ricerca semantica che quella lessicale.

Pipeline di normalizzazione e arricchimento. La pipeline è composta da una sequenza di funzioni, ognuna con uno scopo specifico, applicate in ordine strategico:

1. **Espansione di acronimi e sinonimi (`expand_acronyms`, `expand_technical_terms`):** Il testo viene prima arricchito per rendere esplicite le relazioni concettuali. Un dizionario personalizzato, `ACRONYM_MAP`, viene utilizzato per espandere acronimi comuni tramite espressioni regolari (es. "SQLi" → "sql injection"). Successivamente, una mappa di sinonimi tecnici, `TECHNICAL_SYNONYMS`, normalizza termini semanticamente simili (es. "breach" → "compromise"). Questo passaggio riduce la variabilità lessicale e aumenta la densità semantica del testo, a beneficio di entrambi i motori di ricerca.

2. **Normalizzazione testuale di base (`preprocess_text`):** Vengono applicate operazioni di pulizia standard, come la conversione in minuscolo e la rimozione di entità che introducono rumore, quali indirizzi IP e numeri di versione, identificate tramite espressioni regolari. Tutti i numeri vengono sostituiti con un token generico ("number") per astrarne il valore specifico.

3. **Rimozione di termini generici (`remove_unwanted_terms`):** Per aumentare il rapporto segnale/rumore, vengono eliminate non solo le *stopwords* comuni della lingua inglese, ma anche una lista curata di termini generici del dominio della sicurezza (es. "attack", "pattern", "vulnerability"). Questo passaggio è vitale per l'indice `TF-IDF`, poiché impedisce che termini frequenti ma poco informativi dominino i punteggi di rilevanza.

Tokenizzazione avanzata e generazione di n-grammi. L'output di `full_preprocess` è una stringa di testo pulita, che serve come input per la fase di tokenizzazione avanzata, gestita dalla funzione `get_enhanced_tokens` (Listato 5). Questa funzione è uno dei pilastri del preprocessing, in quanto non si limita a dividere il testo in parole, ma esegue un'analisi linguistica profonda utilizzando la libreria `SpaCy`.

Questa funzione implementa una logica sofisticata:

- Utilizza il *Part-of-Speech (POS) tagging* di `SpaCy` per filtrare e mantenere solo le parole con maggiore contenuto semantico (nomi, verbi, aggettivi, nomi propri), scartando articoli, preposizioni e altre parti del discorso meno significative che potrebbero non essere state catturate dalla lista di *stopwords*.

```python
def get_enhanced_tokens(text: str) -> List[str]:
if not isinstance(text, str): return []
# Utilizzo di SpaCy per un'analisi linguistica accurata
doc = nlp(text)
# 1. Filtro per POS, lemmatizzazione e rimozione di stopwords/punteggiatura
tokens = [token.lemma_.lower() for token in doc if
not token.is_stop and not token.is_punct and
token.pos_ in ['NOUN', 'VERB', 'ADJ', 'PROPN'] and
len(token.lemma_) > 2]
# 2. Generazione di n-grammi per catturare espressioni composte
bigrams = [' '.join(tokens[i:i + 2]) for i in range(len(tokens) - 1)]
trigrams = [' '.join(tokens[i:i + 3]) for i in range(len(tokens) - 2)]

return [t for t in tokens + bigrams + trigrams if t and t.strip()]
```

Listato 5: Funzione di tokenizzazione avanzata con lemmatizzazione e generazione di n-grammi

- Applica la *lemmatizzazione* per ridurre ogni parola alla sua forma base (es. "attacks", "attacking" diventano "attack"), garantendo che le varianti morfologiche di un termine siano trattate come un unico concetto.
- Genera *bi-grammi e tri-grammi* (es. "cross site scripting") a partire dalla lista di token filtrati. Questo è fondamentale per permettere al sistema di trattare espressioni composte come un singolo token, aumentando drasticamente la precisione del *matching* sia lessicale che semantico.

5.4.3 Implementazione dell'indicizzazione parallela

Il testo preprocessato e tokenizzato di ogni pattern CAPEC viene utilizzato per popolare i due indici paralleli. L'intera logica di indicizzazione è incapsulata nella classe `VectorDBManager` per garantire la modularità.

Costruzione dell'indice lessicale (TF-IDF). L'indice lessicale viene costruito utilizzando la classe `TfidfVectorizer` della libreria `Scikit-learn` come illustrato nel Listato 6. Un aspetto implementativo cruciale è che il vettorizzatore viene configurato per utilizzare direttamente la funzione `get_enhanced_tokens` come 'tokenizer' personalizzato, e `full_preprocess` come 'preprocessor'. Questo garantisce che la matrice TF-IDF sia costruita sulla stessa identica base linguistica (testo pulito, lemmi, n-grammi) utilizzata nel resto del sistema.

I parametri `max_df=0.85` e `min_df=2` sono stati impostati per escludere termini che appaiono in più dell'85% dei documenti (troppo comuni) o in meno di 2 documenti (troppo rari o potenziali errori), agendo come un filtro statistico sul rumore.

```
self.tfidf_vectorizer = TfidfVectorizer(
tokenizer=get_enhanced_tokens,
preprocessor=full_preprocess,
max_df=0.85,
min_df=2,
ngram_range=(1, 3),
stop_words='english'
)
# Creazione della matrice TF-IDF
self.tfidf_matrix = self.tfidf_vectorizer.fit_transform(contexts)
```

Listato 6: Configurazione e creazione dell'indice lessicale TF-IDF

Costruzione dell'indice vettoriale (semantico). L'indice vettoriale viene costruito in due passaggi. In primo luogo, viene caricato il modello di *embedding* dominio-specifico `basel/ATTACK-BERT` tramite la libreria `Sentence-Transformers`. Questo modello viene quindi utilizzato per codificare l'intero corpus dei contesti semantici di CAPEC. La procedura di generazione è dettagliata nel Listato 7.

```
# Caricamento del modello di embedding
self.embedder = SentenceTransformer(EMBEDDING_MODEL)

# Codifica dell'intero corpus in batch per efficienza
all_embeddings = self.embedder.encode(
contexts,
batch_size=32,
show_progress_bar=True,
normalize_embeddings=True
)
```

Listato 7: Generazione degli *embedding* semantici per il corpus CAPEC

L'opzione `normalize_embeddings=True` è importante in quanto normalizza tutti i vettori a lunghezza unitaria, rendendoli ottimali per il calcolo della similarità coseno.

Successivamente, questi *embedding*, insieme ai metadati associati a ogni pattern (estratti tramite il metodo `.to_metadata()` della classe `CAPECPattern`), vengono memorizzati in una collezione persistente utilizzando `ChromaDB`. La collezione viene configurata per utilizzare lo spazio di similarità "cosine", garantendo che le ricerche ANN siano allineate con la metrica di valutazione. L'aggiunta dei dati a ChromaDB avviene in batch per ottimizzare le prestazioni di scrittura sul disco.

5.5 Implementazione del modulo di preprocessing e contestualizzazione dei dati

Come descritto nell'architettura (Sezione 4.3), il primo passo operativo della pipeline è la trasformazione dei dati grezzi in profili di comportamento contestualizzati. Questa sezione descrive i dettagli implementativi di tale processo, che si articola in due fasi principali: una fase di caricamento e aggregazione globale, seguita da una fase di sintesi e arricchimento specifico per ogni profilo.

5.5.1 Caricamento, pulizia e aggregazione olistica dei log

Il processo inizia con il caricamento del dataset di log degli honeypot dal file CSV descritto nella sezione 5.3. Per questa operazione viene utilizzata la libreria `Pandas`, che carica l'intero dataset in un DataFrame per una manipolazione efficiente.

Pulizia e filtraggio preliminare del dataset. Una volta caricato, il DataFrame viene sottoposto a una serie di operazioni di pulizia e filtraggio per garantire la qualità e la pertinenza dei dati:

- **Gestione del timestamp:** Viene assicurata la presenza e la coerenza della colonna del timestamp, rinominando `@timestamp` in `timestamp` se necessario e convertendo i valori in un formato datetime standard. Questo è un prerequisito per l'ordinamento cronologico degli eventi.
- **Filtraggio per tipo di honeypot:** Il dataset viene filtrato per mantenere esclusivamente le righe corrispondenti ai tipi di honeypot supportati dalla pipeline (`Cowrie`, `Honeytrap`, `Dionaea`, `Sentrypeer`, `CiscoASA`).
- **Rimozione del rumore:** Vengono rimosse le righe contenenti messaggi di log che corrispondono a pattern di rumore noti (es. "Traceback", "Exception occurred"), i quali non rappresentano attività di attacco ma errori interni o eventi di sistema irrilevanti.
- **Validazione dei dati essenziali:** Vengono eliminate le righe che presentano valori nulli in colonne fondamentali come `type`, `timestamp` e `src_ip`, che sono indispensabili per l'aggregazione e l'analisi.

Il Listato 8 illustra la logica di caricamento e filtraggio implementata tramite Pandas.

Aggregazione olistica per attore. Questo è il passaggio metodologicamente più importante della fase di preprocessing, che sposta il focus dall'evento singolo all'attore. L'implementazione utilizza il metodo `groupby()` di Pandas per partizionare il DataFrame e aggregare tutti i log correlati in "profili di comportamento". La chiave di aggregazione viene scelta in modo adattivo:

```
# Caricamento del dataset con Pandas
df = pd.read_csv(HONEYPOT_CSV_FILE, delimiter=';', ...)

# Filtro per i tipi di honeypot supportati
allowed_types = ['Cowrie', 'Honeytrap', 'Dionaea', 'Sentrypeer', 'Ciscoasa']
df_filtered = df[df['type'].isin(allowed_types)].copy()

# Rimozione di righe contenenti pattern di rumore noti
noise_patterns = ['Traceback', 'Exception occurred', ...]
for pattern in noise_patterns:
df_filtered = df_filtered[~df_filtered['message'].str.contains(pattern, na=False)]

# Eliminazione di righe con valori essenziali mancanti
base_required_cols = ['type', 'timestamp', 'src_ip']
df_filtered.dropna(subset=[c for c in base_required_cols if c in df_filtered.columns],
↪  inplace=True)
```

Listato 8: Pipeline di caricamento, pulizia e filtraggio dei log grezzi

- Per i log di `Cowrie`, si utilizza l'identificativo di sessione (`session_id`) fornito dall'honeypot, che garantisce un'aggregazione precisa delle sessioni interattive.
- Per tutti gli altri tipi di honeypot, si utilizza l'indirizzo IP sorgente dell'attaccante (`src_ip`) come chiave di raggruppamento. Questa scelta permette di consolidare tutte le attività (es. scansioni su porte diverse, tentativi di connessione a servizi multipli) provenienti da un singolo attore in un unico contesto di analisi, rappresentativo della sua "campagna" di ricognizione o attacco.

Il risultato di questa operazione è un dizionario Python (`defaultdict(list)`), in cui ogni chiave è un identificativo univoco dell'attore/sessione e ogni valore è una lista di dizionari. Ciascun dizionario interno rappresenta una riga di log originale, e l'intera lista viene ordinata cronologicamente per `timestamp`, preservando la sequenza temporale delle azioni. L'algoritmo di aggregazione è riportato nel Listato 9.

Questa struttura dati aggregata, che trasforma i dati da una visione incentrata sull'evento a una visione incentrata sull'attore, è l'input fondamentale per la fase successiva di sintesi e arricchimento, che opererà su ciascun profilo di comportamento individualmente.

5.5.2 Implementazione della sintesi e arricchimento dei dati

Una volta che i profili di comportamento sono stati aggregati come descritto nella Sezione 4.3.1, la funzione `generate_llm_input` viene invocata per ciascuno di essi. Questa funzione contiene la logica di sintesi e arricchimento, che è altamente specializzata per ogni tipo di honeypot, al fine di trasformare una lista di eventi di log in una singola descrizione testuale olistica,

```
sessions_to_process = defaultdict(list)

# Esempio di raggruppamento per le sessioni Cowrie
cowrie_df = df_filtered[df_filtered['type'] == 'Cowrie' and df_filtered['session'].notna()]
for session_id, group in cowrie_df.groupby('session'):
sessions_to_process[session_id] = group.sort_values(by='timestamp').to_dict('records')

# Logica simile viene applicata per altri tipi di honeypot usando 'src_ip'
honeytrap_df = df_filtered[df_filtered['type'] == 'Honeytrap']
for src_ip, group in honeytrap_df.groupby('src_ip'):
session_id = f"honeytrap_{src_ip}"
sessions_to_process[session_id] = group.sort_values(by='timestamp').to_dict('records')
```

Listato 9: Logica di aggregazione olistica dei log per sessione o indirizzo IP

ottimizzata per l'analisi da parte dell'LLM. Data la complessità e la ricchezza delle informazioni, l'implementazione per le sessioni `Cowrie` è la più articolata e merita una trattazione dettagliata.

Implementazione della sintesi per le sessioni Cowrie (SSH/Telnet). Le sessioni `Cowrie` registrano interazioni complesse che richiedono una ricostruzione narrativa per essere comprese. L'implementazione itera su tutti gli eventi della sessione (una lista di dizionari Python) per estrarre e aggregare diverse categorie di attività in modo strutturato prima di assemblarle in una descrizione finale.

Il primo passo è l'estrazione di metadati e attività di alto livello. L'algoritmo analizza la lista di eventi per identificare e collezionare:

- **Parametri di connessione:** Vengono estratti le porte di destinazione, i protocolli e le architetture emulate (dal `eventid: cowrie.session.params`), che forniscono il contesto di base della sessione.
- **Stato dell'autenticazione:** Vengono isolati tutti gli eventi di login (`cowrie.login.success` e `cowrie.login.failed`) per determinare se l'accesso è avvenuto, con quale utente, e dopo quanti tentativi falliti.
- **Attività di rete e file system:** Vengono identificate e collezionate in liste separate tutte le attività specifiche come il download di file (`cowrie.session.file_download`), l'upload di file (`cowrie.session.file_upload`) e i tentativi di *tunneling* (`cowrie.direct-tcpip.request`). Per ciascuna, vengono estratte le informazioni chiave come URL, percorsi dei file e destinazioni dei tunnel.

Successivamente, viene implementata la logica più complessa del modulo: il **riepilogo euristico della sequenza di comandi**. Poiché una sessione può contenere centinaia o migliaia di comandi, inviarli tutti all'LLM sarebbe inefficiente e supererebbe il limite di contesto del modello.

Per superare questo limite, è stata sviluppata un'euristica di riepilogo che, invece di un semplice troncamento, mira a preservare le informazioni più salienti, come illustrato nel Listato 10.

```
# Estrazione di tutti i comandi (riusciti e falliti) dalla sessione
commands_raw = [e.get('input', '') for e in session_entries
if e.get('eventid') in ['cowrie.command.input', 'cowrie.command.failed']
and pd.notna(e.get('input'))]
commands = [cmd.strip() for cmd in commands_raw if cmd.strip()]

if commands:
MAX_CMDS_SUMMARY = 15
if len(commands) <= MAX_CMDS_SUMMARY:
# Per sessioni brevi, vengono mostrati tutti i comandi in sequenza
summary_commands = {"Commands sequence": commands}
else:
# Per sessioni lunghe, si applica l'euristica di riepilogo
initial_cmds = commands[:5]   # Primi 5 comandi (fase di staging/ricognizione iniziale)
final_cmds = commands[-5:]     # Ultimi 5 comandi (fase di esecuzione/pulizia)
middle_cmds = commands[5:-5]

# Identificazione dei comandi "notevoli" nella parte centrale della sessione
counts = defaultdict(int)
for cmd in middle_cmds: counts[cmd] += 1
unique_cmds = [cmd for cmd in middle_cmds if counts[cmd] < 3]
complex_pattern = re.compile(r"[|;&<>]|`|\$\(") # Pattern per comandi complessi
complex_cmds = [cmd for cmd in middle_cmds if complex_pattern.search(cmd)]

# Unione dei comandi unici e complessi, preservando l'ordine di apparizione
noteworthy_cmds = list(dict.fromkeys(unique_cmds + complex_cmds))

summary_commands = {
        "Initial Commands": initial_cmds,
        "Noteworthy Commands from Middle": noteworthy_cmds[:5],
        "Final Commands": final_cmds
}
detailed_activity += f" Command Summary: {json.dumps(summary_commands)}."
```

Listato 10: Logica di riepilogo euristico per sequenze di comandi lunghe in sessioni Cowrie

Questa logica si basa sull'ipotesi che i comandi più importanti di una lunga sessione automatizzata siano spesso quelli all'inizio (*staging* e ricognizione), alla fine (esecuzione del *payload* finale o pulizia delle tracce), e quelli più rari o sintatticamente complessi (es. con *piping*, *subshell*) nel mezzo. Questo approccio euristico al riepilogo, a differenza di un semplice troncamento, è progettato per preservare il "segnale" informativo essenziale per l'analisi dell'LLM.

Infine, tutte queste informazioni aggregate vengono assemblate in una descrizione narrativa in linguaggio naturale. La funzione costruisce una serie di frasi che descrivono progressivamente la sessione: inizia con i dettagli della connessione, poi descrive l'esito dell'autenticazione, e infine elenca tutte le attività significative osservate in una sezione di "Specific observations", formattando i riepiloghi complessi (come quello dei comandi) in formato JSON per una maggiore chiarezza. Questo approccio trasforma una lista disordinata di eventi JSON in un "rapporto di intelligence" conciso e leggibile, pronto per essere interpretato dall'LLM.

Sintesi dei profili di rete e servizio (Honeytrap, SentryPeer, Dionaea, CiscoASA). A differenza delle sessioni interattive, i profili di comportamento aggregati per gli honeypot non interattivi consistono in una collezione di eventi atomici (spesso, tentativi di connessione) provenienti da un unico `src_ip`. In questo caso, l'obiettivo della sintesi non è ricostruire una sequenza di azioni, ma identificare i **pattern di comportamento** a partire dall'analisi aggregata degli eventi.

L'implementazione itera su tutti gli eventi del profilo per estrarre e aggregare in collezioni ("set" o "list") le informazioni univoche, come mostrato nel Listato 11.

```
# Logica concettuale per la sintesi di profili di rete/servizio
def generate_llm_input(session_entries: List[Dict], honeypot_type: str) -> str:
# ... (logica per Cowrie) ...
elif honeypot_type == 'Honeytrap':
targeted_ports, protocols_used, payload_hints = set(), set(), set()

for entry in session_entries:
targeted_ports.add(entry.get('dest_port'))
# ... (estrazione di altre informazioni aggregate) ...
if payload_len > 0 and payload_hex:
# ... (invocazione della logica di analisi del payload) ...
payload_hints.add(inferred_protocol)

# Assemblaggio della descrizione finale basata sulle collezioni aggregate
desc_parts.append(f"Observed {len(session_entries)} connection attempts...")
desc_parts.append(f"Targeting {len(targeted_ports)} distinct port(s): {ports_str}.")
if payload_hints:
detailed_activity += f" Inferred Payload Types: {json.dumps(list(payload_hints))}."
# ...
return " ".join(desc_parts)
```

Listato 11: Logica di sintesi aggregata per profili di rete e servizio (es. Honeytrap)

La logica di sintesi viene poi ulteriormente specializzata con euristiche specifiche per ogni tipo di honeypot:

- **`Honeytrap` (Scansioni di rete):** L'analisi si focalizza sul delineare il profilo di ricognizione, aggregando porte e protocolli unici. Per ogni evento con *payload*, viene invocata la funzione `decode_payload` e si applicano euristiche per identificare il tipo di traffico (es. TLS, HTTP), arricchendo il riepilogo con una lista di "Inferred Payload Types". Questo permette di distinguere una semplice scansione di porte da una più sofisticata enumerazione di servizi.

- **`Dionaea` (Emulazione di servizi):** Similmente, vengono aggregati i servizi unici contattati. La logica implementata dà priorità all'estrazione di tentativi di autenticazione (username/password), considerandoli un segnale di intento più forte rispetto a una semplice connessione. Nel caso di interazioni FTP, le sequenze di comandi vengono estratte tramite la funzione `safe_parse_list_str` e riassunte, per distinguere un'interazione approfondita da un semplice *probe*.

- **`SentryPeer` (VoIP):** L'implementazione aggrega i metodi SIP e gli User-Agent. Una delle euristiche più significative analizza la lista di tutti i numeri chiamati nel profilo. Il codice verifica se i numeri sono sequenziali (es. 1001, 1002, 1003) o se seguono un pattern numerico. In caso positivo, il sistema inferisce un pattern di attacco specifico come "*dial plan scanning*" e aggiunge questa interpretazione di alto livello (`pattern_hint`) alla descrizione, un arricchimento che non sarebbe presente nel log grezzo.

- **`CiscoASA` (Web):** Per questo honeypot, l'analisi si concentra sull'estrazione aggregata di richieste HTTP. Utilizzando espressioni regolari, il codice parsifica i *payload* testuali di tutti gli eventi nel profilo per identificare e collezionare gli URL unici richiesti (metodo e percorso). Questa lista di *endpoint* unici è spesso l'indicatore più forte dell'intento dell'attaccante, rivelando se sta cercando vulnerabilità note o esplorando un'API.

Implementazione della decodifica e pulizia dei payload. Un componente fondamentale, riutilizzato in diverse logiche di sintesi, è la funzione `decode_payload`. Questa funzione implementa una catena di tentativi robusta per convertire una stringa esadecimale, spesso rumorosa, in testo leggibile. La logica di decodifica è mostrata nel Listato 12.

Questa funzione è un esempio di ingegneria robusta. La sua logica implementa una catena di *fallbacks* (da UTF-8 a Latin-1), classifica il tipo di contenuto basandosi sulla distribuzione statistica dei byte (euristica dei caratteri non stampabili), e applica espressioni regolari per pulire l'output da artefatti comuni come sequenze di caratteri ripetuti, garantendo che lo snippet di *payload* fornito all'LLM sia il più informativo e pulito possibile.

5.5.3 Dall'evento grezzo alla descrizione semantica: un esempio concreto

Per illustrare concretamente il processo di trasformazione e il valore della contestualizzazione olistica, consideriamo un esempio basato su una sessione reale dell'honeypot `Cowrie`. Nel dataset

```python
def decode_payload(hex_payload: Optional[str]) -> Tuple[str, str]:
# ... (gestione di input vuoti o hash noti di payload vuoti) ...
try:
# Pulizia preliminare della stringa esadecimale
hex_payload_clean = re.sub(r'[^0-9a-fA-F]', '', hex_payload_str)
decoded_bytes = binascii.unhexlify(hex_payload_clean)

# Tentativo di decodifica primario con UTF-8
try:
text = decoded_bytes.decode('utf-8', errors='strict')
content_type = "text"
# Euristica: se troppi caratteri non sono stampabili, classifica come binario
non_printable = sum(1 for byte in decoded_bytes if not (32 <= byte <= 126 or byte in [9, 10,
↪  13]))
if non_printable / len(decoded_bytes) > 0.3:
content_type = "text_mixed_or_binary"
# Fallback a Latin-1 in caso di errore di decodifica
except UnicodeDecodeError:
text = decoded_bytes.decode('latin-1')
content_type = "text_mixed"

# Pulizia finale del testo decodificato
clean_text = ''.join(filter(lambda x: 32 <= ord(x) <= 126 or x in '\n\r\t', text))
clean_text = re.sub(r'(.)\1{20,}', r'\1' * 10 + '...', clean_text) # Rimuove caratteri
↪  ripetuti
return clean_text[:250], content_type
except Exception as e:
return f"[Decode Error: {e}]", "error"
```

Listato 12: Funzione di utilità per la decodifica robusta e la pulizia dei payload esadecimali

grezzo, questa singola campagna di attacco, identificata dalla `session_id` `1183fdaf131c`, è frammentata in numerosi eventi atomici, ognuno registrato come una riga separata. La Tabella 5.1 mostra un estratto semplificato di questi eventi, che rappresentano solo una frazione del "rumore" e del "segnale" presenti nei dati grezzi.

