

Certified Tester Specialist Level Testing with Generative AI (CT-GenAI) Syllabus

versione italiana 1.1

International Software Testing Qualifications Board



Avviso di Copyright

Avviso di Copyright © International Software Testing Qualifications Board (di seguito denominato ISTQB®)

ISTQB® è un marchio registrato di International Software Testing Qualifications Board.

Copyright © 2025, gli autori Abbas Ahmad, Gualtiero Bazzana, Alessandro Collino, Olivier Denoo e Bruno Legeard.

Tutti i diritti riservati. Con la presente gli autori trasferiscono il copyright a ISTQB®. Gli autori (come attuali titolari del copyright) e ISTQB® (come futuro titolare del copyright) hanno concordato le seguenti condizioni di utilizzo:

È possibile copiare estratti di questo documento per uso non commerciale, a condizione che venga riconosciuta la fonte. Qualsiasi Training Provider accreditato (azienda accreditata alla Formazione) può utilizzare il presente Syllabus come base per un corso di formazione, a condizione che gli autori e ISTQB® siano riconosciuti come fonte e proprietari del copyright del Syllabus, e a condizione che qualsiasi pubblicità di tale corso di formazione possa menzionare il Syllabus solo dopo l'accREDITAMENTO ufficiale dei materiali di formazione da parte di un Member Board riconosciuto da ISTQB®.

Qualsiasi individuo o gruppo di individui può utilizzare questo Syllabus come base per articoli e libri, a condizione che gli autori e ISTQB® siano riconosciuti come fonte e proprietari del copyright del Syllabus.

È proibito qualsiasi altro utilizzo di questo Syllabus senza avere prima ottenuto l'approvazione scritta di ISTQB®.

Qualsiasi Member Board riconosciuto da ISTQB® può tradurre questo Syllabus, a condizione che riproduca il suddetto Avviso di Copyright nella versione tradotta del Syllabus.

Storico delle review

Versione	Data	Osservazioni
v1.0	2025/07/25	CT-GenAI v1.0 Release
v1.1	2026/04/27	CT-GenAI v1.1 Aggiornamento minore

Indice dei contenuti

Avviso di Copyright	2
Storico delle review	3
Indice dei contenuti	4
Ringraziamenti	7
0 Introduzione	8
0.1 Scopo di questo Syllabus	8
0.2 Testing del Software con l'AI Generativa	8
0.3 Percorso di Carriera per i Tester	8
0.4 Business Outcome	8
0.5 Obiettivi di Apprendimento Esaminabili, Obiettivi Pratici e Livelli di Conoscenza	9
0.6 L'Esame per il Certificato di Certified Tester Testing with Generative AI	10
0.7 Accreditamento	10
0.8 Gestione degli Standard	10
0.9 Livello di Dettaglio	10
0.10 Come è Organizzato questo Syllabus	11
1 Introduzione all'AI Generativa per il Testing del Software – 100 minuti	13
1.1 Fondamenti e Concetti Chiave dell'AI Generativa	14
1.1.1 Spettro dell'AI: AI simbolica, machine learning classico, deep learning e AI generativa	14
1.1.2 Basi dell'AI generativa e dei LLM	15
1.1.3 LLM di base, ottimizzati con istruzioni e di ragionamento	16
1.1.4 LLM multimodali e modelli linguistico-visivi	17
1.2 Utilizzare l'AI Generativa nel Testing del Software: Principi Fondamentali	17
1.2.1 Capacità principali dei LLM per le attività di test	18
1.2.2 Chatbot AI e applicazioni di testing alimentate da LLM per il testing del software	18
2 Ingegneria dei Prompt per un Testing del Software Efficace – 365 minuti	20
2.1 Sviluppo Efficace dei Prompt	22
2.1.1 Struttura dei prompt per l'AI generativa nel testing del software	22
2.1.2 Tecniche di prompting fondamentali per il testing del software	23
2.1.3 Prompt di sistema e prompt utente	24

2.2	Applicare le Tecniche di Ingegneria dei Prompt alle Attività di Test del Software	25
2.2.1	Analisi dei test con l'AI generativa	25
2.2.2	Progettazione dei test e implementazione dei test con l'AI generativa	27
2.2.3	Regression testing automatizzato con l'AI generativa	28
2.2.4	Monitoraggio dei test e controllo dei test con l'AI generativa	30
2.2.5	Scegliere le tecniche di prompting per il testing del software	31
2.3	Valutare i Risultati dell'AI Generativa e Ottimizzare i Prompt per le Attività di Test del Software	32
2.3.1	Metriche per valutare i risultati dell'AI generativa sulle attività di test	32
2.3.2	Tecniche per valutare e ottimizzare iterativamente i prompt	33
3	Gestire i Rischi dell'AI Generativa nel Testing del Software – 160 minuti	35
3.1	Allucinazioni, Errori di Ragionamento e Bias.....	36
3.1.1	Allucinazioni, errori di ragionamento e bias nell'AI generativa.....	36
3.1.2	Identificare allucinazioni, errori di ragionamento e bias nell'output del LLM.....	36
3.1.3	Tecniche di mitigazione delle allucinazioni, degli errori di ragionamento e dei bias della GenAI nelle attività di test del software	38
3.1.4	Mitigare il comportamento non deterministico dei LLM.....	39
3.2	Rischi della Privacy dei Dati e della Sicurezza dell'AI Generativa nel Testing del Software	39
3.2.1	Rischi della privacy dei dati e della sicurezza associati all'uso dell'AI generativa	39
3.2.2	Privacy dei dati e vulnerabilità nell'AI generativa per i processi e gli strumenti di test	40
3.2.3	Strategie di mitigazione per proteggere la privacy dei dati e migliorare la sicurezza nel testing con l'AI generativa	40
3.3	Consumo Energetico e Impatto Ambientale dell'AI Generativa per il Testing del Software.....	42
3.3.1	L'impatto dell'uso della GenAI sul consumo energetico e sulle emissioni di CO ₂	42
3.4	Normative, Standard e Framework delle Best Practice per l'AI	42
3.4.1	Normative, standard e framework dell'AI rilevanti per la GenAI nel testing del software	43
4	Infrastruttura di Test Alimentata da LLM per il Testing del Software – 110 minuti	44
4.1	Approcci Architetture per l'Infrastruttura di Test Alimentata da LLM	45
4.1.1	Componenti architetture e concetti principali dell'infrastruttura di test alimentata da LLM .	45
4.1.2	Retrieval-Augmented Generation.....	46
4.1.3	Il ruolo degli agenti alimentati da LLM nell'automazione dei processi di test.....	47
4.2	Fine-Tuning e LLMops: Rendere Operativa l'AI Generativa per il Testing del Software	48

4.2.1	Fine-tuning dei LLM per le attività di test	48
4.2.2	LLMOps durante il deployment e la gestione dei LLM per il testing del software	49
5	Implementare e Integrare l'AI Generativa nelle Organizzazioni di Test – 80 minuti	50
5.1	Roadmap per l'Adozione dell'AI Generativa nel Testing del Software	51
5.1.1	Rischi dell'AI ombra	51
5.1.2	Aspetti chiave di una strategia di AI generativa nel testing del software	51
5.1.3	Selezionare i LLM/SLM per attività di test del software	52
5.1.4	Fasi durante l'adozione dell'AI generativa nel testing del software	52
5.2	Gestire le Modifiche quando si Adotta l'AI Generativa per il Testing del Software	53
5.2.1	Competenze e conoscenze essenziali per il testing con l'AI generativa	53
5.2.2	Sviluppo di capacità con l'AI generativa nei team di test	54
5.2.3	Evolgere i processi di test nelle organizzazioni di test abilitate all'AI	54
6	Riferimenti	55
	Standard	55
	Documenti ISTQB®	55
	Riferimenti del glossario	55
	Libri	55
	Articoli	55
	Pagine web	56
7	Appendice A – Obiettivi di Apprendimento/Livello Cognitivo di Conoscenza	57
	Livello 1: Ricordare (K1)	57
	Livello 2: Comprendere (K2)	57
	Livello 3: Applicare (K3)	58
8	Appendice B – Matrice di tracciabilità dei Business Outcome agli Obiettivi di Apprendimento	59
9	Appendice C – Release Note	66
10	Appendice D – Termini Specifici dell'AI Generativa	67
11	Appendice E – Marchi commerciali	70
12	Indice	71

Ringraziamenti

Questo documento è stato formalmente rilasciato dalla General Assembly di ISTQB® il 25/07/2025.

È stato prodotto da un team di International Software Testing Qualifications Board: Abbas Ahmad (product owner), Gualtiero Bazzana, Alessandro Collino, Olivier Denoo e Bruno Legeard (technical manager).

Il team ringrazia Anne Kramer, Jędrzej Kwapinski, Samuel Ouko e Ina Schieferdecker per la loro review tecnica e il team di review e i Member Board per i loro suggerimenti e contributi.

Le seguenti persone hanno partecipato alla revisione, ai commenti e alla votazione di questo syllabus:

Albert Laura, Aneta Derkova, Anne Kramer, Arda Ender Torçuk, Baris Sarialioglu, Claire Van Der Meulen, Daniel van der Zwan, Derek Young, Dietmar Gehring, Francisca Cano Ortiz, Gary Mogyorodi, Gergely Ágnes, Horst Pohlmann, Ina Schieferdecker, Ingvar Nordström, Jan Sabak, Jaroslaw Hryszko, Jędrzej Kwapinski, Joanna Kazun, Karol Frühauf, Katalin Balla, Koray Yitmen, Laura Albert, Linda Vreeswijk, Lucjan Stapp, Lukáš Piška, Mario Winter, Mattijs Kemmink, Matthias Hamburg, Meile Posthuma, Michael Stahl, Márton Siska, Nele Van Asch, Nils Röttger, Nishan Portoyan, Piet de Roo, Piotr Wicherski, Péter Földházi, Péter Sótér, Radosław Smilgin, Ralf Pichler, Renzo Cerquozzi, Rik Marselis, Samuel Ouko, Stephanie Ulrich, Stuart Reid, Tal Pe'er, Tamás Gergely, Thomas Letzkus, Wim Decoutere, Zsolt Hargitai, Mark Rutz, Patrick Quilter, Earl Burba, Taz Daughtrey, Judy McKay, Randall Rice, Thomas Adams, Tom Van Ongeval, Sander Mol, Miroslav Renda, Geng Chen, Chai Afeng, Xinghan Li, Klaudia Dussa-Zieger, Arnd Pehl, Florian Fieber, Ray Gillespie, József Kreisz, Dénes Medzihradszky, Ferenc Hamori, Nicola De Rosa, Massimo Di Carlo, Dario Galiero, Giorgio Pisani, Giancarlo Tomasig, Young jae Choi, Arnika Hryszko, Andrei Brovko, Ilia Kulakov, Praveen, Kostas Pashalidis, Ferdinand Gramsamer, A. Berfin Öztas, Abdullah Gök, Abdurrahman AKIN, Aleyna Zuhail İŞİK, Anıl Şahin, Atakan Erdemgil, Aysel Bilici, Azmi YÜKSEL, Bilal Gelik, Bilge Yazıcı, Burak Gel, Burcu ÖZEL, Büşra İlayda Çevik Köken, Can Polat, Canan Ayten Dörtkol (Polat), Cansu Mercan Daldaban, Denizcan Orhun Karaca, Didem Çiçek Bay, Duygu Yalçinkaya, Efe Can Yemez, ELİF CERAV, Emine Tekiner, Emre Aman, Emre Can Akgül, Esra Küçük, Gençay GENÇ, Gül Çalışır Acan, Gül Nihal SİNGİL, Güler GÖK, Gulhanim Anulur, Hakan GÜVEZ, Haktan Bilgehan Dilber, Halil Ibrahim Tasdemir, Hasan Küçükayar, Hatice Erdoğan, Hatice Kübra Daşdoğan, Hüseyin Sevkî ARI, Hyulya Gyuler, İLKNUR NEŞE TUNCAL, Kaan Eminğlu, Kamil Isik, Koray Danışman, Melisa Canbaz, Merve Guleroglu, Müjde Ceylan, Mustafa Furkan CEYLAN, Nergiz Gençaslan, Nuh Soner Bozkurt, Omer Fatih Poyraz, Onur Ersoy, Özlem Körpe, Özgür Özdemir, Sedat YOLTAY, Selahattin Aliyazıcıoğlu, Sevan Lalikoğlu, Sebastian Malyska, Sevim Öykü Demirel, Tatsiana Beliai, Tayg.

Traduzione italiana a cura di ITA-STQB (www.ita-stqb.org) – info@ita-stqb.org

Revisione della traduzione italiana a cura di: Raffaele Bozzini, Salvatore Reale

0 Introduzione

0.1 Scopo di questo Syllabus

Questo Syllabus costituisce la base per l'International Software Testing Qualifications Board per la qualifica di Testing with Generative AI (CT-GenAI). ISTQB® fornisce questo Syllabus come segue:

1. Ai Member Board, per la traduzione nella loro lingua locale e per l'accreditamento dei Training Provider. I Member Board possono adattare il Syllabus alle loro specifiche esigenze linguistiche e modificare i riferimenti per adattarli alle loro pubblicazioni locali.
2. Agli organismi di certificazione, per derivare le domande d'esame nella loro lingua locale adattate agli obiettivi di apprendimento di questo Syllabus.
3. Ai Training Provider, per produrre materiale didattico e determinare i metodi di insegnamento appropriati.
4. Ai candidati alla certificazione, per prepararsi all'esame di certificazione (come parte di un corso di formazione o in modo indipendente).
5. Alla comunità internazionale di System Engineering e Software Engineering, per promuovere la professione del testing dei sistemi e del software, e come base per libri e articoli.

0.2 Testing del Software con l'AI Generativa

La certificazione di Testing with Generative AI è rivolta a chiunque sia coinvolto nell'utilizzo dell'AI generativa (GenAI) per il testing del software. Ciò include persone che ricoprono ruoli come tester, test analyst, test automation engineer, test manager, user acceptance tester e sviluppatori di software. La certificazione Testing with GenAI è adatta anche a tutti coloro che desiderano una comprensione di base dell'utilizzo della GenAI per il testing del software, come project manager, quality manager, software development manager, business analyst, IT director e consulenti di gestione.

0.3 Percorso di Carriera per i Tester

Lo schema ISTQB® fornisce supporto ai professionisti del testing in tutte le fasi della loro carriera, offrendo una conoscenza ampia e approfondita. Chi ha conseguito la certificazione ISTQB® Certified Tester Testing with Generative AI può essere interessato anche ai Core Advanced Level (Test Analyst, Technical Test Analyst, Test Manager e Test Engineering) e successivamente agli Expert Level (Test Management o Improving the Test Process). È possibile visitare il sito www.istqb.org per le ultime informazioni sul Certified Tester Scheme di ISTQB®.

0.4 Business Outcome

Questo paragrafo elenca i Business Outcome (Risultati di Business) previsti da una persona che ha conseguito la certificazione Testing with Generative AI.

Una persona che ha conseguito la certificazione Testing with Generative AI è in grado di:

GenAI-BO1	Comprendere i concetti fondamentali, le capacità e i limiti dell'AI generativa
GenAI-BO2	Sviluppare competenze pratiche nel prompting di large language model per il testing del software
GenAI-BO3	Acquisire approfondimenti sui rischi e sulle mitigazioni dell'utilizzo dell'AI generativa per il testing del software
GenAI-BO4	Acquisire approfondimenti sulle applicazioni delle soluzioni di AI generativa per il testing del software
GenAI-BO5	Contribuire in modo efficace alla definizione e all'implementazione di una strategia e di una roadmap di AI generativa per il testing del software all'interno di un'organizzazione

0.5 Obiettivi di Apprendimento Esaminabili, Obiettivi Pratici e Livelli di Conoscenza

Gli Obiettivi di Apprendimento (LO, Learning Objective) e gli Obiettivi Pratici (HO, Hands-on Objective) supportano i business outcome e vengono utilizzati per creare gli esami di certificazione Testing with Generative AI.

In generale, tutti i contenuti di questo Syllabus sono esaminabili ai livelli K1, K2 e K3, ad eccezione di Introduzione, Obiettivi Pratici e Appendici. Le domande d'esame verificheranno la conoscenza delle parole chiave al livello K1 (vedi di seguito) o degli Obiettivi di Apprendimento a tutti i livelli K.

I livelli specifici degli Obiettivi di Apprendimento sono specificati all'inizio di ogni capitolo e sono classificati come segue:

- K1: Ricordare
- K2: Capire
- K3: Applicare

Ulteriori dettagli ed esempi di Obiettivi di Apprendimento sono riportati nell'Appendice A.

Tutti i termini elencati come parole chiave sotto i titoli dei capitoli devono essere ricordati (K1), anche se non esplicitamente menzionati negli Obiettivi di Apprendimento.

Gli Obiettivi Pratici specifici (HO) sono specificati all'inizio di ogni capitolo. Ogni HO è collegato a un LO di livello K2 o K3, con l'obiettivo di migliorare l'apprendimento attraverso la pratica. Il livello di un HO è classificato come segue:

- H0: Può includere una live demo di un esercizio o un video registrato. Poiché non viene eseguito dal candidato, non si tratta di un esercizio vero e proprio.
- H1: Esercizio guidato. I candidati seguono una sequenza di passaggi eseguiti dal formatore.

- H2: Esercizio con suggerimenti. Al candidato viene assegnato un esercizio con suggerimenti rilevanti per consentirgli di risolverlo entro il tempo stabilito.

0.6 L'Esame per il Certificato di Certified Tester Testing with Generative AI

L'esame per il Certificato di Certified Tester Testing with Generative AI si baserà su questo Syllabus. Le risposte alle domande d'esame possono richiedere l'uso di materiale basato su più di un paragrafo di questo Syllabus. Tutti i paragrafi del Syllabus sono esaminabili, ad eccezione dell'Introduzione, degli Obiettivi Pratici e delle Appendici. Gli standard, i libri e gli articoli sono inclusi come riferimenti, ma il loro contenuto non è esaminabile, al di là di quanto sintetizzato in questo Syllabus.

Per ulteriori dettagli, consultare il documento Exam Structures and Rules V1.0 per il Certified Tester Testing with Generative AI.

Nota per i Requisiti di Ingresso: il certificato ISTQB® Foundation Level deve essere ottenuto prima di sostenere l'esame di certificazione ISTQB® Certified Tester Testing with Generative AI.

0.7 Accredитamento

Un Member Board ISTQB® può accreditare Training Provider, il cui materiale didattico segue questo Syllabus. I Training Provider dovrebbero ottenere le linee guida per l'accreditamento dal Member Board o dall'ente che effettua l'accreditamento. Un corso accreditato viene riconosciuto essere conforme a questo Syllabus ed è autorizzato a gestire un esame ISTQB® come parte del corso stesso.

Le linee guida per l'accreditamento di questo Syllabus sono definite nel documento ISTQB CT-GenAI Accreditation Guidelines.

0.8 Gestione degli Standard

Esistono standard associati alle caratteristiche di qualità e al testing del software, ovvero quelli citati nel Syllabus Foundation Level, come ad esempio IEEE e ISO. Lo scopo di questi riferimenti è fornire un framework o una fonte di informazioni aggiuntive, se il lettore lo desidera. Si noti che i syllabi utilizzano questi standard come riferimenti. Il contenuto degli standard non è oggetto di esame. Per ulteriori informazioni sugli standard, fare riferimento al Capitolo 6 Riferimenti.

0.9 Livello di Dettaglio

Il livello di dettaglio di questo Syllabus consente di organizzare corsi ed esami consistenti a livello internazionale. Per raggiungere questo obiettivo, il Syllabus consiste di:

- Obiettivi didattici generali che descrivono i propositi della certificazione ISTQB® Certified Tester Testing with Generative AI
- Una lista di termini (parole chiave) che gli studenti devono essere in grado di ricordare
- Obiettivi di apprendimento per ogni area di conoscenza, che descrivono i risultati cognitivi di apprendimento che devono essere raggiunti

- Una descrizione dei concetti chiave, che include i riferimenti a fonti riconosciute come accettate dalla letteratura o dagli standard
- Una descrizione di ogni obiettivo pratico della pratica raccomandata a supporto dell'apprendimento

Il contenuto del Syllabus non è una descrizione dell'intera area di conoscenza del testing con la GenAI; riflette il livello di dettaglio che deve essere coperto dai corsi di formazione ISTQB® Certified Tester Testing with Generative AI. Si focalizza sui concetti e sulle tecniche di test che possono essere applicati a tutti i progetti software quando si utilizza l'AI generativa per il testing.

