Testeur certifié niveau spécialiste

Tester avec l'IA Générative (CT-GenAI)
Syllabus

Version v1.0

International Software Testing Qualifications Board
Comité Français des Tests Logiciels

Avis de copyright

Historique des modifications

Version	Date	Remarques
V1.0FR	2025/08/03	CT-GenAI 1.0 Version française
v1.0	2025/07/25	CT-GenAI v1.0 Version approuvée par l'Assemblée Générale de l'ISTQB

Table des matières

Remerciements

Ce document a été officiellement publié par l'Assemblée générale de l'ISTQB® le 25/07/2025.

Il a été produit par une équipe de l'International Software Testing Qualifications Board® : Abbas Ahmad (product owner), Gualtiero Bazzana, Alessandro Collino, Olivier Denoo, et Bruno Legeard (responsable technique).

L'équipe remercie Anne Kramer, Jedrzej Kwapinski, Samuel Ouko et Ina Schieferdecker pour leur revue technique, ainsi que l'équipe de révision et les membres pour leurs suggestions et leurs contributions.

Les personnes suivantes ont participé à la révision, aux commentaires et au vote de ce 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 Ágnecz, Horst Pohlmann, Ina Schieferdecker, Ingvar Nordström, Jan Sabak, Jaroslaw Hryszko, Jedrzej Kwapinski, Joanna Kazun, Karol Frühauf, Katalin Balla, Koray Yitmen, Laura Albert, Linda Vreeswijk, Lucjan Stapp, Lukáš Piška, Mario Winter, Marton Siska, 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, Radoslaw 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, Giorgio Pisani, Giancarlo Tomasig, Young jae Choi, Arnika Hryszko, Andrei Brovko, Ilia kulakov, Praveen, Kostas Pashalidis, Ferdinand Gramsamer, A. Berfin Öztaş, Abdullah Gök, Abdurrahman AKIN, Aleyna Zuhal IŞIK, 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 Yalçınkaya, Efe Can Yemez, Elif Cerav, Emine Tekiner, Emre Aman, Emre Can Akgül, Esra Kücük, Gençay GENÇ, Gül Çalışır Açan, Gül Nihal Singil, Güler Gök, Gulhanim Anulur, Hakan GÜVEZ, Haktan Bilgehan Dilber, Halil Ibrahim Tasdemir, Hasan Küçükayar, Hatice Erdoğan, Hatice Kübra Daşdoğan, Hüseyin Sevki ARI, Hyulya Gyuler, Ilknur Nese 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.

La traduction française a été réalisée par un groupe d'experts en tests logiciels : Olivier Denoo, Bruno Legeard et Eric Riou du Cosquer. L'équipe remercie Fabrice Bouquet, Etienne Dufour et Elizabeta Fourneret pour leur relecture de cette traduction.

0 Introduction

0.1 Objectif de ce document

Ce syllabus forme la base du module spécialisé en Tester avec l'IA générative de la qualification internationale en tests de logiciels. L'International Software Testing Qualifications Board (ISTQB®) et le Comité Français des Tests Logiciels (ci-après appelé CFTL) fournissent ce syllabus comme suit :

1. Aux Membres de l'ISTQB, afin de traduire dans leur langue locale et d'accréditer les fournisseurs de formation. Les Membres peuvent adapter ce syllabus à leurs besoins linguistiques particuliers et ajouter des références pour l'adapter à leurs publications locales.
2. Aux organismes de certification, afin de produire les questions d'examen dans leur langue locale en fonction des objectifs d'apprentissage pour ce syllabus.
3. Aux organismes de formation, afin de produire le matériel de formation et déterminer les méthodes pédagogiques appropriées.
4. Aux candidats à la certification, pour se préparer à l'examen de certification (dans le cadre d'un cours de formation ou indépendamment).
5. À la communauté internationale de l'ingénierie des logiciels et des systèmes, pour faire progresser les pratiques professionnelles en tests de logiciels et de systèmes, et comme base pour des livres articles.

L'ISTQB® et le CFTL peuvent permettre à d'autres entités d'utiliser ce syllabus à d'autres fins, à condition qu'elles demandent et obtiennent une autorisation écrite préalable.

0.2 Tester avec l'IA générative

La qualification « Tester avec l'IA générative » s'adresse à toute personne impliquée dans l'utilisation de l'IA générative (GenAI) pour les tests logiciels. Cela inclut les personnes occupant des rôles tels que testeurs, analystes de tests, ingénieurs en automatisation des tests, test managers, testeurs d'acceptation utilisateur et développeurs de logiciels. La qualification « Tester avec l'IA générative» est également appropriée pour toute personne souhaitant acquérir des connaissances de base sur l'utilisation de l'IA générative pour les tests logiciels, comme les chefs de projet, les responsables qualité, les responsables du développement logiciel, les analystes métier, les directeurs informatiques et les consultants en management.