Tabella 5.1: Estratto semplificato degli eventi grezzi per la sessione Cowrie `1183fdaf131c`

Timestamp	**Event ID**	**Input**	**Messaggio**
2025-03-07 T12:56:06.111Z	cowrie.session.connect	-	New connection: 122.117.121.3:33276
2025-03-07 T12:56:09.213Z	cowrie.login.failed	-	Login attempt [supper/PhilG3t] failed
2025-03-07 T12:56:10.150Z	cowrie.session.params	linux-tilegx-lsb	Client architecture detected
2025-03-07 T12:56:10.963Z	cowrie.login.success	-	Login attempt [root/adminpass] succeeded
2025-03-07 T12:56:12.492Z	cowrie.command.input	`sh`	CMD: sh
2025-03-07 T12:56:14.912Z	cowrie.command.input	`enable`	CMD: enable
2025-03-07 T12:56:16.566Z	cowrie.command.input	`system`	CMD: system
2025-03-07 T12:56:18.916Z	cowrie.command.input	`ping ;sh`	CMD: ping ;sh
2025-03-07 T12:56:20.178Z	cowrie.command.input	`/bin/busybox...`	CMD: /bin/busybox cat /proc/self/exe...
2025-03-07 T12:56:44.180Z	cowrie.session.closed	-	Connection lost after 3 seconds

Analizzare questi eventi singolarmente sarebbe non solo inefficiente, ma anche fuorviante. L'indizio chiave che rivela la tecnica di attacco è il comando `ping ;sh`. Questo non è un semplice ping, ma un classico test di *Command Injection*: l'attaccante concatena un comando innocuo (`ping`) con un separatore (`;`) e un comando per avviare una shell (`sh`) per verificare se il sistema di destinazione esegue l'intera stringa senza un'adeguata validazione dell'input. Una volta ottenuta la conferma, le azioni successive, come `/bin/busybox cat /proc/self/exe`, rivelano l'obiettivo finale: una fase di *Discovery* per identificare la natura dell'ambiente (in questo caso, un sistema basato su BusyBox, tipico dei dispositivi IoT o *embedded*).

È solo attraverso l'aggregazione olistica e la sintesi implementate nel modulo di *preprocessing* che questa narrazione completa emerge. La funzione `generate_llm_input`, applicata all'intera sequenza di eventi, produce la descrizione testuale per l'LLM riportata nel Listato 13.

Questo esempio illustra chiaramente il valore della trasformazione operata dal modulo:

- **Da dati frammentati a narrazione coerente:** Il sistema ha trasformato una serie di righe di log in una narrazione unica e leggibile, che collega logicamente il tentativo di login fallito, il successo successivo e la sequenza di comandi, fornendo un quadro completo dall'accesso iniziale alla fase di esplorazione.

- **Sintesi e contestualizzazione intelligente:** Invece di presentare ogni singolo evento, il sistema ha sintetizzato le informazioni salienti (numero di tentativi falliti, utente loggato,

```
Interactive TELNET session detected on destination port(s) 23, emulating
↪ architecture(s): 'linux-tilegx-lsb'. After 1 failed attempt(s), login succeeded
↪ as user 'root'. 8 command(s) were attempted/executed in total. Specific
↪ observations: Command Summary: {"Commands sequence": ["sh", "shell", "shell",
↪ "enable", "system", "system", "ping ;sh", "/bin/busybox cat /proc/self/exe ||
↪ cat /bin/echo"]}.
```

Listato 13: Descrizione testuale olistica generata per la sessione `1183fdaf131c`, pronta per l'LLM

numero totale di comandi) e ha incluso dettagli di contesto cruciali come l'architettura emulata (`linux-tilegx-lsb`), che possono aiutare l'LLM a inferire la natura del target.

- **Preservazione delle evidenze chiave in formato strutturato:** La sequenza completa dei comandi, che è l'evidenza più importante, è stata preservata e strutturata in un formato JSON all'interno della sezione "Specific observations". Questo non solo garantisce che l'LLM riceva l'informazione critica in modo chiaro, ma dimostra anche l'efficacia dell'euristica di riepilogo (in questo caso, mostrando l'intera sequenza poiché breve).

Questo output, pulito e contestualizzato, è la base su cui il modulo di interpretazione semantica (descritto nella Sezione 5.6) potrà operare con la massima efficacia per inferire il TTP sottostante. L'LLM, ricevendo questa narrazione, non ha più bisogno di parsare dati grezzi, ma può concentrarsi sul compito di più alto livello: riconoscere il pattern di *Reconnaissance* che sfrutta una vulnerabilità di *Command Injection*.

5.6 Implementazione del modulo di interpretazione semantica

Il modulo di interpretazione semantica traduce la descrizione testuale del comportamento di un attore, prodotta dal modulo di *preprocessing*, in un'analisi strutturata in formato JSON. Questo componente è interamente basato su un LLM generativo *open weight*. La sua implementazione ha richiesto di affrontare tre sfide principali:

1. **Selezione del modello:** individuare un modello adeguato che bilanci prestazioni e requisiti computazionali.
2. **Caricamento efficiente:** eseguire tale modello su hardware con risorse limitate.
3. **Ingegnerizzazione del *prompt*:** sviluppare un *prompt* robusto e adattivo in grado di guidare il modello verso l'*output* desiderato.

5.6.1 Selezione, caratteristiche e configurazione del modello

La scelta del modello è un passaggio critico che influenza l'intera pipeline. Per questo studio è stato selezionato `mistralai/Mistral-7B-Instruct-v0.2`, un modello linguistico *open weight* sviluppato da Mistral AI, per via del suo eccellente compromesso tra dimensioni, prestazioni e capacità di seguire istruzioni.

Origini e prestazioni del modello base. `Mistral-7B-Instruct-v0.2`, con i suoi 7.3 miliardi di parametri, è una versione *instruction-tuned* del modello base `Mistral-7B` [29], [88]. Già al suo rilascio, il modello base si è distinto nel panorama degli LLM *open weight* per le sue performance competitive. Come documentato in diversi benchmark di *natural language processing*, `Mistral-7B` ha superato modelli con un numero di parametri significativamente superiore (come `Llama 2 13B`), dimostrando una particolare efficacia in compiti complessi che richiedono ragionamento e generazione di codice [31], [88]. Questa elevata efficienza parametrica lo ha reso un candidato ideale per applicazioni che richiedono prestazioni elevate senza l'onere computazionale di modelli molto più grandi.

Specializzazione "Instruct" e ottimizzazioni architetturali. La variante `-Instruct-v0.2` è stata specificamente ottimizzata per l'aderenza a istruzioni complesse. Questo è il risultato di un addestramento mirato su dataset pubblicamente disponibili per *task* di conversazione e *question answering*, che ne potenzia l'idoneità per applicazioni interattive e guidate da *prompt* dettagliati, come quella richiesta dal nostro sistema [29], [31].

Dal punto di vista architetturale, il modello `Mistral-7B` integra due innovazioni chiave che ne migliorano l'efficienza, particolarmente rilevanti per l'analisi di log che possono essere lunghi e complessi:

- ***Grouped-query attention* (GQA):** Questa modifica all'architettura dell'attention permette un'inferenza più rapida e riduce i requisiti di memoria durante la fase di decodifica, accelerando la generazione del testo.
- ***Sliding window attention* (SWA):** A differenza dei modelli Transformer classici, la SWA consente al modello di gestire sequenze di testo molto lunghe (fino a una finestra di contesto teorica di 32.000 token) con costi computazionali contenuti. Questa caratteristica è particolarmente vantaggiosa per il nostro caso d'uso, in quanto le descrizioni aggregate dei log, specialmente per sessioni `Cowrie` complesse, possono superare i limiti di contesto di modelli più vecchi.

Queste ottimizzazioni sono descritte nel *paper* originale di Mistral AI [88]. Inoltre, il modello è rilasciato sotto la licenza Apache 2.0, che ne consente un utilizzo flessibile e senza restrizioni per scopi di ricerca e applicativi [89].

Implementazione del caricamento tramite quantizzazione a 4-bit. Una volta selezionato il modello `Mistral-7B-Instruct-v0.2`, la sfida implementativa principale è stata quella di renderlo operativo su un ambiente con risorse computazionali limitate, come una singola GPU disponibile in piattaforme cloud come Google Colaboratory. Un modello da 7 miliardi di parametri, se caricato con la precisione standard a 32-bit (FP32), richiederebbe circa 28 GB di VRAM solo per i pesi, senza contare la memoria necessaria per gli stati dell'ottimizzatore e i gradienti. Per superare questo ostacolo, è stata adottata una strategia di **quantizzazione a 4-bit**.

Quantizzazione a 4-bit con bitsandbytes. La quantizzazione è una tecnica che riduce la precisione numerica dei pesi del modello, diminuendone drasticamente l'occupazione di memoria con un impatto minimo sulla performance inferenziale. L'implementazione di questa tecnica è stata gestita tramite la libreria `bitsandbytes`, configurata attraverso l'oggetto `bitsandbytesconfig` della libreria `transformers`. Questa configurazione permette di definire in modo granulare la strategia di quantizzazione. Il Listato 14 illustra la configurazione specifica utilizzata nel progetto.

```
# Configurazione per la quantizzazione del modello a 4-bit
bnb_config = bitsandbytesconfig(
load_in_4bit=True,
bnb_4bit_quant_type="nf4",
bnb_4bit_use_double_quant=True,
bnb_4bit_compute_dtype=torch.bfloat16
)

# Caricamento del modello tramite la classe AutoModelForCausalLM
model = AutoModelForCausalLM.from_pretrained(
LLM_MODEL,
quantization_config=bnb_config,
device_map="auto",
torch_dtype=torch.bfloat16,
trust_remote_code=True
)
```

Listato 14: Configurazione della quantizzazione a 4-bit e caricamento del modello Mistral-7B

Le scelte di configurazione sono state deliberate per massimizzare l'efficienza senza sacrificare l'accuratezza:

- `load_in_4bit=True`: È l'istruzione principale che attiva la quantizzazione, indicando alla libreria di caricare i pesi del modello utilizzando solo 4 bit per parametro.
- `bnb_4bit_quant_type="nf4"`: Specifica l'uso del tipo di dato "Normal Float 4-bit" (NF4). Questo tipo di quantizzazione è particolarmente efficace per i modelli neurali,

poiché è ottimizzato per pesi che seguono una distribuzione normale, come è tipico nei modelli pre-addestrati.

- `bnb_4bit_use_double_quant=True`: Abilita una seconda fase di quantizzazione per le costanti di quantizzazione stesse, un'ottimizzazione che permette di risparmiare, in media, circa 0.4 bit aggiuntivi per parametro, riducendo ulteriormente l'impronta di memoria.
- `bnb_4bit_compute_dtype=torch.bfloat16`: Questo parametro è cruciale per preservare l'accuratezza. Sebbene i pesi siano *memorizzati* a 4-bit, i *calcoli* (come le moltiplicazioni matrice-vettore durante l'inferenza) vengono eseguiti a una precisione maggiore (in questo caso, 16-bit bfloat). Questo approccio, noto come *mixed-precision*, garantisce che la riduzione della memoria non si traduca in una degradazione significativa delle performance del modello.

Il parametro `device_map="auto"` della libreria `Accelerate` gestisce la distribuzione automatica del modello tra le risorse disponibili (GPU e CPU), ottimizzando ulteriormente l'uso della memoria.

Configurazione del tokenizer. Parallelamente al caricamento del modello, viene inizializzato il ***tokenizer*** associato (`AutoTokenizer`). Questo componente è responsabile della conversione del *prompt* testuale in una sequenza di token numerici che il modello può processare. Una configurazione cruciale ha riguardato la gestione del *padding token*. Poiché i modelli generativi come `Mistral-7B` non hanno un *padding token* predefinito, è stato impostato esplicitamente in modo che corrispondesse all'*end-of-sentence (EOS) token*. Questa è una pratica standard per garantire che il modello gestisca correttamente sequenze di lunghezza variabile durante la tokenizzazione.

Creazione della pipeline di inferenza. Una volta caricato e quantizzato il modello, l'interazione con esso viene astratta attraverso una *pipeline* di generazione di testo, creata tramite la funzione `pipeline` della libreria `transformers`. Questa astrazione di alto livello incapsula l'intera catena di inferenza:

1. La tokenizzazione del *prompt* di input, gestita dall'`AutoTokenizer` associato al modello.
2. L'esecuzione dell'inferenza vera e propria sul modello caricato in memoria.
3. La decodifica dei token di output in una stringa di testo leggibile.

La *pipeline* viene inizializzata una sola volta all'avvio del sistema (`initialize_system`), garantendo che il modello rimanga residente in memoria per un'elaborazione rapida delle richieste successive, un requisito fondamentale per un'analisi efficiente di un gran numero di log.

5.6.2 Implementazione del *prompt engineering* adattivo

Una volta configurata l'infrastruttura per l'esecuzione del modello, il passo successivo consiste nel guidare la sua inferenza per ottenere un'analisi accurata e strutturata. Questo obiettivo è stato raggiunto tramite l'implementazione di un meccanismo di *prompt engineering* dinamico e adattivo, gestito dalla funzione `_get_llm_prompt`. Questa funzione non si limita all'inserimento dei dati in un *template* statico, bensì costruisce un *prompt* su misura per ogni richiesta, orchestrando diverse strategie.

Struttura e sintassi del *prompt template*. Il *prompt* si fonda su un *template* di base che rispetta la sintassi specifica richiesta dai modelli "Instruct" come `Mistral-7B`. L'intera sequenza di istruzioni è incapsulata tra i *token* di inizio sequenza (`<s>`), i tag di istruzione (`[INST]` e `[/INST]`), e il *token* di fine sequenza (`</s>`), come raccomandato dalla documentazione ufficiale per garantire un'interpretazione ottimale del *prompt* [31]. All'interno di questa struttura, il contenuto è organizzato utilizzando intestazioni in formato Markdown (es. `### Your Role`, `### Instructions`), una scelta che migliora la capacità del modello di distinguere gerarchicamente le diverse parti della richiesta: il suo ruolo, la missione, le regole critiche e il formato di *output*. Un estratto del template di base é mostrato nel Listato 15.

Il *template* integra elementi avanzati per guidare il ragionamento:

- ***Threat hierarchy*:** Viene imposta una gerarchia esplicita (Execution > Credential Access > Reconnaissance) per risolvere le ambiguità in scenari multi-fase, istruendo il modello a dare priorità all'azione più severa (*apex threat*).
- ***Chain-of-thought* (CoT):** Si richiede un processo di ragionamento interno obbligatorio (Evidence Ingestion → Intent Inference → Abstraction) per evitare risposte superficiali.
- **Regole critiche:** Vengono definiti vincoli negativi (es. "DO NOT SUMMARIZE") per forzare la produzione di definizioni astratte utili al *retrieval*.

Adattività contestuale e *few-shot learning* dinamico. Uno dei contributi più significativi dell'implementazione riguarda il superamento dei limiti dei *prompt* statici. L'approccio adottato trasforma il paradigma di interazione da un generico *zero-shot* (in cui il modello deve indovinare il *task* senza esempi) a un più robusto *one-shot learning* dinamico. Questa logica è implementata attraverso una struttura condizionale nel codice Python che, prima di ogni inferenza, assembla il *prompt* finale concatenando al *template* di base due moduli informativi specifici per il tipo di honeypot in esame (es. `Cowrie`, `Honeytrap`, `Dionaea`).

Questa strategia si articola in due componenti fondamentali:

1. **Iniezione di regole di contesto specifiche (`### Honeypot Context`):** Ogni tipologia di honeypot cattura dati con semantiche e formati radicalmente diversi. Un *prompt* generico

```
base_template = """<s>[INST]
### Your Role: Threat Intelligence Lexicographer
You are an AI tasked with defining cybersecurity attack patterns for a formal knowledge base like CAPEC. Your role is **not to narrate a specific event**, but to provide a formal, abstract, and encyclopedic **DEFINITION**...

### Core Mission: Identify the Apex Threat and Define It
1. **Identify the Apex Threat**: From the input log, identify the single most severe and significant action...
2. **Define the Technique**: Write a formal, general-purpose definition...

### Chain-of-Thought Process
You MUST follow this internal reasoning process for ALL tasks:
1. **Evidence Ingestion**: List all actions...
2. **Input Validity Check**: ...
3. **Intent Inference**: Determine the *true intent*...
4. **Apex Threat Identification**: ...
5. **Contextual Coherence Check**: ...
6. **Abstraction & Definition**: ...

### Threat Hierarchy (Highest to Lowest Priority)
1. **Execution**: Any form of code or command execution.
2. **Credential Access**: Attempts to use, steal, or guess credentials.
3. **Reconnaissance & Discovery**: All other information gathering activities.

### Critical Rules
- RULE 1 (EVIDENCE-BOUND REASONING): You MUST NOT infer actions not present in the log...
- RULE 2 (DO NOT SUMMARIZE): ...
- RULE 3 (STRICT ABSTRACTION): ...

### Output JSON Structure
Your output must be ONLY a single valid JSON object...
"""
```

Listato 15: Estratto del *template* di base utilizzato per il *prompt engineering*

rischierebbe di essere troppo vago. Il sistema, invece, inietta un blocco di testo che agisce come una "lente interpretativa" specializzata.

- Per **`Cowrie`** (SSH/Telnet), le regole istruiscono il modello a focalizzarsi sulla sequenza temporale dei comandi eseguiti e sui *download* di file, ignorando i metadati di connessione irrilevanti.
- Per **`Honeytrap`** (Network), le regole spostano l'attenzione sull'analisi dei *payload* esadecimali decodificati e sulle porte target, fornendo euristiche per distinguere una scansione generica da un *exploit* specifico.
- Per **`SentryPeer`** (VoIP), il contesto definisce esplicitamente come interpretare i metodi SIP (es. `REGISTER` vs `INVITE`) in ottica di frode telefonica.

Queste istruzioni riducono lo spazio di ricerca del modello, prevenendo allucinazioni derivanti da un'errata interpretazione della natura dei log.

2. **Esemplificazione mirata (`### Example`):** Il *prompt* non si limita a descrivere il *task*, ma lo *dimostra*. Viene incluso un esempio completo (coppia Log Input → JSON Output Ideale) selezionato specificamente per l'honeypot corrente. Questo passaggio è cruciale per tre motivi:
   - **Allineamento del formato:** Mostra al modello l'esatta struttura JSON richiesta, abbattendo significativamente gli errori di sintassi nel *parsing* successivo.
   - **Calibrazione del livello di astrazione:** L'esempio "insegna" al modello quanto deve essere astratta la descrizione. Vedendo come una lista di comandi *raw* (es. `wget`, `chmod`) viene trasformata in un concetto formale (es. "Ingress Tool Transfer"), il modello apprende a replicare questo salto logico sui nuovi dati.
   - ***Tone setting*:** Impone lo stile linguistico formale e tecnico richiesto per i campi `detailed_description` e `justification`.

Il Listato 16 mostra concretamente come questi blocchi vengono selezionati dinamicamente nel caso di una sessione `Cowrie`, trasformando un modello generalista in un analista specializzato per quella specifica sessione.

Questa combinazione di istruzioni contestuali ed esempi specifici "specializza" temporaneamente il modello, migliorando drasticamente la pertinenza e l'accuratezza dell'analisi senza la necessità di un costoso *fine-tuning*.

5.6.3 Esecuzione dell'inferenza e *parsing* dell'output

L'esecuzione dell'inferenza è gestita dalla funzione `_generate_llm_analysis`, che orchestra la chiamata alla *pipeline* di generazione di testo di `Hugging Face`.

```
### Honeypot Context: Cowrie
- **Input Type**: Interactive SSH/Telnet session logs.
- **Context-Specific Rules**:
1. The execution of any script or binary (`sh`, `bash`, `./filename`) ... is ALWAYS the Apex Threat (Execution).
2. If no execution occurs, a sequence of commands for system enumeration ... is the Apex Threat (Reconnaissance).

### Example
**Input Log:**
`Interactive SSH session. Login succeeded as 'pi'. Commands: ["uname -r", "ifconfig", "ps aux"].`
**Output JSON:**
```json
{
        "pattern_name": "Reconnaissance: System Information Gathering",
        "detailed_description": "System Information Gathering is a technique where an adversary executes a series
        ↪  of built-in commands to obtain detailed information about a host's configuration. This can include
        ↪  discovering the operating system version, network interfaces, and running processes. This activity
        ↪  allows the adversary to map the system's environment to plan subsequent actions.",
        "technical_keywords": ["reconnaissance", "discovery", "system information gathering", "os fingerprinting",
        ↪  "network configuration discovery", "process discovery", "host enumeration", "post-exploitation"],
        "justification": "The sequence of commands including 'uname -r', 'ifconfig', and 'ps aux' is the
        ↪  definitive evidence of a system information gathering technique."
}
```
```

Listato 16: Esempio di iniezione contestuale e *one-shot* specifica per l'honeypot Cowrie

Configurazione dei parametri di generazione. I parametri di generazione sono stati scelti per bilanciare determinismo e flessibilità (Listato 17).

```
# Chiamata alla pipeline di generazione testo con parametri specifici
response = llm_pipeline(
prompt,
max_new_tokens=500,       # Limita la lunghezza per evitare output sproporzionati
temperature=0.2,          # Bassa temperatura per risposte più focalizzate e meno casuali
top_p=0.9,                # Nucleus sampling per mantenere diversità
do_sample=True,             # Abilita il sampling (necessario per temperature > 0)
repetition_penalty=1.1,   # Penalizza leggermente la ripetizione di token
pad_token_id=tokenizer.eos_token_id,
eos_token_id=tokenizer.eos_token_id
)[0]['generated_text']
```

Listato 17: Chiamata all'inferenza con parametri ottimizzati per la stabilità

Una `temperature` bassa (0.2) riduce la casualità, rendendo l'*output* più deterministico e focalizzato, mentre l'attivazione del `do_sample` con un `top_p` elevato (0.9) permette al modello di esplorare un vocabolario leggermente più ampio, evitando di rimanere bloccato in *loop* ripetitivi, un comportamento ulteriormente mitigato dalla `repetition_penalty`.

***Parsing* robusto dell'*output* JSON.** L'*output* grezzo di un LLM è una stringa di testo, che può occasionalmente contenere artefatti o non rispettare perfettamente il formato richiesto. Per

garantire la stabilità della *pipeline*, la funzione `_parse_llm_response` implementa una logica di *parsing* robusta:

1. **Isolamento della risposta:** Viene prima isolata la porzione di testo generata dal modello, scartando il *prompt* di input che viene restituito dalla *pipeline*.