Il Syllabus utilizza la terminologia (cioè il nome e il significato) dei termini utilizzati nel testing del software e nella Quality Assurance secondo il Glossario ISTQB®.

0.10 Come è Organizzato questo Syllabus

Sono presenti 5 capitoli con contenuti esaminabili. Nell'istestazione principale di ogni capitolo è specificato il tempo per il capitolo; la tempistica non viene fornita al di sotto del livello del capitolo. Per i corsi di formazione accreditati, il Syllabus richiede un minimo di 13,6 ore di lezione, distribuite nei 5 capitoli come segue:

- Capitolo 1: 100 minuti - Introduzione all'AI Generativa per il Testing del Software
 - Il tester apprende le nozioni di base dei large language model (LLM), comprese la tokenizzazione e le capacità multimodali.
 - Il tester esplora le applicazioni dell'AI Generativa (GenAI) nel testing del software, distinguendo i chatbot AI dagli strumenti di test alimentati da LLM, e sperimentando con la tokenizzazione, le finestre di contesto e i prompt multimodali.
- Capitolo 2: 365 minuti - Ingegneria dei Prompt per un Testing del Software Efficace
 - Il tester apprende come creare prompt efficaci e strutturati per la GenAI nel testing del software.
 - Il tester acquisisce esperienza pratica con le tecniche di ingegneria dei prompt per le attività di test del software e le applica.
- Capitolo 3: 160 minuti - Gestire i Rischi dell'AI Generativa nel Testing del Software
 - Il tester apprende come identificare e mitigare allucinazioni, errori di ragionamento e bias durante il testing con la GenAI.
 - Il tester apprende come affrontare i problemi relativi alla privacy dei dati e alla sicurezza della GenAI nel testing del software.
 - Il tester apprende il consumo energetico e l'impatto ambientale della GenAI nel testing del software.
 - Il tester apprende le normative, gli standard e le best practice dell'AI per un uso etico, trasparente e sicuro della GenAI nel testing del software.
- Capitolo 4: 110 minuti - Infrastruttura di Test Alimentata da LLM per il Testing del Software

- Il tester esplora l'architettura della GenAI come Retrieval-Augmented Generation e gli agenti GenAI.
 - Il tester apprende il processo di ottimizzazione dei LLM per le attività di test del software.
 - Il tester apprende i concetti di Large Language Model Operations (LLMOps) per il deployment e la gestione dei LLM nel testing del software.
- Capitolo 5: 80 minuti - Implementare e Integrare l'AI Generativa nelle Organizzazioni di Test
 - Il tester apprende una roadmap strutturata per l'integrazione della GenAI nei processi di test.
 - Il tester apprende la trasformazione organizzativa per l'integrazione della GenAI nei processi di test.

1 Introduzione all'AI Generativa per il Testing del Software – 100 minuti

Parole Chiave

Nessuna

Parole Chiave Specifiche dell'AI Generativa

chatbot AI, finestra di contesto, deep learning, embedding, funzionalità, LLM di base (foundation), AI generativa, generative pre-trained transformer, LLM ottimizzato con istruzioni (instruction-tuned), large language model, machine learning, modello multimodale, LLM di ragionamento (reasoning), AI simbolica, tokenizzazione, transformer

Obiettivi di Apprendimento e Obiettivi Pratici per il Capitolo 1:

1.1 Fondamenti e Concetti Chiave dell'AI Generativa

- | | | |
|-------------|------|---|
| GenAI-1.1.1 | (K1) | Richiamare i diversi tipi di AI: AI simbolica, machine learning classico, deep learning e AI generativa |
| GenAI-1.1.2 | (K2) | Spiegare le basi dell'AI generativa e dei large language model |
| HO-1.1.2 | (H1) | Esercitarsi nella tokenizzazione e nella valutazione del conteggio dei token quando si utilizza un LLM per un'attività di test del software |
| GenAI-1.1.3 | (K2) | Distinguere tra LLM di base, ottimizzati con istruzioni e di ragionamento |
| GenAI-1.1.4 | (K2) | Scrivere ed eseguire un determinato prompt che affronta un'attività di test utilizzando un modello LLM multimodale |
| HO-1.1.4 | (H1) | Scrivere ed eseguire un prompt per un LLM multimodale utilizzando input testuali e immagini per un'attività di test del software |

1.2 Utilizzare l'AI Generativa nel Testing del Software: Principi Fondamentali

- | | | |
|-------------|------|--|
| GenAI-1.2.1 | (K2) | Fornire esempi delle capacità principali dei LLM per le attività di test |
| GenAI-1.2.2 | (K2) | Confrontare i modelli di interazione quando si utilizza la GenAI per il testing del software |

1.1 Fondamenti e Concetti Chiave dell'AI Generativa

L'Intelligenza Artificiale Generativa (GenAI) è un ramo dell'intelligenza artificiale che utilizza modelli di grandi dimensioni e pre-addestrati per generare output simili a quelli umani, come testo, immagini o codice. I large language model (LLM) sono modelli GenAI pre-addestrati su grandi set di dati testuali, che consentono loro di determinare il contesto e produrre risposte rilevanti in base ai prompt dell'utente.

I concetti chiave includono la tokenizzazione (ovvero la suddivisione del testo in unità per un'elaborazione efficiente), le finestre di contesto (che limitano la quantità di informazioni considerate contemporaneamente per mantenere la pertinenza) e i modelli multimodali (in grado di elaborare più tipi di dati come testo, immagini e audio per interazioni avanzate).

Nel testing del software, questi LLM possono supportare attività quali la revisione e il miglioramento dei criteri di accettazione, la generazione di test case o test script, l'identificazione di potenziali difetti, l'analisi dei modelli di difetti, la generazione di dati di test sintetici o il supporto alla generazione di documentazione, durante l'intero processo di test.

1.1.1 Spettro dell'AI: AI simbolica, machine learning classico, deep learning e AI generativa

L'Intelligenza Artificiale (AI) è un campo ampio che comprende diversi tipi di tecnologie, ciascuna con il proprio modo unico di risolvere i problemi, come l'AI simbolica, il machine learning classico, il deep learning e la GenAI (tra le altre tecnologie che esulano dall'ambito di questo Syllabus):

- L'AI simbolica utilizza un sistema rule-based per imitare il processo decisionale umano. Essenzialmente, l'AI simbolica rappresenta la conoscenza utilizzando simboli e regole logiche.
- Il machine learning classico è un approccio data-driven che richiede la preparazione dei dati, la selezione delle funzionalità e l'addestramento dei modelli, e può essere utilizzato per attività quali la categorizzazione dei difetti e la previsione dei problemi del software.
- Il deep learning utilizza strutture di machine learning chiamate reti neurali per apprendere automaticamente le funzionalità dai dati. I modelli di deep learning sono in grado di trovare modelli in set di dati molto grandi e complessi, come immagini, video, audio o testo, senza che gli utenti debbano definire manualmente le funzionalità, anche se nella pratica può essere comunque necessario il coinvolgimento umano in attività quali l'annotazione dei dati, il perfezionamento dei modelli o la validazione dei risultati.
- L'AI generativa utilizza tecniche di deep learning per creare nuovi contenuti (testo, immagini, codice) apprendendo e imitando modelli dai dati di addestramento. Modelli come i LLM possono generare testo, scrivere codice e simulare ragionamenti o risolvere problemi nell'ambito del loro addestramento.

In sintesi, il campo dell'AI si è evoluto in diverse direzioni, ognuna con diversi punti di forza e limiti. Il vantaggio principale dell'utilizzo della GenAI per il testing del software è che utilizza modelli pre-addestrati che possono essere applicati direttamente alle attività di test senza la necessità di una fase di addestramento aggiuntiva, anche se questo comporta alcuni rischi (vedere la sezione 3.1).

1.1.2 Basi dell'AI generativa e dei LLM

In base al modello di deep learning del generative pre-trained transformer, i LLM vengono addestrati su set di dati molto grandi, inclusi libri, articoli e siti web. Gli small language model (SLM) sono modelli compatti con meno parametri rispetto ai large language model, progettati per fornire soluzioni GenAI leggere e mirate.

I LLM sono in grado di gestire le sfumature linguistiche e di generare contenuti coerenti. Due concetti chiave che aiutano i LLM a elaborare e generare contenuti sono la tokenizzazione e gli embedding. La tokenizzazione e gli embedding convertono il linguaggio in una forma numerica che il modello può elaborare in modo efficace.

- La tokenizzazione nei language model è il processo di suddivisione del testo in unità più piccole chiamate token. I token possono essere piccoli come un carattere o grandi come una parola o una parte di parola. Quando un LLM elabora una frase, prima tokenizza l'input in modo che ogni token possa essere compreso individualmente, mantenendo al contempo il contesto generale.
- Gli embedding sono rappresentazioni numeriche dei token che codificano le loro relazioni semantiche, sintattiche e contestuali in un formato adatto all'elaborazione da parte dei modelli di AI generativa. Ogni token viene trasformato in un vettore in uno spazio ad alta dimensionalità, catturando informazioni sfumate sul suo significato e utilizzo. I token con significati o ruoli contestuali simili hanno embedding posizionati vicini tra loro in questo spazio. Questa vicinanza consente ai LLM di comprendere le relazioni tra le parole, mantenere il contesto e generare risposte coerenti e contestualmente appropriate.

I LLM utilizzano un'architettura di rete neurale nota come modello transformer. I modelli transformer eccellono nelle attività linguistiche elaborando il contesto di sequenze di testo estese e apprendendo come i token sono correlati tra loro. Durante l'inferenza, i LLM prevedono il token successivo in una sequenza, utilizzando queste relazioni apprese per generare un testo coerente e contestualmente appropriato. Il modello transformer può essere utilizzato per generare nuovo testo statisticamente plausibile, basato sui dati di addestramento e sul prompt. Ma plausibile non significa necessariamente corretto.

I LLM mostrano un comportamento non deterministico dovuto principalmente alla natura probabilistica dei loro meccanismi di inferenza e alle impostazioni degli iperparametri. Questa casualità intrinseca può portare a variazioni negli output anche quando viene fornito più volte lo stesso input.

Nel campo dei LLM, la finestra di contesto si riferisce alla quantità di testo precedente, misurata in token, che il modello può considerare durante la generazione delle risposte. Una finestra di contesto più ampia consente al modello di mantenere la coerenza su passaggi più lunghi, ad esempio durante l'analisi di test log di grandi dimensioni. Tuttavia, aumentando il numero di token nella finestra di contesto aumenta anche la complessità computazionale e il tempo di elaborazione necessario affinché il modello funzioni in modo efficace.

Obiettivo Pratico HO-1.1.2 (H1): Esercitarsi nella tokenizzazione e nella valutazione del conteggio dei token quando si usa un LLM per un'attività di test del software

Questa attività pratica è progettata per aiutare i partecipanti a sviluppare una comprensione pratica della tokenizzazione e delle sue implicazioni quando si lavora con i LLM. L'esercizio è diviso in due parti principali:

- Tokenizzazione: utilizzare un tokenizer per suddividere un campione di testo in singoli token. Esaminare l'output per vedere come sono rappresentate le parole, la punteggiatura e le frasi e identificare i modelli o le sfumature nella tokenizzazione.
- Valutazione del conteggio dei token: misurare il numero di token generati da vari testi in input. Analizzare come il conteggio dei token influisce sulle prestazioni del modello, in particolare in relazione ai limiti della finestra di contesto del modello e alle considerazioni sull'efficienza.

Al termine di questo esercizio, i partecipanti saranno in grado di anticipare meglio come le diverse strutture di testo e lunghezze di input possono influenzare le interazioni con i LLM.

1.1.3 LLM di base, ottimizzati con istruzioni e di ragionamento

I Large Language Model vengono sviluppati attraverso fasi di addestramento progressivamente specializzate per migliorarne l'efficacia in un'ampia gamma di attività. Queste fasi danno origine a tre categorie principali: LLM di base (foundation), LLM ottimizzati con istruzioni (instruction-tuned) e LLM di ragionamento (reasoning).

- LLM di base: sono modelli di uso generale addestrati su set di dati vasti e diversificati che comprendono testo, codice, immagini e altre modalità. Il loro ampio pre-addestramento consente loro di supportare varie attività in diversi ambiti, quali l'elaborazione del linguaggio naturale, la visione artificiale e il riconoscimento vocale. Sebbene potenti e flessibili, i LLM di base richiedono in genere un ulteriore adattamento per soddisfare i requisiti specifici delle attività.
- LLM ottimizzati con istruzioni: derivati dai modelli di base, i LLM ottimizzati con istruzioni vengono ottimizzati utilizzando set di dati che abbinano i prompt alle risposte attese. Questa fase migliora il loro allineamento con le istruzioni umane, migliorando l'usabilità nelle applicazioni del mondo reale. Il processo di ottimizzazione prevede l'ottimizzazione dell'aderenza all'attività, del rispetto delle istruzioni e della coerenza delle risposte, migliorando così la capacità del modello di interpretare e agire efficacemente sull'intenzione dell'utente.
- LLM di ragionamento: i modelli di ragionamento estendono i modelli ottimizzati con istruzioni enfatizzando le capacità cognitive strutturate come l'inferenza logica, la risoluzione di problemi in più fasi e il ragionamento a catena. Questi modelli vengono ulteriormente addestrati o ottimizzati su attività accuratamente selezionate che richiedono comprensione contestuale, passaggi di ragionamento intermedi e sintesi di informazioni complesse. Di conseguenza, sono più adatti per attività ad alto carico cognitivo, comprese quelle in ambiti tecnici.

Nel contesto delle applicazioni GenAI per il testing del software, vengono utilizzati sia LLM ottimizzati con istruzioni (talvolta indicati come non-reasoning) sia LLM orientati al reasoning. La scelta dipende dalla complessità e dalle esigenze di ragionamento della specifica attività di testing da svolgere.

1.1.4 LLM multimodali e modelli linguistico-visivi

I LLM multimodali estendono il tradizionale modello transformer per elaborare più modalità di dati, inclusi testo, immagini, suoni e video. Questi modelli sono addestrati su set di dati grandi e diversificati che consentono loro di apprendere le relazioni tra diversi tipi di dati. Per gestire varie modalità, la tokenizzazione viene adattata per ogni tipo di dati: ad esempio, le immagini vengono convertite in embedding utilizzando modelli linguistico-visivi prima di essere elaborate nel modello transformer.

I modelli linguistico-visivi, un sottoinsieme dei LLM multimodali, integrano in modo specifico informazioni visive e testuali per eseguire attività quali la didascalia delle immagini, la risposta a domande visive e l'analisi della coerenza tra input testuali e visivi.

Nel testing del software, i LLM multimodali, in particolare i LLM potenziati con modelli linguistico-visivi, offrono opportunità significative. Sono in grado di analizzare elementi visivi delle applicazioni, come screenshot e wireframe della GUI (Graphical User Interface), insieme alle descrizioni testuali associate, come defect report o user story. Questa capacità consente ai tester di identificare discrepanze tra i risultati attesi e gli elementi visivi effettivi su uno screenshot. Inoltre, i LLM potenziati con modelli linguistico-visivi possono generare test case avanzati e realistici che incorporano sia dati testuali che segnali visivi, aumentando così la copertura complessiva.

Obiettivo pratico HO-1.1.4 (H1): Scrivere ed eseguire un determinato prompt che riguarda un'attività di test utilizzando un modello LLM multimodale

Questo esercizio prevede la scrittura e l'esecuzione di un determinato prompt per un LLM multimodale utilizzando come input sia testo che immagini per risolvere un'attività di test in due passaggi:

- Revisionare gli input: revisionare il prompt e i dati di input (testo e immagini).
- Eseguire il prompt e verificare il risultato effettivo: utilizzare un LLM multimodale per inserire sia immagini che testo e controllare la risposta del LLM.

Questo esercizio dimostra come utilizzare i LLM multimodali per un'attività che coinvolge come input sia testo che immagini in use case di testing del software, compreso riconoscere i vantaggi e le potenziali sfide coinvolti.

1.2 Utilizzare l'AI Generativa nel Testing del Software: Principi Fondamentali

La GenAI offre capacità trasformatrice in varie attività di test. I LLM eccellono nell'elaborazione del linguaggio naturale e del codice, nella generazione di testi e codici coerenti, nella risposta a domande, nella sintesi di informazioni, nella traduzione di lingue e nell'analisi di immagini in un contesto multimodale.

I professionisti del test in tutti i ruoli possono utilizzare la GenAI in due modi complementari: attraverso chatbot GenAI che forniscono risposte immediate alle query e attraverso applicazioni alimentate da LLM integrate negli strumenti di test.

1.2.1 Capacità principali dei LLM per le attività di test

I LLM sono in grado di interpretare requisiti, specifiche, screenshot, codice, test case e defect report, rendendoli strumenti per comprendere e chiarire le informazioni necessarie durante tutto il processo di test e generando elementi del testware. Di seguito sono riportate alcune delle capacità principali dei LLM rilevanti per il testing del software:

- Analisi e miglioramento dei requisiti: i LLM possono aiutare ad analizzare i requisiti e altri elementi della base di test, identificando ambiguità, incongruenze o informazioni mancanti. Possono generare domande significative per aiutare a chiarire i requisiti durante le discussioni con gli stakeholder.
- Supporto alla creazione di test case: i LLM possono aiutare a generare test case e suggerire obiettivi di test basati sui requisiti di sistema, sulle user story o su qualsiasi altro elemento della base di test.
- Generazione dell'oracolo del test: i LLM possono aiutare a generare i risultati attesi.
- Generazione di dati di test: i LLM possono generare set di dati, impostare valori limite e creare diverse combinazioni di dati di test.
- Supporto alla test automation: i LLM possono aiutare a generare test script dalla descrizione dei test case e migliorare i test script esistenti suggerendo modifiche e identificando tecniche di test appropriate.
- Analisi dei risultati dei test: i LLM possono aiutare ad analizzare i risultati dei test creando riepiloghi e classificando le anomalie in base alla severità e alla priorità.
- Creazione di testware: i LLM possono aiutare a creare vari documenti, inclusi test plan, test report e defect report, e mantenerli aggiornati man mano che il progetto evolve.

Queste capacità dimostrano come i LLM possono influire su vari aspetti del testing del software durante l'intero processo di test.

1.2.2 Chatbot AI e applicazioni di testing alimentate da LLM per il testing del software

I chatbot AI e le applicazioni di testing alimentate da LLM possono entrambi assistere i tester, sebbene differiscano in termini di funzionalità, flessibilità e approcci di integrazione.

I chatbot AI forniscono un'interfaccia conversazionale user-friendly che consente ai tester di comunicare direttamente con i LLM. Questa interazione in linguaggio naturale permette ai tester di inserire domande, comandi o prompt e ricevere risposte immediate e contestualizzate. Attraverso tecniche come il prompt chaining, i tester possono ottimizzare iterativamente gli output, rendendo i chatbot particolarmente efficaci per le attività di routine, il testing esplorativo e persino l'onboarding di nuovi tester, fornendo un accesso rapido alle conoscenze e alle pratiche di testing.

Questi chatbot AI sono particolarmente utili in scenari che richiedono un feedback rapido, chiarimenti sui concetti di test o l'esplorazione dinamica dei requisiti e dei potenziali test case. La loro interfaccia intuitiva li rende accessibili anche agli stakeholder non tecnici, ampliando la base di utenti potenziali e incoraggiando un'adozione più ampia.

Le applicazioni di testing alimentate da LLM, al contrario, prevedono l'integrazione delle capacità dei LLM tramite API per eseguire attività di testing ben definite e spesso automatizzate. Queste applicazioni offrono una maggiore personalizzazione e scalabilità, consentendo alle organizzazioni e ai fornitori di strumenti di integrare l'AI generativa nei framework di test esistenti. Ciò consente l'automazione di attività ripetitive o complesse, come la generazione di test case, l'analisi dei difetti o la sintesi dei dati di test. In implementazioni più avanzate, le organizzazioni possono creare agenti AI progettati specificamente per svolgere determinati ruoli di testing (vedere il Capitolo 4).

Indipendentemente dal modo in cui il tester interagisce con i LLM, sia attraverso chatbot che applicazioni integrate alimentate da LLM, l'implementazione di successo dell'AI generativa nel testing richiede una solida ingegneria dei prompt (vedere il Capitolo 2). Prompt progettati con cura e istruzioni chiare e specifiche sono essenziali per migliorare la probabilità che gli output generati dai LLM siano accurati, rilevanti e in linea con gli obiettivi del testing. Questa pratica aiuta a massimizzare il valore derivato dall'AI generativa e consente un supporto più coerente e affidabile per un'ampia gamma di attività di testing.

2 Ingegneria dei Prompt per un Testing del Software Efficace – 365 minuti

Parole Chiave

criteri di accettazione, test script, test case, condizione di test, dati di test, progettazione dei test, test report

Parole Chiave Specifiche dell'AI Generativa

few-shot prompting, meta prompting, elaborazione del linguaggio naturale, one-shot prompting, prompt, prompt chaining, ingegneria dei prompt, prompt di sistema, prompt utente, zero-shot prompting

Obiettivi di Apprendimento e Obiettivi Pratici per il Capitolo 2:

2.1 Sviluppo Efficace dei Prompt

GenAI-2.1.1	(K2)	Fornire esempi della struttura dei prompt utilizzati nell'AI generativa per il testing del software
HO-2.1.1	(H0)	Osservare diversi prompt forniti per attività di test del software, identificando i componenti di ruolo, contesto, istruzioni, dati di input, vincoli e formato di output in ognuno di essi
GenAI-2.1.2	(K2)	Differenziare le tecniche fondamentali di prompting per il testing del software
HO-2.1.2a	(H0)	Osservare dimostrazioni di prompt chaining, few-shot prompting e meta prompting applicati ad attività di test del software
HO-2.1.2b	(H1)	Identificare quali tecniche di ingegneria dei prompt vengono utilizzate in determinati esempi
GenAI-2.1.3	(K2)	Distinguere tra prompt di sistema e prompt utente

2.2 Applicare le Tecniche di Ingegneria dei Prompt alle Attività di Test del Software

GenAI-2.2.1	(K3)	Applicare l'AI generativa alle attività di analisi dei test
HO-2.2.1a	(H2)	Esercitarsi nel prompting multimodale per generare criteri di accettazione per una user story basata su un wireframe della GUI
HO-2.2.1b	(H2)	Esercitarsi con il prompt chaining e la verifica umana per analizzare progressivamente una determinata user story e affinare i criteri di accettazione
GenAI-2.2.2	(K3)	Applicare l'AI generativa alle attività di progettazione dei test e di implementazione dei test
HO-2.2.2a	(H2)	Esercitarsi nella generazione di test case funzionali dalle user story con l'AI generativa utilizzando prompt chaining, prompt strutturati e meta-prompting
HO-2.2.2b	(H2)	Utilizzare la tecnica del few-shot prompting per generare condizioni di test in stile Gherkin e test case dalle user story

HO-2.2.2c	(H2)	Utilizzare il prompt chaining per prioritizzare i test case all'interno di una determinata test suite, tenendo conto delle loro priorità e dipendenze specifiche
GenAI-2.2.3	(K3)	Applicare l'AI generativa al regression testing automatizzato
HO-2.2.3a	(H2)	Esercitarsi con il few-shot prompting per creare e gestire test script keyword-driven
HO-2.2.3b	(H2)	Esercitarsi nell'ingegneria dei prompt strutturati per l'analisi dei test report
GenAI-2.2.4	(K3)	Applicare l'AI generativa alle attività di controllo e monitoraggio dei test
HO-2.2.4	(H0)	Osservare le metriche di monitoraggio dei test preparate dall'AI generativa a partire dai dati di test
GenAI-2.2.5	(K3)	Selezionare e applicare tecniche di prompting appropriate per un determinato contesto e una determinata attività di test
HO-2.2.5	(H1)	Selezionare e applicare tecniche di prompting appropriate al contesto per una determinata attività di test

2.3 Valutare i Risultati dell'AI Generativa e Ottimizzare i Prompt per le Attività di Test del Software

GenAI-2.3.1	(K2)	Comprendere le metriche per valutare i risultati dell'AI generativa su attività di test
HO-2.3.1	(H0)	Osservare come le metriche possono essere utilizzate per valutare il risultato dell'AI generativa su un'attività di test
GenAI-2.3.2	(K2)	Fornire esempi di tecniche per valutare e ottimizzare iterativamente i prompt
HO-2.3.2	(H1)	Valutare e ottimizzare un prompt per una determinata attività di test

2.1 Sviluppo Efficace dei Prompt

Una progettazione efficace dei prompt migliora la possibilità che gli strumenti GenAI eseguano le attività di test del software in modo accurato ed efficiente e che i tester ottengano risultati utili dai LLM. Un prompt strutturato include diversi componenti (vedere la sezione 2.1.1). Ognuno di questi componenti contribuisce alla chiarezza e alla precisione di un prompt che comunica in modo efficace i requisiti e le aspettative ai LLM.

Diverse tecniche di ingegneria dei prompt migliorano l'efficacia dei prompt nel testing del software. Tecniche quali il prompt chaining, il few-shot prompting e il meta prompting aiutano ad affrontare sfide di testing complesse (vedere la sezione 2.1.2).

La combinazione di prompt strutturati (vedere la sezione 2.1.1) con tecniche di prompting fondamentali ha lo scopo di ottenere buoni risultati quando si interroga un LLM per attività di testing del software (vedere la sezione 2.1.3).

2.1.1 Struttura dei prompt per l'AI generativa nel testing del software

Un prompt strutturato per il testing del software include tipicamente sei componenti:

- **Ruolo:** il ruolo definisce la prospettiva o la persona che il modello di GenAI dovrebbe assumere quando genera una risposta. Specificare il ruolo aiuta il LLM a determinare le proprie responsabilità e ad adottare un tono o un approccio appropriato, ad esempio agendo come tester, test manager o test automation engineer.
- **Contesto:** il contesto fornisce le informazioni di base necessarie al modello di GenAI per determinare le condizioni di test. Ciò include dettagli sull'oggetto di test, sulla funzionalità specifica da testare e su qualsiasi informazione contestuale rilevante.
- **Istruzioni:** le istruzioni sono direttive fornite alla GenAI che delineano l'attività specifica da svolgere. Istruzioni chiare, imperative e concise includono una descrizione dell'attività e tutti i requisiti rilevanti per l'attività.
- **Dati di input:** i dati di input includono tutte le informazioni necessarie per eseguire l'attività, come user story, criteri di accettazione, screenshot, codice, test case esistenti o esempi di output. Fornire dati di input dettagliati e strutturati aiuta il LLM a generare risultati più accurati e sensibili al contesto.
- **Vincoli:** i vincoli delineano eventuali restrizioni o considerazioni speciali a cui il LLM dovrebbe attenersi. I vincoli aiutano a specificare come le istruzioni dovrebbero essere applicate ai dati di input.
- **Formato di output:** le specifiche di output indicano il formato, la struttura o le caratteristiche attese della risposta. Questi indicatori aiutano a definire l'output del LLM.

Questi componenti costituiscono la struttura di base del prompt. Questa struttura dovrebbe essere combinata con l'implementazione di tecniche di prompting (vedere la sezione 2.1.2), a seconda dell'attività da svolgere e del LLM da utilizzare.

Obiettivo Pratico HO-2.1.1 (H0): Osservare e analizzare i componenti del prompt

In una dimostrazione, vengono sperimentati diversi prompt strutturati su un chatbot AI, ognuno adattato a specifiche attività di testing del software. Questi prompt seguono un formato strutturato composto da sei componenti chiave: ruolo, contesto, istruzioni, dati di input, vincoli e formato di output. La dimostrazione mira a facilitare l'osservazione e l'analisi di questi prompt strutturati, evidenziando come ogni componente contribuisca a fornire approfondimenti accurati, rilevanti e utilizzabili a un LLM utilizzato per un'attività di testing del software.

2.1.2 Tecniche di prompting fondamentali per il testing del software

Negli ultimi anni sono state proposte molte tecniche di prompting per i LLM per diversi use case della GenAI (Schulhoff 2024). Tra queste, tre tecniche di prompting fondamentali sono comunemente utilizzate per le attività di test con GenAI in combinazione con la struttura del prompt a 6 componenti descritta in precedenza (vedere la sezione 2.1.1): prompt chaining, few-shot prompting e meta prompting.

- Il prompt chaining consiste nel suddividere un'attività in una serie di passaggi intermedi (prompt multipli). Il risultato di ogni passaggio viene controllato e perfezionato manualmente o automaticamente prima di procedere al passaggio successivo. Questo approccio garantisce una maggiore accuratezza, poiché ogni risposta informa il prompt successivo. Il prompt chaining è particolarmente utile nei processi di test in cui le attività sono complesse e richiedono la scomposizione in sottoattività e il controllo sistematico degli output intermedi del LLM. Consente inoltre interazioni dinamiche nei processi di test.
- Il few-shot prompting consiste nel fornire al LLM degli esempi nel prompt. Mentre il zero-shot prompting (senza esempi) si basa sulle conoscenze preesistenti del modello per generare una risposta, il one-shot prompting fornisce un esempio per dimostrare il risultato atteso per un dato input. I prompt few-shot contengono più di un esempio (in genere alcuni) per consolidare ulteriormente il comportamento di risposta desiderato del modello.

Questa tecnica aiuta a guidare il modello fornendo un riferimento chiaro e garantendo che i risultati siano coerenti e in linea con le aspettative. Il few-shot prompting è particolarmente efficace per attività in cui gli esempi possono illustrare il comportamento richiesto, consentendo al modello di generalizzare in modo efficace e produrre risultati affidabili.

- Il meta prompting utilizza la capacità dell'AI di generare o perfezionare i propri prompt. In un ciclo iterativo, il LLM può generare prompt che possono essere valutati e perfezionati dal tester. Questo approccio ottimizza la qualità dei prompt sfruttando la conoscenza del LLM riguardo i prompt ottimizzati. Il meta prompting è particolarmente vantaggioso quando l'efficienza e l'ottimizzazione dei prompt sono fondamentali, poiché riduce l'effort manuale necessario per progettare prompt efficaci. Un altro vantaggio del meta prompting è che se il tester non è sicuro di come creare un prompt efficace, può collaborare con il LLM per co-crearlo. Ciò riflette una forma di accoppiamento con lo strumento GenAI in cui il tester e l'AI lavorano insieme in modo interattivo per raggiungere un obiettivo condiviso. Questo concetto di accoppiamento evidenzia un nuovo modo di collaborare con gli strumenti di AI, migliorando sia la produttività che l'apprendimento non solo nell'ingegneria dei prompt, ma anche nel pair programming e nel pair testing.

Queste tecniche di prompting possono essere utilizzate efficacemente in combinazione per migliorare i risultati del LLM (vedere la sezione 2.2.5).

Obiettivo Pratico HO-2.1.2a (H0): Osservare e discutere il prompt chaining , il few-shot prompting e il meta prompting in attività di test del software

I partecipanti sperimenteranno il prompt chaining, il few-shot prompting e il meta prompting su un chatbot AI, ognuno dei quali applicato a specifiche attività di test del software. La dimostrazione mira a esplorare e discutere queste tecniche di prompting nel contesto del testing del software, sottolineando come ogni tecnica contribuisca all'accuratezza e alla completezza degli output del LLM.

Obiettivo Pratico HO-2.1.2b (H1): Identificare le tecniche di ingegneria dei prompt in determinati esempi

I partecipanti leggeranno una serie di esempi di prompt relativi al testing del software per identificare le tecniche fondamentali di prompting applicate. L'attenzione sarà focalizzata sul riconoscimento di tecniche quali il prompt chaining, il one-shot prompting e il meta-prompting, evidenziandone le caratteristiche distintive e le applicazioni pratiche.

Questa attività mira ad approfondire la comprensione dei partecipanti su come le diverse tecniche di prompting migliorano l'uso efficace della GenAI nel testing del software.

2.1.3 Prompt di sistema e prompt utente

I prompt di sistema e i prompt utente hanno scopi diversi nelle interazioni con i LLM, ognuno dei quali con un ruolo distinto nel plasmare la conversazione. Il prompt di sistema in genere è definito dallo sviluppatore o dal tester, per guidare il comportamento complessivo del LLM, e non è visibile o modificabile dall'utente del chatbot nella maggior parte delle interfacce.

Un prompt di sistema funge da set di comandi predefiniti che definisce il comportamento, la personalità e i parametri operativi del LLM. I parametri operativi determinano il modo in cui il LLM risponde, ad esempio utilizzando un tono formale, mantenendo le risposte concise, rispettando le regole specifiche del dominio o evitando determinati comportamenti. Il prompt di sistema imposta le regole per l'intera conversazione. Può contenere parti di un prompt strutturato come il ruolo, il contesto e i vincoli.

Il prompt di sistema rimane costante durante tutta la sessione di interazione e stabilisce il framework fondamentale per il modo in cui il LLM dovrebbe rispondere. Ad esempio, un prompt di sistema potrebbe dire: "Sei un assistente professionale per il testing del software. Rispondi sempre in modo chiaro, usa un linguaggio formale e focalizzati sulle pratiche allineate a ISTQB. Evita speculazioni e cita i principi del testing quando rilevante".

Il prompt utente, invece, rappresenta l'input o la domanda effettiva dell'utente del chatbot. Cambia ad ogni interazione e può includere istruzioni, domande o attività specifiche che l'utente del chatbot desidera che il LLM affronti. A differenza del prompt di sistema, i prompt utente sono direttamente visibili e costituiscono il contesto immediato per ogni risposta.

Ad esempio, un prompt utente potrebbe essere: "Elenca le differenze principali tra il testing black-box e il testing white-box con degli esempi".

L'utilizzo tipico prevede l'impostazione del prompt di sistema una volta all'inizio della conversazione, quindi l'invio di prompt utente successivi per ogni interazione. Il LLM genera risposte considerando sia il prompt di sistema immutabile sia il prompt utente attuale insieme. Per un'implementazione efficace, i prompt di sistema dovrebbero essere chiari e specifici riguardo al ruolo del LLM e ai possibili vincoli. Possono anche contenere il contesto e istruzioni generali, ad esempio riguardo all'output previsto.

I prompt utente devono essere mirati e ben strutturati, includendo istruzioni esplicite, nonché ulteriori informazioni contestuali rilevanti e istruzioni sul formato dell'output.

2.2 Applicare le Tecniche di Ingegneria dei Prompt alle Attività di Test del Software

Applicare le tecniche di ingegneria dei prompt al testing del software consente alla GenAI di supportare attività di test quali l'analisi dei test, la progettazione dei test, la test automation, la prioritizzazione dei test case, il rilevamento dei difetti, l'analisi della copertura e il monitoraggio e il controllo dei test. Utilizzando e combinando tecniche quali il prompt chaining, il few-shot prompting e il meta prompting, i team possono adattare i prompt AI agli obiettivi specifici dei test, rendendo gli output più precisi, rilevanti ed efficaci. Un input di alta qualità è fondamentale per ottenere risultati GenAI significativi.

2.2.1 Analisi dei test con l'AI generativa

La GenAI può supportare le attività di analisi dei test generando e prioritizzando le condizioni di test, identificando i difetti nella base di test e fornendo analisi di copertura. I dati di input includono requisiti, user story, specifiche tecniche, wireframe della GUI e altre informazioni rilevanti. L'output consiste in prodotti di lavoro tipici dell'analisi dei test, come condizioni di test prioritizzate (ad esempio, criteri di accettazione).

Di seguito sono riportate alcune attività tipiche di analisi dei test che possono essere supportate dalla GenAI:

- **Identificare potenziali difetti nella base di test:** la GenAI può aiutare ad analizzare la base di test per individuare incongruenze, ambiguità o informazioni incomplete che potrebbero portare a difetti. Confrontando modelli di requisiti simili o applicando le conoscenze acquisite da precedenti defect report, il LLM può segnalare potenziali anomalie e suggerire miglioramenti.
- **Generare condizioni di test basate sulla base di test,** ad esempio su requisiti/user story: i LLM possono analizzare i requisiti e le user story per generare condizioni di test. Utilizzando l'elaborazione del linguaggio naturale, possono interpretare il significato dei requisiti e suddividerli in istruzioni misurabili e testabili. Ciò può aiutare a tradurre i requisiti in condizioni di test specifiche.
- **Prioritizzazione delle condizioni di test in base al livello di rischio:** con informazioni sulla probabilità del rischio e sull'impatto del rischio di failure per ciascuna condizione di test, un LLM può aiutare a prioritizzare l'effort di test. Considerando aspetti quali la conformità normativa, le funzionalità rivolte agli utenti (ad esempio, la funzionalità di login o l'elaborazione dei pagamenti) e i dati storici sui difetti, il LLM può raccomandare livelli di priorità.

- **Supportare l'analisi della copertura:** mappando i requisiti e le user story alle condizioni di test, un LLM può eseguire un'analisi della copertura per determinare se tutti gli aspetti della base di test sono coperti. Ciò è particolarmente utile per progetti con requisiti complessi, in cui le lacune nella copertura possono portare a difetti che sfuggono.
- **Suggerire tecniche di test:** la GenAI può suggerire tecniche di test rilevanti (ad esempio, analisi ai valori limite, partizionamento di equivalenza) in base al tipo di requisito o alla user story sottoposta a test. Ciò può aiutare i tester ad applicare le tecniche di test più efficaci per condizioni di test specifiche.

La qualità e la pertinenza degli input forniti al LLM in relazione all'attività da completare influiscono direttamente sull'accuratezza e sulla precisione dell'output generato dal LLM.

Obiettivo Pratico 2.2.1a (H2): Esercitarsi nella creazione di prompt multimodali strutturati per generare criteri di accettazione per una user story basata su un wireframe della GUI

Questo è un esercizio per fare pratica nella scrittura di prompt strutturati utilizzando input multimodali (testo e immagini). L'obiettivo è di generare criteri di accettazione di alta qualità (cioè ben formati, chiari e completi) da una user story e da un wireframe della GUI. È possibile aggiungere altri elementi di testo per fornire contesto, come vincoli sui campi di input o regole di business da applicare all'elaborazione dei dati.

I risultati ottenuti dal LLM vengono confrontati per valutare l'impatto delle diverse formulazioni del prompt strutturato (ruolo, contesto, istruzioni, dati di input testuali e di immagini, vincoli e formato di output) per un'attività di analisi dei test.

Questo esercizio fornisce un'esperienza pratica sull'importanza della strutturazione del prompt, sul contributo di istruzioni precise e sull'importanza dei dati contestuali sia testuali che di immagini per ottenere risultati accurati e rilevanti dal LLM.

Obiettivo Pratico 2.2.1b (H2): Esercitarsi nel prompt chaining e nella verifica umana per analizzare progressivamente una determinata user story e affinare i criteri di accettazione

Questo è un esercizio per fare pratica nel prompt chaining per analizzare una determinata user story e affinare i criteri di accettazione, prima identificando le ambiguità, poi valutando la testabilità e infine valutando la completezza. Questo esercizio incoraggia un approccio step-by-step, affinando l'analisi ad ogni passo per garantire che i criteri di accettazione siano ben formati e attuabili per raggiungere gli obiettivi di test. Ad ogni passo, i risultati forniti dal LLM vengono verificati manualmente e corretti, se necessario, regolando l'output o attraverso un processo di prompt chaining con il LLM. In questo modo, la fase successiva utilizza un risultato pulito della fase precedente per affrontare un altro aspetto del miglioramento dei criteri di accettazione.

Questo esercizio fornisce un'esperienza pratica dei vantaggi derivanti dalla suddivisione di un'attività complessa in sottoattività, con la verifica umana dei risultati di ogni fase.

2.2.2 Progettazione dei test e implementazione dei test con l'AI generativa

Come descritto nel [ISTQB_CTFL_SYL], la progettazione dei test comporta l'elaborazione e l'affinamento delle condizioni di test, che vengono poi tradotte in test case e altro testware. L'implementazione dei test comporta la creazione o l'acquisizione del testware necessario per eseguire i test.

Sia i test manuali che i test script automatizzati possono essere creati, prioritizzati e organizzati all'interno di una schedulazione dell'esecuzione dei test con il supporto della GenAI. La GenAI può supportare in modo significativo questo ampio gruppo di attività di test assistendo nella creazione e nella valutazione di vario testware, inclusi test case, dati di test, test script e ambienti di test.