2. **Estrazione del blocco JSON:** Invece di assumere che l'intera risposta sia JSON, il codice cerca esplicitamente la prima parentesi graffa di apertura (‘‘) e l'ultima di chiusura (‘‘), estraendo solo la sottostringa contenuta tra di esse. Questo rende il *parsing* resiliente a testo aggiuntivo che il modello potrebbe generare prima o dopo il blocco JSON.

3. **Decodifica e gestione degli errori:** La sottostringa viene quindi decodificata utilizzando il *parser* standard `json.loads`. L'intera operazione è avvolta in un blocco `try...except` che, in caso di qualsiasi errore (es. JSON malformato o assente), intercetta l'eccezione e restituisce un dizionario di *default* con un messaggio di errore. Questo approccio garantisce che un fallimento isolato dell'LLM non causi l'interruzione dell'intera *pipeline* di analisi dei log.

Questa catena di elaborazione assicura che l'interazione con l'LLM sia non solo efficace, ma anche robusta e integrata in modo affidabile nel flusso automatizzato.

5.7 Implementazione del modulo di *matching* ibrido

Il modulo di *matching* ibrido rappresenta la fase conclusiva della *pipeline*, implementando la logica di ricerca e classificazione. Questo componente traduce in codice i principi astratti della ricerca ibrida e della *Reciprocal Rank Fusion*, descritti nell'architettura concettuale (Sezione 4.6). L'intera logica è incapsulata all'interno della classe `HybridAnalyzer`, il cui metodo principale, `analyze`, orchestra l'intero processo, dalla preparazione della *query* alla fusione finale delle classifiche.

5.7.1 Preparazione della *query* e coerenza dello spazio di rappresentazione

Il processo di *matching* inizia dopo che il modulo LLM ha prodotto la sua analisi in formato JSON. Per poter essere utilizzata efficacemente dai motori di ricerca, questa analisi strutturata deve essere trasformata in una rappresentazione testuale unificata. Questo compito è affidato al metodo `_get_query_text_from_llm` (Listato 18).

Un dettaglio implementativo di importanza cruciale è che la stringa di *query* viene processata dalla stessa funzione di *preprocessing* linguistico, `full_preprocess`, utilizzata per la costruzione della *knowledge base* CAPEC (descritta nella Sezione 5.4.2). Questa scelta garantisce la *coerenza dello spazio di rappresentazione* tra la *query* e i documenti indicizzati. Per la ricerca lessicale, assicura che la tokenizzazione e la normalizzazione dei termini nella *query* corrispondano

```python
def _get_query_text_from_llm(self, llm_analysis: Dict) -> str:
# Aggregazione delle componenti testuali dell'analisi JSON
query_components = [
llm_analysis.get('pattern_name', ''),
llm_analysis.get('detailed_description', ''),
' '.join(llm_analysis.get('technical_keywords', []))
]
# Unione dei componenti in un'unica stringa
unified_query_text = ' '.join(filter(None, query_components))

# Applicazione della stessa pipeline di preprocessing del corpus CAPEC
return full_preprocess(unified_query_text)
```

Listato 18: Preparazione della *query* unificata con coerenza di *preprocessing*

esattamente a quelle dell'indice TF-IDF. Per la ricerca semantica, garantisce che l'*embedding* della *query* venga generato a partire da un testo con le stesse caratteristiche linguistiche (es. acronimi espansi, *stopwords* rimosse) degli *embedding* dei documenti target, un prerequisito fondamentale per un calcolo di similarità coseno significativo.

5.7.2 Esecuzione della ricerca parallela e generazione delle classifiche

Ultimata la preparazione della *query*, il sistema esegue due ricerche in parallelo, ciascuna mirata a generare una classifica di candidati basata su un diverso segnale di rilevanza.

Recupero della classifica semantica tramite ricerca vettoriale. La ricerca semantica ha lo scopo di identificare i pattern CAPEC concettualmente più simili alla descrizione dell'attacco fornita dall'LLM. L'implementazione si articola in tre passaggi:

1. **Codifica della *query* in *embedding*:** La stringa di *query*, già pre-elaborata, viene passata al modello Transformer dominio-specifico `ATT&CK-BERT`, caricato tramite la libreria `Sentence-Transformers`. Il metodo `encode()` del modello trasforma il testo in un vettore numerico ad alta dimensionalità. L'opzione `normalize_embeddings=True` viene utilizzata per proiettare il vettore risultante sulla superficie di un'ipersfera di raggio unitario, una normalizzazione standard che ottimizza il vettore per il calcolo successivo della similarità coseno.

2. **Interrogazione del database vettoriale:** L'*embedding* della *query* viene quindi utilizzato per interrogare la collezione `ChromaDB` tramite il suo metodo `query()`. Questo metodo implementa internamente un algoritmo di *Approximate Nearest Neighbor* (ANN), specificamente HNSW, per trovare in modo efficiente i `n_results` vettori più vicini (in questo caso, 100 candidati). Poiché la collezione è stata configurata con lo spazio di similarità "cosine", la "vicinanza" corrisponde a una maggiore similarità coseno.

3. **Costruzione della mappa dei *rank*:** Il metodo `query()` di `ChromaDB` restituisce una lista di ID di documenti già ordinata per similarità, dalla più alta alla più bassa. Questa lista ordinata viene trasformata in un dizionario Python, `semantic_rank_map`, che funge da *lookup table*, mappando ogni ID di pattern CAPEC alla sua posizione (*rank*, da 1 a N) all'interno della classifica semantica.

Il Listato 19 mostra l'implementazione di questi passaggi.

```
# 1. Codifica della query in un embedding normalizzato
query_embedding = self.embedder.encode(query_text, normalize_embeddings=True)

# 2. Interrogazione di ChromaDB per ottenere N candidati
CANDIDATE_COUNT = 100
semantic_results = db_manager.collection.query(
query_embeddings=[query_embedding.tolist()],
n_results=CANDIDATE_COUNT
)

# 3. Creazione della mappa dei rank semantici
semantic_ids = semantic_results.get('ids', [[]])[0]
semantic_rank_map = {doc_id: i + 1 for i, doc_id in enumerate(semantic_ids)}
```

Listato 19: Esecuzione della ricerca vettoriale e generazione della mappa dei *rank*

Recupero della classifica lessicale tramite ricerca TF-IDF. Parallelamente alla ricerca semantica, il sistema esegue una ricerca basata su *keyword* per identificare i pattern CAPEC che contengono i termini tecnici più rilevanti estratti dall'analisi dell'LLM. Questa seconda "opinione" è fondamentale per ancorare il *matching* alla terminologia specifica del dominio, agendo come correttivo alla natura talvolta eccessivamente astratta della ricerca semantica. L'implementazione di questa ricerca sfrutta la matrice `TF-IDF` e il vettorizzatore costruiti durante l'inizializzazione della *knowledge base*.

Il processo si articola in tre passaggi logici:

1. **Vettorizzazione della *query* nello spazio TF-IDF:** La stessa stringa di *query* unificata e pre-elaborata, `query_text`, viene passata al metodo `transform()` del `TfidfVectorizer` precedentemente addestrato sul corpus CAPEC. È cruciale sottolineare che viene utilizzato il metodo `transform()` e non `fit_transform()`. Questo garantisce che la *query* venga vettorizzata utilizzando il vocabolario e i pesi IDF (*Inverse Document Frequency*) già appresi dall'intero corpus CAPEC, senza alterare il modello statistico esistente. Il risultato è un vettore sparso che rappresenta la *query* nello stesso identico spazio vettoriale dei documenti indicizzati, un prerequisito per un confronto matematicamente valido.

2. **Calcolo della similarità coseno su vettori sparsi:** Viene quindi calcolata la similarità coseno tra il vettore `TF-IDF` della *query* e la matrice `TF-IDF` dell'intero corpus CAPEC. Questa operazione, implementata tramite la funzione `cosine_similarity` della libreria `Scikit-learn`, è ottimizzata per operare in modo efficiente su matrici sparse. A differenza della ricerca vettoriale densa, che opera su poche centinaia di dimensioni, qui il confronto avviene in uno spazio ad altissima dimensionalità (pari alla dimensione del vocabolario), ma dove la maggior parte dei valori è zero. La funzione calcola il prodotto scalare normalizzato, producendo un array di punteggi di similarità, uno per ogni pattern CAPEC nel database. Un punteggio elevato in questa fase indica una forte sovrapposizione di termini rari e significativi tra la *query* e un pattern CAPEC.

3. **Costruzione della mappa dei *rank* lessicali:** L'array di punteggi grezzi di similarità `TF-IDF` viene utilizzato per ordinare gli ID dei pattern CAPEC in ordine decrescente di rilevanza. Come per la ricerca semantica, questa lista ordinata viene trasformata in un dizionario Python, `keyword_rank_map`. Questa struttura dati funge da *lookup table*, mappando ogni ID di pattern CAPEC alla sua esatta posizione (*rank*, da 1 a N) all'interno della classifica puramente lessicale.

Il Listato 20 illustra l'implementazione concreta di questi passaggi, mostrando come, a partire da una stringa di testo, si ottiene una classifica ordinata basata sulla rilevanza delle *keyword*.

```
# Funzione interna alla classe VectorDBManager
def compute_keyword_similarity(self, query: str) -> Dict[str, float]:
# 1. Vettorizzazione della query usando il TfidfVectorizer esistente
query_vec = self.tfidf_vectorizer.transform([full_preprocess(query)])

# 2. Calcolo della similarità coseno contro l'intera matrice TF-IDF
cos_sim = cosine_similarity(query_vec, self.tfidf_matrix).flatten()

# Restituisce un dizionario di punteggi grezzi {capec_id: score}
return {self.capec_ids[i]: float(cos_sim[i]) for i in range(len(self.capec_ids))}

# Logica nel metodo analyze di HybridAnalyzer per creare la mappa dei rank
# ...
all_keyword_scores = db_manager.compute_keyword_similarity(query_text)
# Ordinamento per punteggio per ottenere la classifica
sorted_keyword_ids = sorted(all_keyword_scores, key=all_keyword_scores.get, reverse=True)
# 3. Creazione della mappa dei rank lessicali
keyword_rank_map = {doc_id: i + 1 for i, doc_id in enumerate(sorted_keyword_ids)}
```

Listato 20: Esecuzione della ricerca lessicale TF-IDF e generazione della mappa dei *rank*

Al termine di questa fase, il sistema dispone di due mappe di *rank* parallele, `semantic_rank_map` e `keyword_rank_map`, pronte per essere fuse nella fase successiva.

5.7.3 Fusione delle classifiche tramite *reciprocal rank fusion*

A questo punto del processo, il sistema dispone di due classifiche indipendenti, rappresentate dalle mappe di *rank* `semantic_rank_map` e `keyword_rank_map`. Ciascuna classifica offre una prospettiva diversa sulla rilevanza dei pattern CAPEC, ma i loro punteggi grezzi (similarità coseno e `TF-IDF`) operano su scale e con distribuzioni statistiche diverse, rendendo una loro combinazione diretta, come una somma pesata, metodologicamente problematica e potenzialmente inaffidabile.

Per superare questa sfida, è stata implementata una tecnica di fusione più robusta e agnostica rispetto ai punteggi: la *Reciprocal Rank Fusion* (RRF). L'RRF non opera sui punteggi, ma sulla *posizione* (*rank*) di ogni documento in ciascuna lista. L'intuizione fondamentale è che un documento posizionato in alto in entrambe le classifiche ha maggiori probabilità di essere rilevante, indipendentemente dai suoi punteggi assoluti.

Logica implementativa della RRF. L'implementazione, contenuta nel metodo `analyze` della classe `HybridAnalyzer`, traduce in codice l'algoritmo teorico definito nell'Equazione 4.1 (Sezione 4.6). Il processo si articola in tre passaggi chiave:

1. **Unione dell'insieme dei candidati:** Per prima cosa, viene creato un insieme unico (`set`) di tutti gli ID di pattern CAPEC che appaiono in almeno una delle due classifiche (fino a un massimo di `CANDIDATE_COUNT` per lista). Questo garantisce che nessun potenziale candidato venga perso durante la fusione e definisce lo spazio dei documenti da ri-classificare.

2. **Calcolo del punteggio fuso (RRFScore):** Il sistema itera su ogni ID di documento nell'insieme dei candidati e calcola il punteggio sommando i reciproci dei *rank*, applicando la costante di smorzamento k. Come mostrato nel Listato 21, se un documento non è presente in una delle classifiche, il codice gestisce implicitamente il termine come zero non sommando nulla per quel ramo.

3. **Ordinamento finale e generazione dell'output:** Infine, il dizionario contenente i punteggi RRF fusi viene utilizzato per ordinare la lista dei candidati in ordine decrescente. Ciò genera la classifica ibrida definitiva.

L'utilizzo della costante `k_rrf` è una scelta implementativa importante. Come suggerito dalla letteratura, un valore come 60 serve a smorzare l'impatto dei documenti con un *rank* molto alto (es. posizione 1 o 2), evitando che un singolo risultato eccellente in una sola lista domini la classifica finale. Questo favorisce i documenti che ottengono un buon posizionamento, anche se non perfetto, in *entrambe* le classifiche, rendendo la fusione più stabile e robusta.

5.7.4 Esempio concreto di fusione RRF in azione

Per illustrare concretamente il funzionamento della fusione RRF, consideriamo l'esempio della sessione `Cowrie 1183fdaf131c`, già discussa nella Sezione 5.5.3, relativa a un attacco di

```
# La costante k_rrf è un iperparametro della classe HybridAnalyzer
self.k_rrf = 60 # Valore comune in letteratura per stabilizzare la fusione

# ... all'interno del metodo analyze ...

# 1. Unione di tutti gli ID candidati unici dalle due classifiche
all_candidate_ids = set(semantic_ids) | set(sorted_keyword_ids[:CANDIDATE_COUNT])

# 2. Calcolo del punteggio RRF per ogni candidato
fused_scores = defaultdict(float)
for doc_id in all_candidate_ids:
# Aggiunge il contributo del rank semantico, se presente
if doc_id in semantic_rank_map:
fused_scores[doc_id] += 1 / (self.k_rrf + semantic_rank_map[doc_id])
# Aggiunge il contributo del rank lessicale, se presente
if doc_id in keyword_rank_map:
fused_scores[doc_id] += 1 / (self.k_rrf + keyword_rank_map[doc_id])

# 3. Ordinamento finale basato sui punteggi RRF
sorted_fused_ids = sorted(fused_scores, key=fused_scores.get, reverse=True)
```

Listato 21: Algoritmo di *Reciprocal Rank Fusion* per la combinazione delle classifiche

command injection. Per l'analisi di questa sessione, i due motori di ricerca hanno prodotto classifiche divergenti (si riporta un estratto significativo):

- La **ricerca semantica** ha identificato come miglior candidato `CAPEC-6` (*Argument Injection*) posizionandolo al *rank* #1. Tuttavia, `CAPEC-88` (*OS Command Injection*) è stato posizionato subito dopo al *rank* #2.
- La **ricerca lessicale**, basata sulle *keyword* estratte ("command injection", "shell commands"), ha invece premiato `CAPEC-88` posizionandolo al *rank* #1, mentre `CAPEC-6` è scivolato molto in basso, al *rank* #28, probabilmente per la mancanza di corrispondenza esatta con i termini tecnici specifici.

Applicando la formula RRF (con $k = 60$), il sistema calcola i punteggi fusi che premiano il consenso:

- **Per `CAPEC-88`:** RRFScore = $\frac{1}{60+2} + \frac{1}{60+1} \approx 0.0161 + 0.0164 = 0.0325$
- **Per `CAPEC-6` (vincitore semantico):** RRFScore = $\frac{1}{60+1} + \frac{1}{60+28} \approx 0.0164 + 0.0114 = 0.0278$
- **Per `CAPEC-248` (*Command Injection*):** RRFScore = $\frac{1}{60+4} + \frac{1}{60+2} \approx 0.0156 + 0.0161 = 0.0317$

La Tabella 5.2 mostra l'estratto della classifica finale prodotta dal sistema. Questo esempio dimostra tangibilmente la robustezza della fusione: `CAPEC-6`, nonostante fosse il primo risultato semantico, viene penalizzato dalla mancanza di supporto lessicale e scende all'ottava posizione. Al contrario, `CAPEC-88` emerge come miglior candidato (*rank* #1) grazie al suo posizionamento eccellente e consistente in entrambe le classifiche (#2 e #1).

Tabella 5.2: Estratto della tabella di *debug* della fusione RRF per la sessione `1183fdaf131c`.

Rank finale	**CAPEC ID**	**Rank sem.**	**Rank key.**	**Punteggio RRF**
1	`CAPEC-88`	#2	#1	0.032522
2	`CAPEC-248`	#4	#2	0.031754
3	`CAPEC-108`	#3	#3	0.031746
4	`CAPEC-640`	#9	#4	0.030118
...	...	...	...	...
8	`CAPEC-6`	#1	#28	0.027757
...	...	...	...	...

Infine, per garantire la piena *interpretabilità* del processo di *matching*, l'*output* finale del sistema non si limita alla *top-k* dei risultati. Viene generata anche una tabella di *debug* completa, come quella mostrata, che permette a un analista di comprendere esattamente perché un particolare pattern CAPEC è stato classificato in una certa posizione, aumentando la fiducia nella soluzione.

5.8 Sintesi dell'implementazione e sfide superate

Il presente capitolo ha trasposto l'architettura concettuale, descritta nel Capitolo 3, in una soluzione software concreta e operativa, dettagliando le scelte tecnologiche e le metodologie ingegneristiche adottate per realizzare ogni modulo della *pipeline*.

Partendo da un ambiente di sviluppo basato su `Python` e sul suo ricco ecosistema di librerie *open source*, sono state affrontate e risolte diverse sfide ingegneristiche chiave. In primo luogo, è stata implementata la costruzione di una *knowledge base* ibrida, che sfrutta un *preprocessing* linguistico avanzato per indicizzare i pattern CAPEC in due modalità parallele, semantica e lessicale. Successivamente, è stata descritta l'implementazione del modulo di *preprocessing* dei log, evidenziando la complessa logica di aggregazione olistica per ricostruire il comportamento degli attaccanti e le tecniche di sintesi specifiche per ogni tipo di honeypot.

Il cuore implementativo del sistema è stato poi analizzato in dettaglio: è stato illustrato come un modello LLM *open weight* di grandi dimensioni (`Mistral-7B-Instruct-v0.2`) sia stato reso efficiente tramite *quantizzazione a 4-bit*, e come venga guidato da un *prompt engineering* avanzato e adattivo per tradurre i dati grezzi in un'analisi JSON strutturata. Infine, è stata descritta l'implementazione del motore di *matching*, che utilizza la robusta tecnica della *Reciprocal Rank Fusion* per combinare i risultati della ricerca semantica e lessicale.

Ogni scelta implementativa, dall'uso di modelli dominio-specifici come `ATT&CK-BERT` alla gestione robusta del *parsing* dell'*output* dell'LLM, è stata motivata dalla necessità di creare un sistema non solo efficace, ma anche interpretabile e riproducibile. Avendo ora descritto in dettaglio sia l'architettura concettuale (Capitolo 3) sia la sua realizzazione pratica (Capitolo 4), il passo successivo è misurarne l'efficacia. Il capitolo seguente, dedicato alla *valutazione sperimentale*, si concentrerà sulla validazione della *pipeline*, presentando gli obiettivi della valutazione, le metriche adottate, il protocollo di test e un'analisi approfondita dei risultati ottenuti dall'applicazione del sistema su un dataset di log reali.

Capitolo 6

Valutazione sperimentale della soluzione proposta

Dopo aver delineato l'architettura concettuale della *pipeline* di analisi nel Capitolo 3 e averne descritto in dettaglio la realizzazione pratica nel Capitolo 4, il presente capitolo si concentra sulla validazione empirica della soluzione proposta. L'obiettivo primario è misurare in modo rigoroso e multidimensionale l'efficacia del sistema nell'affrontare il complesso *task* di mappare automaticamente i log grezzi, eterogenei e rumorosi, provenienti da un ecosistema multi-honeypot, ai corrispondenti pattern di attacco del framework MITRE CAPEC.

Data l'assenza di *benchmark* pubblici consolidati e dataset annotati per questo specifico dominio applicativo, si è reso necessario definire e implementare un protocollo sperimentale su misura. Tale protocollo è stato ingegnerizzato per garantire la riproducibilità dei risultati e per fornire una valutazione trasparente e critica delle *performance* del sistema. La metodologia adottata, descritta nelle sezioni seguenti, combina un'analisi quantitativa basata su metriche di *information retrieval* standard, calcolate su un dataset di test appositamente costruito e annotato manualmente (denominato "*Gold Standard*"), con un'analisi qualitativa approfondita di casi studio specifici. Questo approccio misto mira a fornire una comprensione olistica non solo del "cosa" il sistema è in grado di fare, ma anche del "perché" e del "come" ottiene i suoi risultati.

6.1 Obiettivi e criteri della valutazione

L'esperimento è stato progettato per indagare in modo approfondito quattro aspetti fondamentali delle *performance* del sistema. Un principio guida centrale, che permea l'intera valutazione, è la filosofia di progettazione del sistema stesso: il suo scopo primario non è semplicemente elencare tutti i possibili eventi osservati in un log, ma piuttosto identificare e prioritizzare l'*apex threat*, ovvero l'azione di attacco più specifica, significativa e grave che emerge dal comportamento aggregato dell'attaccante.

Questa filosofia di progettazione influenza direttamente la scelta delle metriche di valutazione, ponendo un'enfasi primaria su quelle che misurano la capacità del sistema di posizionare correttamente la minaccia principale al primo posto. Di conseguenza, metriche come la *Precision@1* (P@1) e il *Mean Reciprocal Rank* (MRR) verranno considerate indicatori di successo primari, poiché riflettono direttamente l'obiettivo operativo di fornire a un analista di sicurezza una valutazione immediata e accurata della minaccia più critica.

In questo quadro, gli obiettivi specifici della valutazione sono i seguenti:

- **Valutare l'efficacia assoluta del sistema.** Il primo obiettivo è stabilire una *baseline* di *performance* quantitativa, misurando con quale accuratezza il sistema riesce a identificare l'*apex threat* e a produrre una lista di candidati CAPEC complessivamente pertinente e ben ordinata.

- **Quantificare il contributo dell'approccio ibrido.** Un secondo obiettivo cruciale è validare empiricamente la scelta architetturale di fondere una ricerca semantica e una lessicale. Attraverso uno studio di ablazione, si intende isolare e misurare il valore aggiunto della sinergia ottenuta tramite *Reciprocal Rank Fusion* rispetto ai due approcci utilizzati singolarmente.

- **Contestualizzare le *performance* in un confronto SOTA.** Per posizionare il sistema nel panorama tecnologico attuale, le sue *performance* verranno confrontate con quelle di una *baseline state-of-the-art*, rappresentata dal modello avanzato `Gemini 2.5 Pro`. Questo confronto mira a valutare la competitività della soluzione proposta, basata sul modello *open weight* `Mistral-7B`, rispetto a un modello di frontiera di maggiori dimensioni.

- **Analizzare la robustezza su diverse tipologie di dati.** Infine, la valutazione intende investigare come le *performance* del sistema varino in funzione del tipo di honeypot che ha generato il log. L'analisi cercherà di determinare se la natura dei dati di *input* (es. sessioni interattive vs. log di rete) influenzi l'efficacia dell'analisi, rivelando i punti di forza e le aree di potenziale miglioramento del sistema.