Di seguito sono riportate alcune attività tipiche di progettazione dei test e di implementazione dei test che possono essere supportate dalla GenAI:

- **Generazione di test case:** l'elaborazione del linguaggio naturale consente alla GenAI di creare draft di test case basati su requisiti funzionali e non funzionali. Quando vengono richieste informazioni adeguate, un LLM può suggerire precondizioni e input di test, risultati attesi e criteri di copertura, producendo test case che soddisfano diversi obiettivi di test, dalla verifica funzionale di base al testing end-to-end complesso.
- **Sintesi dei dati di test:** la GenAI può creare dati di test sintetici rappresentativi e rispettosi della privacy dei dati che assomigliano ai dati di produzione, coprendo situazioni estreme e condizioni di test diverse. Questi dati di test sintetici possono essere utilizzati per il testing funzionale e non-funzionale. I dati di test generati dall'AI possono essere adattati ai requisiti dell'applicazione, simulando scenari realistici senza esporre informazioni sensibili.
- **Generazione di test script automatizzati:** la GenAI può generare procedure di test manuali e test script automatizzati da test case strutturati, interpretando i passi del test e traducendoli in codice compatibile con vari framework di test automation. Questi test script possono essere aggiornati o estesi in base a nuovi requisiti.
- **Schedulazione dell'esecuzione dei test e prioritizzazione:** la GenAI può analizzare i test case e le loro interdipendenze, ottimizzando la schedulazione dell'esecuzione dei test in base alla priorità, ai rischi associati, alla disponibilità delle risorse e agli obiettivi di test.

Obiettivo Pratico 2.2.2a (H2): Esercitarsi nella generazione di test case funzionali da user story con l'AI utilizzando il prompt chaining, i prompt strutturati e il meta prompting

Questo esercizio si focalizza sullo sviluppo di test case funzionali da user story con la GenAI, utilizzando tecniche di prompt chaining, prompt strutturati e meta prompting per garantire una copertura approfondita. Il primo passo è quello di creare un prompt che istruisca l'AI a generare test case funzionali basati su determinati criteri di accettazione seguendo un formato di output specifico. Il secondo passo consiste nel verificare la completezza dei test case generati. Qui, il prompt verifica che ogni criterio di accettazione sia coperto facendo generare all'AI una tabella che riassume la copertura. Infine, il terzo passo consiste nel creare un meta-prompt per aiutare nella creazione di procedure di test end-to-end. Questo meta-prompt aiuta a ottimizzare il prompt per generare test end-to-end completi, incoraggiando miglioramenti iterativi per massimizzare l'efficacia.

Questo esercizio migliora la comprensione dell'utilizzo dei LLM per la generazione di test case, la validazione della copertura e il testing end-to-end.

Obiettivo Pratico 2.2.2b (H2): Utilizzare la tecnica del few-shot prompting per generare condizioni di test e test case in stile Gherkin dalle user story

Questo esercizio riguarda l'uso del few-shot prompting per generare test case in stile Gherkin da determinate user story. Partendo da una review di esempi predefiniti e della sintassi Gherkin, il passo 1 consiste nel selezionare n esempi da includere nel prompt, ognuno con una user story, condizioni di test e test case attesi in stile given-when-then per modellare il risultato atteso. Questo prompt viene quindi applicato a una nuova user story, generando scenari Gherkin che riflettono le condizioni di test originali. Se i risultati sono imprecisi, il prompt o gli esempi dovrebbero essere ottimizzati.

Questo esercizio aiuta ad acquisire esperienza nell'applicazione delle tecniche di few-shot prompting alle attività realistiche di progettazione dei test e di implementazione dei test.

Obiettivo Pratico 2.2.2c (H2): Utilizzare il prompt chaining per prioritizzare i test case all'interno di una determinata test suite, tenendo conto delle loro priorità e dipendenze specifiche

Questo esercizio si focalizza sull'utilizzo della GenAI per migliorare la prioritizzazione dei test case all'interno di una determinata test suite con l'analisi del rischio associata e le dipendenze tra i test case. La sessione inizia con una breve panoramica dei diversi approcci di test, come quello risk-based, coverage-based e requirements-based, e una review della test suite in questione. I partecipanti saranno quindi coinvolti nella creazione di prompt per generare piani di prioritizzazione attuabili per varie strategie di prioritizzazione dei test. I risultati del LLM in base al prompt e ai dati di input forniti dovrebbero essere verificati manualmente per rilevare eventuali errori nel ragionamento del LLM.

L'obiettivo di questo esercizio è sperimentare con la GenAI su attività di test che richiedono capacità di ragionamento multicriteriale (in questo caso, i diversi rischi e dipendenze da considerare per la prioritizzazione dei test case).

2.2.3 Regression testing automatizzato con l'AI generativa

Man mano che ogni nuova iterazione o release viene completata, il numero di regression test case da eseguire spesso aumenta, rendendoli candidati ideali per l'automazione, in particolare nelle pipeline di Continuous Integration / Continuous Delivery (CI/CD) a causa dell'alta frequenza di esecuzione dei test. La GenAI può semplificare questo processo assistendo nella creazione, manutenzione e ottimizzazione di regression test suite automatizzate. Adattandosi dinamicamente alle modifiche della base di codice ed eseguendo l'analisi degli impatti, la GenAI è in grado di identificare quali aree del software sono più suscettibili ad essere influenzate dalle recenti modifiche, focalizzando l'effort dei regression test dove sono più necessari.

Di seguito sono riportate alcune attività tipiche di regression testing automatizzato e di reporting dei test che possono essere supportate dal prompting della GenAI:

- **Implementazione di test script automatizzati con automazione keyword-driven:** i LLM possono essere utilizzati per implementare test script in base su framework di test automation keyword-driven, in cui parole chiave predefinite rappresentano passi del test comuni. La GenAI

può mappare queste parole chiave per specifici test case, generare test script e assistere i tester e i test automation engineer nel loro lavoro.

- **Analisi degli impatti e ottimizzazione dei test:** la GenAI può essere utilizzata per analizzare le modifiche al codice al fine di identificare le aree ad alto rischio, consentendo così di eseguire un regression testing mirato dove è più necessario.
- **Test auto-riparativi e adattivi:** la GenAI può essere utilizzata per regolare automaticamente i test script per gestire piccole modifiche alla UI o alle API, evitando inutili failure dovuti a piccole modifiche e garantendo che le test suite rimangano stabili nel tempo.
- **Reporting dei test automatizzato e approfondimenti:** la GenAI consente la generazione di test report dettagliati e disponibili tempestivamente con metriche di successo, failure e approfondimenti chiave, fornendo agli stakeholder dashboard che evidenziano le tendenze del testing e offrono approfondimenti predittivi sui potenziali punti di failure.
- **Reporting dei difetti e analisi delle root cause migliorati:** la GenAI può facilitare la compilazione automatica di defect report completi con test log, screenshot e dati dell'ambiente di test.

Queste attività possono essere applicate a una varietà di regression test, inclusi i regression test funzionali e non funzionali. Tuttavia, i tester devono essere consapevoli che la GenAI può commettere errori. L'output generato deve quindi essere controllato attentamente, a seconda del rischio associato (vedere il Capitolo 3).

Inoltre, la GenAI può assistere i regression test automatizzati end-to-end basati su GUI e API, ognuno dei quali con le proprie sfide e soluzioni distintive. I test della GUI diventano spesso instabili a causa di modifiche ricorrenti alla user interface. La GenAI può adattare automaticamente i test script per gestire cambiamenti come i locator dinamici e le interazioni modificate, riducendo la necessità di intervento manuale. I regression test delle API affrontano sfide quali la modifica dei formati di richiesta/risposta, degli endpoint e dell'autenticazione. La GenAI è in grado di adattare automaticamente i test script alle specifiche delle API in evoluzione e di generare dati di test diversificati, mantenendo una copertura completa e riducendo la necessità di aggiornamenti manuali.

Obiettivo Pratico 2.2.3a (H2): Esercitarsi nel few-shot prompting per creare e gestire test script keyword-driven

Questo esercizio si focalizza sullo sviluppo e l'automazione di test script per una determinata applicazione web utilizzando un framework di test automation della GUI. L'esercizio è strutturato in due sezioni principali: test automation e debugging dei test script. La prima parte dell'esercizio fornisce una guida alla creazione della documentazione per una libreria di parole chiave, alla generazione di test script iniziali, alla validazione di questi test script da parte dell'AI e all'espansione della copertura con test script aggiuntivi. La seconda parte pone l'accento sul supporto al debugging, utilizzando i prompt di sistema per creare un assistente AI in grado di controllare e correggere i test script.

Questo esercizio combina la test automation tradizionale con la validazione assistita dall'AI, dimostrando come il few-shot prompting possa essere efficacemente utilizzato per creare, mantenere e eseguire il debug di test script keyword-driven.

Obiettivo Pratico 2.2.3b (H2): Esercitarsi nella scrittura di prompt strutturati per l'analisi dei test report nel contesto del regression testing

Questo esercizio illustra un approccio metodico all'analisi dei regression test report, utilizzando prompt strutturati. Il processo inizia con un'analisi dei risultati dei test forniti e un confronto con le specifiche di test. Si prosegue poi con il clustering di difetti simili, la manutenzione di un elenco di anomalie note e un controllo incrociato delle scoperte. Ogni passo è collegato al successivo in un'unica conversazione del LLM.

L'approccio step-by-step dimostra come i prompt strutturati possano essere utilizzati per trasformare i risultati dei regression test e i test log in informazioni fruibili, supportando così un'efficace analisi dei test report nel contesto del regression testing.

2.2.4 Monitoraggio dei test e controllo dei test con l'AI generativa

Le attività di monitoraggio dei test richiedono il recupero di grandi quantità di dati (a volte non strutturati), che sono spesso già disponibili negli strumenti di test management che la GenAI può aiutare ad analizzare e sintetizzare.

La GenAI facilita una serie di attività di monitoraggio dei test e controllo dei test, tra cui:

- **Monitoraggio dei test e analisi delle metriche:** la GenAI può facilitare l'automazione del monitoraggio dei test, nonché l'analisi delle tendenze per prevedere potenziali rischi e avvisare i team di eventuali deviazioni dal piano. Ciò consente ai team di rimanere informati e di intervenire per mantenere gli standard di qualità.
- **Controllo dei test:** la GenAI può aiutare nel controllo dei test fornendo approfondimenti per riprioritizzare i test, adeguare le schedulazioni dei test e riallocare le risorse secondo necessità. Ciò favorisce che il testing rimanga flessibile e focalizzato sulle aree ad alta priorità.
- **Approfondimenti sul completamento dei test e apprendimento continuo:** la GenAI può aiutare generando test completion report, evidenziando i successi e le lezioni apprese. Ciò consente ai team di affinare le strategie di test e migliorare i processi di test futuri.
- **Visualizzazione e reporting delle metriche di test migliorati:** la GenAI può aiutare nella creazione di dashboard dinamiche e riepiloghi in linguaggio naturale, garantendo a tutti gli stakeholder l'accesso alle metriche rilevanti. Questa assistenza fornisce le informazioni necessarie per prendere decisioni rapide e offre una visione chiara dell'avanzamento dei test.

Obiettivo Pratico 2.2.4 (H0): Osservare le metriche di monitoraggio dei test preparate dall'AI partendo dai dati di test

Questa dimostrazione illustra come la GenAI possa assistere i team di test trasformando i dati di test in metriche di monitoraggio dei test utilizzabili, facilitando così un processo decisionale informato. Partendo dai dati di test estratti dagli strumenti di test, un LLM li elabora per generare metriche chiave come l'avanzamento dei test, le tendenze dei difetti o la copertura, evidenziando i potenziali rischi. Queste

metriche generate dall'AI possono poi essere visualizzate su una dashboard e riassunte in linguaggio naturale per una facile comprensione da parte di tutti gli stakeholder.

Questa dimostrazione illustra come la GenAI trasforma i dati di test in approfondimenti pratici, aiutando i team di test a monitorare l'avanzamento dei test, gestire la qualità e adattarsi rapidamente alle modifiche.

2.2.5 Scegliere le tecniche di prompting per il testing del software

La tabella seguente mostra l'idoneità delle tre tecniche di prompting menzionate nella sezione 2.1.2 in base alle caratteristiche dell'attività di test.

Tecnica di prompting	Use case consigliato	Funzionalità principali e applicazioni
Prompt chaining	Attività complesse che richiedono precisione con verifica umana ad ogni passaggio	Scomponere le attività in passaggi più piccoli, utile per l'analisi dei test, la progettazione dei test e la test automation, dove ogni passo del test viene controllato per verificarne la correttezza funzionale.
Few-shot prompting	Attività ripetitive o con un formato di output specifico/vincolato	Fornisce esempi alla GenAI per la generazione ripetitiva con un modello specifico, ad esempio in test case in stile Gherkin (ad esempio scenario-based), testing keyword-driven o reporting dei test con un formato di output specifico.
Meta prompting	Attività flessibili e dinamiche, utili per creare prompt per nuove attività	Descrizione generale dell'obiettivo e dell'attività da eseguire, che guida il LLM nella creazione del prompt. Utile per tutti i tipi di attività complesse come l'analisi dei test report e il rilevamento delle anomalie.

È anche possibile utilizzare più tecniche per un singolo use case. Ad esempio, il meta prompting può essere utilizzato per creare un prompt iniziale. Questo prompt generato può contenere esempi che devono essere adattati e può essere migliorato (few-shot prompting). Infine, può essere utile dividere l'attività in sottoattività più piccole per consentire la validazione dei passaggi intermedi (prompt chaining).

Obiettivo Pratico 2.2.5 (H1): Selezionare tecniche di prompting appropriate al contesto per determinate attività di test

Questo esercizio si focalizza sulla selezione di tecniche di prompting appropriate per diverse attività di test. Ai partecipanti vengono assegnate diverse attività di test con diverse sfide. Per ogni attività di test, i partecipanti dovrebbero valutare la natura dell'attività - se richiede precisione o una struttura ripetitiva - e suggerire la tecnica o le tecniche di prompting più adatta/e al contesto e alle esigenze specifiche dell'attività. Le scelte vengono discusse in gruppo.

Questo esercizio è progettato per approfondire la comprensione di come diverse tecniche di prompting possano essere utilizzate efficacemente nell'effort di test pratici.

2.3 Valutare i Risultati dell'AI Generativa e Ottimizzare i Prompt per le Attività di Test del Software

Valutare le prestazioni della GenAI nel testing del software richiede una serie chiara di metriche per valutare la qualità, la rilevanza e l'efficacia degli output generati (Li 2024). Queste metriche, sia generali che specifiche per le attività, aiutano a ottimizzare il prompting del LLM.

2.3.1 Metriche per valutare i risultati dell'AI generativa sulle attività di test

È possibile utilizzare diverse metriche per valutare la qualità e l'efficienza dei risultati della GenAI su un'attività di test:

Metrica	Descrizione	Esempio
Accuratezza	Misura la correttezza complessiva dell'output generato rispetto a test case scritti da esperti, requisiti o altri standard.	Il grado in cui i test case generati coprono tutti i requisiti specificati.
Precisione	Valuta la correttezza dell'output generato rispetto a un obiettivo specifico.	Il grado in cui i test case generati identificano correttamente le anomalie.
Richiamo	Misura la capacità di un modello di identificare tutte le istanze rilevanti all'interno di un set di dati.	Il grado in cui i test case generati coprono il partizionamento di equivalenza valido e non valido di una classe di dati.
Rilevanza e adeguatezza contestuale	Determina se l'output generato è applicabile e appropriato per un determinato contesto.	Il grado in cui i test case generati sono coerenti con la base di test e integrano i requisiti specifici del dominio.
Diversità	Consente che sia coperta un'ampia gamma di input e scenari, evitando ripetizioni.	Il grado in cui i test case generati coprono vari comportamenti degli utenti e in cui esplorano i casi limite.
Tasso di successo dell'esecuzione	Misura la proporzione di artefatti di test generati che possono essere eseguiti con successo così come sono nell'ambiente di test.	Determina quanti dei test script generati possono essere eseguiti senza errori di sintassi o problemi del formato di output in un ambiente di test.
Efficienza temporale	Valuta il tempo risparmiato rispetto all'effort di test manuali.	Tempo richiesto dall'AI per generare test case rispetto al tempo che impiegherebbe un essere umano per creare manualmente test equivalenti.

Oltre a queste metriche generali, è possibile personalizzare metriche specifiche per le attività per valutare l'efficacia della GenAI nel supportare attività di test specifiche.

Per valutare efficacemente queste metriche, i tester possono eseguire review manuali o automatizzarle, ad esempio confrontando l'output del LLM con un riferimento predefinito. Data la natura non deterministica della GenAI, le metriche devono essere basate su dati statisticamente rilevanti.

Obiettivo Pratico 2.3.1 (H0): Osservare come le metriche possano essere utilizzate per valutare il risultato dell'AI generativa su un'attività di test

Durante una dimostrazione su una determinata attività di test, vengono mostrate metriche adatte all'attività per valutare i risultati della GenAI, nonché la loro applicazione concreta ai risultati ottenuti con un LLM su quell'attività di test.

Questa dimostrazione illustra l'importanza delle metriche di valutazione nel fornire fiducia sui risultati dell'AI generativa per il testing del software.

2.3.2 Tecniche per valutare e ottimizzare iterativamente i prompt

Sulla base delle metriche presentate in precedenza, vengono utilizzate tecniche specifiche per la valutazione e l'ottimizzazione dei prompt al fine di migliorare i risultati dell'AI:

- **Modifica iterativa dei prompt:** si parte da un prompt di base e lo si modifica iterativamente in base ai risultati effettivi, aggiungendo gradualmente più contesto o modificando la formulazione (ad esempio per quanto riguarda la terminologia) per migliorare la specificità e la rilevanza.
- **Testing A/B dei prompt:** creare più versioni dei prompt e valutare quale versione produce risultati migliori sulla base di metriche predefinite. Questo approccio aiuta a determinare quale formulazione o struttura del prompt produce i risultati più accurati e rilevanti.
- **Analisi dell'output:** esaminare l'output generato dall'AI per individuare imprecisioni o incongruenze, ad esempio rispetto alla base di test. Comprendere i tipi di errori e incongruenze può aiutare a ottimizzare i prompt per evitare difetti simili nelle iterazioni future.
- **Integrare il feedback degli utenti:** raccogliere input dai tester sull'utilità e la chiarezza dell'output generato, ad esempio per quanto riguarda il livello di dettaglio dei test generati. Analizzare i loro approfondimenti e utilizzarli per ottimizzare i prompt in modo da soddisfare meglio le esigenze del testing nel mondo reale.
- **Regolare la lunghezza e la specificità dei prompt:** sperimentare diverse lunghezze e livelli di dettaglio dei prompt. A volte l'aggiunta di più contesto può migliorare la qualità della risposta. In altri casi, prompt più brevi possono produrre una migliore generalizzazione.

Utilizzando queste tecniche, i team di test possono organizzare sessioni di valutazione e ottimizzazione dei prompt per garantire il miglioramento continuo dei prompt della GenAI. La condivisione delle pratiche all'interno del team di test o dell'organizzazione di test non solo aiuta a standardizzare le tecniche di prompt e a mantenere una qualità costante, ma promuove anche una cultura dell'apprendimento e del

miglioramento iterativo. Questo approccio collaborativo contribuisce all'evoluzione delle metodologie di test della GenAI consentendo ai team di test di basarsi su approfondimenti collettivi, evitare errori ripetuti e affinare l'uso degli strumenti GenAI in modo più efficace nel tempo, ad esempio condividendo librerie di prompt.

Obiettivo Pratico 2.3.2 (H1): Valutare e ottimizzare un prompt per una determinata attività di test

Questo esercizio si focalizza sull'applicazione delle tecniche di ottimizzazione dei prompt a una determinata attività di test. I partecipanti cominceranno con un prompt iniziale e lo ottimizzeranno iterativamente per migliorare i risultati generati dall'AI. Utilizzeranno tecniche come il testing A/B e la verifica umana per valutare e migliorare la qualità dei prompt. L'obiettivo è di far sperimentare ai partecipanti come l'ottimizzazione iterativa porti a una generazione di test case più efficace e contestualmente rilevante.

Al termine dell'esercizio, i partecipanti avranno eseguito diverse iterazioni di ottimizzazione dei prompt e valutato ciascuna iterazione utilizzando le metriche discusse per migliorare la qualità dell'output dell'AI.

3 Gestire i Rischi dell'AI Generativa nel Testing del Software – 160 minuti

Parole Chiave

sicurezza, vulnerabilità, privacy dei dati

Parole Chiave Specifiche dell'AI Generativa

allucinazione, temperatura, errore di ragionamento, bias, manipolazione del contesto

Obiettivi di Apprendimento e Obiettivi Pratici per il Capitolo 3:

3.1 Allucinazioni, Errori di Ragionamento e Bias

- | | | |
|-------------|------|---|
| GenAI-3.1.1 | (K1) | Richiamare le definizioni di allucinazioni, errori di ragionamento e bias nei sistemi di AI generativa |
| GenAI-3.1.2 | (K3) | Identificare allucinazioni, errori di ragionamento e bias nell'output del LLM |
| HO-3.1.2a | (H1) | Sperimentare con le allucinazioni nel testing con la GenAI |
| HO-3.1.2b | (H1) | Sperimentare con gli errori di ragionamento nel testing con la GenAI |
| GenAI-3.1.3 | (K2) | Riassumere le tecniche di mitigazione per le allucinazioni, gli errori di ragionamento e i bias della GenAI nelle attività di test del software |
| GenAI-3.1.4 | (K1) | Richiamare le tecniche di mitigazione per il comportamento non deterministico dei LLM |

3.2 Rischi della Privacy dei Dati e della Sicurezza dell'AI Generativa nel Testing del Software

- | | | |
|-------------|------|--|
| GenAI-3.2.1 | (K2) | Spiegare i principali rischi della privacy dei dati e della sicurezza associati all'uso dell'AI generativa nel testing del software |
| GenAI-3.2.2 | (K2) | Fornire esempi di privacy dei dati e vulnerabilità nell'uso dell'AI generativa nel testing del software |
| GenAI-3.2.3 | (K2) | Riassumere le strategie di mitigazione per proteggere la privacy dei dati e migliorare la sicurezza nell'AI generativa per il testing del software |
| HO-3.2.3 | (H0) | Riconoscere i rischi della privacy dei dati e della sicurezza in un determinato caso di studio di AI generativa per il testing |

3.3 Consumo Energetico e Impatto Ambientale dell'AI Generativa per il Testing del Software

- | | | |
|-------------|------|--|
| GenAI-3.3.1 | (K2) | Spiegare l'impatto delle caratteristiche delle attività e dell'utilizzo dei modelli sul consumo energetico dell'AI generativa nel testing del software |
| HO-3.3.1 | (H1) | Utilizzare un simulatore per calcolare l'energia e le emissioni di CO ₂ per determinate attività di test con l'AI generativa |

3.4 Normative, Standard e Framework delle Best Practice per l'AI

GenAI-3.4.1 (K1) Richiamare esempi di normative, standard e framework delle best practice dell'AI rilevanti per l'AI generativa nel testing del software

3.1 Allucinazioni, Errori di Ragionamento e Bias

I sistemi di GenAI, in particolare i LLM, sono soggetti a determinati difetti, tra cui allucinazioni, errori di ragionamento e bias. Questi difetti riducono la qualità dell'output della GenAI nelle attività di test, con il risultato che il testware generato non soddisfa le aspettative dei tester. Queste allucinazioni, errori di ragionamento e bias devono essere identificati dai tester nell'output del LLM e devono essere adottate misure per mitigare tali rischi.

Il comportamento non deterministico dei LLM (vedere la sezione 1.1.2) rende difficile correggere questo tipo di difetti; essi possono sembrare corretti per un output del LLM, ma riapparire in un'altra conversazione con lo stesso LLM.

3.1.1 Allucinazioni, errori di ragionamento e bias nell'AI generativa

Le allucinazioni si verificano quando un LLM genera un output che appare fattualmente errato o irrilevante per una determinata attività. Nel testing del software, le allucinazioni possono manifestarsi quando i LLM creano test case fittizi o irrilevanti, generano test script errati o non funzionanti, o suggeriscono test case che verificano criteri di accettazione inesistenti. Ciò può fuorviare i tester e compromettere la validità degli output dei test.

Gli errori di ragionamento si verificano quando i LLM interpretano erroneamente strutture logiche, come relazioni di causa-effetto, logica condizionale o processi di risoluzione dei problemi step-by-step, portando a conclusioni errate. A differenza degli esseri umani, i LLM non hanno un vero ragionamento logico e si basano sul riconoscimento di pattern, il che può portare a una logica errata quando eseguono attività come il ragionamento matematico (Mirzadeh 2024). La pianificazione dei test e la prioritizzazione dei test case sono esempi di attività di test che richiedono un ragionamento logico e in cui i LLM possono commettere errori di ragionamento.

I bias dei LLM (Gallegos 2024) derivano dai dati su cui è stato addestrato il modello. Questi bias possono portare a output che favoriscono determinati tipi di informazioni, approcci o ipotesi. Ad esempio, i LLM addestrati principalmente su dati in lingua inglese potrebbero sottorappresentare le prospettive non inglesi. Nel testing del software, i bias possono influenzare le risposte dei LLM quando, ad esempio, si generano dati di test o si affinano i criteri di accettazione per i test case.

Le allucinazioni, gli errori di ragionamento e i bias nell'output della GenAI derivano dalla natura dei dati di addestramento e dai limiti intrinseci del modello transformer (vedere il Capitolo 1). Riconoscere e affrontare queste sfide aumenta la qualità dei risultati dell'AI generativa nei processi di test.

3.1.2 Identificare allucinazioni, errori di ragionamento e bias nell'output del LLM

L'integrazione efficace dei sistemi di GenAI nel testing del software richiede la capacità di rilevare allucinazioni, errori di ragionamento e bias nell'output del LLM. A seconda del tipo di problema, è possibile applicare diversi approcci di rilevamento. Di seguito sono riportati gli approcci più comuni applicati attraverso la review o una combinazione di review e verifica automatizzata:

Rilevamento delle allucinazioni:

- Verifica incrociata: confrontare l'output generato dall'AI con la documentazione esistente, i requisiti e il comportamento noto del sistema. Gli strumenti automatizzati possono aiutare a incrociare l'output con fonti di dati consolidate per segnalare eventuali discrepanze.
- Consulenza di esperti del dominio: coinvolgere esperti in materia per validare l'accuratezza dei contenuti generati. La loro esperienza è essenziale per cogliere sfumature e approfondimenti che i sistemi automatizzati potrebbero trascurare.
- Controlli della coerenza: verificare che gli output generati siano coerenti tra loro e con le informazioni note. I sistemi automatizzati possono aiutare a identificare modelli e segnalare incongruenze.

Rilevamento degli errori di ragionamento:

- Validazione logica: valutare il flusso logico (ad esempio, la coerenza, l'uniformità e il ragionamento strutturato all'interno del testo generato) dei contenuti generati dall'AI per verificarne la coerenza e la correttezza attraverso cicli di review. Gli strumenti automatizzati possono essere d'aiuto, ma i casi complessi possono richiedere il giudizio umano.
- Testing dell'output: ad esempio, l'esecuzione dei test case o dei test script generati sugli oggetti di test per verificare i risultati dei test. Questo può essere parzialmente o completamente automatizzato, a seconda del tipo di testware generato.

Rilevamento dei bias:

- Revisionare se il testware generato (ad esempio, codice di test o dati di test sintetici), rappresenti in modo equo e accurato la strategia di test definita e la copertura dei requisiti.
- Valutare i bias relativi ai tipi di test, come i test non funzionali sottorappresentati nell'output generato dal LLM.

L'implementazione effettiva di questi metodi di rilevamento dipenderà dal livello di rischio stimato di allucinazioni, errori di ragionamento o bias nell'attività di test eseguita con la GenAI.

Obiettivo Pratico 3.1.2a (H1): Sperimentare con le allucinazioni dell'AI generativa relative a un'attività di test del software

Questo esercizio si focalizza sulla sperimentazione di esempi di allucinazioni della GenAI in relazione all'insieme di conoscenze del testing del software. I partecipanti cercheranno di confrontare almeno due LLM con una situazione in cui i LLM inventano elementi irrilevanti, ad esempio aggiungono criteri non correlati che non esistono nei dati contestuali forniti. Vengono testate variazioni nel prompting per esaminare l'influenza del prompting sulle allucinazioni.

Questo esercizio aumenta la comprensione dell'identificazione delle allucinazioni della GenAI nel testing del software.

Obiettivo Pratico 3.1.2b (H1): Sperimentare con gli errori di ragionamento dell'AI generativa in un'attività di pianificazione dei test

Questo esercizio si focalizza sulla presentazione di un esempio di errore di ragionamento della GenAI. Un esempio di problema da risolvere nell'area della pianificazione dei test, come la stima dell'effort di test e la prioritizzazione dei test case (vedere [ISTQB_CTFL] - Capitolo 5). L'esercizio è progettato con una certa complessità dei dati di input, che richiede capacità di problem-solving ed evidenzia i limiti dei LLM per questo scopo. Il risultato del LLM sarà confrontato con il risultato esatto che dovrebbe essere ottenuto. Saranno provati tre diversi tipi di LLM (LLM, SLM e modello di ragionamento) e saranno utilizzate variazioni del prompt per cercare di migliorare i risultati.

Questo esercizio aumenta la comprensione di come identificare gli errori di ragionamento della GenAI nelle attività di test del software che richiedono capacità logiche di problem-solving.

3.1.3 Tecniche di mitigazione delle allucinazioni, degli errori di ragionamento e dei bias della GenAI nelle attività di test del software

Per ridurre al minimo i risultati indesiderati della GenAI nel testing del software, è possibile adottare diverse strategie volte a ridurre le allucinazioni, gli errori di ragionamento e i bias. È più probabile che questi problemi accadano quando i prompt non sono progettati correttamente (vedere il Capitolo 2) o quando mancano dati di input contestuali rilevanti per una determinata attività di test. Le tecniche chiave per mitigare i rischi associati alle allucinazioni, agli errori di ragionamento e ai bias dell'AI includono:

- Fornire un contesto completo: assicurarsi che il prompt contenga tutte le informazioni rilevanti (vedere la sezione 2.1.1), offrendo un contesto completo per guidare l'AI nella produzione di risultati accurati.
- Dividere i prompt in segmenti gestibili: suddividere i prompt complessi in passaggi più piccoli utilizzando tecniche di prompt chaining (vedere la sezione 2.1.2), verificando sistematicamente ogni output prima di passare al successivo. Questo approccio step-by-step può aiutare a individuare tempestivamente gli errori di ragionamento nel processo di generazione.
- Utilizzare formati di dati chiari e interpretabili: evitare formati che potrebbero essere ambigui o difficili da interpretare per la GenAI. Formati strutturati e semplici aiutano il modello a focalizzarsi sugli aspetti essenziali dell'attività.
- Selezionare il modello di GenAI appropriato per l'attività: utilizzare un LLM specificamente addestrato per l'attività da svolgere (vedere la sezione 5.1.3).
- Confrontare i risultati tra i modelli: quando opportuno, valutare il prompt con diversi LLM e confrontare gli output aiuta a individuare gli errori dell'output e a selezionare i risultati più affidabili.

Il Capitolo 4 introduce due tecniche complementari per migliorare i risultati del LLM: la Retrieval-Augmented Generation e il Fine-Tuning.

3.1.4 Mitigare il comportamento non deterministico dei LLM

Il comportamento non deterministico intrinseco dei LLM (Shuyin 2023) può portare a variazioni nell'output, anche quando viene fornito lo stesso input. Ciò deriva dai processi di campionamento probabilistico utilizzati durante l'inferenza. Di conseguenza, ottenere risultati coerenti e riproducibili quando si utilizzano i LLM può essere difficile, in particolare per output lunghi, il che aumenta il rischio di variabilità.

Sebbene non sia possibile garantire una riproducibilità completa, alcune strategie possono aiutare a ridurre la variabilità:

- Regolare le impostazioni del parametro di temperatura del LLM: abbassare la temperatura durante la generazione della risposta (inferenza) restringe la distribuzione di probabilità, riducendo la casualità e ottenendo output più coerenti. Tuttavia, ciò limiterà anche la creatività e la diversità delle risposte, rendendo gli output più ripetitivi o eccessivamente deterministici.
- Impostare semi (seed) casuali: alcune implementazioni del LLM consentono di impostare un valore del seme per il generatore di numeri casuali (random number generator), garantendo l'utilizzo della stessa sequenza pseudo-casuale (ovvero valori casuali deterministici), il che migliora la riproducibilità.

La riduzione del rischio di allucinazioni e di errori di ragionamento nell'output del LLM comporta la risoluzione di questo comportamento non deterministico, ad esempio automatizzando alcuni aspetti della verifica dell'output per garantire un processo di valutazione strutturato e coerente.

3.2 Rischi della Privacy dei Dati e della Sicurezza dell'AI Generativa nel Testing del Software

La GenAI nel testing introduce rischi relativi alla privacy dei dati e alla sicurezza a causa della gestione di informazioni sensibili e potenziali vulnerabilità nell'infrastruttura di test alimentata da LLM. Una solida protezione dei dati è essenziale per prevenire violazioni, accessi non autorizzati e divulgazione di dati riservati.

3.2.1 Rischi della privacy dei dati e della sicurezza associati all'uso dell'AI generativa

La GenAI è in grado di elaborare grandi quantità di dati che possono contenere informazioni sensibili o personali. Ciò solleva le seguenti preoccupazioni in materia di privacy dei dati:

- Divulgazione involontaria dei dati: i modelli di GenAI possono generare output che rivelano accidentalmente informazioni sensibili.
- Mancanza di controllo sull'utilizzo dei dati: gli strumenti GenAI possono archiviare ed elaborare dati sensibili senza il consenso esplicito o il controllo dell'utente. Ciò può comportare un potenziale uso improprio o un accesso non autorizzato.
- Rischi di conformità: l'utilizzo di strumenti GenAI senza il rispetto delle normative sulla protezione dei dati, come il General Data Protection Regulation (GDPR, Regolamento (UE) 2016/679), potrebbe portare a controversie legali.

Inoltre, durante il testing con la GenAI sorgono rischi specifici della sicurezza, quali:

- L'infrastruttura di test alimentata da LLM può essere vulnerabile ad attacchi alla sicurezza, come violazioni dei dati o accessi non autorizzati.
- Gli attori malintenzionati possono sfruttare le vulnerabilità dei LLM , come gli attacchi manipolativi (vedere la sezione 3.2.2), per alterarne il comportamento o estrarre informazioni sensibili.
- Gli attacker possono introdurre intenzionalmente dati di input dannosi per fuorviare i LLM e comprometterne l'accuratezza o la sicurezza.

3.2.2 Privacy dei dati e vulnerabilità nell'AI generativa per i processi e gli strumenti di test

La tabella seguente fornisce alcuni esempi di vettori di attacco nei processi e negli strumenti di test della GenAI.

Vettore di attacco	Descrizione	Esempio
Manipolazione del contesto	Invio di richieste progettate per estrarre dati di addestramento confidenziali.	Il superamento della finestra di contesto del LLM con prompt lunghi che sovraccaricano la memoria dell'AI potrebbe portare quest'ultima a rivelare frammenti casuali dei propri dati di addestramento ed esporre potenzialmente informazioni sensibili.
Manipolazione delle richieste	Introduzione di dati che interrompono l'output dell'AI.	Immagini che attirano l'AI in un contesto diverso, provocando così allucinazioni, ad esempio sui criteri di accettazione.
Avvelenamento dei dati	Manipolazione dei dati di addestramento.	Fornire valutazioni false durante la valutazione dei risultati di un test report generato dall'AI.
Generazione di codice dannoso	Manipolazione di un LLM per generare backdoor (ad esempio, chiamate a comandi esterni) durante l'uso.	Generazione di codice per aprire un canale di comunicazione con un IP specifico e dannoso.

3.2.3 Strategie di mitigazione per proteggere la privacy dei dati e migliorare la sicurezza nel testing con l'AI generativa

Con la diffusione della GenAI e i rischi intrinseci che essa comporta, stanno emergendo normative e standard per mitigarli (vedere la sezione 3.4.1).

Le normative sulla protezione dei dati come il GDPR non limitano esplicitamente le applicazioni della GenAI, ma forniscono garanzie che possono limitare ciò che è possibile fare, in particolare per quanto riguarda la liceità e le limitazioni alle finalità di raccolta, trattamento e conservazione dei dati.

Per mitigare tali rischi, le organizzazioni dovrebbero implementare solide misure di privacy dei dati, tra cui:

- Minimizzazione dei dati: evitare il trattamento di dati sensibili, salvo nei casi consentiti dalla legge, e utilizzare solo la quantità necessaria di dati non sensibili nel testing con la GenAI per ridurre i rischi della privacy dei dati.
- Anonimizzazione e pseudonimizzazione dei dati: mascherare o sostituire le informazioni sensibili con dati non identificabili.
- Archiviazione e trasmissione sicura dei dati: implementare una crittografia avanzata e controlli di accesso.
- Formazione delle risorse: le organizzazioni dovrebbero stabilire programmi e politiche di formazione chiari per garantire un uso responsabile degli strumenti GenAI, promuovere pratiche etiche e mitigare i potenziali rischi.

È possibile prendere in considerazione ulteriori strategie di mitigazione quando si implementa la GenAI per il testing:

- Review sistematica dell'output generato: la valutazione umana è essenziale per garantire la qualità e l'accuratezza delle attività di test alimentate dalla GenAI.
- Valutazione mediante confronto con un altro LLM: ciò comporta l'utilizzo di diversi LLM su una determinata attività per valutare gli output confrontando le loro risposte.
- Scelta di un ambiente sicuro e operativo: a seconda del livello di confidenzialità richiesto, le organizzazioni possono optare per diverse soluzioni sicure: utilizzare un'offerta commerciale sicura di un fornitore di LLM, operare il LLM in un cloud sicuro o installare il LLM nell'infrastruttura dell'organizzazione.
- Audit di sicurezza e valutazioni di vulnerabilità regolari: identificare e affrontare le debolezze dei sistemi di GenAI.
- Rimanere aggiornati sulle best practice in materia di sicurezza: tenersi aggiornati sulle ultime linee guida e tecnologie di sicurezza.

Le strategie sono spesso complementari tra loro e una loro combinazione è necessaria per garantire la sicurezza dei dati durante l'utilizzo della GenAI. Si consiglia vivamente di coinvolgere senior Security Engineer, consulenti legali, il Chief Technology Officer (CTO) o il Chief Information Security Officer (CISO), se presenti nell'organizzazione.

Obiettivo Pratico 3.2.3 (H0): Riconoscere i rischi della privacy dei dati e della sicurezza in un determinato caso di studio di AI Generativa per il testing

Questa dimostrazione illustra come possono sorgere rischi di privacy dei dati e di sicurezza quando si utilizza la GenAI nel testing del software. I partecipanti esploreranno casi di studio per identificare potenziali minacce, come vulnerabilità del modello, accesso non autorizzato ai dati o uso dannoso degli output generati. Esploreranno strategie di mitigazione, tra cui la gestione sicura dei dati, controlli di accesso robusti e pratiche di monitoraggio dell'AI, riflettendo al contempo sulle implicazioni etiche e pratiche.

Al termine, i partecipanti comprenderanno i principi della privacy dei dati e impareranno a riconoscere e affrontare i rischi di sicurezza nelle condizioni di test della GenAI.

3.3 Consumo Energetico e Impatto Ambientale dell'AI Generativa per il Testing del Software

Studi come (Luccioni 2024a) dimostrano che l'addestramento e l'elaborazione dei modelli di LLM richiedono un uso intensivo di un gran numero di risorse informatiche specializzate. I LLM sono resi disponibili come servizi web-based e il loro utilizzo aumenta il carico sui dispositivi, sulle reti e sui data center, con un conseguente aumento del consumo energetico.

3.3.1 L'impatto dell'uso della GenAI sul consumo energetico e sulle emissioni di CO₂

L'impatto ambientale della GenAI non dovrebbe essere sottovalutato, poiché il consumo energetico aumenta notevolmente con l'aumentare dell'utilizzo. La complessità dell'attività e le risorse computazionali richieste influenzano il consumo energetico. Ad esempio, la generazione di una singola immagine utilizzando un potente modello di AI può consumare tanta energia quanto il ricaricare completamente uno smartphone, mentre la generazione di testo consuma solo una piccola percentuale della carica di uno smartphone (Heikkilä 2023).