Le sezioni successive di questo capitolo descriveranno il *setup* sperimentale progettato per rispondere a questi obiettivi, le metriche adottate e l'analisi dettagliata dei risultati ottenuti.

6.2 Setup sperimentale e metodologia

Per rispondere in modo rigoroso agli obiettivi di valutazione delineati, è stato definito un protocollo sperimentale completo. È importante sottolineare che l'intero esperimento è stato concepito fin dall'inizio come un'*analisi comparativa*. Di conseguenza, la *baseline* SOTA (`Gemini 2.5 Pro`) non è solo un sistema con cui confrontare i risultati finali, ma è parte integrante del processo di creazione del *ground truth* stesso, attraverso la metodologia del *pooling*.

Questo garantisce che la valutazione sia equa e direttamente focalizzata sul confronto delle capacità di *ranking* dei diversi sistemi. Il protocollo copre la costruzione del dataset di test, la definizione dei sistemi a confronto e la selezione delle metriche di valutazione.

6.2.1 Costruzione del dataset di valutazione

Data l'assenza di un corpus di riferimento standard, il primo passo fondamentale è stato la creazione di un dataset di valutazione su misura, sufficientemente vario e rappresentativo da permettere una valutazione significativa.

Selezione del sottoinsieme di test

A partire dal dataset completo di log raccolti, è stato selezionato un sottoinsieme mirato di 25 profili di comportamento unici. La selezione non è stata casuale, ma ha seguito un approccio di campionamento *stratificato*, con l'obiettivo di garantire una rappresentazione bilanciata di tutte e cinque le tipologie di honeypot supportate dalla *pipeline*. Sono stati quindi selezionati 5 esempi distinti per ciascun tipo: `Cowrie`, `Honeytrap`, `Dionaea`, `Sentrypeer` e `CiscoASA`.

Inoltre, all'interno di ogni strato, gli esempi sono stati scelti per coprire un'ampia gamma di scenari di attacco osservati, da attività di ricognizione a basso impatto, come semplici scansioni di porte, fino a sessioni interattive complesse con esecuzione di comandi *post-authentication*. Questa eterogeneità è cruciale per testare la robustezza del sistema di fronte a segnali di minaccia di diversa natura e complessità.

Generazione del "*Gold Standard*" tramite *pooling* e *adjudication*

Il passaggio più critico e laborioso dell'intero processo sperimentale è stata la creazione del *ground truth*, o "*Gold Standard*". Per garantire un processo di valutazione efficiente e, soprattutto, perfettamente equo tra il sistema proposto e il suo principale termine di paragone, è stata adottata una metodologia standard di *information retrieval* nota come *pooling*.

Il principio è semplice: invece di giudicare l'intero corpus di oltre 500 pattern CAPEC per ogni log, il giudizio umano si concentra solo sui candidati più promettenti, come identificati dai sistemi stessi. Concretamente, per ciascuno dei 25 profili di comportamento, è stata creata una "*pool* di giudizio" consolidando i primi 5 candidati proposti dal sistema ibrido proposto e dalla *baseline* SOTA. L'insieme dei pattern unici risultante da questa unione ha costituito lo spazio di documenti da sottoporre al giudizio dell'esperto.

Successivamente, un esperto umano ha eseguito il processo di *adjudication* (giudizio) su *ogni* pattern all'interno di questa *pool*:

1. **Analisi delle prove:** Per ogni profilo, sono state esaminate attentamente sia la descrizione testuale aggregata del log, sia la lista consolidata di candidati CAPEC presenti nella *pool*.

2. **Assegnazione della rilevanza graduata:** Invece di una classificazione binaria (corretto/errato), è stata adottata una scala di rilevanza graduata a 3 livelli. Per ogni pattern CAPEC all'interno della *pool*, è stato assegnato un punteggio:

    - **Rilevanza 2 (*apex threat*):** Assegnato al singolo pattern CAPEC che descrive in modo più accurato, specifico e completo l'azione di attacco principale e più grave osservata nel log.
    - **Rilevanza 1 (Plausibile/Secondario):** Assegnato a pattern che sono tecnicamente corretti ma descrivono aspetti secondari, componenti della minaccia principale, o categorie più generiche.
    - **Rilevanza 0 (Irrilevante):** Assegnato a tutti gli altri pattern che sono stati giudicati errati o non pertinenti alle prove contenute nel log.

L'output di questo meticoloso processo è un file, `gold_standard.json`, che mappa ogni identificativo di sessione a un dizionario contenente le coppie {CAPEC-ID: punteggio di rilevanza}. È importante notare che, se un pattern CAPEC non appare nella *pool* (ovvero non è stato classificato nella *top-5* da nessuno dei due sistemi principali), viene implicitamente considerato con rilevanza 0. Sebbene questo approccio possa potenzialmente escludere pattern rilevanti non identificati da nessuno dei sistemi, esso garantisce un confronto perfettamente equo e rigoroso tra le *performance* relative del sistema proposto e della *baseline* SOTA, che è uno degli obiettivi primari di questa valutazione. Le configurazioni di ablazione, pur non contribuendo alla creazione della *pool*, vengono poi valutate rispetto a questo stesso *Gold Standard* per garantire la coerenza dell'analisi.

6.2.2 Sistemi a confronto

Per rispondere in modo esauriente agli obiettivi di valutazione, l'esperimento è stato strutturato come un'analisi comparativa tra diverse configurazioni del sistema, permettendo così di isolare e misurare l'impatto di specifiche scelte architetturali. I sistemi messi a confronto sono:

1. **Sistema proposto (ibrido):** Rappresenta la configurazione completa e finale della soluzione descritta in questa tesi. Esso impiega l'interprete LLM basato su `Mistral-7B` per generare una *query* strutturata, che viene poi utilizzata per interrogare simultaneamente un indice semantico (`ATT&CK-BERT`) e uno lessicale (`TF-IDF`). Le due classifiche risultanti vengono infine fuse tramite l'algoritmo *Reciprocal Rank Fusion* per produrre il *ranking* finale.

2. ***Baseline state-of-the-art* (SOTA):** Per contestualizzare le *performance* del sistema proposto, è stata introdotta una *baseline* di riferimento ad alte prestazioni. Questo ruolo è stato affidato a `Gemini 2.5 Pro`, un modello linguistico di grandi dimensioni e di frontiera. Per garantire un confronto equo e rigoroso, è stato definito un protocollo specifico per generare i risultati di questa *baseline*, mimando un approccio di *zero-shot reasoning* su una base di

conoscenza fornita in contesto. La metodologia di generazione per la *baseline* SOTA è stata la seguente:

- **Ruolo e istruzioni:** Al modello è stato assegnato il ruolo di "SOTA Threat Intelligence Analyst" con il compito di identificare l'*apex threat* e, successivamente, di trovare altri 4 pattern CAPEC correlati ma più generici.
- ***Knowledge base in-context*:** L'intero corpus dei pattern CAPEC (ID, nome e descrizione) è stato fornito direttamente nel contesto del *prompt*, costringendo il modello a eseguire il ragionamento e il *matching* basandosi esclusivamente sulle informazioni fornite.
- ***Input* e *output*:** Al modello sono stati forniti la stessa descrizione del log usata dal sistema proposto e un formato di *output* JSON strutturato da rispettare. Per garantire la determinicità, la temperatura del modello è stata impostata a 0.

Questo processo ha prodotto il file `oracle_results.json`, utilizzato per il confronto. Tale approccio mira a valutare la competitività della soluzione proposta, che si basa su un'architettura specializzata e un modello *open weight*, rispetto a un modello generalista di maggiori dimensioni a cui viene affidato un *task end-to-end*.

3. **Configurazioni di ablazione:** Per condurre lo studio di ablazione (*ablation study*) e rispondere direttamente all'obiettivo di quantificare il contributo dell'approccio ibrido, sono state valutate due versioni "ridotte" del sistema, disabilitando una delle due componenti di ricerca:

   - **Solo semantico:** In questa configurazione, viene utilizzato esclusivamente il *ranking* prodotto dalla ricerca vettoriale basata su similarità coseno, ignorando completamente i risultati della ricerca lessicale.
   - **Solo lessicale:** In questa configurazione, viene utilizzato esclusivamente il *ranking* prodotto dalla ricerca basata su `TF-IDF`, ignorando i risultati della ricerca semantica.

Il confronto tra le *performance* di queste quattro configurazioni permette di ottenere una visione completa, non solo dell'efficacia assoluta del sistema, ma anche del contributo relativo di ciascun suo componente fondamentale.

6.2.3 Metriche di valutazione

La selezione delle metriche è stata guidata dalla necessità di catturare diverse dimensioni della qualità di un sistema di *ranking*, in piena coerenza con gli obiettivi di valutazione. Come precedentemente menzionato, la filosofia progettuale del sistema, focalizzata sull'identificazione dell'*apex threat*, conferisce un'importanza preminente alle metriche che misurano la capacità di prioritizzare correttamente la minaccia principale. Per questo motivo, le metriche sono state raggruppate in tre categorie funzionali.

Metriche "*strict*" (*focus* sull'*apex threat*)

Queste metriche sono considerate le più importanti in quanto misurano direttamente il successo del sistema nel suo obiettivo primario: fornire una valutazione rapida e accurata della minaccia più grave.

- **Precision@1 (P@1):** Misura la percentuale di volte in cui il primo risultato (*rank*=1) restituito dal sistema corrisponde esattamente all'*apex threat* (definito come il pattern con rilevanza 2 nel *Gold Standard*). Un valore elevato di P@1 indica un'eccellente capacità di prioritizzazione immediata, che è cruciale in uno scenario operativo di sicurezza. Per questa metrica, cruciale ma sensibile alla variabilità su piccoli campioni, viene inoltre calcolato l'intervallo di confidenza di Clopper-Pearson al 95%, al fine di fornire una stima più robusta e statisticamente fondata della sua reale efficacia.

- **Mean Reciprocal Rank (MRR):** Calcola la media del reciproco del *rank* del *primo* risultato pertinente (con rilevanza > 0) trovato nella lista. Questa metrica, pur essendo meno stringente di P@1, premia fortemente i sistemi che posizionano un risultato corretto nelle primissime posizioni, mentre penalizza severamente quelli che lo posizionano in basso. Un MRR vicino a 1.0 indica che il sistema è quasi sempre in grado di fornire un'indicazione utile immediatamente. Formalmente, per un insieme di N *query* (i profili di comportamento), l'MRR è calcolato come:

$$\text{MRR} = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{\text{rank}_i} \tag{6.1}$$

  dove rank_i è la posizione del primo documento pertinente per la *query* i-esima.

Metriche *multi-label* (*focus* sulla copertura)

Queste metriche forniscono una visione più ampia della qualità della lista dei risultati, valutando la sua capacità di includere un *set* di candidati pertinenti nel loro complesso. Per questo studio, la profondità di analisi è stata fissata a $k = 5$.

- **Precision@5 (P@5):** Misura la proporzione di risultati pertinenti (rilevanza > 0) all'interno dei primi 5 risultati restituiti. Valuta la "purezza" della lista dei risultati: una Precision@5 elevata significa che la lista è densa di informazioni utili e contiene pochi falsi positivi ("rumore").

- **Recall@5 (R@5):** Misura la proporzione di *tutti* i pattern pertinenti identificati nel *Gold Standard* che sono stati effettivamente recuperati dal sistema nei primi 5 risultati. Valuta la "completezza" o la capacità di copertura del sistema entro una soglia di attenzione ragionevole per un analista.

Metrica "*graded*" (*focus* sulla qualità complessiva del *ranking*)

Questa metrica rappresenta la sintesi più completa e sofisticata della *performance*, in quanto sfrutta appieno la granularità della scala di rilevanza graduata.

- **Normalized Discounted Cumulative Gain (nDCG@5):** Calcolata per $k = 5$, la nDCG valuta la qualità complessiva dell'ordinamento dei primi 5 risultati. Essa aggrega i punteggi di rilevanza dei documenti in una lista, ponderandoli in base a due fattori: il *guadagno* (gain), che cresce esponenzialmente con la rilevanza, e lo *sconto* (discount), che penalizza i documenti in posizioni più basse. Matematicamente, il *Discounted Cumulative Gain* alla posizione k è definito come:

$$\mathrm{DCG}_k = \sum_{i=1}^{k} \frac{2^{\mathrm{rel}_i} - 1}{\log_2(i+1)} \tag{6.2}$$

  dove rel_i è il punteggio di rilevanza (0, 1, o 2) del documento alla posizione i. Per ottenere un punteggio confrontabile tra diverse *query*, il DCG viene normalizzato dividendolo per l'*Ideal Discounted Cumulative Gain* (IDCG), che rappresenta il DCG massimo possibile per quella *query*:

$$\mathrm{nDCG}_k = \frac{\mathrm{DCG}_k}{\mathrm{IDCG}_k} \tag{6.3}$$

  Il risultato è un punteggio tra 0 e 1, dove 1 rappresenta un ordinamento perfetto. La nDCG@5 è considerata la metrica olistica principale per confrontare le *performance* complessive dei sistemi.

6.3 Risultati complessivi e *performance* aggregate

La prima fase dell'analisi si concentra sulla valutazione dell'efficacia assoluta del sistema proposto, al fine di rispondere al primo obiettivo di valutazione: stabilire una *baseline* quantitativa delle sue *performance*. A tal fine, sono state calcolate le metriche aggregate sull'intero dataset di 25 profili di comportamento. I risultati, che forniscono una visione d'insieme delle capacità del sistema, sono riassunti nella Tabella 6.1.

L'analisi di questi risultati aggregati rivela un quadro decisamente positivo delle *performance* del sistema. La metrica olistica **nDCG@5**, che rappresenta l'indicatore più completo della qualità del *ranking*, si attesta su un valore di **0.741**. Questo punteggio, su una scala da 0 a 1, indica che l'ordinamento prodotto dal sistema è di alta qualità, riuscendo non solo a recuperare i pattern pertinenti, ma anche a posizionarli in cima alla lista con un ordinamento che rispecchia fedelmente la loro rilevanza graduata.

Tuttavia, sono le metriche "*strict*", quelle focalizzate sull'identificazione dell'*apex threat*, a fornire le indicazioni più significative. Il sistema ha ottenuto un punteggio di **Precision@1 (P@1) pari a 0.760**. Questo significa che nel 76% dei casi (19 su 25), il sistema è stato in grado

Tabella 6.1: Performance aggregate del sistema ibrido proposto sul dataset di test (N=25). L'intervallo di confidenza (CI) per P@1 è calcolato con il metodo di Clopper-Pearson ($\alpha = 0.05$).

Metrica	**Valore Medio**
nDCG@5	**0.741**
MRR	**0.973**
P@1	**0.760**
95% CI for P@1	*[0.549, 0.906]*
Precision@5	0.632
Recall@5	0.511

di identificare correttamente la minaccia principale e di posizionarla come primo e più importante risultato. Si tratta di un risultato notevole, che suggerisce una forte affidabilità del sistema nel suo compito primario di prioritizzazione. L'intervallo di confidenza al 95%, calcolato tra 0.549 e 0.906, pur riflettendo la variabilità attesa su un campione di 25 elementi, conferma con alta probabilità statistica che la *performance* reale del sistema in questo *task* è significativamente elevata.

Ancora più impressionante è il valore del **Mean Reciprocal Rank (MRR)**, che raggiunge un eccellente **0.973**. Un punteggio così vicino a 1.0 implica che, anche nei rari casi in cui l'*apex threat* non si trova in prima posizione, un risultato pertinente (con rilevanza > 0) è quasi sempre presente nelle primissime posizioni del *ranking*. Questa caratteristica è di fondamentale importanza pratica: essa garantisce che un analista di sicurezza, ispezionando i risultati, trovi quasi istantaneamente un'indicazione utile e corretta sulla natura dell'attacco.

Le metriche di copertura, Precision@5 e Recall@5, offrono un quadro complementare. Un valore di **Precision@5 di 0.632** indica che, in media, oltre il 63% dei primi 5 risultati proposti sono pertinenti, dimostrando che la lista dei candidati è relativamente "pulita". Il **Recall@5 di 0.511** suggerisce che il sistema è in grado di recuperare, entro i primi 5 risultati, poco più della metà di tutti i pattern di attacco rilevanti identificati nel *Gold Standard*.

Per visualizzare come queste *performance* si distribuiscono al variare della profondità del *ranking*, la Figura 6.1 mostra l'andamento delle curve medie di Precisione e *Recall* per k da 1 a 5.

Come illustrato nella Figura 6.1, le due metriche mostrano un comportamento complementare e informativo. La curva di **Precision@k** (in blu) è stata calcolata in modo da misurare la purezza generale della lista a diverse profondità k, considerando pertinente qualsiasi risultato con rilevanza > 0. Questa scelta permette di evidenziare un dettaglio ulteriore rispetto alla metrica P@1. Si osserva infatti che il valore a $k = 1$ (0.96 nel grafico) è superiore al valore P@1 "*strict*" (0.760) riportato in tabella. Questo risultato è significativo: indica che nei casi in cui il sistema non posiziona l'*apex threat* al primo posto, riesce comunque a fornire un pattern secondario ma pertinente, confermando l'utilità pratica della lista fin dal primo risultato, come già suggerito dall'altissimo MRR. Dal primo al quinto risultato, la precisione media decresce gradualmente, assestandosi a 0.632, un valore che denota comunque una buona qualità complessiva.

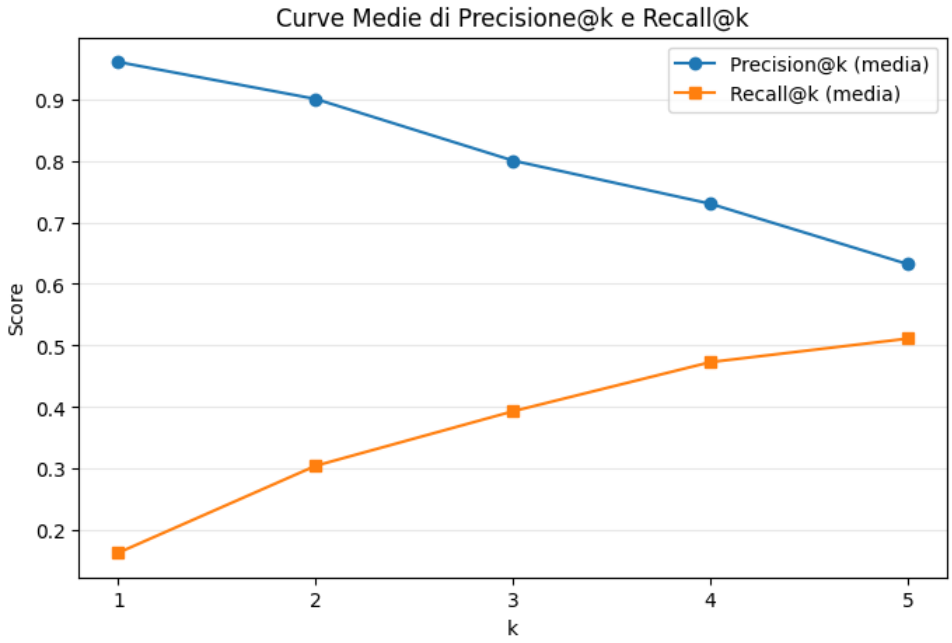

Figura 6.1: Curve medie di Precision@k e Recall@k per il sistema ibrido, calcolate sull'intero dataset di test.

Specularmente, la curva di **Recall@k** (in arancione) mostra una crescita costante e significativa, partendo da un valore basso a $k = 1$ (circa 0.16) e salendo fino a 0.511 a $k = 5$. Questo dimostra che ogni posizione aggiuntiva nel *ranking* contribuisce in modo sostanziale a recuperare una frazione maggiore dell'insieme totale dei pattern pertinenti. Il *trade-off* tra precisione e *recall* è evidente: mentre la massima pertinenza si concentra nelle primissime posizioni, è necessario esaminare l'intera *top-5* per raggiungere una copertura (*recall*) superiore al 50%.

Complessivamente, questi risultati aggregati delineano il profilo di un sistema non solo efficace nell'identificare la minaccia principale, ma anche robusto nel fornire una lista di candidati utile e ben ordinata.

6.4 Analisi dell'approccio ibrido - *ablation study*

Dopo aver stabilito una solida *baseline* di *performance* per il sistema completo, il passo successivo è validare empiricamente una delle scelte architetturali più importanti di questa tesi: l'adozione di un approccio di ricerca ibrido. Come discusso nel capitolo dello Stato dell'Arte (si veda la Sottosezione 3.2.5), la letteratura più recente converge in modo inequivocabile verso soluzioni ibride che uniscono il recupero lessicale e quello semantico per superare i limiti dei singoli approcci. Per rispondere al secondo obiettivo di valutazione – *L'implementazione di una fusione ibrida offre un vantaggio misurabile rispetto ai componenti singoli?* – e per allineare questo lavoro con le migliori pratiche emergenti, è stato condotto uno studio di ablazione (*ablation study*).

In questo esperimento, le *performance* del sistema ibrido completo sono state poste a confronto con due configurazioni ablate: una basata esclusivamente sulla componente di ricerca semantica (vettoriale) e un'altra fondata unicamente sul recupero lessicale (`TF-IDF`). La valutazione è stata eseguita sullo stesso dataset di 25 profili e rispetto allo stesso *Gold Standard*, utilizzando le metriche chiave focalizzate sulla qualità del *ranking*. I risultati di questo confronto sono presentati nella Tabella 6.2.

Tabella 6.2: Risultati dello studio di ablazione. Confronto delle performance tra il sistema ibrido completo e le sue componenti di ricerca singole (semantica e lessicale).

Metrica	Sistema Ibrido	Ricerca Vettoriale	Ricerca Lessicale
P@1	**0.760**	0.120	0.120
MRR	**0.973**	0.456	0.490
nDCG@5	**0.741**	0.275	0.254

I risultati dello studio di ablazione sono netti e non lasciano spazio a interpretazioni ambigue. Si osserva un **drastico e inequivocabile crollo delle *performance*** in entrambe le configurazioni "pure" rispetto al sistema ibrido. Questo divario non rappresenta un semplice miglioramento incrementale, ma evidenzia come la fusione sinergica dei due approcci di ricerca sia un **componente essenziale e abilitante** per l'efficacia complessiva della soluzione.

Analizzando i dati nel dettaglio, la metrica più critica, **P@1**, crolla da un robusto **0.760** nel sistema ibrido a un modesto **0.120** in entrambe le configurazioni di ablazione. Questo significa che, se presi singolarmente, sia l'approccio semantico che quello lessicale riescono a identificare l'*apex threat* come primo risultato solo nel 12% dei casi (3 su 25). Questo risultato dimostra in modo lampante che nessuno dei due approcci, da solo, è sufficientemente affidabile per il *task* di prioritizzazione delle minacce.

Una dinamica simile, sebbene con magnitudo diverse, si osserva per le altre metriche. L'**MRR**, che misura la capacità di trovare un risultato pertinente nelle prime posizioni, precipita da **0.973** a 0.456 per la ricerca vettoriale e 0.490 per quella lessicale. Questi valori, inferiori a 0.5, indicano che in media, quando si basano su un singolo motore di ricerca, bisogna scendere oltre la seconda posizione per trovare il primo risultato utile, un calo significativo di efficienza per un analista.