Anche se è difficile ottenere dati precisi sull'impatto ambientale della GenAI (Luccioni 2024b), è chiaro che queste operazioni ad alta intensità energetica contribuiscono collettivamente a emissioni significative di CO₂ (Berthelot 2024). Sebbene una singola ricerca o una singola attività di generazione di testo possano sembrare trascurabili, il loro effetto cumulativo tra milioni di utenti in tutto il mondo comporta un notevole impatto ambientale.

L'adozione di best practice, come la limitazione delle interazioni non necessarie con i modelli, è fondamentale per mitigare i rischi ambientali posti dalla GenAI.

Obiettivo Pratico 3.3.1 (H1): Utilizzare un simulatore per calcolare l'energia e le emissioni di CO₂ per determinate attività di test con l'AI generativa

Questo esercizio si focalizza sulla valutazione del consumo energetico e delle emissioni di CO₂ associate a varie attività di AI generativa nell'ambito del testing del software. I partecipanti utilizzeranno simulazioni per calcolare queste metriche ed esaminare come le diverse caratteristiche delle attività e l'utilizzo dei modelli influenzano l'impatto ambientale.

Osservando come diversi fattori influenzano il consumo energetico e le emissioni, i partecipanti comprenderanno i driver del consumo energetico con i LLM.

3.4 Normative, Standard e Framework delle Best Practice per l'AI

La GenAI sta trasformando il testing del software assistendo i tester in una varietà di attività di test (vedere il Capitolo 2). Tuttavia, queste opportunità comportano anche rischi significativi, quali errori di

ragionamento, privacy dei dati, vulnerabilità e impatto ambientale (vedere le sezioni 3.1, 3.2 e 3.3). Per affrontare questi rischi è necessario prendere in considerazione le normative generali, gli standard e i framework delle best practice per l'AI.

3.4.1 Normative, standard e framework dell'AI rilevanti per la GenAI nel testing del software

Di seguito è riportata una panoramica delle linee guida principali rilevanti per l'utilizzo della GenAI nel testing del software:

Nome / Tipo	Descrizione	Applicazione nel testing del software
ISO/IEC 42001:2023 Tecnologia dell'informazione - Intelligenza artificiale - Sistema di gestione Tipo: Standard	Specifica i requisiti per la gestione dei sistemi di AI all'interno di un'organizzazione.	Promuove che la GenAI nel testing aderisca alle pratiche raccomandate, promuovendo la coerenza e l'affidabilità.
ISO/IEC 23053:2022 Framework per i sistemi di intelligenza artificiale (AI) che utilizzano il machine learning Tipo: Standard	Fornisce un framework per i processi del ciclo di vita dell'AI, enfatizzando la safety e la trasparenza.	Fornisce un framework per la qualità dei dati, la trasparenza e la safety quando si utilizza la GenAI per il testing.
Legge dell'UE sull'AI Tipo: Normativa	Stabilisce un framework legale che affronta i rischi dell'AI, classificando le applicazioni in base al livello di rischio. Fonte: (Legge sull'AI 2024)	Impone la conformità in materia di trasparenza, responsabilità e mitigazione dei bias per la GenAI utilizzata nel testing.
Framework per il risk management dell'AI del NIST (USA) Tipo: Framework	Offre linee guida per la gestione dei rischi dell'AI, focalizzandosi sull'equità, sulla trasparenza e sulla sicurezza. Fonte: (NIST AI RMF 1.0)	Supporta l'equità e mitiga i rischi nella GenAI, prevenendo risultati di test distorti (bias).

Poiché le tecnologie di AI e il suo panorama normativo continuano ad evolversi, è fondamentale che le organizzazioni di test si mantengano aggiornate sullo sviluppo di normative, standard, leggi nazionali e framework delle best practice, come quelli riportati in questa tabella.

4 Infrastruttura di Test Alimentata da LLM per il Testing del Software – 110 minuti

Parole Chiave

infrastruttura di test

Parole Chiave Specifiche dell'AI Generativa

fine-tuning, agente alimentato da LLM, Large Language Model Operations, retrieval-augmented generation, database vettoriali

Obiettivi di Apprendimento e Obiettivi Pratici per il Capitolo 4:

4.1 Approcci Architettureali per l'Infrastruttura di Test Alimentata da LLM

- | | | |
|-------------|------|---|
| GenAI-4.1.1 | (K2) | Spiegare i componenti architettureali e i concetti principali dell'infrastruttura di test alimentata da LLM |
| GenAI-4.1.2 | (K2) | Riassumere la Retrieval-Augmented Generation |
| HO-4.1.2 | (H1) | Sperimentare con la Retrieval-Augmented Generation per una determinata attività di test |
| GenAI-4.1.3 | (K2) | Spiegare il ruolo e l'applicazione degli agenti alimentati da LLM nell'automazione dei processi di test |
| HO-4.1.3 | (H0) | Osservare come un agente alimentato da LLM assiste nell'automazione di un'attività di test ripetitiva |

4.2 Fine-Tuning e LLMops: Rendere Operativa l'AI Generativa per il Testing del Software

- | | | |
|-------------|------|--|
| GenAI-4.2.1 | (K2) | Spiegare il fine-tuning dei language model per attività di test specifiche |
| HO-4.2.1 | (H0) | Osservare un esempio di processo di fine-tuning per una determinata attività di test e un determinato language model |
| GenAI-4.2.2 | (K2) | Spiegare LLMops e il suo ruolo nel deployment e nella gestione di LLM per attività di test |

4.1 Approcci Architetture per l'Infrastruttura di Test Alimentata da LLM

I chatbot AI e gli strumenti di test alimentati da LLM sono due tipi di infrastrutture di test che utilizzano i LLM (vedere la sezione 1.2.2).

Oltre all'architettura di base di un'infrastruttura di test alimentata da LLM (vedere la sezione 4.1.1), la Retrieval-Augmented Generation (vedere la sezione 4.1.2) e le architetture di agenti alimentati da LLM (vedere la sezione 4.1.3) estendono la funzionalità e l'utilità dell'utilizzo dei LLM nel testing del software.

4.1.1 Componenti architetture e concetti principali dell'infrastruttura di test alimentata da LLM

Un'infrastruttura di test alimentata da LLM si riferisce a un sistema che integra un LLM nel processo di test per migliorare l'automazione, il ragionamento e il processo decisionale. A differenza di un chatbot AI tradizionale, che si focalizza principalmente sulle interazioni delle conversazioni, uno strumento di test alimentato da LLM è progettato per supportare il testing del software elaborando query relative ai test, analizzando i requisiti, generando test case e valutando gli output.

L'architettura tipica di un'infrastruttura di test alimentata da LLM segue una progettazione multi-componente che facilita l'interazione sicura ed efficiente con il LLM. L'architettura è composta da componenti front-end e back-end, insieme a fonti di dati esterne e un LLM integrato:

- Il front-end funge da user interface dove i tester interagiscono con il sistema inserendo query o comandi.
- Il back-end elabora gli input degli utenti e gestisce funzioni critiche quali l'autenticazione, il recupero dei dati, la preparazione di prompt e l'interazione con il LLM.
- Il LLM, che può essere ospitato come servizio di terze parti (accessibile tramite API) o come modello personalizzato in-house, genera risposte basate su prompt strutturati.

Questa architettura va oltre il tradizionale modello client-server incorporando componenti di elaborazione intelligenti, come LLM e back-end multi-sorgente:

1. Il LLM non è solo un server, ma un componente di elaborazione intelligente che interpreta e ragiona sulla base del testware.
2. A differenza dei chatbot rule-based che seguono risposte scriptate, un'infrastruttura di test alimentata da LLM genera approfondimenti di test in modo dinamico dal contesto, come requisiti, codice o risultati dei test.
3. Il back-end integra più fonti di dati, quali:
 - Database relazionali (per i dati strutturati utilizzati nel testing, come i test case).
 - Database vettoriali (per il recupero semantico di contenuti correlati utilizzando embedding; vedere la sezione 4.1.2).
4. Il back-end migliora l'output grezzo del LLM attraverso la post-elaborazione, garantendo che le risposte siano in linea con le condizioni di test del processo di test prima di presentarle al front-end.

4.1.2 Retrieval-Augmented Generation

La Retrieval-Augmented Generation (RAG) migliora i LLM incorporando ulteriori fonti di dati nel loro processo di generazione delle risposte (Zhao 2024), aumentando così la rilevanza e l'accuratezza dei loro output.

La RAG combina sistemi di recupero con language model per generare risposte sensibili al contesto. Durante la pre-elaborazione, i documenti di grandi dimensioni vengono suddivisi in blocchi più piccoli (ad esempio, 256-512 token) per garantire un recupero mirato e la compatibilità con la finestra di contesto del modello. Ogni blocco viene pulito, elaborato e codificato in un vettore ad alta dimensionalità (embedding) utilizzando modelli pre-addestrati. Questi embedding, che possono essere archiviati in database vettoriali, consentono un recupero efficiente similarity-based in runtime (inferenza). Una query dell'utente viene codificata, i blocchi rilevanti vengono recuperati in base alla similarità semantica e questi blocchi vengono utilizzati come contesto per il language model per generare una risposta fondata.

Una risposta rilevante è essenzialmente un output generato dal language model che è profondamente radicato in informazioni rilevanti, accurate e contestualmente appropriate raccolte durante il processo di recupero. Ciò promuove che la risposta non sia basata solo sull'addestramento preesistente del modello, ma anche arricchita con dati precisi pertinenti al prompt. Questa sinergia tra recupero e generazione migliora l'accuratezza e la rilevanza delle risposte, rendendole più affidabili e informative per l'utente.

Nella fase di elaborazione del prompt dell'utente, un sistema RAG funziona attraverso un processo in due fasi:

1. **Recupero:** data una query dell'utente, il sistema recupera le informazioni rilevanti dai database vettoriali creati in precedenza. Questo recupero si basa in genere sulla somiglianza semantica tra gli embedding del prompt e quelli dei blocchi.
2. **Generazione:** le informazioni recuperate vengono quindi inviate al LLM, che genera una risposta che combina le sue conoscenze esistenti con i dati appena acquisiti, ottenendo un output più accurato e contestualmente appropriato.

La RAG nel testing del software consente all'infrastruttura di test alimentata da LLM di accedere alle fonti di dati aziendali quali database, documentazione e repository per recuperare informazioni contestuali in tempo reale, garantendo che le attività di test quali l'analisi dei test o la progettazione dei test siano allineate con le specifiche più recenti, i requisiti e i dati di test esistenti.

Obiettivo Pratico 4.1.2 (H1): Sperimentare con la Retrieval-Augmented Generation per una determinata attività di test

Questo esercizio pratico si focalizza sull'applicazione delle tecniche di RAG per una determinata attività di test. I partecipanti sperimenteranno con un sistema RAG incorporando documenti e osserveranno come genera risposte più o meno accurate sulla base di informazioni complesse. I partecipanti confronteranno l'output del LLM con e senza RAG sull'attività di test assegnata. Questo esercizio punta a identificare i punti di forza e i limiti del sistema RAG nella gestione di diversi tipi di attività di test.

Esaminando i dati recuperati e i risultati generati, i partecipanti acquisiranno una comprensione approfondita del ruolo della RAG nel miglioramento dei processi di test alimentati da LLM.

4.1.3 Il ruolo degli agenti alimentati da LLM nell'automazione dei processi di test

Gli agenti alimentati da LLM (Wang 2024) sono applicazioni GenAI specializzate alimentate da LLM e progettate per l'elaborazione semi-autonoma o autonoma di attività definite. Fondamentalmente, questi agenti si affidano ai LLM per la comprensione e la generazione del linguaggio naturale, integrata dalla possibilità di elaborare istruzioni, recuperare il contesto e intraprendere azioni intelligenti.

A differenza dei tradizionali chatbot AI che si focalizzano esclusivamente sulle interazioni domanda-risposta, gli agenti alimentati da LLM possono eseguire attività o "agire" richiamando un insieme predefinito di funzioni, comunemente denominate "strumenti". Questa capacità consente loro di interagire con sistemi esterni e di manipolarli, rendendoli altamente versatili nell'esecuzione delle attività. Il grado di autonomia degli agenti alimentati da LLM può variare:

- Gli agenti autonomi operano in modo indipendente, eseguendo attività con un intervento umano minimo utilizzando regole predefinite, apprendimento rinforzato e cicli di feedback adattivi.
- Gli agenti semi-autonomi eseguono attività con una supervisione umana periodica per garantire che l'output soddisfi gli obiettivi definiti dall'utente.

Le architetture multi-agente prevedono un sistema collaborativo in cui diversi agenti, ciascuno con ruoli specializzati, comunicano e si coordinano per risolvere problemi complessi in modo più efficiente rispetto a un singolo agente. Questo effort coordinato tra più agenti di AI è noto come "orchestrazione", dove gli agenti alimentati da LLM possono automatizzare le attività di test emulando il ragionamento umano e il processo decisionale.

Nei moderni strumenti di testing, gli agenti alimentati da LLM vengono spesso presentati agli utenti tramite assistenti AI integrati nel flusso di lavoro del testing. Questi assistenti possono automatizzare le attività di test lungo l'intero ciclo di vita del test, trasformando le user story o i requisiti in artefatti di test. Ciò include l'esecuzione in modo semi-autonomo dell'analisi, della progettazione, dell'implementazione, dell'esecuzione e del reporting dei test. Questo permette un flusso di lavoro di automazione continuo e completo che riduce l'effort manuale e accorcia i cicli di feedback, spostando la test automation da un'esecuzione script-based a un'automazione agent-based e orientata agli obiettivi.

Tuttavia, questi agenti soffrono degli stessi problemi di possibili allucinazioni, errori di ragionamento e bias osservati quando si utilizzano i LLM (vedere la sezione 3.1). Questi agenti possono produrre risultati errati o fuorvianti, che possono indebolire l'affidabilità dei processi di test automatizzati. Questi rischi possono essere mitigati implementando procedure di verifica automatizzate per i risultati degli agenti o utilizzando agenti semi-autonomi per attività critiche.

Obiettivo Pratico 4.1.3 (H0): Osservare come un agente alimentato da LLM assiste nell'automazione di un'attività di test ripetitiva

La dimostrazione si focalizza su un'attività di test eseguita da un agente alimentato da LLM. Verranno mostrati i dati di input passati all'agente, il suo comportamento e i risultati delle sue azioni per illustrare i vari aspetti dell'integrazione di soluzioni agent-based in un processo di test.

Questa dimostrazione mostra un esempio concreto di un agente alimentato da LLM nel contesto di un'attività di test.

4.2 Fine-Tuning e LLMOps: Rendere Operativa l'AI Generativa per il Testing del Software

Due pratiche principali per rendere operativa l'infrastruttura di test alimentata da LLM per il testing includono il fine-tuning dei LLM e la gestione della pipeline operativa attraverso LLMOps (Mailach 2024).

4.2.1 Fine-tuning dei LLM per le attività di test

Il fine-tuning adatta un Language Model (LM) pre-addestrato, come un LLM o uno Small Language Model (SLM, vedere la sezione 1.1.2), per eseguire attività specifiche o adattarlo a domini particolari (Parthasarathy 2024). Ciò comporta un ulteriore addestramento del modello su un set di dati mirato, consentendogli di apprendere conoscenze e sfumature specifiche del dominio. Grazie al fine-tuning, le prestazioni del modello vengono migliorate per applicazioni specializzate, rendendolo più accurato e rilevante per lo use case previsto.

In pratica, il fine-tuning è adatto per dotare i LLM generici di capacità di ragionamento specializzato rilevanti per un dominio specifico o per adottare un vocabolario unico per quel campo. Il fine-tuning può essere applicato anche a modelli più piccoli, noti come SLM, che richiedono meno risorse. Usando il fine-tuning su un SLM, è possibile ottenere livelli di prestazioni più elevati per attività specifiche senza lo stesso sovraccarico computazionale richiesto dai LLM. Questo confronto evidenzia la flessibilità e l'efficienza nell'utilizzo sia dei LLM che dei SLM in base ai requisiti specifici dell'attività.

Ad esempio, nel testing del software, il fine-tuning può consentire a un LLM o a un SLM di generare test case dalle user story in un formato di output specifico per il contesto dell'organizzazione. Addestrando il modello sulle user story dell'organizzazione e sui corrispondenti test case, il modello si allinea al processo di test e alla terminologia specifici dell'organizzazione.

Il fine-tuning di un modello di GenAI per il testing del software presenta diverse sfide:

- Evitare risultati distorti (bias) o imprecisi garantendo l'uso di set di dati di addestramento di alta qualità e specifici per l'attività.
- Mitigare l'overfitting (il modello diventa troppo specializzato per i dati di addestramento, con un impatto negativo sulle sue prestazioni su dati nuovi e sconosciuti) per mantenere la generalizzazione in diversi scenari.
- Affrontare l'opacità (mancanza di trasparenza nel modo in cui un LLM prende le sue decisioni o produce i suoi output) nel ragionamento del modello, il che complica il debugging e la validazione.
- Gestire le significative risorse computazionali richieste per il processo di fine-tuning (per i LLM).

Obiettivo Pratico 4.2.1 (H0): Osservare un esempio di processo di fine-tuning per una determinata attività di test e un LLM/SLM

Questa dimostrazione mostra i vari passaggi coinvolti nel fine-tuning di un LLM per una determinata attività di test. Si inizia con la selezione di un LLM o SLM appropriato. Successivamente, viene presentato un set di dati che è stato adattato per l'attività di test assegnata. Quindi viene mostrata una

soluzione di esempio per il processo di ottimizzazione (ad esempio un framework di machine learning). Infine, viene inviato un prompt al modello ottimizzato e viene discussa la qualità dell'output generato.

Questa dimostrazione del processo di fine-tuning di un LLM/SLM per un'attività di test mostra diversi aspetti fondamentali di questo processo e affronta in particolare la qualità dei dati di addestramento.

4.2.2 LLMOps durante il deployment e la gestione dei LLM per il testing del software

LLMOps, ovvero Large Language Model Operations, si riferisce all'insieme di pratiche, strumenti e processi progettati per semplificare lo sviluppo, il deployment e la manutenzione dei LLM in ambienti di produzione (Sinha 2024).

L'uso dell'AI generativa nei processi di test di un'organizzazione può essere realizzato in diversi modi, che influenzeranno le decisioni LLMOps da prendere. Di seguito sono riportati tre possibili approcci:

- Utilizzare un chatbot AI: le considerazioni principali per questo approccio includono la gestione dei rischi relativi alla privacy dei dati e alla sicurezza, ottimizzando al contempo i costi. Le organizzazioni potrebbero utilizzare piattaforme LLM-as-a-Service se vengono fornite le garanzie necessarie, oppure implementare un'infrastruttura in-house utilizzando LLM con licenza open source per un maggiore controllo. Una valutazione rigorosa delle garanzie dei fornitori o delle capacità interne è fondamentale per mitigare i rischi relativi alla privacy dei dati e alla sicurezza (vedere la sezione 3.2) e garantire l'efficienza operativa.
- Utilizzare uno strumento di test con funzionalità di AI generativa: questo approccio presenta considerazioni simili a quelle relative ai chatbot AI, quali la privacy dei dati, la sicurezza e i costi operativi. Inoltre, le organizzazioni devono valutare la sicurezza dei dati e le garanzie delle prestazioni offerte dal fornitore dello strumento di test. Questi strumenti di test sono in genere complementari ai processi di test esistenti, che richiedono un'analisi costi-benefici approfondita e una valutazione del rischio.
- Sviluppo in-house di uno strumento di test basato sull'AI generativa: questo approccio enfatizza il controllo completo dei rischi relativi alla privacy dei dati e alla sicurezza, nonché un'attenta pianificazione dell'utilizzo delle risorse dell'AI, quali le risorse computazionali, l'archiviazione dei dati e la formazione del personale. Le organizzazioni devono inoltre stabilire processi strutturati per validare e mantenere gli sviluppi specifici della GenAI. Lo sviluppo di soluzioni in-house richiede competenze nell'implementazione e nel deployment di un'infrastruttura di test alimentata da LLM.

Questi approcci non si escludono a vicenda, poiché un'organizzazione potrebbe utilizzare un chatbot AI per alcune attività e sviluppare strumenti personalizzati per altre. Pertanto, possono essere implementati contemporaneamente a seconda delle specifiche attività di test coinvolte. Inoltre, possono incorporare tecnologie aggiuntive, come RAG e fine-tuning di LLM/SLM, per migliorare l'efficacia e l'adattabilità dei processi di test con la GenAI.

5 Implementare e Integrare l'AI Generativa nelle Organizzazioni di Test – 80 minuti

Parole Chiave

Nessuna

Parole Chiave Specifiche dell'AI Generativa

AI ombra

Obiettivi di Apprendimento e Obiettivi Pratici per il Capitolo 5:

5.1 Roadmap per l'Adozione dell'AI Generativa nel Testing del Software

- | | | |
|-------------|------|---|
| GenAI-5.1.1 | (K1) | Richiamare i rischi dell'AI ombra |
| GenAI-5.1.2 | (K2) | Spiegare gli aspetti chiave da considerare durante la definizione di una strategia di AI generativa per il testing del software |
| GenAI-5.1.3 | (K2) | Riassumere i criteri fondamentali per la selezione di LLM/SLM per attività di test del software in un determinato contesto |
| HO-5.1.3 | (H1) | Stimare i costi ricorrenti dell'utilizzo dell'AI generativa per una determinata attività di test |
| GenAI-5.1.4 | (K1) | Richiamare le fasi principali dell'adozione dell'AI generativa in un'organizzazione di test |

5.2 Gestire le Modifiche quando si Adotta l'AI Generativa per il Testing del Software

- | | | |
|-------------|------|---|
| GenAI-5.2.1 | (K2) | Spiegare le competenze e le aree di conoscenza essenziali richieste ai tester per lavorare in modo efficace con l'AI generativa nei processi di test |
| GenAI-5.2.2 | (K1) | Richiamare le strategie per coltivare le competenze in materia di AI all'interno dei team di test per supportare l'adozione dell'AI generativa nelle attività di test |
| GenAI-5.2.3 | (K1) | Riconoscere come i processi di test e le responsabilità cambiano all'interno di un'organizzazione di test quando si adotta l'AI generativa |

5.1 Roadmap per l'Adozione dell'AI Generativa nel Testing del Software

Una strategia di test con la GenAI deve considerare attentamente alcuni aspetti fondamentali quali gli obiettivi di test da raggiungere, la selezione appropriata del LLM, i problemi relativi ai dati di input utilizzati per il prompting e la conformità agli standard e alle normative in materia di AI. Sulla base di questa strategia, l'organizzazione può stabilire una roadmap e monitorare i progressi nell'integrazione della GenAI nei processi di test.

5.1.1 Rischi dell'AI ombra

L'AI ombra può comportare rischi in materia di sicurezza, conformità e privacy dei dati:

- Debolezze della sicurezza delle informazioni e della privacy dei dati: gli strumenti di AI personali possono non disporre di una sicurezza robusta, con conseguenti potenziali violazioni dei dati.
- Problemi di conformità e normative: l'utilizzo di strumenti di AI non approvati può comportare la non conformità agli standard e alle normative del settore (vedere la sezione 3.4.1), con potenziali conseguenze legali.
- Proprietà intellettuale poco chiara: l'uso di strumenti di AI con accordi di licenza poco chiari può esporre gli utenti del LLM a controversie in materia di proprietà intellettuale, soprattutto se i dati protetti da copyright vengono elaborati senza la debita autorizzazione.

Una strategia e dei passaggi per l'integrazione e il deployment della GenAI possono aiutare le organizzazioni di test a evitare il rischio dell'AI ombra.

5.1.2 Aspetti chiave di una strategia di AI generativa nel testing del software

Per implementare con successo una strategia di GenAI nel testing, le organizzazioni devono considerare attentamente diversi fattori chiave per garantire un'integrazione fluida e risultati ottimali. Questo inizia con la definizione di obiettivi di test misurabili per la GenAI, come l'aumento della produttività dei test, l'accorciamento dei cicli di test e il miglioramento della qualità dei test. La selezione dei giusti LLM è fondamentale (vedere la sezione 5.1.3) e dovrebbe essere allineata con questi obiettivi di test, garantendo al contempo la compatibilità con l'infrastruttura di test esistente e soddisfacendo i requisiti di scalabilità del sistema.

La qualità dei dati gioca un ruolo fondamentale, poiché l'efficacia del testing alimentato da LLM dipende da dati di input accurati e rilevanti, protetti da solide procedure di sicurezza. Mantenere un'elevata qualità dei dati di input è quindi fondamentale per ottenere risultati affidabili e corretti.

È necessario fornire programmi di formazione completi per garantire che i team di test dispongano delle competenze tecniche ed etiche necessarie per utilizzare in modo efficace gli strumenti GenAI. Oltre alla formazione, è necessario raccogliere metriche specifiche per misurare l'efficacia dei risultati della GenAI (vedere la sezione 2.3.1).

Per garantire la conformità agli standard normativi e l'adesione alle linee guida etiche, le organizzazioni dovrebbero stabilire linee guida di processo per l'utilizzo della GenAI, comprese le regole per l'uso di dati sensibili, gli obblighi di trasparenza (ad esempio, cosa è stato generato utilizzando la GenAI) e i quality gate con review del testware generato.

5.1.3 Selezionare i LLM/SLM per attività di test del software

Esiste un'ampia gamma di LLM/SLM, ognuno con diverse capacità funzionali (ad esempio, input multimodale, capacità di ragionamento), funzionalità tecniche (ad esempio, dimensione della finestra di contesto) e tipi di licenza (ad esempio, commerciale o open source). Sebbene siano disponibili molti benchmark per valutare i LLM/SLM per attività quali l'elaborazione del linguaggio naturale, la generazione di codice o l'analisi delle immagini, solo pochi sono specificamente incentrati sulle attività di test del software (Wenhan 2024). Pertanto, la selezione dei LLM/SLM per le attività di test richiede un'attenta valutazione di diversi criteri fondamentali:

- Prestazioni del modello: valutare le prestazioni del modello per le attività di test previste rispetto ai benchmark dell'organizzazione utilizzando metriche come quelle presentate nella sezione 2.3.1.
- Potenziale di fine-tuning: valutare se è possibile e utile utilizzare il fine-tuning del modello linguistico (LLM o SLM) con dati specifici del dominio per migliorare le prestazioni per un determinato use case, aumentando l'accuratezza e la rilevanza in contesti specializzati.
- Costi ricorrenti: considerare i costi ricorrenti dell'utilizzo di LLM/SLM, inclusi i costi di licenza e le spese operative, per garantire che rientrino nel budget dell'organizzazione per le attività di test previste.
- Comunità e supporto: scegliere modelli con un supporto attivo da parte della comunità e una documentazione dettagliata per facilitare l'implementazione e la risoluzione dei problemi.

Valutando attentamente questi criteri, le organizzazioni di test possono selezionare uno o più LLM/SLM che soddisfano le loro esigenze specifiche e i vincoli organizzativi.

Obiettivo Pratico 5.1.3 (H1): Stimare i costi ricorrenti dell'utilizzo dell'AI generativa per una determinata attività di test

Questo esercizio si focalizza sullo stimare i costi ricorrenti dell'utilizzo della GenAI per una specifica attività di test sulla base di varie ipotesi. Queste includono fattori quali il numero di token nei dati di input e output, i prompt utilizzati e la frequenza dell'attività. Verranno esaminati e confrontati i modelli di prezzo di diversi fornitori di LLM/SLM, tra cui almeno una soluzione commerciale e un modello con licenza open source.

Questo esercizio offre l'opportunità di calcolare e sperimentare con i costi ricorrenti della GenAI utilizzando condizioni di test pratiche, aiutando a comprendere le implicazioni finanziarie dei diversi approcci e fornitori.

5.1.4 Fasi durante l'adozione dell'AI generativa nel testing del software

L'adozione della GenAI all'interno di un'organizzazione di test prevede tre fasi principali:

1. Scoperta: la prima fase si focalizza sulla consapevolezza e sullo sviluppo delle capacità. Le attività includono la formazione dei team di test sui concetti della GenAI, fornire l'accesso a

LLM/SLM e la sperimentazione con use case iniziali per familiarizzare i tester con la GenAI e accrescere la loro sicurezza.

2. Avvio e definizione dell'utilizzo: una volta stabilita la comprensione di base, la seconda fase si focalizza sull'identificazione e la prioritizzazione degli use case pratici per la GenAI nel testing del software. Questa fase include la valutazione dell'infrastruttura di test alimentata da LLM, lo sviluppo di competenze e garantire l'allineamento con le esigenze dell'organizzazione (vedere il [ISTQB_CTFL_SYL] sezione 6).
3. Utilizzo e iterazione: in questa fase avanzata, le organizzazioni integrano completamente la GenAI nei propri processi di test. Viene effettuato un monitoraggio continuo dell'avanzamento della GenAI per il testing del software e degli strumenti correlati, nonché la misurazione e la gestione della trasformazione per garantire vantaggi sostenibili e scalabilità.

Queste fasi possono essere eseguite in parallelo per diversi use case. Ad esempio, l'analisi dei test report può essere in una fase più avanzata della roadmap, mentre la test automation è nelle fasi iniziali. È anche importante riconoscere e affrontare le preoccupazioni iniziali, come il timore di una ricollocazione, che possono influire sull'adozione e sul morale del team.

5.2 Gestire le Modifiche quando si Adotta l'AI Generativa per il Testing del Software

L'implementazione con successo della GenAI in un'organizzazione di test richiede un approccio strutturato ai processi di gestione delle modifiche. Gli aspetti fondamentali includono lo sviluppo di competenze GenAI essenziali e l'evoluzione dei ruoli tradizionali nel testing per abbracciare i processi di test abilitati all'AI. La trasformazione coinvolge sia competenze tecniche che aspetti organizzativi.

5.2.1 Competenze e conoscenze essenziali per il testing con l'AI generativa

L'integrazione con successo della GenAI nel testing richiede la padronanza delle tecniche di ingegneria dei prompt, la comprensione delle finestre di contesto del modello e lo sviluppo di metodi di review dei test. I tester devono combinare le competenze del dominio e dei test con le competenze di AI per valutare il testing LLM-driven in attività quali la generazione di test case, l'analisi dei defect report e la generazione di dati di test.

Le competenze principali includono la valutazione delle capacità del LLM, la comprensione delle tecniche di affinamento del prompt e la valutazione del testware generato dall'AI. Le conoscenze essenziali includono la comprensione dei rischi intrinseci della GenAI, insieme alla consapevolezza delle strategie di mitigazione comuni. I tester dovrebbero comprendere le implicazioni della sicurezza dei dati derivanti dalla condivisione di testware con i LLM, implementare un'adeguata sanificazione dei dati (rimuovendo o mascherando informazioni sensibili, personali o riservate) e seguire le pratiche di ingegneria dei prompt che preservano la privacy dei dati. Le considerazioni ambientali includono l'ottimizzazione della selezione dei modelli e dei modelli di utilizzo per ridurre il sovraccarico computazionale, la selezione di modelli di dimensioni adeguate per le attività di test e il bilanciamento dei vantaggi dell'automazione con GenAI con l'impatto sui costi e sul consumo energetico.

5.2.2 Sviluppo di capacità con l'AI generativa nei team di test

Un approccio pratico è essenziale per formare strategicamente i team di test nella GenAI per il testing. Ciò include la pratica con vari LLM/SLM, seguendo percorsi di apprendimento strutturati e sviluppando gradualmente il know-how attraverso la condivisione all'interno dell'organizzazione. L'obiettivo della formazione è lo sviluppo di competenze pratiche attraverso esercizi guidati, il peer learning e la graduale integrazione dell'AI nelle attività di test quotidiane.

I membri del team di test passano dalla padronanza della creazione di prompt basilari all'utilizzo di tecniche più mirate, come i prompt specifici per i test. Un prompt pattern è un modello riutilizzabile per creare prompt efficaci che guidano la GenAI verso output coerenti e affidabili. Le comunità interne di pratica supportano la condivisione continua delle conoscenze, con meeting regolari per evidenziare le applicazioni di successo della GenAI, discutere le sfide e affinare le best practice. Queste comunità promuovono il miglioramento continuo condividendo librerie di prompt pattern e documentando le lezioni apprese dalla GenAI per l'implementazione dei test in tutti i progetti e domini.

5.2.3 Evolvere i processi di test nelle organizzazioni di test abilitate all'AI

L'integrazione della GenAI trasforma i processi di test tradizionali dei tester e dei test manager all'interno delle organizzazioni di test.

I tester si evolvono da specialisti nella progettazione dei test e nell'esecuzione dei test a specialisti in test assistiti dall'AI, combinando la loro esperienza nelle tecniche di test con le competenze necessarie per guidare e verificare il testware generato dall'AI. Le loro attività di test si espandono fino a includere la review dell'output complessivo basato sull'AI, l'ottimizzazione dei prompt e la manutenzione delle librerie di prompt specifiche per i test.

Le responsabilità dei test manager vengono aggiornate per includere lo sviluppo di una strategia di test basata sull'AI, il risk management basato sull'AI e il monitoraggio e il controllo dei processi di test basati sull'AI. I test manager si focalizzano sul bilanciamento tra le capacità umane e dell'AI, sulla definizione di framework di governance dell'AI per gli use case e sulla garanzia che i loro team di test mantengano sia le competenze di testing tradizionale che le competenze in materia di AI. I test manager non solo guideranno i tester umani, ma coordineranno anche gli agenti di test alimentati dalla GenAI, richiedendo nuove competenze di management per supervisionare team ibridi di persone e strumenti GenAI.

6 Riferimenti

Standard

- **ISO/IEC 42001:2023** (2023), Information technology — Artificial intelligence — Management system
- **ISO/IEC 23053:2022** (2022), Framework for Artificial Intelligence (AI) Systems Using Machine Learning (ML)

Documenti ISTQB®

- **[ISTQB_CTFL_SYL]** ISTQB® Foundation Level Syllabus v4.0, 2023

Riferimenti del glossario

- **ISTQB® Glossary** <https://glossary.istqb.org/>

Libri

- Winteringham M. (2024) Software Testing with Generative AI, Manning Publications (5 Mar. 2025), ISBN-13: 978-1633437364, 10 Dec. 2024 - 304 pagine

Articoli

- (Berthelot 2024) Berthelot, Adrien, et al. "Estimating the environmental impact of Generative-AI services using an LCA-based methodology." *Procedia CIRP* 122 (2024): 707-712.
- (Gallegos 2024) Gallegos, Isabel O., et al. "Bias and fairness in large language models: A survey." *Computational Linguistics* (2024): 1-79.
- (Li 2024) Yihao Li, Pan Liu, Haiyang Wang, Jie Chu, W. Eric Wong, Evaluating Large Language Models for Software Testing, *Computer Standards & Interfaces* (2024), doi: <https://doi.org/10.1016/j.csi.2024.103942>
- (Luccioni 2024a) Luccioni, Sasha, Yacine Jernite e Emma Strubell. "Power hungry processing: Watts driving the cost of AI deployment?" *The 2024 ACM Conference on Fairness, Accountability, and Transparency*. 2024.
- (Mailach 2024) Mailach, Alina, et al. "Practitioners' Discussions on Building LLM-based Applications for Production." *arXiv preprint arXiv:2411.08574* (2024).
- (Mirzadeh 2024) Mirzadeh, Iman et al. "GSM-Symbolic: Understanding the Limitations of Mathematical Reasoning in Large Language Models." *ArXiv abs/2410.05229* (2024)
- (NIST AI RMF 1.0) National Institute of Standards and Technology. Artificial Intelligence Risk Management Framework (AI RMF 1.0). NIST AI 100-1, U.S. Department of Commerce, 2023, <https://doi.org/10.6028/NIST.AI.100-1>.

- (Parthasarathy 2024) Parthasarathy, Venkatesh Balavadhani, et al. "The ultimate guide to fine-tuning LLMs from basics to breakthroughs: An exhaustive review of technologies, research, best practices, applied research challenges and opportunities." arXiv preprint arXiv:2408.13296 (2024).
- (Schulhoff 2024) Schulhoff, S., "The Prompt Report: A Systematic Survey of Prompting Techniques", Art. no. arXiv:2406.06608, 2024. doi:10.48550/arXiv.2406.06608.
- (Shuyin 2023) Ouyang, Shuyin, et al. "LLM is Like a Box of Chocolates: the Non-determinism of ChatGPT in Code Generation." arXiv preprint arXiv:2308.02828 (2023).
- (Sinha 2024) Sinha, Megha, Sreekanth Menon e Ram Sagar. "LLMOps: Definitions, Framework and Best Practices." 2024 International Conference on Electrical, Computer and Energy Technologies (ICECET. IEEE, 2024).
- (Wang 2024) Wang, Yanlin, et al. "Agents in Software Engineering: Survey, Landscape, and Vision." arXiv preprint arXiv:2409.09030 (2024).
- (Wenhan 2024) Wang, Wenhan, et al. "TESTEVAL: Benchmarking Large Language Models for Test Case Generation." arXiv preprint arXiv:2406.04531 (2024).
- (Zhao 2024) Zhao, Penghao, et al. "Retrieval-augmented generation for ai-generated content: A survey." arXiv preprint arXiv:2402.19473 (2024).

Pagine web

(AI Act 2024) European Commission. "European Approach to Artificial Intelligence." *Shaping Europe's Digital Future*, European Commission, <https://digital-strategy.ec.europa.eu/en/policies/european-approach-artificial-intelligence>. Ultimo accesso effettuato il 24 novembre 2024.

(Heikkilä 2023) Heikkilä, M. (2023, December 1). Making an image with generative AI uses as much energy as charging your phone. MIT Technology Review. Estratto da <https://www.technologyreview.com/2023/12/01/1084189/making-an-image-with-generative-ai-uses-as-much-energy-as-charging-your-phone/>

(Luccioni 2024b) Luccioni, S. (2024, February 22). Generative AI's environmental costs are soaring. Nature. Estratto da <https://www.nature.com/articles/d41586-024-00478-x>

(Google Dev Glossary 2024) Google Developers. (n.d.). Machine learning glossary: Generativa AI. Estratto il 24 novembre 2024 da <https://developers.google.com/machine-learning/glossary/generative>

(MIT 2024) "Glossary of Terms: Generative AI Basics." *MIT Sloan Teaching & Learning Technologies*, MIT Sloan School of Management, <https://mitsloanedtech.mit.edu/ai/basics/glossary>. Ultimo accesso effettuato il 24 novembre 2024.

I riferimenti precedenti rimandano a informazioni disponibili su Internet e altrove. Sebbene tali riferimenti siano stati verificati al momento della pubblicazione di questo Syllabus, ISTQB® non può essere ritenuta responsabile qualora essi non fossero più disponibili.

7 Appendice A – Obiettivi di Apprendimento/Livello Cognitivo di Conoscenza

Gli obiettivi di apprendimento specifici applicabili a questo Syllabus sono specificati all'inizio di ogni capitolo. Ogni argomento del Syllabus sarà esaminato in base al relativo obiettivo di apprendimento.

Gli obiettivi di apprendimento iniziano con un verbo di azione corrispondente al livello cognitivo di conoscenza, come elencato di seguito.

Livello 1: Ricordare (K1)

Il candidato sarà in grado di ricordare, riconoscere e richiamare un termine o un concetto.

Verbi di azione: Richiamare, riconoscere.

Esempi
Richiamare i concetti della piramide di test.
Riconoscere gli obiettivi tipici del testing.

Livello 2: Comprendere (K2)

Il candidato è in grado di selezionare le ragioni o le spiegazioni delle affermazioni relative all'argomento ed è in grado di riassumere, confrontare, classificare e fornire esempi per il concetto del testing.

Verbi di azione: Classificare, confrontare, differenziare, distinguere, spiegare, fornire esempi, interpretare, riassumere.

Esempi	Note
Classificare gli strumenti di test in base al loro scopo e alle attività di test che supportano.	
Confrontare i diversi livelli di test.	Può essere utilizzato per cercare somiglianze, differenze o entrambe le cose.
Differenziare il testing dal debugging.	Cerca le differenze tra i concetti.
Distinguere tra rischi di progetto e rischi di prodotto.	Consente di classificare separatamente due (o più) concetti.
Spiegare l'impatto del contesto sul processo di test.	
Fornire esempi del perché il testing è necessario.	