Infine, la metrica olistica **nDCG@5** subisce un collasso analogo, passando da **0.741** a 0.275 per la componente semantica e 0.254 per quella lessicale. Questo crollo di quasi il 65% della qualità complessiva del *ranking* conferma che i modelli "puri" non solo falliscono nel prioritizzare l'*apex threat*, ma producono anche liste di candidati globalmente meno pertinenti e peggio ordinate.

In conclusione, lo studio di ablazione fornisce una validazione empirica decisiva a favore dell'architettura ibrida. Dimostra, in pieno accordo con le tendenze evidenziate nello stato dell'arte, che la ricerca semantica e quella lessicale catturano segnali complementari e ortogonali di rilevanza. La ricerca vettoriale eccelle nel comprendere il contesto e l'intento (il "cosa"), mentre la ricerca lessicale garantisce la precisione su termini tecnici specifici (il "come"). È

solo attraverso la loro fusione intelligente, orchestrata dall'algoritmo *Reciprocal Rank Fusion*, che il sistema è in grado di capitalizzare i punti di forza di entrambi, mitigandone le reciproche debolezze e ottenendo un livello di *performance* robusto e affidabile.

6.5 Analisi comparativa con la *baseline* SOTA

Uno degli obiettivi centrali di questa valutazione è contestualizzare le *performance* del sistema proposto, confrontandole direttamente con quelle di una *baseline state-of-the-art* (SOTA). Come descritto nella Sezione 6.2.2, questo ruolo è stato affidato a `Gemini 2.5 Pro`, interrogato secondo un protocollo rigoroso per garantire un confronto equo. L'analisi che segue mira a rispondere al terzo obiettivo di valutazione: *Come si posizionano le performance del nostro sistema rispetto a quelle della baseline SOTA?*

I risultati aggregati del confronto sono presentati nella Tabella 6.3.

Tabella 6.3: Confronto delle performance aggregate tra il sistema proposto e la baseline SOTA (Gemini 2.5 Pro)

Metrica	**Sistema Proposto**	**Baseline SOTA**
nDCG@5	0.741	**0.874**
MRR	**0.973**	0.960
P@1	**0.760**	0.720
Precision@5	0.632	**0.872**
Recall@5	0.511	**0.715**

I risultati aggregati mostrano un quadro articolato e di grande interesse. Come prevedibile, il modello SOTA, in virtù della sua maggiore capacità di ragionamento e comprensione contestuale, ottiene *performance* medie superiori sulla maggior parte delle metriche, in particolare su quelle che valutano la qualità e la copertura complessiva della lista. La *baseline* SOTA raggiunge un eccellente **nDCG@5 di 0.874**, superando il sistema proposto di circa 13 punti percentuali. Mostra inoltre una **Precision@5 (0.872)** e un **Recall@5 (0.715)** nettamente superiori, a indicare che il modello più grande è in grado di produrre una lista *top-5* più densa di risultati pertinenti e di coprire una porzione maggiore del *ground truth*.

Un risultato sorprendente e di notevole importanza emerge, però, dall'analisi delle metriche "*strict*", quelle focalizzate sul compito primario di identificare l'*apex threat*. Il sistema proposto non solo si dimostra estremamente competitivo, ma **supera la *baseline* SOTA** sia in termini di **P@1 (0.760 vs 0.720)** sia di **MRR (0.973 vs 0.960)**. Questo risultato è significativo: suggerisce che, per il *task* altamente specifico di prioritizzazione della minaccia più grave, l'architettura specializzata del sistema proposto – che disaccoppia l'interpretazione semantica dal *retrieval* ibrido – è più efficace di un approccio monolitico *end-to-end*, anche se quest'ultimo è basato su un modello LLM di frontiera. La *pipeline* specializzata sembra guidare l'analisi in modo più robusto verso l'identificazione della minaccia principale.

È fondamentale, a questo punto, analizzare questi risultati non solo in termini di *performance* finali, ma anche alla luce delle profonde differenze architetturali e dei limiti intrinseci dell'approccio utilizzato per la *baseline* SOTA. Sebbene il punteggio nDCG@5 medio sia superiore, la metodologia di fornire l'intero corpus CAPEC *in-context* a un modello LLM di grandi dimensioni per un *task end-to-end* nasconde inefficienze e fragilità significative, che ne rendono l'applicazione in uno scenario reale problematica e che spiegano le sue *performance* non ottimali nel compito cruciale della prioritizzazione.

I limiti principali di questo approccio monolitico sono:

- **Costo economico proibitivo:** La principale barriera all'adozione di un simile approccio SOTA in uno scenario reale è di natura economica. I modelli di frontiera come `Gemini 2.5 Pro` sono accessibili tramite API a pagamento, il cui costo è direttamente proporzionale al numero di *token* elaborati in *input* e in *output*. Nel nostro protocollo sperimentale, fornire l'intero corpus CAPEC (versione 3.9) *in-context* significa inserire nel *prompt* circa **820,000 *token*** per *ogni singola richiesta*. Un simile volume di dati, moltiplicato per le migliaia di eventi di sicurezza che un sistema reale deve analizzare quotidianamente, si tradurrebbe in costi operativi insostenibili, rendendo l'approccio economicamente irrealizzabile su larga scala. Al contrario, l'architettura del sistema proposto, basata su un modello *open source* eseguito localmente, sposta questo "costo" in una fase di indicizzazione *una tantum* e offline. L'inferenza LLM *online* opera su un *input* di poche centinaia di *token* (il solo log), abbattendo i costi marginali per inferenza a un livello prossimo allo zero e garantendo la sostenibilità economica della soluzione.

- **Il problema del "*Lost in the Middle*":** La letteratura scientifica ha ampiamente dimostrato che le *performance* dei modelli Transformer degradano quando devono trovare informazioni rilevanti poste al centro di un lungo contesto. Questo fenomeno, noto come "*Lost in the Middle*", è stato quantificato empiricamente da Liu et al. [90], i quali hanno osservato una caratteristica curva di *performance* a "U", dove i modelli accedono con maggiore affidabilità alle informazioni poste all'inizio o alla fine del *prompt*, ma falliscono significativamente nel recuperare quelle "sepolte" nel mezzo. Fornire l'intero corpus CAPEC *in-context* espone la *baseline* SOTA a questo esatto problema, spiegando la sua potenziale incapacità di identificare il pattern più specifico se questo non si trova in una posizione privilegiata. Il sistema proposto mitiga questo problema a monte, utilizzando motori di ricerca specializzati che agiscono come un primo filtro, riducendo drasticamente il "pagliaio" informativo.

- **Mancanza di ancoraggio lessicale robusto:** Un LLM generalista, anche se potente, ragiona primariamente a livello semantico. Può fallire nel dare il giusto peso a parole chiave, acronimi o comandi tecnici rari ma estremamente discriminanti. Ad esempio, la presenza della stringa esatta "wget" seguita da "chmod +x" è un segnale fortissimo che un approccio lessicale come `TF-IDF` cattura con precisione. La *baseline* SOTA, ragionando

olisticamente, potrebbe perdere questa sfumatura a favore di un'interpretazione semantica più generica. Il sistema proposto, tramite la fusione RRF, garantisce che questo forte segnale lessicale non venga mai ignorato, "ancorando" il risultato alla realtà tecnica del log.

- **Fragilità e mancanza di determinismo:** L'*output* di un LLM in un *task* di *zero-shot reasoning* su un contesto così vasto può essere fragile e meno deterministico. Sebbene la temperatura sia stata impostata a 0, piccole variazioni nel *prompt* o nella struttura del contesto potrebbero portare a risultati diversi. L'architettura del sistema proposto è intrinsecamente più robusta: la fase di recupero dell'informazione è deterministica e riproducibile, e il ruolo dell'LLM è circoscritto a un compito di interpretazione e strutturazione dei dati, riducendone la potenziale variabilità.

Queste considerazioni sono fondamentali per interpretare correttamente i risultati delle metriche "*strict*". Il fatto che il sistema proposto, pur essendo computazionalmente più leggero e architetturalmente più semplice, superi la *baseline* SOTA su P@1 e MRR, non è casuale. È la diretta conseguenza di un'architettura ingegnerizzata che disaccoppia i compiti e utilizza gli strumenti giusti per ogni fase: l'LLM per la comprensione del linguaggio naturale e i motori di ricerca specializzati per un recupero efficiente e robusto. Questo suggerisce che, per il *task* specifico di identificare l'*apex threat*, un approccio specializzato è superiore a un approccio monolitico basato puramente sul ragionamento *in-context*, anche se quest'ultimo impiega un modello di frontiera.

Per investigare più a fondo questa dinamica a livello dei singoli casi, la Figura 6.2 visualizza la differenza di *performance* (Delta nDCG) per ogni singola sessione.

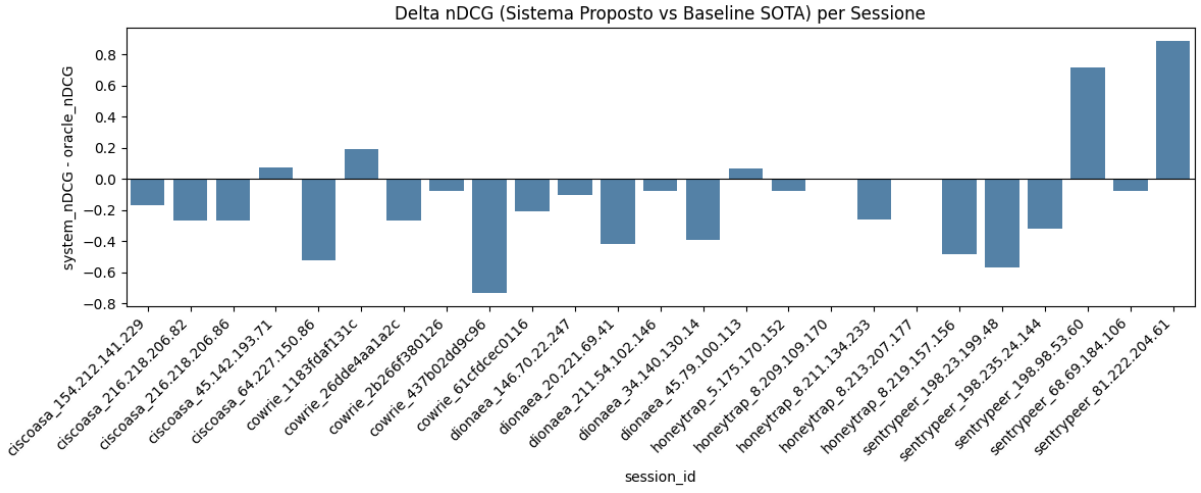

Figura 6.2: Differenza di performance (Delta nDCG@5) tra il sistema proposto e la baseline SOTA per ciascuna delle 25 sessioni. Barre positive indicano una performance superiore del sistema proposto.

Il grafico del Delta nDCG@5 conferma l'analisi aggregata ma aggiunge una profondità cruciale. La maggior parte delle barre si trova al di sotto della linea dello zero, a conferma della superiorità

media della *baseline* SOTA nella qualità complessiva del *ranking*. Si osservano, tuttavia, diversi casi degni di nota. In due sessioni (‘honeytrap_8.209.109.170‘ e ‘honeytrap_8.213.207.177‘), entrambi i sistemi hanno raggiunto una *performance* perfetta (nDCG@5 = 1.0), risultando in un delta nullo. Più importante, in ben cinque sessioni, il sistema proposto ha ottenuto una *performance* superiore a quella della *baseline* SOTA (barre positive). I casi più eclatanti sono ‘sentrypeer_198.98.53.60‘ (+0.720) e ‘sentrypeer_81.222.204.61‘ (+0.890), dove il sistema proposto ha ottenuto un nDCG@5 nettamente superiore. Questi casi, che verranno analizzati in dettaglio nella sezione qualitativa, suggeriscono che esistono scenari specifici in cui l’approccio ibrido e specializzato del sistema proposto riesce a cogliere sfumature che sfuggono anche a un modello SOTA.

Infine, per determinare se la differenza media di *performance* osservata tra i due sistemi sia statisticamente rilevante, è stato applicato il **test non parametrico dei ranghi con segno di Wilcoxon** sulla distribuzione dei 25 punteggi nDCG@5. Questo test è appropriato per campioni appaiati di piccole dimensioni senza assumere una distribuzione normale dei dati. I risultati del test sono i seguenti:

- **Statistica del test (W):** 58.0
- **P-value:** 0.0148

Dato un livello di significatività $\alpha = 0.05$, il *p-value* ottenuto (**0.0148**) è inferiore a α. Si può quindi rigettare l’ipotesi nulla e concludere che **la differenza tra le mediane delle *performance* dei due sistemi è statisticamente significativa**. In sintesi, sebbene la *baseline* SOTA si dimostri superiore nella generazione di una lista di candidati complessivamente di alta qualità, il sistema proposto si rivela non solo competitivo, ma addirittura più efficace nel compito cruciale di prioritizzare la minaccia principale, dimostrando il valore della sua architettura specializzata.

6.6 Analisi dettagliata delle *performance* per tipo di honeypot

Dopo aver valutato le *performance* aggregate del sistema, un’analisi più profonda richiede di disaggregare i risultati per indagare come il sistema si comporta sulle diverse tipologie di dati di *input*. Questo passaggio è fondamentale per rispondere al quarto obiettivo di valutazione: *Le performance del sistema varino in funzione del tipo di honeypot che ha generato il log?* L’analisi che segue esplora questa domanda attraverso diverse lenti, esaminando i profili di *performance* generali, l’andamento della precisione e la distribuzione dei risultati per ogni singola sessione.

6.6.1 Profili di *performance* comparati

Per ottenere una visione d’insieme, qualitativa e comparativa, delle *performance* tra le cinque categorie di honeypot, è stato generato un profilo di *performance* normalizzato, visualizzato tramite un *radar plot* nella Figura 6.3. In questo grafico, ogni asse corrisponde a una metrica

chiave (P@1, MRR, nDCG@5, Recall@5, Precision@5). I valori medi di ciascuna metrica per ogni tipo di honeypot sono stati normalizzati su una scala da 0 a 1, dove 1 rappresenta la *performance* migliore osservata per quella specifica metrica tra tutte le categorie, e 0 la peggiore. Questo processo di normalizzazione permette di astrarre dai valori assoluti e di confrontare le "forme" dei profili di *performance*, evidenziando i punti di forza e di debolezza relativi di ciascuna categoria.

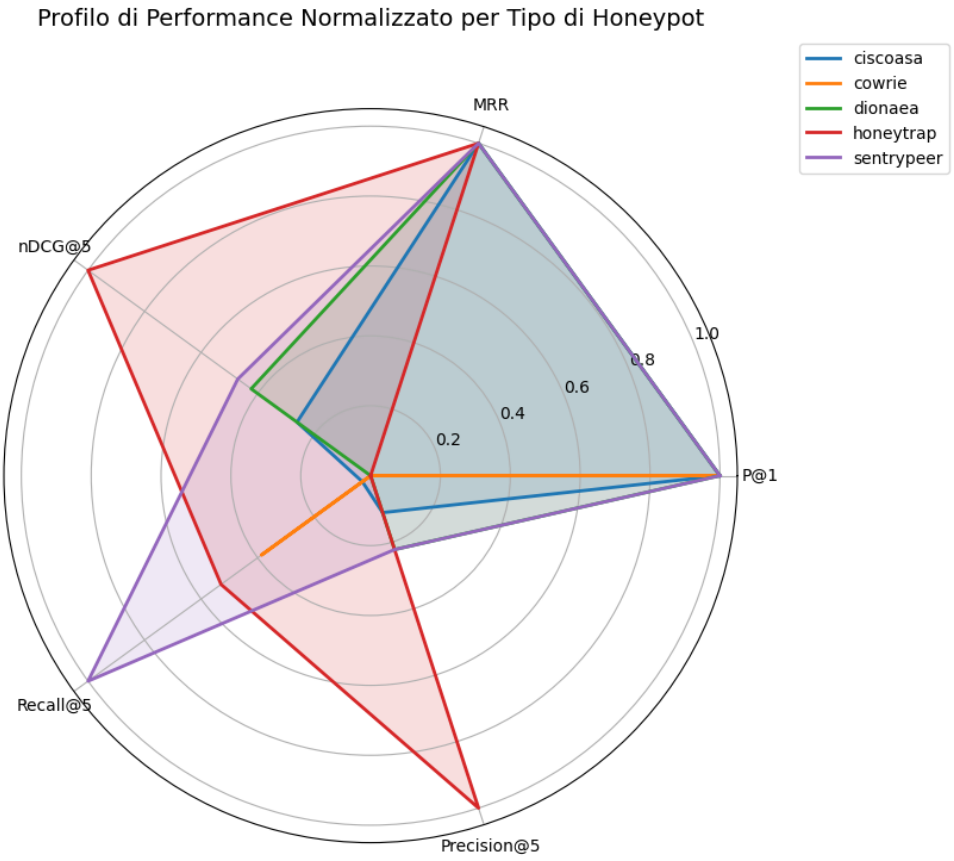

Figura 6.3: Profilo di performance normalizzato per tipo di honeypot. Ogni asse rappresenta una metrica chiave, normalizzata su una scala da 0 a 1 per permettere il confronto delle forme dei profili.

L'analisi della Figura 6.3 rivela immediatamente delle eterogeneità sostanziali:

- **`Honeytrap` (profilo rosso):** Mostra un profilo con un comportamento duale e molto interessante. Da un lato, domina in modo assoluto sull'asse della **Precision@5**, raggiungendo il massimo normalizzato. Questo indica che le sue liste di risultati *top-5* sono le più "pulite" e dense di pattern pertinenti. Dall'altro lato, mostra la sua principale debolezza sull'asse del **P@1**, dove il suo profilo collassa verso il centro, registrando il valore normalizzato di 0. Questo significa che, pur essendo eccellente nel produrre una buona lista di candidati, è il meno efficace tra tutti i tipi di honeypot nell'identificare l'*apex threat* come primo risultato.
- **`SentryPeer` (profilo viola):** Presenta un profilo molto ampio e robusto, che si distingue per il raggiungimento del valore massimo assoluto sull'asse del **Recall@5**. Questo

suggerisce che, per i log di tipo VoIP, il sistema è particolarmente abile nel recuperare una grande frazione del totale dei pattern pertinenti entro i primi 5 risultati. Mantiene inoltre *performance* massime anche su P@1 e MRR, indicando un'eccellente e affidabile capacità di prioritizzazione.

- **`Dionaea` (profilo verde) e `CiscoASA` (profilo blu):** Mostrano profili molto simili tra loro, caratterizzati da un ottimo bilanciamento. Entrambi raggiungono il massimo su P@1 e MRR, denotando una forte affidabilità nell'identificare l'*apex threat*. La loro estensione è più contenuta sugli assi di Precision@5 e Recall@5, indicando che, pur essendo molto precisi nel prioritizzare, le loro liste di risultati tendono ad essere leggermente meno complete e pure rispetto a quelle di `Honeytrap` o `SentryPeer`.

- **`Cowrie` (profilo arancione):** Evidenzia il profilo più contratto, con la *performance* normalizzata più bassa sull'asse del **MRR**. Questo dato è particolarmente significativo e conferma le sfide poste da questa tipologia di log. Avere un MRR normalizzato a 0 significa che, tra tutti i tipi di honeypot, `Cowrie` è quello per cui, in media, si deve scendere più in basso nel *ranking* per trovare il primo risultato pertinente. Questo, unito a *performance* non massime su quasi tutti gli altri assi, suggerisce una difficoltà sistemica nell'analizzare e classificare correttamente le complesse e ambigue sessioni interattive catturate da questo honeypot.

In sintesi, l'analisi dei profili suggerisce una prima tassonomia: il sistema mostra una *performance* duale su `Honeytrap` (ottima purezza, debole prioritizzazione), è molto robusto e completo su `SentryPeer`, affidabile e preciso su `Dionaea` e `CiscoASA`, e affronta la sfida analitica maggiore con i log di `Cowrie`.

6.6.2 Dinamiche di *ranking* e andamento della precisione

Se il *radar plot* fornisce una visione d'insieme statica dei profili di *performance*, l'analisi dell'andamento della *Precision@k* per ciascun tipo di honeypot permette di comprendere le dinamiche interne del processo di *ranking*. La Figura 6.4 illustra come la "purezza" della lista dei risultati (ovvero la proporzione di candidati pertinenti) varia man mano che si scende dalla prima alla quinta posizione.

L'analisi comparativa delle curve rivela comportamenti marcatamente differenti, che rafforzano e approfondiscono le osservazioni emerse dal *radar plot*:

- **`Honeytrap` (curva rossa):** Mostra un andamento eccezionale. La precisione è perfetta (1.0) per le prime tre posizioni, indicando che i primi tre candidati restituiti per i log `Honeytrap` sono *sempre* pertinenti. La *performance* degrada solo leggermente a $k = 4$ (0.95) e $k = 5$ (0.88). Questo profilo di "lenta degradazione" è tipico di un sistema che ha un'alta confidenza nell'identificare un *set* di candidati molto forte. Spiega perché questa

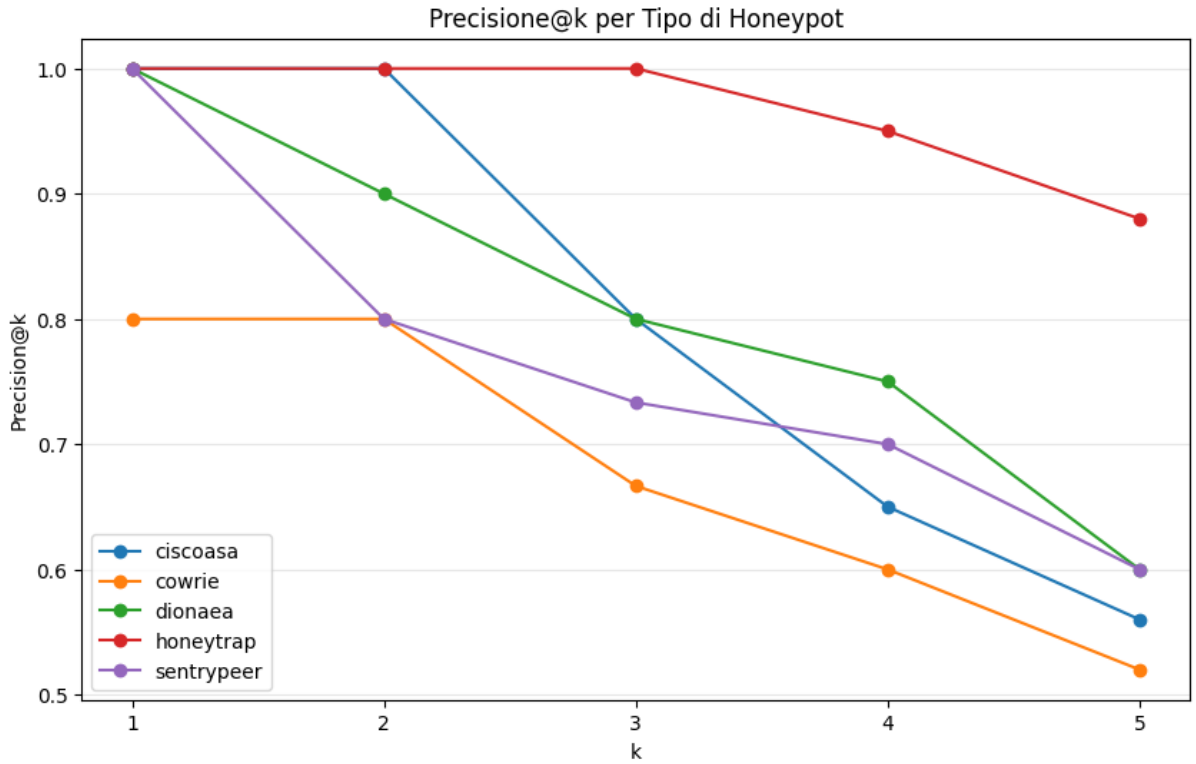

Figura 6.4: Andamento della Precision@k media per ciascun tipo di honeypot. La metrica Precision@k considera pertinente qualsiasi risultato con rilevanza > 0.

categoria domina sull'asse della Precision@5 nel *radar plot*: la sua lista di risultati è di altissima qualità in profondità.