Esempi	Note
Ricavare la root cause dei difetti da un determinato profilo di failure.	
Riassumere le attività del processo di review del prodotto del lavoro.	

Livello 3: Applicare (K3)

Il candidato è in grado di eseguire una procedura quando si trova di fronte a un compito che gli è familiare, oppure è in grado di selezionare la procedura corretta e applicarla a un determinato contesto.

Verbi di azione: Applicare, implementare, preparare, utilizzare

Esempi	Note
Applicare l'analisi ai valori limite per ricavare i test case da determinati requisiti.	Dovrebbe riferirsi a una procedura / tecnica / processo ecc.
Implementare metodi di raccolta delle metriche per supportare i requisiti tecnici e di management.	
Preparare test di installabilità per le app mobile.	
Utilizzare la tracciabilità per monitorare l'avanzamento del test, in modo da poter verificare la completezza e la coerenza con gli obiettivi del test, la strategia di test e il test plan.	Potrebbe essere utilizzato in un LO che richiede che il candidato sia in grado di utilizzare una tecnica o una procedura. Simile ad "applicare".

Riferimento

(Per i livelli cognitivi degli obiettivi di apprendimento)

Anderson, L. W. and Krathwohl, D. R. (eds) (2001) A Taxonomy for Learning, Teaching, and Assessing: A Revision of Bloom's Taxonomy of Educational Objectives, Allyn & Bacon

8 Appendice B – Matrice di tracciabilità dei Business Outcome agli Obiettivi di Apprendimento

Questo capitolo elenca la tracciabilità tra i Business Outcome e gli Obiettivi di Apprendimento del Certified Tester Testing with Generative AI. Gli Obiettivi Pratici non sono menzionati in questa tabella in quanto ogni HO è associato ad un singolo LO. La tracciabilità tra un HO e un BO avviene tramite il LO a cui l'HO è associato.

Business Outcome: Certified Tester Testing with Generative AI			BO1	BO2	BO3	BO4	BO5
GenAI-BO1	Comprendere i concetti fondamentali, le capacità e i limiti dell'AI generativa		8				
GenAI-BO2	Sviluppare competenze pratiche nel prompting di large language model per il testing del software			10			
GenAI-BO3	Acquisire approfondimenti sui rischi e sulle mitigazioni dell'utilizzo dell'AI generativa per il testing del software				11		
GenAI-BO4	Acquisire approfondimenti sulle applicazioni delle soluzioni di AI generativa per il testing del software					19	
GenAI-BO5	Contribuire in modo efficace alla definizione e all'implementazione di una strategia e di una roadmap di AI generativa per il testing del software all'interno di un'organizzazione						13

Business Outcome: Certified Tester Testing with Generative AI			BO1	BO2	BO3	BO4	BO5
LO Unico	Obiettivo di Apprendimento	Livello K					
1	Introduzione all'AI Generativa per il Testing del Software						
1.1	Fondamenti e Concetti Chiave dell'AI Generativa						
GenAI-1.1.1	Richiamare i diversi tipi di AI: AI simbolica, machine learning classico, deep learning e AI generativa	K1	X				
GenAI-1.1.2	Spiegare le basi dell'AI generativa e dei large language model	K2	X				
GenAI-1.1.3	Distinguere tra LLM di base, ottimizzati con istruzioni e di ragionamento	K2	X				
GenAI-1.1.4	Riassumere i principi di base dei LLM multimodali e dei modelli linguistico-visivi	K2	X				
1.2	Utilizzare l'AI Generativa nel Testing del Software: Principi Fondamentali						
GenAI-1.2.1	Fornire esempi delle capacità principali dei LLM per le attività di test	K2	X			X	
GenAI-1.2.2	Confrontare i modelli di interazione quando si utilizza la GenAI per il testing del software	K2	X			X	

Business Outcome: Certified Tester Testing with Generative AI			BO1	BO2	BO3	BO4	BO5
2	Ingegneria dei Prompt per un Testing del Software Efficace						
2.1	Sviluppo Efficace dei Prompt						
GenAI-2.1.1	Fornire esempi della struttura dei prompt utilizzati nell'AI generativa per il testing del software	K2		X			
GenAI-2.1.2	Differenziare le tecniche fondamentali di prompting per il testing del software	K2		X			
GenAI-2.1.3	Distinguere tra prompt di sistema e prompt utente	K2		X			
2.2	Applicare le Tecniche di Ingegneria dei Prompt alle Attività di Test del Software						
GenAI-2.2.1	Applicare l'AI generativa alle attività di analisi dei test	K3		X			
GenAI-2.2.2	Applicare l'AI generativa alle attività di progettazione dei test e di implementazione dei test	K3		X			
GenAI-2.2.3	Applicare l'AI generativa al regression testing automatizzato	K3		X			
GenAI-2.2.4	Applicare l'AI generativa alle attività di controllo e monitoraggio dei test	K3		X			

Business Outcome: Certified Tester Testing with Generative AI			BO1	BO2	BO3	BO4	BO5
GenAI-2.2.5	Selezionare e applicare tecniche di prompting appropriate per un determinato contesto e una determinata attività di test	K3		X		X	
2.3	Valutare i Risultati dell'AI Generativa e Ottimizzare i Prompt per le Attività di Test del Software						
GenAI-2.3.1	Comprendere le metriche per valutare i risultati dell'AI generativa su attività di test	K2		X	X	X	
GenAI-2.3.2	Fornire esempi di tecniche per valutare e ottimizzare iterativamente i prompt	K2		X	X	X	
3	Gestire i Rischi dell'AI Generativa nel Testing del Software						
3.1	Allucinazioni, Errori di Ragionamento e Bias						
GenAI-3.1.1	Richiamare le definizioni di allucinazioni, errori di ragionamento e bias nei sistemi di AI generativa	K1	X		X	X	
GenAI-3.1.2	Identificare allucinazioni, errori di ragionamento e bias nell'output del LLM	K3			X	X	
GenAI-3.1.3	Riassumere le tecniche di mitigazione per le allucinazioni, gli errori di ragionamento e i bias della GenAI nelle attività di test del software	K2			X	X	

Business Outcome: Certified Tester Testing with Generative AI			BO1	BO2	BO3	BO4	BO5
GenAI-3.1.4	Richiamare le tecniche di mitigazione per il comportamento non deterministico dei LLM	K1	X		X	X	
3.2	Rischi della Privacy dei Dati e della Sicurezza dell'AI Generativa nel Testing del Software						
GenAI-3.2.1	Spiegare i principali rischi della privacy dei dati e della sicurezza associati all'uso dell'AI generativa nel testing del software	K2			X	X	
GenAI-3.2.2	Fornire esempi di privacy dei dati e vulnerabilità nell'uso dell'AI generativa nel testing del software	K2			X	X	
GenAI-3.2.3	Riassumere le strategie di mitigazione per proteggere la privacy dei dati e migliorare la sicurezza nell'AI generativa per il testing del software	K2			X	X	
3.3	Consumo Energetico e Impatto Ambientale dell'AI Generativa per il Testing del Software						
GenAI-3.3.1	Spiegare l'impatto delle caratteristiche delle attività e dell'utilizzo dei modelli sul consumo energetico dell'AI generativa nel testing del software	K2			X	X	
3.4	Normative, Standard e Framework delle Best Practice per l'AI						
GenAI-3.4.1	Richiamare esempi di normative, standard e framework delle best practice dell'AI rilevanti per l'AI generativa nel testing del software	K1			X	X	X

Business Outcome: Certified Tester Testing with Generative AI			BO1	BO2	BO3	BO4	BO5
4	Infrastruttura di Test Alimentata da LLM per il Testing del Software						
4.1	Approcci Architettureali per l'Infrastruttura di Test Alimentata da LLM						
GenAI-4.1.1	Spiegare i componenti architettureali e i concetti principali dell'infrastruttura di test alimentata da LLM	K2				X	X
GenAI-4.1.2	Riassumere la Retrieval-Augmented Generation	K2				X	X
GenAI-4.1.3	Spiegare il ruolo e l'applicazione degli agenti alimentati da LLM nell'automazione dei processi di test	K2				X	X
4.2	Fine-Tuning e LLMOps: Rendere Operativa l'AI Generativa per il Testing del Software						
GenAI-4.2.1	Spiegare il fine-tuning dei language model per attività di test specifiche	K2				X	X
GenAI-4.2.2	Spiegare LLMOps e il suo ruolo nel deployment e nella gestione di LLM per attività di test	K2				X	X
5	Implementare e Integrare l'AI Generativa nelle Organizzazioni di Test						
5.1	Roadmap per l'Adozione dell'AI Generativa nel Testing del Software						

Business Outcome: Certified Tester Testing with Generative AI			BO1	BO2	BO3	BO4	BO5
GenAI-5.1.1	Richiamare i rischi dell'AI ombra	K1					X
GenAI-5.1.2	Spiegare gli aspetti chiave da considerare durante la definizione di una strategia di AI generativa per il testing del software	K2					X
GenAI-5.1.3	Riassumere i criteri fondamentali per la selezione di LLM/SLM per attività di test del software in un determinato contesto	K2					X
GenAI-5.1.4	Richiamare le fasi principali dell'adozione dell'AI generativa in un'organizzazione di test	K1					X
5.2	Gestire le Modifiche quando si Adotta l'AI Generativa per il Testing del Software						
GenAI-5.2.1	Spiegare le competenze e le aree di conoscenza essenziali richieste ai tester per lavorare in modo efficace con l'AI generativa nei processi di test	K2					X
GenAI-5.2.2	Richiamare le strategie per coltivare le competenze in materia di AI all'interno dei team di test per supportare l'adozione dell'AI generativa nelle attività di test	K1					X
GenAI-5.2.3	Riconoscere come i processi di test e le responsabilità cambiano all'interno di un'organizzazione di test quando si adotta l'AI generativa	K1					X

9 Appendice C – Release Note

Questa versione è la V1.1. Le release note per questa versione possono essere scaricate.

5

10 Appendice D – Termini Specifici dell'AI Generativa

Nome del termine	Definizione
Agente alimentato da LLM	Un'applicazione che integra il ragionamento, il processo decisionale e la memoria del LLM, utilizzando strumenti per eseguire attività.
AI Generativa (GenAI)	Un tipo di sistema di intelligenza artificiale che utilizza modelli di machine learning per generare (nuovi) contenuti intellettuali simili a quelli creati dall'uomo.
AI ombra	L'uso di strumenti o sistemi di GenAI all'interno di un'organizzazione senza approvazione formale o supervisione.
AI simbolica	Un approccio all'AI che utilizza simboli, regole e conoscenze strutturate per modellare il ragionamento all'interno di un sistema.
Allucinazione	Informazioni errate create da un LLM.
Chatbot AI	Un agente conversazionale che utilizza LLM per elaborare le query e generare risposte testuali simili a quelle umane, consentendo una comunicazione interattiva con gli utenti.
Database vettoriale	Un database ottimizzato per l'archiviazione e l'interrogazione di rappresentazioni vettoriali ad alta dimensionalità dei dati.
Deep learning	ML che utilizza reti neurali con più strati.
Elaborazione del linguaggio naturale (ELN o anche natural language processing, NLP)	Elaborazione di dati codificati in linguaggio naturale da parte dei computer per recuperare informazioni e rappresentare conoscenze.
Embedding	Tecnica utilizzata per rappresentare i token come vettori densi in uno spazio continuo, appresa durante l'addestramento per catturare le relazioni semantiche, sintattiche e contestuali.
Few-shot prompting	Una tecnica in cui a un modello vengono forniti alcuni esempi all'interno del prompt per guidarlo nella generazione di risposte appropriate.
Finestra di contesto	L'intervallo di testo, misurato in token, che un language model considera durante la generazione delle risposte, influenzando la rilevanza e la coerenza dei suoi output.
Fine-tuning	Processo di apprendimento supervisionato che utilizza un set di dati di esempi etichettati per aggiornare i pesi del LLM e adattarli ad attività o domini specifici.
Funzionalità	Un attributo misurabile individuale dei dati di input utilizzati per l'addestramento da parte di un algoritmo di ML e per la previsione da parte di un modello di ML.

Generative pre-trained transformer (GPT)	Un tipo di modello di deep learning transformer-based pre-addestrato su grandi quantità di dati di testo per comprendere e generare testi simili a quelli umani.
Ingegneria dei prompt	Il processo di progettazione e ottimizzazione dei prompt di input per guidare i prompt LLM verso la produzione degli output desiderati.
Large language model (LLM)	Un programma per computer che utilizza raccolte molto grandi di dati linguistici per comprendere e produrre testi in modo simile a quello umano.
LLM di base (o foundation LLM)	Modelli di uso generale pre-addestrati su un'ampia gamma di dati di testo, in grado di prevedere la parola successiva sulla base dei modelli linguistici appresi. Sinonimo: Base LLM
LLM di ragionamento (reasoning)	Un LLM che si basa su modelli ottimizzati con istruzioni, ottimizzando la propria capacità di emulare i processi di ragionamento umani.
LLM ottimizzato con istruzioni (instruction-tuned)	Un LLM di base addestrato a seguire le istruzioni, spesso rafforzato da feedback per incoraggiare risposte corrette.
LLMOps	Pratiche e strumenti focalizzati sul deployment, il monitoraggio e la manutenzione di LLM in ambienti di produzione.
Machine learning (ML)	Il processo che utilizza tecniche computazionali per consentire ai sistemi di apprendere dai dati o dall'esperienza (ISO/IEC TR 29119-11).
Meta prompting	La creazione di istruzioni di livello superiore che generano prompt specifici per esplorare o automatizzare le capacità.
Modello linguistico-visivo	Un sistema di GenAI che elabora congiuntamente dati visivi e testuali per eseguire attività collegando e generando contenuti in entrambe le modalità.
Modello multimodale	Modelli di GenAI in grado di elaborare e generare contenuti in diversi tipi di dati, come testo, immagini e audio.
One-shot prompting	Una tecnica di scrittura di prompt in cui il prompt contiene un esempio per guidare la risposta del LLM.
Prompt	Input in linguaggio naturale fornito per suscitare una risposta specifica nell'AI Generativa e nei large language model.
Prompt chaining	Una tecnica di prompting che prevede l'utilizzo dell'output di un prompt come input per un altro, creando una sequenza di prompt.
Prompt di sistema	Un insieme predefinito di istruzioni, solitamente nascosto agli utenti del chatbot, che stabilisce in modo coerente il contesto, il

	tono e i limiti delle risposte di un LLM e ne guida il comportamento durante le interazioni.
Prompt utente	Un'istruzione o una query inserita da un utente in un Large Language Model (LLM) che indirizza la risposta del modello per eseguire attività specifiche o fornire le informazioni desiderate.
Retrieval-augmented generation (RAG)	Una tecnica che combina le capacità di generazione di LLM con un retriever per recuperare dati rilevanti per generare risposte accurate e contestualmente rilevanti.
Small language Model (SLM)	Language Model progettati e addestrati intenzionalmente per essere piccoli, offrendo un equilibrio tra efficienza e comprensione del linguaggio specifico per l'attività.
Temperatura	Un parametro che controlla la casualità o la creatività degli output di un LLM.
Tokenizzazione	Il processo di suddivisione del testo in unità più piccole per l'elaborazione da parte dei language model.
Transformer	Un'architettura di modello di deep learning che utilizza meccanismi di self-attention per catturare dipendenze a lungo raggio nelle sequenze di input.
Zero-shot prompting	Una tecnica di scrittura di prompt in cui il prompt non contiene esempi, ma si affida alle conoscenze preesistenti del modello per generare una risposta.

11 Appendice E – Marchi commerciali

ISTQB® è un marchio registrato di International Software Testing Qualifications Board

12 Indice

Tutti i termini relativi al testing sono definiti nel Glossario ISTQB® (<http://glossary.istqb.org/>).

- 10
- 11
- 12
- | | | | |
|----|---|----|---|
| 13 | AI generativa; 8; 9; 11; 13; 20; 21; 33; 35; 36; | 44 | ingegneria dei prompt; 11; 20; 21; 22; 24; 25; |
| 14 | 42; 49; 50; 56; 61; 65 | 45 | 53 |
| 15 | AI ombra; 50 | 46 | large language model; 13 |
| 16 | AI simbolica; 13; 14; 60 | 47 | LLM; 11; 12; 13; 14; 15; 16; 17; 18; 22; 23; |
| 17 | allucinazione; 35 | 48 | 24; 25; 26; 27; 28; 30; 32; 33; 35; 36; 37; |
| 18 | bias; 11; 35; 36; 37; 38; 47; 62 | 49 | 38; 39; 40; 41; 42; 44; 45; 46; 47; 48; 49; |
| 19 | chatbot; 17; 47; 49 | 50 | 50; 51; 52; 53; 54; 55; 56; 60; 63; 64; 67; |
| 20 | chatbot AI; 11; 13; 23; 24; 45; 49 | 51 | 68; 69 |
| 21 | condizione di test; 20; 25; 31 | 52 | LLM di base; 68 |
| 22 | criteri di accettazione; 14; 20; 22; 25; 26; 27; | 53 | LLM di base (foundation); 13 |
| 23 | 36 | 54 | LLMOps; 12; 44; 47; 49; 56; 64; 68 |
| 24 | database vettoriali; 44 | 55 | machine learning; 13; 14; 60 |
| 25 | dati di test; 14; 18; 20; 21; 27; 29; 30; 31; 36; | 56 | meta prompting; 20; 22; 23; 24; 25; 27; 31 |
| 26 | 45; 46; 53 | 57 | modelli visione-linguaggio; 17 |
| 27 | deep learning; 13; 14; 15; 60; 68 | 58 | modello multimodale; 13 |
| 28 | elaborazione del linguaggio naturale; 20; 25; | 59 | one-shot prompting; 20 |
| 29 | 52 | 60 | privacy dei dati; 11; 27; 35; 39; 40; 41; 42; 49; |
| 30 | embedding; 13; 46 | 61 | 51; 53; 63 |
| 31 | few-shot prompting; 20; 21; 22; 23; 24; 25; | 62 | progettazione dei test; 18; 20; 25; 27; 28; 31; |
| 32 | 28; 29 | 63 | 46; 54; 61 |
| 33 | finestra di contesto; 13; 15; 46; 52 | 64 | prompt; 11; 13; 15; 17; 20; 21; 22; 23; 24; 25; |
| 34 | fine-tuning; 44; 47; 48; 49; 56; 64 | 65 | 26; 27; 28; 33; 34; 38; 45; 46; 53; 54; 67; |
| 35 | funzionalità; 13; 14 | 66 | 68; 69 |
| 36 | GenAI; 1; 3; 8; 9; 10; 11; 12; 13; 14; 15; 17; | 67 | prompt chaining; 20; 21; 22; 23; 24; 25; 26; |
| 37 | 20; 21; 22; 23; 24; 25; 26; 27; 28; 29; 30; | 68 | 27; 28; 38 |
| 38 | 31; 32; 33; 35; 36; 37; 38; 39; 40; 41; 42; | 69 | prompt dell'utente; 14 |
| 39 | 43; 44; 47; 48; 49; 50; 51; 52; 53; 54; 59; | 70 | prompt di sistema; 20; 24; 25 |
| 40 | 60; 61; 62; 63; 64; 65; 67; 68 | 71 | prompt utente; 20; 24; 25; 61 |
| 41 | generative pre-trained transformer; 13; 15 | 72 | RAG; 46; 49; 69 |
| 42 | infrastruttura di test; 39; 44; 45; 46; 47; 49; | 73 | ragionamento; 11; 13; 14; 28; 35; 36; 37; 38; |
| 43 | 51; 53; 64 | 74 | 39; 42; 45; 47; 48; 52; 60; 62; 67 |

75	Retrieval-Augmented Generation; 12; 38; 44;	83	test report; 20; 21; 30; 53
76	45; 46; 64; 69	84	token; 15; 46; 52; 67
77	sicurezza; 11; 35; 39; 40; 41; 43; 49; 51; 53;	85	tokenizzazione; 11; 13; 14; 15; 16; 17; 69
78	63	86	transformer; 13; 15; 17; 36; 68
79	SLM; 15; 48; 49; 50; 52; 53; 54	87	vulnerabilità; 35; 41
80	temperatura; 35; 39	88	zero-shot prompting; 20
81	test automation; 8; 20; 22; 25; 28; 29; 31; 53		
82	test case; 18; 20; 25; 27; 28; 31; 34; 36; 53		
89			