- **`Dionaea` (curva verde):** Anche `Dionaea` mostra una *performance* molto robusta. Parte da una precisione perfetta a $k = 1$, che si mantiene alta a $k = 2$ (0.9) per poi decrescere più gradualmente. Questo suggerisce che il sistema è molto affidabile nell'identificare i primi candidati corretti, anche se con una confidenza leggermente inferiore rispetto a `Honeytrap` nelle posizioni più basse.

- **`CiscoASA` (curva blu) e `SentryPeer` (curva viola):** Queste due categorie mostrano un comportamento simile tra loro. Partono entrambe con una precisione perfetta a $k = 1$, ma subiscono una discesa più rapida rispetto a `Honeytrap` e `Dionaea`. Ad esempio, a $k = 3$, la loro precisione si attesta intorno a 0.8 e 0.73 rispettivamente. Questo andamento suggerisce che, pur essendo molto efficaci nell'identificare l'*apex threat* (come conferma il P@1 elevato), il sistema ha più difficoltà a mantenere la stessa purezza del *ranking* nelle posizioni successive.

- **`Cowrie` (curva arancione):** Questa categoria, come anticipato, mostra la curva più problematica. È l'unica a non partire da una precisione perfetta a $k = 1$ (il suo valore è 0.8). La curva, inoltre, mostra il calo più netto e costante, raggiungendo a $k = 5$ il valore di precisione più basso di tutto il gruppo (0.52). Questo andamento conferma in

modo quantitativo la maggiore difficoltà analitica posta dalle sessioni interattive. Per i log `Cowrie`, il sistema non solo ha una probabilità leggermente inferiore di trovare subito un risultato pertinente, ma la qualità della sua lista di candidati si degrada più rapidamente rispetto a tutte le altre categorie.

In sintesi, l'analisi delle curve di precisione permette di classificare il comportamento del sistema su una scala di "confidenza di *ranking*". Il sistema dimostra una confidenza massima e stabile sui log `Honeytrap`, una confidenza alta ma decrescente su `Dionaea`, `CiscoASA` e `SentryPeer`, e una confidenza inferiore e più fragile sui log `Cowrie`, confermando che la complessità e l'ambiguità semantica dei dati di *input* sono il fattore principale che influenza la qualità del *ranking*.

6.6.3 Distribuzione delle *performance* a livello di sessione

Dopo aver analizzato i profili medi, è utile scendere al livello delle singole 25 sessioni di test per comprendere la distribuzione e la variabilità delle *performance*. La Figura 6.5 visualizza ogni sessione come un punto su un piano definito dalle due metriche "*strict*" più importanti: l'MRR (asse x) e la metrica olistica nDCG@5 (asse y). Ogni punto è colorato in base al tipo di honeypot e la sua dimensione è proporzionale alla Precision@5, fornendo una visione multi-dimensionale densa di informazioni.

Dall'analisi dello *scatter plot* emergono tre osservazioni fondamentali:

1. **Dominanza dell'MRR ed efficacia nella prioritizzazione:** La prima osservazione macroscopica è l'estremo affollamento dei punti sul lato destro del grafico. Ben 24 delle 25 sessioni si posizionano su un valore di **MRR pari o molto vicino a 1.0**. Questo dato è estremamente significativo e conferma, a livello granulare, una delle principali forze del sistema: la sua quasi infallibile capacità di posizionare un risultato pertinente (con rilevanza > 0) come primo o, al massimo, secondo candidato. Questa robustezza nella prioritizzazione iniziale è un risultato diretto della sinergia dell'approccio ibrido, che riesce quasi sempre a catturare un segnale di rilevanza forte, sia esso semantico o lessicale.

2. **Variabilità della qualità complessiva del *ranking*:** A fronte di un MRR quasi costantemente perfetto, si osserva una **dispersione molto più ampia lungo l'asse verticale del nDCG@5**. I punteggi variano da valori perfetti (1.0) fino a valori molto bassi (~ 0.2). Poiché l'MRR è quasi sempre alto, questa variabilità non è causata dal fallimento nel trovare risultati pertinenti, ma dalla capacità del sistema di *ordinarli correttamente* in base alla loro rilevanza graduata. Un nDCG@5 basso a fronte di un MRR alto significa che il sistema ha trovato un pattern plausibile (rilevanza 1) come primo risultato, ma ha posizionato l'*apex threat* (rilevanza 2) in una posizione inferiore, o non l'ha incluso affatto nella *top-5*.

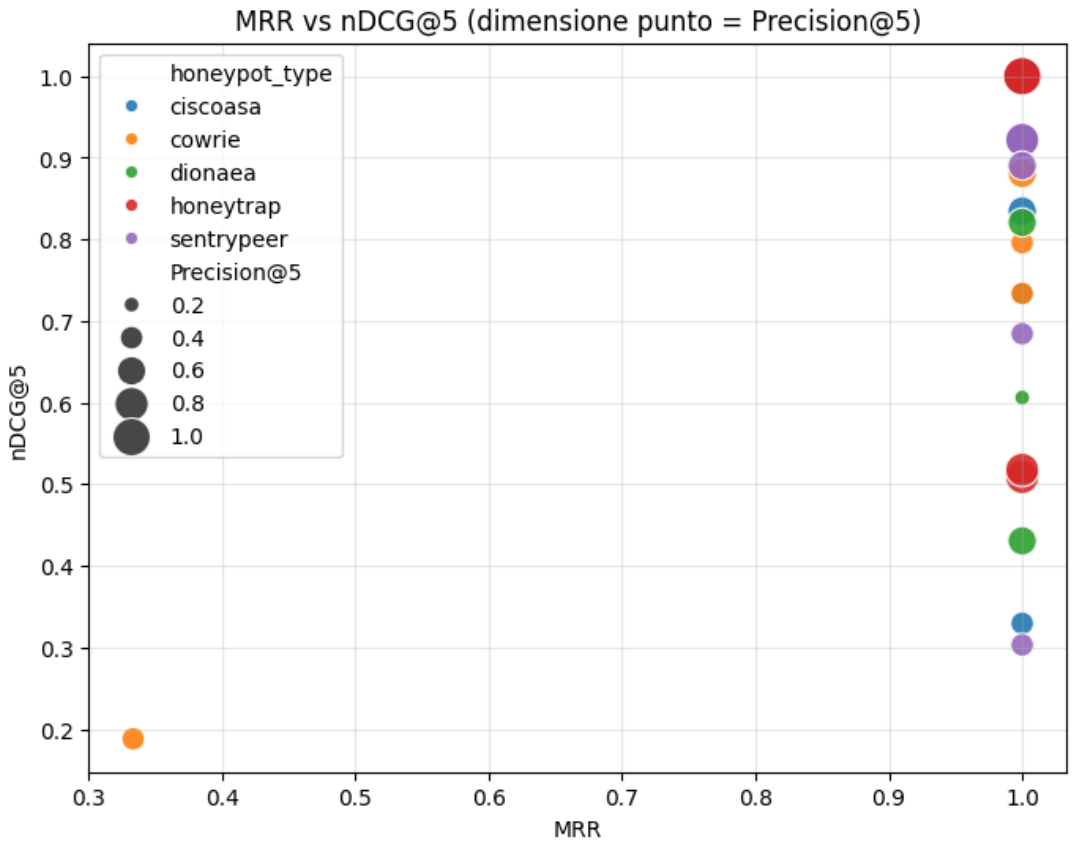

Figura 6.5: Correlazione tra MRR e nDCG@5 per ogni sessione, raggruppata per tipo di honeypot. La dimensione di ogni punto è proporzionale al valore di Precision@5.

3. **Clusterizzazione per tipo di honeypot:** La colorazione dei punti rivela *pattern* di *performance* distinti per le diverse categorie.

    - I log **`Cowrie`** (arancione) confermano la loro natura problematica: includono l'unico vero ***outlier*** del dataset ('cowrie_437b02dd9c96'), l'unico punto con un MRR basso (0.33) e un nDCG@5 infimo (0.19). Questo rappresenta l'unico caso in cui il sistema ha fallito quasi completamente nel fornire un risultato utile nelle prime posizioni. Gli altri punti `Cowrie`, pur avendo MRR perfetto, mostrano una certa dispersione sul nDCG@5.
    - I log **`Honeytrap`** (rosso) e **`Dionaea`** (verde) mostrano un comportamento simile, con MRR sempre a 1.0 ma una notevole variabilità verticale, indicando che la sfida per questi log non è trovare un pattern, ma distinguere quello principale da quelli secondari.
    - I punti **`CiscoASA`** (blu) e, in particolare, **`SentryPeer`** (viola) formano dei *cluster* più compatti e tendenzialmente posizionati più in alto sull'asse del nDCG@5, suggerendo una *performance* mediamente più stabile e di alta qualità per queste categorie.

In conclusione, l'analisi disaggregata per sessione conferma in modo definitivo che la sfida principale per il sistema non risiede tanto nel recupero di informazioni pertinenti – un compito in cui eccelle – quanto nella fine distinzione gerarchica tra minaccia primaria e secondaria. La variabilità delle *performance* è fortemente legata alla tipologia del log, con le sessioni interattive di `Cowrie` che si confermano come il caso d'uso più complesso e sfidante per l'intera *pipeline* di analisi.

6.7 Analisi qualitativa e casi studio

L'analisi quantitativa ha fornito una misura oggettiva delle *performance*, ma per comprendere appieno le dinamiche interne del sistema e il "perché" dei risultati osservati, è necessario scendere nel dettaglio dei singoli processi decisionali. In questa sezione vengono analizzati quattro casi studio rappresentativi, selezionati per illustrare diversi aspetti del comportamento del sistema.

6.7.1 Caso studio 1: il valore della sinergia ibrida (successo inequivocabile)

Il primo caso analizzato riguarda la sessione `honeytrap_8.213.207.177`. Questo esempio è stato selezionato perché rappresenta una validazione empirica perfetta della teoria alla base della *Reciprocal Rank Fusion*: mostra una situazione in cui entrambi i singoli motori di ricerca falliscono nel prioritizzare l'*apex threat*, ma la loro combinazione produce il risultato corretto.

Analisi delle evidenze nel log. Il log grezzo, proveniente da un sensore `Honeytrap`, registra 7 tentativi di connessione TCP provenienti da un singolo indirizzo IP verso la porta di destinazione **11019**. L'elemento discriminante, evidenziato dal modulo di *preprocessing*, non è il volume delle connessioni, ma il loro contenuto. Ben 6 di queste connessioni contengono *payload* dati. Il sistema ha inferito protocolli eterogenei: `["HTTP Request", "TLS Handshake", "TPKT/RDP"]`. I *payload* grezzi mostrano stringhe complesse come "`v d-+...h2http/1.1spdy/3.1`" e richieste HTTP strutturate con *User-Agent* specifici ("`Mozilla/5.0...`"). Questa evidenza suggerisce un'intenzione che va oltre la semplice scansione di porte (*port scanning*): l'attaccante sta "dialogando" con il servizio utilizzando diversi dialetti protocollari per sollecitarne una risposta e identificarne la natura precisa.

Interpretazione semantica dell'LLM. L'interprete LLM (`Mistral-7B`) coglie perfettamente questa sfumatura. Nella sua analisi JSON, identifica il pattern come "*Reconnaissance: Application Fingerprinting*". La giustificazione prodotta è puntuale:

> *The presence of six connection attempts targeting a single port with multiple inferred application-layer payloads ('HTTP Request', 'TLS Handshake') is strong evidence of an application fingerprinting attempt.*

L'LLM estrae inoltre *keyword* tecniche cruciali come "`application fingerprinting`", "`service fingerprinting`" e "`active probing`", orientando la ricerca verso il concetto di identificazione attiva del servizio.

Il "conflitto" tra i motori di ricerca. Nonostante l'ottima interpretazione dell'LLM, la fase di *retrieval* rivela i limiti dei singoli approcci. Il *Gold Standard* definisce come *apex threat* (Rilevanza 2) il pattern **`CAPEC-541`** (***Application Fingerprinting***). Tuttavia, analizzando la tabella di *debug*, emerge che nessuno dei due indici lo ha posizionato al primo posto:

- **Ricerca semantica:** Ha assegnato a `CAPEC-541` solo il *rank* #7. Al primo posto (#1) ha preferito `CAPEC-310` *(*Scanning for Vulnerable Software*)*. Sebbene semanticamente correlato, `CAPEC-310` è un pattern più generico e meno accurato per descrivere l'azione specifica di "*fingerprinting*" su una singola porta. Il modello vettoriale sembra aver privilegiato il concetto astratto di "scansione" rispetto alla specificità tecnica.
- **Ricerca lessicale (TF-IDF):** Ha assegnato a `CAPEC-541` il *rank* #3. Al secondo posto (#2) ha posizionato `CAPEC-287` *(*TCP SYN Scan*)*. Qui, la ricerca lessicale è stata probabilmente tratta in inganno dalla frequenza di termini di rete generici ("TCP", "connection") presenti nella descrizione, favorendo un pattern di scansione di basso livello non coerente con l'uso di *payload* applicativi complessi.

La risoluzione tramite RRF. Se il sistema si fosse basato su un approccio "Solo semantico" o "Solo lessicale", avrebbe fallito nella metrica P@1. È qui che interviene l'algoritmo RRF. L'algoritmo penalizza i documenti che hanno un *ranking* molto basso in una delle due liste e premia quelli che mostrano una **consistenza** di posizionamento.

- Il candidato **`CAPEC-310`** (vincitore semantico) è stato penalizzato dal suo scarso riscontro lessicale (*rank* #11), ottenendo un punteggio RRF di 0.030478.
- Il candidato **`CAPEC-287`** (forte lessicalmente) è stato penalizzato dal suo basso *rank* semantico (#9), ottenendo un punteggio RRF di 0.030622.
- Il candidato **`CAPEC-541`**, pur non essendo primo in nessuna lista, era **consistente**: *rank* #7 (semantico) e *rank* #3 (lessicale). La fusione di questi due posizionamenti solidi ha generato il punteggio RRF più alto (**0.030798**), proiettandolo al ***rank* Finale #1**.

Validazione del risultato. Il risultato finale del sistema è una *top-5* eccezionalmente accurata:

1. **`CAPEC-541`** (*Apex Threat* - Rilevanza 2)
2. **`CAPEC-287`** (Pertinente - Rilevanza 1)
3. **`CAPEC-310`** (Pertinente - Rilevanza 1)
4. **`CAPEC-224`** (Pertinente - Rilevanza 1)
5. **`CAPEC-300`** (Pertinente - Rilevanza 1)

Tutti i pattern restituiti sono presenti nel *Gold Standard*. Inoltre, il confronto con l'*output* dell'oracolo (`Gemini 2.5 Pro`) conferma pienamente la correttezza dell'analisi: anche il modello SOTA ha identificato `CAPEC-541` come *apex threat*, con una giustificazione quasi identica ("*attacker sent probes using multiple protocols... to elicit responses*"). Questo caso studio dimostra come l'architettura proposta sia in grado di raggiungere la stessa accuratezza di un modello SOTA, ma attraverso un processo ingegnerizzato che mitiga attivamente le debolezze dei singoli componenti di *retrieval*.

6.7.2 Caso studio 2: ambiguità tassonomica e recupero parziale (quasi-successo)

Il secondo caso analizzato, relativo alla sessione `ciscoasa_64.227.150.86`, offre uno spunto di riflessione cruciale sulla gestione delle sfumature semantiche e sulla distinzione tra P@1 e utilità pratica. In questo scenario, il sistema non riesce a posizionare l'*apex threat* al primo posto (P@1=0), ma fornisce comunque al primo *rank* un risultato altamente pertinente (Rilevanza 1).

Analisi delle evidenze nel log. L'*input*, proveniente da un honeypot `CiscoASA`, è estremamente sintetico ma significativo. Vengono registrate 3 richieste HTTP uniche: `["GET /aaa9", "GET /aab8", "GET /"]`. A differenza di una scansione di vulnerabilità standard che cerca file noti (es. `.env`, `config`), qui l'attaccante sta richiedendo percorsi apparentemente casuali o generati proceduralmente (`/aaa9`, `/aab8`). Questa tecnica è tipica del *fuzzing* o della ricerca di percorsi amministrativi non pubblicati ("*hidden paths*") che potrebbero essere stati lasciati attivi per errore su un dispositivo di rete.

Interpretazione semantica dell'LLM. L'LLM decodifica correttamente l'intento, dimostrando ancora una volta la sua capacità di astrazione.

- **Pattern inferito:** "*Reconnaissance: Specific Administrative Path Scanning*"
- **Giustificazione:** "*The sequence of requests targeting the '/aaa9' and '/aab8' directories is indicative of an attempt to discover administrative paths on a Cisco ASA device.*"

L'interpretazione è ineccepibile: l'LLM ha capito che non si tratta di un attacco generico, ma di un tentativo mirato di scoprire risorse non standard.

La sfida del *matching*: "*Detect*" vs "*Use*". Il *Gold Standard* identifica come *apex threat* (Rilevanza 2) il pattern **`CAPEC-143`** **(*Detect Unpublicized Web Pages*)**. Questo pattern descrive esattamente l'azione di enumerare e indovinare URL per trovare pagine nascoste. Tuttavia, il sistema ha classificato al *rank* #1 il pattern **`CAPEC-36`** **(*Using Unpublished Interfaces or Functionality*)**, con una confidenza relativa del 100%. Analizzando la tabella di *debug*, si nota che CAPEC-36 ha dominato la classifica semantica (*rank* #1). Questo è avvenuto probabilmente perché la descrizione generata dall'LLM ("*Specific Administrative Path Scanning*") e le *keyword* estratte ("*administrative path*", "*enumeration*") hanno una sovrapposizione semantica fortissima con la definizione di `CAPEC-36`, che riguarda l'interazione con interfacce non documentate.

Esiste una sottile differenza tassonomica tra i due: `CAPEC-143` si concentra sull'*atto della scoperta* (*Detect*), mentre `CAPEC-36` si concentra sull'*utilizzo* (*Use*) di ciò che è stato scoperto o che si presume esista. Dato che il log mostra delle richieste `GET` (tentativi di utilizzo/accesso), il confine è labile. Il sistema ha privilegiato l'interpretazione "tentativo di utilizzo di interfaccia nascosta" (`CAPEC-36`), mentre l'esperto umano ha privilegiato l'interpretazione "tecnica di scoperta" (`CAPEC-143`).

Confronto con il SOTA e conclusioni. È interessante notare che anche la *baseline* SOTA (`Gemini 2.5 Pro`) ha "fallito" nel prioritizzare l'*apex threat* esatto. L'*oracolo* ha posizionato al primo posto **`CAPEC-170`** **(*Web Application Fingerprinting*)**, relegando l'*apex threat* corretto (`CAPEC-143`) alla seconda posizione. Il nostro sistema, pur mancando `CAPEC-143` nella *top-5* (un limite di *recall*), ha fornito:

1. **CAPEC-36** (*rank* 1, Rilevanza 1 - Pertinente)
2. **CAPEC-170** (*rank* 2, Rilevanza 1 - Pertinente)

Questo caso viene definito un "quasi-successo" perché, sebbene la metrica rigorosa P@1 sia 0, l'analista riceve comunque al primo posto un pattern (`CAPEC-36`) che descrive correttamente la natura della minaccia (interazione con interfacce nascoste). Dimostra che, in presenza di ambiguità tassonomiche nel framework CAPEC, il sistema tende a convergere verso concetti semanticamente adiacenti e operativamente utili.

6.7.3 Caso studio 3: errata specificità e ambiguità del contesto (fallimento istruttivo)

Il terzo caso, relativo alla sessione `cowrie_437b02dd9c96`, rappresenta l'unico vero *outlier* negativo evidenziato nello *scatter plot* (MRR=0.33, nDCG@5=0.19). L'analisi di questo fallimento è fondamentale per comprendere i limiti attuali del sistema nella gestione di attacchi che utilizzano comandi di sistema standard in contesti ambigui.

Analisi delle evidenze nel log. La sessione, catturata da un honeypot `Cowrie`, mostra un attaccante che ottiene l'accesso *root* ed esegue una serie di 9 comandi eterogenei. La sequenza include comandi Linux standard (`ifconfig`, `uname -a`, `cat /proc/cpuinfo`), ma anche comandi specifici per router MikroTik (`/ip cloud print`) e controlli su dispositivi *hardware* specifici (`/dev/ttyGSM*`, `/dev/modem*`). L'intento è chiaramente una ricognizione manuale (*footprinting*) volta a identificare se il sistema compromesso è un *server* generico o un dispositivo IoT/Router di valore, e a verificare la presenza di processi concorrenti (`ps | grep '[Mm]iner'`).

Interpretazione semantica dell'LLM. L'LLM analizza correttamente le azioni, classificando l'evento come "*Reconnaissance: System Scanning*". La descrizione generata è accurata:

> *The adversary is using multiple commands to gather information about the operating system, network interfaces... This information gathering could be used for further exploitation.*

Tuttavia, l'LLM non coglie esplicitamente la natura ibrida del target (Linux/MikroTik) e genera *keyword* tecniche piuttosto generiche come "`system scanning`", "`post-exploitation`" e "`file enumeration`".

L'anomalia nel *retrieval*: l'errore di CAPEC-529. Il *Gold Standard* identifica come *apex threat* **CAPEC-169 (*Footprinting*)**, un pattern di alto livello ("Meta") che comprende tutte le attività di ricognizione osservate. Il sistema, invece, posiziona al *rank* #1 il pattern **CAPEC-529 (*Malware-Directed Internal Reconnaissance*)**. Perché questo errore? `CAPEC-529` descrive

un *malware* che, una volta infettato un *host*, esegue autonomamente comandi di ricognizione. Sebbene semanticamente simile (i comandi eseguiti sono gli stessi), il *contesto* è errato: qui l'azione è manuale (sessione interattiva), non automatizzata da *malware*. Tuttavia, la descrizione dell'LLM ("*post-exploitation*", "*information gathering*") ha creato una forte risonanza semantica con la definizione di `CAPEC-529` nel database vettoriale (*rank* semantico #7). La mancanza di *keyword* specifiche che escludessero il contesto "*malware*" ha permesso a questo pattern di salire in cima alla classifica.

Recupero tardivo e confronto con il SOTA. Il sistema non ha fallito completamente nel recupero, ma nel *ranking*.

- Al *rank* #3 troviamo **`CAPEC-312`** (***Active OS Fingerprinting***), che è un risultato pertinente (Rilevanza 1) e un "figlio" diretto dell'*apex threat* corretto.
- Al *rank* #4 troviamo **`CAPEC-639`** (***Probe System Files***), anch'esso pertinente (Rilevanza 1).

L'algoritmo RRF, lavorando su segnali deboli (i punteggi di fusione sono molto bassi e vicini tra loro, da 0.029 a 0.026), non è riuscito a discriminare tra la specificità errata (*Malware*) e quella corretta (*OS Fingerprinting*).

Il confronto con la *baseline* SOTA è netto e istruttivo: `Gemini 2.5 Pro` ha identificato correttamente **`CAPEC-169`** come primo risultato. Il modello più grande, avendo accesso all'intera gerarchia CAPEC *in-context*, ha probabilmente riconosciuto che l'attività osservata era un insieme di tecniche diverse (*OS fingerprinting*, *process footprinting*, *file probing*) e ha correttamente dedotto che la categoria "padre" (*Footprinting*) era la descrizione più appropriata e onnicomprensiva, evitando di impegnarsi su una sottocategoria troppo specifica o contestualmente errata.

Questo caso evidenzia una sfida aperta per l'approccio ibrido: la difficoltà nel navigare all'interno della gerarchia di astrazione di CAPEC quando i segnali nel log sono generici (comandi standard) e potrebbero mappare a molteplici pattern specifici.

6.7.4 Caso studio 4: superiorità semantica sul SOTA (la distinzione tra ricognizione e accesso)

L'ultimo caso studio, relativo alla sessione `sentrypeer_198.98.53.60`, rappresenta il risultato più significativo dell'intera valutazione comparativa. In questo scenario, il sistema proposto ha ottenuto un punteggio nDCG@5 di **0.922**, mentre la *baseline* SOTA si è fermata a un modesto **0.202**, generando un Delta positivo di **+0.72**. L'analisi di questo divario rivela come l'architettura proposta sia in grado di cogliere sfumature tecniche critiche che possono sfuggire anche ai modelli generalisti più avanzati.

Analisi delle evidenze nel log. L'*input* proviene da un sensore `SentryPeer` (VoIP) e registra 2 interazioni SIP provenienti da un singolo IP. Il metodo utilizzato è esclusivamente `REGISTER`. I

metadati mostrano uno *User-Agent* generico (“`bc-uc`”) e un tentativo di registrazione sull’interno “`1000`”. In ambito VoIP, il metodo `REGISTER` è funzionalmente equivalente a una richiesta di *login*: il *client* chiede al *server* di associare il proprio indirizzo IP a un’estensione telefonica valida per ricevere chiamate. Sebbene possa essere usato per enumerare gli interni (ricognizione), la sua funzione primaria è l’autenticazione. Un attaccante che invia `REGISTER` sta tentando attivamente di falsificare la propria identità o di “rubare” una sessione.

Interpretazione semantica dell’LLM. L’LLM del sistema proposto classifica l’evento con estrema precisione, identificando il pattern “*Credential Access: Unauthorized SIP Registration*”. La giustificazione è illuminante:

> *The repeated use of the SIP ’REGISTER’ method... indicates an attempt to perform unauthorized registration, which falls under the category of credential access.*

L’LLM non si limita all’osservazione dell’azione tecnica, bensì ne inferisce correttamente l’obiettivo strategico: non semplice curiosità (Ricognizione), ma tentativo di intrusione (Accesso alle credenziali). Le *keyword* generate (“`account takeover`”, “`authentication bypass`”, “`session credential falsification`”) riflettono questa gravità.

Il successo del recupero ibrido. Il *Gold Standard* assegna la massima rilevanza (2) a **`CAPEC-196`** (***Session Credential Falsification through Forging***). Il motore di ricerca ibrido traduce perfettamente l’intuizione dell’LLM in un *ranking* corretto:

- **`CAPEC-196`** viene posizionato al *rank* #1. La tabella di *debug* mostra che questo risultato è guidato da una forte componente lessicale (*rank Key* #3) supportata da una buona rilevanza semantica (*rank Sem* #9). I termini “*Credential*”, “*Session*” e “*Falsification*” presenti nella descrizione dell’LLM hanno creato un ponte diretto con la definizione del CAPEC.
- La *top-5* è completata da altri pattern di manipolazione delle credenziali altamente pertinenti, come **`CAPEC-226`** (*Manipulation*) e **`CAPEC-195`** (*Principal Spoof*), tutti presenti nel *Gold Standard*.

Il fallimento interpretativo della *baseline* SOTA. Il confronto con l’*oracolo* è rivelatore. `Gemini 2.5 Pro` ha identificato come *apex threat* **`CAPEC-575`** (***Account Footprinting***), promuovendolo erroneamente però al *rank* #1 nella sua lista (ma essendo questo un pattern di rilevanza 1 nel *Gold Standard*, il punteggio nDCG ne ha sofferto drasticamente). La giustificazione dell’*oracolo* recita:

> *This is the Apex Threat because the attacker is using SIP REGISTER requests to probe for the existence of specific user accounts...*

Qui risiede l'errore cruciale: il modello SOTA ha interpretato l'azione come una mera attività di *footprinting* (ricognizione), sottostimando la gravità dell'attacco. Ha visto il "*probing*", ma ha mancato l'"*account takeover*". Di conseguenza, la sua *top-5* è popolata da pattern generici di ricognizione (`CAPEC-54` *Query System*, `CAPEC-169` *Footprinting*) che, pur essendo tecnicamente veri, non catturano l'intento primario e più pericoloso dell'attaccante.

Questo caso studio dimostra che la specializzazione del *prompt* ("*Threat Intelligence Lexicographer*") e la struttura della *pipeline* proposta guidano il sistema verso un'interpretazione della minaccia più accurata e severa, superando la tendenza dei modelli generalisti a fornire classificazioni più superficiali o conservative.

6.8 Confronto con lo stato dell'arte e posizionamento della soluzione

Per validare definitivamente il contributo scientifico di questa tesi, è essenziale posizionare la soluzione proposta all'interno del panorama attuale della ricerca, confrontandola direttamente con i lavori più rilevanti analizzati nel Capitolo 3. Mentre l'analisi precedente ha misurato le *performance* assolute e relative alla *baseline* SOTA, questa sezione analizza le differenze *architetturali* e *metodologiche* rispetto alla letteratura esistente, evidenziando come la *pipeline* ibrida proposta risolva limitazioni specifiche che altri approcci, pur validi nel loro dominio, non affrontano.

6.8.1 Matrice comparativa delle funzionalità

La Tabella 6.4 sintetizza le caratteristiche distintive del sistema proposto rispetto ai principali "*competitor*" accademici identificati. Il confronto si articola su cinque dimensioni critiche: la natura dei dati di *input* gestiti, il *target* del *mapping*, la metodologia di *core* (*LLM-based*, ibrida, ecc.), la strategia di fusione dei segnali di ricerca e il ruolo attribuito all'LLM.

Dall'analisi della Tabella 6.4 emergono tre vantaggi competitivi chiave che distinguono la proposta di questa tesi: la robustezza nella gestione di *input* non strutturati ("sporchi"), la superiorità metodologica della fusione tramite *ranking* (RRF) rispetto alla combinazione di *score*, e il posizionamento strategico dell'LLM come interprete intermedio.

6.8.2 Analisi dei vantaggi competitivi

Gestione del "*gap* semantico" su *input* eterogenei. La maggior parte delle soluzioni avanzate di *matching* presenti in letteratura, come **Bonomi et al.** [57] e **Sauze-Kadar e Loubier** [85], operano su dati di *input* "puliti" e semanticamente densi, tipicamente descrizioni di vulnerabilità CVE. In questi casi, il divario linguistico tra la descrizione della vulnerabilità (*input*) e il pattern di attacco (*target*) è ridotto, poiché entrambi sono testi tecnici redatti da esperti. Al contrario,

Tabella 6.4: Matrice comparativa tra la soluzione proposta e i lavori più rilevanti dello stato dell'arte

Riferimento	***Input* dati**	***Target***	**Metodologia *core***	**Strategia fusione**	**Ruolo LLM**
Ozkok et al. [82]	Log grezzi (Honeypot)	ATT&CK	*Zero-shot prompting*	*Assente* (Solo LLM)	*Oracolo* (*End-to-end*)
Boffa et al. [83]	Log *shell* (omogenei)	ATT&CK	*Fine-tuned* BERT	*Assente* (Modello unico)	Classificatore
Rafiey et al. [84]	Testo CVE (pulito)	ATT&CK	*Few-shot prompting*	*Assente* (Solo LLM)	*Oracolo* (*End-to-end*)
Bonomi et al. [57]	Testo CVE (pulito)	CAPEC	Ibrida (Sem. + Lessicale)	Somma pesata di *score*	*Nessuno*
Sauze-Kadar et al. [85]	Testo CVE (pulito)	CAPEC	Ibrida adattiva	Selezione dinamica	*Nessuno*
Webb et al. [86]	Standard (Testo)	ATT&CK / CAPEC	RAG standard	*Assente* (Solo semantico)	Sintetizzatore
Tejero et al. [87]	Log IoT (strutturati)	CAPEC	*Fine-tuned* LLaMA	*Assente* (Solo LLM)	Classificatore
Sistema proposto	**Log grezzi (eterogenei)**	**CAPEC**	**Ibrida + LLM *upstream***	**Reciprocal Rank Fusion**	**Interprete semantico**

il sistema proposto affronta la sfida ben più ardua di partire da log grezzi di honeypot. Come evidenziato da **Boffa et al.** [83], i log (specialmente *shell*) sono frammentati e poveri di contesto esplicito. Tuttavia, mentre `LogPrécis` si limita a un "*fingerprinting*" tattico di alto livello per un singolo tipo di log, la nostra soluzione generalizza il problema gestendo un ecosistema eterogeneo (`Cowrie`, `Honeytrap`, `SentryPeer`) e mappandolo a un livello di granularità molto più fine (CAPEC *Attack Patterns*). Il modulo di *preprocessing* olistico e l'uso dell'LLM come "normalizzatore semantico" permettono di colmare quel *gap* che impedirebbe l'applicazione diretta delle tecniche di Bonomi et al. ai log grezzi.

Superiorità della *reciprocal rank fusion* sui metodi ibridi tradizionali. Nel panorama delle soluzioni ibride, il lavoro di **Bonomi et al.** rappresenta il termine di paragone più vicino. Il loro approccio combina ricerca semantica (`SBERT`) e *keyword search* sommando i punteggi di similarità ($S_{overall} = S_{base} + S_{keyword}$). Sebbene efficace su domini chiusi, questo metodo di "*score weighting*" soffre di una debolezza intrinseca: la sensibilità alle diverse distribuzioni statistiche dei punteggi prodotti da algoritmi diversi (es. la similarità coseno ha una distribuzione diversa dal BM25/TF-IDF), che richiede una difficile calibrazione dei pesi. La soluzione proposta in questa tesi supera questo limite adottando la ***Reciprocal Rank Fusion*** (RRF). Come dimostrato nello studio di ablazione (Sezione 6.4), l'RRF opera sui *rank* (posizioni) anziché sui punteggi assoluti. Questo rende il sistema agnostico rispetto alla scala dei punteggi e molto più robusto nel premiare il "consenso" tra i motori. In scenari come il Caso studio 1 (`Honeytrap`), dove i singoli motori divergevano nei punteggi ma concordavano nel posizionare il *target* nella *top-k*, l'RRF ha garantito il successo dove una somma lineare avrebbe potuto fallire a causa di *outlier* nei punteggi.

L'LLM come interprete *upstream* vs. *oracolo end-to-end*. Un'altra distinzione critica riguarda il ruolo dell'LLM. Approcci come quelli di **Ozkok et al.** [82] e **Rafiey e Namadchian** [84] utilizzano l'LLM come un "*oracolo*": gli forniscono i dati e chiedono direttamente la classificazione finale. Questo approccio, come evidenziato dal nostro confronto con la *baseline* SOTA (`Gemini 2.5 Pro`), soffre del problema del "*Lost in the Middle*" quando il contesto è ampio e delle allucinazioni quando il *mapping* richiede precisione tecnica. La nostra architettura inverte questo paradigma: l'LLM non viene usato per *cercare* nella *knowledge base*, ma solo per *interpretare* l'evento e generare una *query* strutturata. Il compito di ricerca è delegato a motori deterministici (vettoriale + `TF-IDF`). Questo disaccoppiamento ha un duplice beneficio: riduce drasticamente le allucinazioni (l'LLM deve solo descrivere ciò che vede, non ricordare migliaia di CAPEC) e garantisce che il *matching* finale sia sempre ancorato a documenti reali esistenti nella *knowledge base*, come dimostrato dalla stabilità dell'MRR (0.973) rispetto alla variabilità intrinseca delle risposte generative pure.

6.8.3 Confronto con soluzioni strumentali e industriali

Oltre al confronto con la letteratura scientifica, è fondamentale posizionare la soluzione proposta rispetto agli strumenti operativi standard utilizzati nei *Security Operations Center* e nell'analisi forense. Attualmente, la gestione dei log di sicurezza e la *threat intelligence* sono dominate da piattaforme `SIEM` (*Security Information and Event Management*) e `XDR` (*Extended Detection and Response*). Tuttavia, l'integrazione di capacità di *mapping* automatico verso standard granulari come CAPEC rimane una funzionalità largamente assente o implementata in modo rudimentale.

La Tabella 6.5 sintetizza le differenze chiave tra il sistema proposto e tre categorie di strumenti rappresentativi: soluzioni basate su regole (es. `Wazuh`, `Suricata`), piattaforme di analisi dati (es. `Elastic/T-Pot standard`) e moderni assistenti AI commerciali (es. `Security Copilot`).

Tabella 6.5: Confronto funzionale tra il sistema proposto e gli standard strumentali di settore

Feature	**SIEM basati su regole** (es. `Wazuh`, `Suricata`)	**AI *security* commerciali** (es. `MS Copilot`, `Splunk`)	**Sistema proposto**
Logica di rilevamento	Deterministica (Regex, Firme). Fragile a variazioni sintattiche e offuscamento.	Probabilistica / Generativa (LLM "*Black Box*").	**Ibrida interpretabile** (Comprensione semantica + ancoraggio lessicale).
Target* del *mapping	MITRE ATT&CK (Alto livello). Raramente CAPEC.	Generico / ATT&CK.	**MITRE CAPEC** (Dettaglio del meccanismo di attacco).
Gestione nuove minacce	Richiede aggiornamento manuale delle regole.	Dipende dal *training set* del modello (*knowledge cutoff*).	***Zero-shot* / Adattivo** grazie al *prompt engineering* e al *retrieval* dinamico.
Privacy e costi	Locale, basso costo.	*Cloud-based*, costo per *token* elevato, problemi di privacy dati.	**Locale (*On-premise*)**, costo inferenza basso (modelli quantizzati).

Superamento dei limiti delle regole statiche (Regex). Gli strumenti standard integrati in piattaforme come `T-Pot` (es. `Suricata` o le *dashboard* `Kibana` basate su filtri) operano secondo una logica strettamente deterministica: un evento viene classificato solo se corrisponde esattamente a una firma o a un'espressione regolare predefinita. Se un attaccante modifica leggermente la sintassi di un comando (es. usando `w'g'et` invece di `wget`) o utilizza un *tool* legittimo in modo anomalo (*living-off-the-land binaries*), questi sistemi spesso falliscono (falsi negativi) o generano allarmi generici. Il sistema proposto, grazie all'interpretazione semantica dell'LLM (Sezione 6.6, Caso studio 3), è in grado di "capire" l'intento dell'azione

indipendentemente dalla sintassi specifica, colmando il *gap* di flessibilità che affligge i SIEM tradizionali.

Il *gap* di granularità: ATT&CK vs CAPEC. Lo standard di fatto nell'industria per il *mapping* delle minacce è il framework MITRE ATT&CK, che descrive le *tattiche* (obiettivi, es. "*Defense Evasion*") e le *tecniche* (come, es. "*Obfuscated Files*"). Tuttavia, strumenti come `Wazuh` o i *plugin* di `Elastic Security` si fermano quasi sempre a questo livello. La nostra soluzione scende a un livello di granularità superiore mappando su MITRE CAPEC, che descrive i *meccanismi di attacco* (es. "Come sfruttare una *race condition* specifica"). Come dimostrato nell'analisi qualitativa (es. Caso studio 4), la distinzione tra un generico "*Credential Access*" (ATT&CK) e uno specifico "*Session Credential Falsification*" (CAPEC) è cruciale per l'ingegneria della sicurezza e la *patch analysis*. Il sistema proposto abilita questo livello di dettaglio in modo automatico, un compito che manualmente richiederebbe ore di analisi esperta.

Trasparenza ("*glass box*") vs. "*black box*". Infine, rispetto alle emergenti soluzioni di AI generativa commerciale (come i "`Security Copilot`"), il sistema proposto offre un vantaggio architetturale in termini di trasparenza e controllo. Mentre i *tool* commerciali forniscono una risposta generata da un modello opaco in *cloud* (con i rischi di allucinazione discussi nel confronto SOTA), la nostra *pipeline* disaccoppia l'interpretazione dal recupero. La tabella di *debug* RRF (mostrata nei casi studio) fornisce all'analista una prova tangibile del "perché" un certo pattern è stato scelto, mostrando i contributi specifici della similarità semantica e della corrispondenza di *keyword*. Questo rende il sistema una "*glass box*" (scatola trasparente), requisito fondamentale per l'adozione in contesti forensi e operativi critici, garantendo al contempo la sovranità dei dati sensibili che non lasciano mai l'infrastruttura locale.

6.9 Considerazioni finali sulla valutazione

L'insieme delle analisi condotte in questo capitolo, spaziando dalle metriche quantitative aggregate allo studio di ablazione, fino all'analisi qualitativa dei casi d'uso e al confronto strutturale con lo stato dell'arte, fornisce un quadro completo delle capacità del sistema proposto. I dati sperimentali confermano che l'architettura ibrida, supportata da un'interpretazione semantica *upstream*, non solo raggiunge *performance* competitive con i modelli di frontiera, ma offre vantaggi decisivi in termini di interpretabilità, gestione di *input* rumorosi e sostenibilità operativa.

Le implicazioni di questi risultati, unitamente a una discussione critica sui limiti metodologici emersi e alle prospettive di sviluppo futuro, saranno oggetto del capitolo conclusivo.

Capitolo 7

Conclusioni e sviluppi futuri

Il presente lavoro di tesi ha affrontato una delle sfide più attuali nell'ambito della *cyber threat intelligence*: l'analisi automatizzata e l'interpretazione semantica di grandi volumi di log di sicurezza eterogenei. Partendo dalla constatazione che l'analisi manuale dei dati provenienti da ecosistemi *multi-honeypot* come `T-Pot` è divenuta insostenibile, l'obiettivo primario è stato progettare, implementare e validare una metodologia capace di mappare tali eventi grezzi verso lo standard di riferimento MITRE CAPEC.

7.1 Sintesi del lavoro svolto

La ricerca ha seguito un percorso metodologico che ha integrato i più recenti avanzamenti nel campo degli LLM con tecniche consolidate di *information retrieval*, dando vita a un'architettura ibrida e modulare che supera i limiti dei singoli approcci.

In primo luogo, è stata affrontata la problematica della *contestualizzazione dei dati*. Rifiutando l'approccio basato sull'analisi del singolo evento atomico, è stato sviluppato un modulo di *preprocessing* olistico capace di aggregare i log per attore e ricostruire la sequenza temporale delle azioni. Questo ha permesso di trasformare dati frammentati in narrazioni coerenti del comportamento dell'attaccante.

Il cuore innovativo della soluzione risiede nel nuovo paradigma di utilizzo dell'LLM. Invece di impiegare il modello (`Mistral-7B-Instruct`) come un "oracolo onnisciente" — approccio soggetto ad allucinazioni e obsolescenza — il modello è stato impiegato come un *interprete semantico upstream*. Attraverso un *prompt engineering* adattivo, che sfrutta tecniche di *chain-of-thought* e *few-shot learning*, il sistema guida il modello nell'astrarre i dettagli tecnici del log in una descrizione strutturata, disaccoppiando la comprensione del linguaggio naturale dalla conoscenza enciclopedica.

Infine, il problema del *mapping* è stato risolto mediante un **motore di ricerca ibrido**. Riconoscendo i limiti dei singoli approcci, è stata implementata una strategia di fusione basata su *Reciprocal Rank Fusion*. Questa tecnica combina i risultati di un indice semantico (basato

su `ATT&CK-BERT`) e di un indice lessicale (`TF-IDF`), garantendo una robustezza superiore nel recupero dei pattern CAPEC più pertinenti.

7.2 Risultati conseguiti

La valutazione sperimentale, arricchita dall'analisi comparativa con la letteratura e gli standard industriali, ha fornito risposte empiriche solide alle domande di ricerca che hanno guidato questo studio.

- **Efficacia assoluta:** Il sistema ha dimostrato un'elevata efficacia nel suo compito primario: la prioritizzazione delle minacce. Con un *Mean Reciprocal Rank* (MRR) di **0.973**, la soluzione garantisce quasi sistematicamente la presenza di un pattern pertinente nelle primissime posizioni del *ranking*. La metrica **P@1 di 0.760** conferma che, nella grande maggioranza dei casi, la minaccia più grave (*apex threat*) viene identificata correttamente al primo tentativo, offrendo un supporto decisionale immediato all'analista e superando le limitazioni dei sistemi basati su firme statiche.

- **Contributo dell'approccio ibrido:** Lo studio di ablazione ha fornito la validazione più forte dell'architettura proposta. Il crollo delle *performance* osservato nelle configurazioni "solo semantico" e "solo lessicale" (con un P@1 che scende a 0.120) dimostra che la fusione dei due segnali non è opzionale, ma costitutiva per la gestione di dati reali e rumorosi. L'algoritmo RRF si è rivelato essenziale per mitigare le ambiguità dei singoli motori, agendo come un meccanismo di consenso robusto.

- **Posizionamento rispetto allo stato dell'arte:** Il confronto non si è limitato alle metriche quantitative, dove il sistema ha superato la *baseline* `Gemini 2.5 Pro` nella prioritizzazione "*strict*" (P@1 e MRR), ma ha evidenziato vantaggi architetturali decisivi. La soluzione proposta colma il *gap* della gestione di *log grezzi*, utilizzando l'LLM come interprete semantico anziché come *oracolo end-to-end*. Questo *design* mitiga il problema del "*Lost in the Middle*" e le allucinazioni tipiche dei modelli generativi puri. Inoltre, rispetto ai SIEM commerciali, il sistema offre una granularità di analisi superiore e una trasparenza decisionale ("*glass box*") fondamentale per l'operatività forense.

- **Robustezza e variabilità:** L'analisi disaggregata ha evidenziato che le *performance* riflettono la complessità intrinseca delle diverse fonti dati. Il sistema eccelle nell'analisi di log strutturati o basati su protocolli specifici (`Honeytrap`, `SentryPeer`), dove i segnali tecnici sono chiari. Mostra una maggiore variabilità nell'analisi di sessioni interattive complesse (`Cowrie`), dove l'ambiguità dei comandi di sistema standard può talvolta condurre a classificazioni tassonomiche imprecise, pur rimanendo semanticamente pertinenti.

7.3 Limiti dello studio

Per garantire il rigore scientifico della trattazione, è necessario discutere i limiti intrinseci al protocollo sperimentale adottato, che rappresentano altrettante opportunità per la ricerca futura.

***Bias* del valutatore (*rater bias*).** La creazione del *Gold Standard* e il relativo processo di *adjudication* sono stati condotti interamente dall'autore del presente lavoro. Sebbene l'assegnazione dei punteggi sia stata rigorosamente guidata dai criteri di rilevanza graduata definiti nella metodologia (scala 0-1-2) al fine di massimizzare l'oggettività, l'assenza di una validazione incrociata con valutatori esterni indipendenti introduce inevitabilmente un potenziale *bias* soggettivo. Tale limitazione, spesso intrinseca ai progetti di ricerca individuali, implica che le decisioni sui casi più ambigui riflettano l'interpretazione di un singolo analista.

Rappresentatività campionaria. La valutazione è stata condotta su un dataset di 25 profili di comportamento. Sebbene la selezione stratificata abbia garantito la copertura di tutte le tipologie di honeypot supportate, la dimensione del campione limita la generalizzabilità statistica dei risultati su larga scala. Eventi rari o tecniche di evasione sofisticate non rappresentate in questo sottoinsieme potrebbero presentare sfide non emerse in questa fase. Tuttavia, la significatività statistica rilevata dal test di Wilcoxon suggerisce che i *trend* osservati sono robusti all'interno del dominio analizzato.

Dipendenza dalla qualità dell'LLM *upstream*. L'intera *pipeline* dipende criticamente dalla capacità dell'LLM di generare una descrizione iniziale accurata. Come evidenziato nei casi di fallimento, se l'LLM interpreta correttamente l'azione ma utilizza una terminologia che si allinea erroneamente a pattern specifici nel database vettoriale (es. "*post-exploitation*" che attiva "*Malware Reconnaissance*"), il motore di ricerca può essere tratto in inganno.

7.4 Sviluppi futuri

Il lavoro svolto apre la strada a numerose direzioni di ricerca e miglioramento.

1. **Evoluzione verso architetture basate su agenti (*AI agents*).** L'attuale *pipeline* è sequenziale. Un'evoluzione naturale consiste nel trasformare il sistema in un agente AI autonomo. Invece di una singola passata, l'agente potrebbe interagire iterativamente con la *knowledge base*: se l'LLM rileva ambiguità tra due pattern CAPEC simili, potrebbe formulare autonomamente *query* di approfondimento per discriminare meglio tra le opzioni, simulando il processo investigativo umano.

2. ***Fine-tuning* supervisionato (SFT) su dati di dominio.** Sebbene il *prompt engineering* avanzato abbia dato ottimi risultati, l'adozione di tecniche di *supervised fine-tuning*

(SFT) o *low-rank adaptation* (LoRA) su un corpus di log annotati potrebbe specializzare ulteriormente il modello linguistico. Questo permetterebbe al modello di apprendere le sfumature del linguaggio degli honeypot (es. comandi offuscati in `Cowrie`) senza la necessità di istruzioni di contesto estese, riducendo la latenza.

3. **Integrazione multimodale e *cross-standard*.** Una futura estensione potrebbe prevedere il *mapping* simultaneo verso più standard, come MITRE ATT&CK e CWE, creando un grafo di conoscenza che colleghi l'osservazione (log) alla debolezza (CWE) e alla tattica avversaria (ATT&CK). Inoltre, l'architettura potrebbe essere estesa per gestire *input* multimodali, integrando l'analisi di *screenshot* o *dump* di memoria catturati dagli honeypot ad alta interazione.

4. **Validazione su larga scala e in tempo reale.** Per consolidare i risultati, è necessario estendere la valutazione su un dataset di dimensioni maggiori e con un processo di annotazione multi-valutatore per ridurre il *bias* soggettivo. Parallelamente, dal punto di vista ingegneristico, il prossimo passo logico è l'integrazione del prototipo come *plugin* all'interno di piattaforme SIEM *open source* (es. `Wazuh` o `Elastic Security`), per testarne l'efficacia in un ambiente di produzione reale e misurarne l'impatto sul *mean time to respond* (MTTR) degli analisti.

In conclusione, questa tesi dimostra che l'integrazione intelligente tra *LLM* e sistemi di *information retrieval* classici rappresenta una via percorribile ed efficace per modernizzare l'analisi della *threat intelligence*, offrendo una soluzione che bilancia accuratamente l'innovazione dell'IA generativa con il rigore e il controllo necessari nell'ambito della sicurezza informatica.

Bibliografia

[1] Z. Morić, V. Dakić e D. Regvart, "Advancing Cybersecurity with Honeypots and Deception Strategies", in *Informatics*, MDPI AG, vol. 12, 2025, p. 14.

[2] Palo Alto Networks. "What Is a Honeypot?", visitato il giorno 23 set. 2025. indirizzo: https://www.paloaltonetworks.co.uk/cyberpedia/honeypots

[3] M. Rana, "Fortifying Cyber Defenses: Leveraging Honeypots for Proactive Threat Mitigation and DoS Attack Prevention", *Journal of Information Systems Engineering and Management*, vol. 10, pp. 443–452, mar. 2025. DOI: 10.52783/jisem.v10i19s.3056

[4] CrowdStrike. "Honeypots in Cybersecurity Explained", visitato il giorno 23 set. 2025. indirizzo: https://www.crowdstrike.com/en-us/cybersecurity-101/exposure-management/honeypots/

[5] Sophos. "What Is a Honeypot in Cybersecurity?", visitato il giorno 23 set. 2025. indirizzo: https://www.sophos.com/en-us/cybersecurity-explained/honeypots

[6] S. Mehta, D. Pawade, Y. Nayyar, I. Siddavatam, A. Tiwart e A. Dalvi, "Cowrie honeypot data analysis and predicting the directory traverser pattern during the attack", in *2021 International Conference on Innovative Computing, Intelligent Communication and Smart Electrical Systems (ICSES)*, IEEE, 2021, pp. 1–4.

[7] Fortinet. "What Are Honeypots (Computing)?", visitato il giorno 23 set. 2025. indirizzo: https://www.fortinet.com/resources/cyberglossary/what-is-honeypot

[8] Cowrie Development Team. "Cowrie SSH/Telnet Honeypot", visitato il giorno 23 set. 2025. indirizzo: https://github.com/cowrie/cowrie

[9] Telekom Security. "T-Pot - The All In One Multi Honeypot Platform", visitato il giorno 23 set. 2025. indirizzo: https://github.com/telekom-security/tpotce

[10] CybersecTools. "Honeytrap by Till Mannw", visitato il giorno 23 set. 2025. indirizzo: https://cybersectools.com/tools/honeytrap-by-till-mannw

[11] The dionaea project. "Dionaea documentation", visitato il giorno 23 set. 2025. indirizzo: https://dionaea.readthedocs.io/

[12] Suricata. "Suricata documentation", visitato il giorno 23 set. 2025. indirizzo: https://docs.suricata.io/en/suricata-8.0.1/what-is-suricata.html

[13] N. O. Jaffal, M. Alkhanafseh e D. Mohaisen, "Large Language Models in Cybersecurity: A Survey of Applications, Vulnerabilities, and Defense Techniques", *AI*, vol. 6, n. 9, p. 216, 2025.

[14] G. Ikuomenisan e Y. Morgan, "Meta-Review of Recent and Landmark Honeypot Research and Surveys", *Journal of Information Security*, vol. 13, pp. 181–209, gen. 2022. DOI: 10.4236/jis.2022.134011

[15] E. Karaarslan, E. Güler, E. E. Yüce e C. Coban, "Towards Log Analysis with AI Agents: Cowrie Case Study", *arXiv preprint arXiv:2509.05306*, 2025.

[16] Proofpoint. "Cos'è l'honeypot?", visitato il giorno 23 set. 2025. indirizzo: https://www.proofpoint.com/it/threat-reference/honeypot

[17] J. Zhang et al., "When LLMs meet cybersecurity: A systematic literature review", *Cybersecurity*, vol. 8, n. 1, p. 55, 2025.

[18] DataSunrise. "Modelli LLM per i Casi d'Uso in Cybersecurity", visitato il giorno 23 set. 2025. indirizzo: https://www.datasunrise.com/it/centro-di-conoscenza/modelli-llm-per-casi-uso-sicurezza-cibernetica/

[19] United States Cybersecurity Institute. "How Large Language Models (LLMs) in Cybersecurity Secure Our Future?", visitato il giorno 23 set. 2025. indirizzo: https://www.uscsinstitute.org/cybersecurity-insights/blog/how-large-language-models-in-cybersecurity-secure-our-future

[20] Splunk. "Introducing DECEIVE: A Proof-of-Concept Honeypot Powered by AI", visitato il giorno 23 set. 2025. indirizzo: https://www.splunk.com/en_us/blog/security/deceive-ai-honeypot-concept.html

[21] Elastic. "What are large language models (LLMs)?", visitato il giorno 14 mag. 2025. indirizzo: https://www.elastic.co/what-is/large-language-models

[22] IBM. "What are large language models (LLMs)?", visitato il giorno 14 mag. 2025. indirizzo: https://www.ibm.com/think/topics/large-language-models

[23] Carlow University. "What Are Large Language Models and Multimodal Models?", visitato il giorno 14 mag. 2025. indirizzo: https://ctrl.carlow.edu/ai/whatare

[24] Palo Alto Networks. "What Are Large Language Models (LLMs)?", visitato il giorno 14 mag. 2025. indirizzo: https://www.paloaltonetworks.com/cyberpedia/large-language-models-llm

[25] A. Vaswani et al., "Attention is all you need", *Advances in neural information processing systems*, vol. 30, 2017.

[26] Magnimind Academy. “The Mechanism of Attention in Large Language Models: A Comprehensive Guide”, visitato il giorno 14 mag. 2025. indirizzo: https://magnimindacademy.com/blog/the-mechanism-of-attention-in-large-language-models-a-comprehensive-guide/

[27] S. Dodson. “Domain specific generative AI: pre-training, fine-tuning, and RAG”, visitato il giorno 15 mag. 2025. indirizzo: https://www.elastic.co/search-labs/blog/domain-specific-generative-ai-pre-training-fine-tuning-rag

[28] PromptLayer. “What is Instruction tuning?”, visitato il giorno 14 mag. 2025. indirizzo: https://www.promptlayer.com/glossary/instruction-tuning

[29] Mistral AI. “Mistral-7B-Instruct-v0.2”, visitato il giorno 17 mag. 2025. indirizzo: https://huggingface.co/mistralai/Mistral-7B-Instruct-v0.2

[30] OneAdvanced. “Large language models (Part 2): Understanding the mechanism”, visitato il giorno 14 mag. 2025. indirizzo: https://www.oneadvanced.com/news-and-opinion/large-language-models-part-2-understanding-the-mechanism/

[31] Prompt Engineering Guide. “Mistral 7B LLM”, visitato il giorno 14 mag. 2025. indirizzo: https://www.promptingguide.ai/models/mistral-7b

[32] Prompt Engineering Guide. “Prompt Engineering Guide”, visitato il giorno 14 mag. 2025. indirizzo: https://www.promptingguide.ai/

[33] T. Brown et al., “Language models are few-shot learners”, *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[34] J. Wei et al., “Chain-of-thought prompting elicits reasoning in large language models”, *Advances in neural information processing systems*, vol. 35, pp. 24 824–24 837, 2022.

[35] Qualys Community. “The Impact of LLMs on Cybersecurity: New Threats and Solutions”, visitato il giorno 14 mag. 2025. indirizzo: https://blog.qualys.com/product-tech/2025/02/07/the-impact-of-llms-on-cybersecurity-new-threats-and-solutions

[36] Infosecurity Europe. “Top Eight Large Language Models Benchmarks for Cybersecurity Practices”, visitato il giorno 14 mag. 2025. indirizzo: https://www.infosecurityeurope.com/en-gb/blog/future-thinking/top-8-llm-benchmarks-for-cybersecurity-practices.html

[37] BreachLock. “What are Large Language Models and How Are They Used in Cyber Security?”, visitato il giorno 14 mag. 2025. indirizzo: https://www.breachlock.com/resources/blog/what-are-large-language-models-and-how-are-they-used-in-cyber-security/

[38] ProjectPro. “10 Biggest Limitations of Large Language Models”, visitato il giorno 14 mag. 2025. indirizzo: https://www.projectpro.io/article/llm-limitations/1045

[39] The MITRE Corporation. “About CAPEC - Common Attack Pattern Enumeration and Classification”, visitato il giorno 6 mag. 2025. indirizzo: https://capec.mitre.org/about/index.html

[40] R. A. Martin e S. Barnum, “Understanding How They Attack Your Weaknesses: CAPEC”, The MITRE Corporation, rapp. tecn. ADA558048, mag. 2011, Presented at the 23rd Systems and Software Technology Conference (SSTC), 16-19 May 2011, Salt Lake City, UT. Approved for public release; distribution unlimited. visitato il giorno 6 mag. 2025. indirizzo: https://apps.dtic.mil/sti/tr/pdf/ADA558048.pdf

[41] S. Craig. “CAPEC: Making Heads or Tails of Attack Patterns”, visitato il giorno 6 mag. 2025. indirizzo: https://www.ibm.com/think/x-force/capec-making-heads-or-tails-of-attack-patterns

[42] The MITRE Corporation. “New to CAPEC?”, visitato il giorno 6 mag. 2025. indirizzo: https://capec.mitre.org/about/new_to_capec.html

[43] The MITRE Corporation. “CAPEC Glossary”, visitato il giorno 6 mag. 2025. indirizzo: https://capec.mitre.org/about/glossary.html

[44] Dana Epp. “3 ways to use Common Attack Patterns to abuse an API”, visitato il giorno 6 mag. 2025. indirizzo: https://danaepp.com/3-ways-to-use-common-attack-patterns-to-abuse-an-api

[45] The MITRE Corporation. “CAPEC – Schema Documentation - Schema Version 3.5”, visitato il giorno 6 mag. 2025. indirizzo: https://capec.mitre.org/documents/schema/index.html

[46] R. A. Martin, “Making security measurable and manageable”, in *MILCOM 2008-2008 IEEE Military Communications Conference*, IEEE, 2008, pp. 1–9.

[47] K. Kanakogi et al., “Tracing CAPEC attack patterns from CVE vulnerability information using natural language processing technique”, *Proceedings of the 2021 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, 2021.

[48] The MITRE Corporation. “CAPEC – ATT&CK Comparison”, visitato il giorno 6 mag. 2025. indirizzo: https://capec.mitre.org/about/attack_comparison.html

[49] AttackForge. “AttackForge”, visitato il giorno 10 mag. 2025. indirizzo: https://attackforge.com/

[50] The MITRE Corporation. “CAPEC - Organization Usage”, visitato il giorno 6 mag. 2025. indirizzo: https://capec.mitre.org/community/usage.html

[51] The MITRE Corporation. “News & Events - 2020 Archive - CAPEC”, visitato il giorno 6 mag. 2025. indirizzo: https://capec.mitre.org/news/archives/news2020.html

[52] MITRE Corporation. “CAPEC-658: ATT&CK Related Patterns (Version 3.9)”, visitato il giorno 10 mag. 2025. indirizzo: https://capec.mitre.org/data/definitions/658.html

[53] CWE/CAPEC Board, “CWE/CAPEC Board Charter”, MITRE Corporation, Technical Report, 2022. visitato il giorno 11 mag. 2025. indirizzo: https://cwe.mitre.org/documents/board/CWE-CAPEC_board_charter.pdf

[54] IriusRisk. “CAPEC Threat Modeling”, visitato il giorno 6 mag. 2025. indirizzo: https://www.iriusrisk.com/resources-blog/capec-threat-modeling

[55] ITU Online. “Common Attack Pattern Enumeration And Classification (CAPEC): Enhancing Threat Modeling And Defense Strategies”, visitato il giorno 6 mag. 2025. indirizzo: https://www.ituonline.com/comptia-securityx/comptia-securityx-1/common-attack-pattern-enumeration-and-classification-capec-enhancing-threat-modeling-and-defense-strategies/

[56] The MITRE Corporation. “Summary of Use Cases”, visitato il giorno 6 mag. 2025. indirizzo: https://capec.mitre.org/about/use_cases.html

[57] S. Bonomi, A. Ciavotta, S. Lenti e A. Palma, “Beyond the Surface: An NLP-based Methodology to Automatically Estimate CVE Relevance for CAPEC Attack Patterns”, *arXiv preprint arXiv:2501.07131*, 2025.

[58] K. Kanakogi et al., “Comparative evaluation of NLP-based approaches for linking CAPEC attack patterns from CVE vulnerability information”, *Applied Sciences*, vol. 12, n. 7, p. 3400, 2022.

[59] B. Abdeen, E. Al-Shaer, A. Singhal, L. Khan e K. Hamlen, “Smet: Semantic mapping of CVE to ATT&CK and its application to cybersecurity”, in *IFIP Annual Conference on Data and Applications Security and Privacy*, Springer, 2023, pp. 243–260.

[60] M. Vanamala, X. Yuan, W. Smith e J. Bennett, “Interactive Visualization Dashboard for Common Attack Pattern Enumeration Classification”, in *Proceedings of the International Conference on Software Engineering and Applications (ICSEA 2022)*, vol. 2022, 2022, p. 79.

[61] C. Hankin, P. Malacaria et al., “Attack dynamics: An automatic attack graph generation framework based on system topology, CAPEC, CWE, and CVE databases”, *Computers & Security*, vol. 123, p. 102 938, 2022.

[62] G. Salton e M. J. McGill, *Introduction to Modern Information Retrieval*. USA: McGraw-Hill, Inc., 1986, ISBN: 0070544840.

[63] C. D. Manning, *Introduction to information retrieval*. Syngress Publishing, 2008.

[64] G. Salton, A. Wong e C.-S. Yang, “A vector space model for automatic indexing”, *Communications of the ACM*, vol. 18, n. 11, pp. 613–620, 1975.

[65] R. Prashanth, “Semantic similarity estimation for domain specific data using BERT and other techniques”, *arXiv preprint arXiv:2506.18602*, 2025.

[66] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado e J. Dean, “Distributed representations of words and phrases and their compositionality”, *Advances in neural information processing systems*, vol. 26, 2013.

[67] Y. Shi, Y. Zheng, K. Guo, W. Li e L. Zhu, “Word similarity fails in multiple sense word embedding”, in *International Conference on Computational Science*, Springer, 2018, pp. 489–498.

[68] J. Devlin, M.-W. Chang, K. Lee e K. Toutanova, “BERT: Pre-training of deep bidirectional transformers for language understanding”, in *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, 2019, pp. 4171–4186.

[69] N. Reimers e I. Gurevych, “Sentence-BERT: Sentence embeddings using siamese bert-networks”, *arXiv preprint arXiv:1908.10084*, 2019.

[70] Y. A. Malkov e D. A. Yashunin, “Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs”, *IEEE transactions on pattern analysis and machine intelligence*, vol. 42, n. 4, pp. 824–836, 2018.

[71] K. Dhungana. “An Overview of ChromaDB: The Vector Database”, visitato il giorno 25 set. 2025. indirizzo: https://medium.com/@kbdhunga/an-overview-of-chromadb-the-vector-database-206437541bdd

[72] J. Rayo, R. de La Rosa e M. Garrido, “A Hybrid Approach to Information Retrieval and Answer Generation for Regulatory Texts”, *arXiv preprint arXiv:2502.16767*, 2025.

[73] IBM. “IBM X-Force 2025 Threat Intelligence Index”, visitato il giorno 25 set. 2025. indirizzo: https://www.ibm.com/thought-leadership/institute-business-value/en-us/report/2025-threat-intelligence-index

[74] CrowdStrike. “2025 Global Threat Report”, visitato il giorno 25 set. 2025. indirizzo: https://www.securityweek.com/wp-content/uploads/2025/02/CrowdStrikeGlobalThreatReport2025.pdf

[75] E. Mezzi, F. Massacci e K. Tuma, “Large language models are unreliable for cyber threat intelligence”, in *International Conference on Availability, Reliability and Security*, Springer, 2025, pp. 343–364.

[76] M. Boffa, G. Milan, L. Vassio, I. Drago, M. Mellia e Z. B. Houidi, “Towards NLP-based processing of honeypot logs”, in *2022 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, IEEE, 2022, pp. 314–321.

[77] G. Yazi, “Large language models (LLMs) for cybersecurity: A systematic review”, *WORLD*, vol. 13, n. 1, pp. 057–069, 2024.

[78] Y. Andrew, C. Lim e E. Budiarto, “Mapping linux shell commands to MITRE ATT&CK using NLP-based approach”, in *2022 International Conference on Electrical Engineering and Informatics (ICELTICs)*, IEEE, 2022, pp. 37–42.

[79] Z. Ma, A. R. Chen, D. J. Kim, T.-H. Chen e S. Wang, “LLMParser: An exploratory study on using large language models for log parsing”, in *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*, 2024, pp. 1–13.

[80] G. Domalis, “GPT-2C: A GPT-2 parser for Cowrie honeypot logs”, *ArXiv*, 2021.

[81] E. Karlsen, X. Luo, N. Zincir-Heywood e M. Heywood, “Benchmarking large language models for log analysis, security, and interpretation”, *Journal of Network and Systems Management*, vol. 32, n. 3, p. 59, 2024.

[82] M. B. Ozkok, B. Birinci, O. Cetin, B. Arief e J. Hernandez-Castro, “Honeypot’s best friend? Investigating ChatGPT’s ability to evaluate honeypot logs”, in *Proceedings of the 2024 European Interdisciplinary Cybersecurity Conference*, 2024, pp. 128–135.

[83] M. Boffa et al., “LogPrécis: Unleashing language models for automated malicious log analysis: Précis: A concise summary of essential points, statements, or facts”, *Computers & Security*, vol. 141, p. 103 805, 2024.

[84] P. Rafiey e A. Namadchian, “Mapping Vulnerability Description to MITRE ATT&CK Framework by LLM”, *arXiv preprint*, 2025.

[85] M. Sauze-Kadar e T. Loubier, “A Multi-Model Approach to Enhance Automatic Matching of Vulnerabilities to Attack Patterns”, *arXiv preprint*, 2025.

[86] B. K. Webb, S. Purohit e R. Meyur, “Cyber knowledge completion using large language models”, *arXiv preprint arXiv:2409.16176*, 2024.

[87] J. J. Tejero-Fernández e A. Sánchez-Macián, “Evaluating Language Models For Threat Detection in IoT Security Logs”, *arXiv preprint arXiv:2507.02390*, 2025.

[88] A. Q. Jiang et al., “Mistral 7B”, *arXiv preprint arXiv:2310.06825*, vol. 3, 2023.

[89] Clarifai. “mistral-7B-Instruct”, visitato il giorno 14 mag. 2025. indirizzo: https://clarifai.com/mistralai/completion/models/mistral-7B-Instruct

[90] N. F. Liu et al., “Lost in the middle: How language models use long contexts”, *Transactions of the Association for Computational Linguistics*, vol. 12, pp. 157–173, 2024.