0.3 Parcours de carrière pour les testeurs

Le programme ISTQB® soutient les professionnels du test à toutes les étapes de leur carrière en leur offrant des connaissances à la fois approfondies et étendues. Les personnes qui obtiennent la certification ISTQB® Testeur Certifié Tester avec l'IA Générative (CT-GenAI) peuvent également être intéressées par les niveaux avancés principaux (Analyste de Test, Analyste Technique de Test, Test Manager et Ingénieur de Test) puis par le niveau expert (Management des Tests ou Amélioration du Processus de Test). Veuillez

consulter le site www.istqb.org pour obtenir les dernières informations sur le programme Testeur Certifié de l'ISTQB® ainsi que le site du CFTL.

0.4 Objectifs métiers

Cette section répertorie les objectifs métier (BO – Business Objective) attendus d'un candidat ayant obtenu la certification « Tester avec l'IA générative ».

Un candidat ayant obtenu la certification « Tester avec l'IA générative » est capable de :

GenAI-BO1	Comprendre les concepts fondamentaux, les capacités et les limites de l'IA générative.
GenAI-BO2	Développer des compétences pratiques pour requêter de grands modèles de langage pour les tests logiciels.
GenAI-BO3	Mieux comprendre les risques et les mesures d'atténuation liés à l'utilisation de l'IA générative pour tester des logiciels.
GenAI-BO4	Mieux comprendre les applications des solutions d'IA générative pour tester les logiciels.
GenAI-BO5	Contribuer efficacement à la définition et à la mise en œuvre d'une stratégie et d'une feuille de route en matière d'IA générative pour les tests logiciels au sein d'une organisation.

0.5 Objectifs d'apprentissage évaluables, objectifs pratiques et niveau cognitif des connaissances

Les objectifs d'apprentissage (LO – Learning objective) soutiennent les objectifs métier et sont utilisés pour créer des examens de certification pour le test avec l'IA générative.

En général, tout le contenu de ce syllabus peut être évalué aux niveaux K1, K2 et K3, à l'exception de l'introduction, des objectifs pratiques et des annexes. Les questions d'examen confirmeront la connaissance des mots-clés au niveau K1 (voir ci-dessous) ou des objectifs d'apprentissage à tous les niveaux K.

Les niveaux spécifiques des objectifs d'apprentissage sont indiqués au début de chaque chapitre et classés comme suit :

- K1 : Mémoriser
- K2 : Comprendre
- K3 : Appliquer

De plus amples détails et des exemples d'objectifs d'apprentissage sont fournis à l'annexe A.

Tous les termes répertoriés comme mots-clés juste en dessous des titres des chapitres doivent être mémorisés, même s'ils ne sont pas explicitement mentionnés dans les objectifs d'apprentissage.

Les objectifs spécifiques d'apprentissage pratique (HO – Hands-on Objective) sont indiqués au début de chaque chapitre. Chaque HO est lié à un LO de niveau K2 ou K3, dans le but d'affiner l'apprentissage par la pratique.

Le niveau de chaque objectif d'apprentissage pratique est classé comme suit :

- H0 : peut inclure une démonstration en direct d'un exercice ou une vidéo enregistrée. Comme cela n'est pas effectué par le candidat, il ne s'agit pas strictement d'un exercice.
- H1 : exercice guidé. Les candidats suivent une séquence d'étapes effectuées par le formateur.
- H2 : exercice avec des indices. Le candidat reçoit un exercice accompagné d'indices pertinents afin de pouvoir le résoudre dans le temps imparti.

0.6 L'examen de certification pour Tester avec l'IA générative

L'examen de certification pour tester avec l'IA générative est basé sur ce syllabus. Les réponses aux questions de l'examen peuvent nécessiter l'utilisation d'exigences basées sur plus d'une section de ce syllabus. Toutes les sections du syllabus sont examinables, à l'exception de l'introduction et des annexes. Les standards et les livres sont inclus comme références, mais leur contenu n'est pas examinable, au-delà de ce qui est résumé dans le syllabus lui-même à partir de ces standards et livres.

Pour plus de détails, veuillez-vous reporter au document Structures et règles des examens V1.0 pour la certification « Tester avec IA générative ».

Exigences d'admission: le certificat ISTQB® de niveau Fondation doit être obtenu avant de passer l'examen de certification ISTQB® de testeur certifié Tester avec IA générative.

0.7 Accréditation

Un Membre de l'ISTQB® peut accréditer des organismes de formation dont le contenu des cours est conforme à ce syllabus. Les organismes de formation doivent obtenir les directives d'accréditation auprès du Membre ou de l'organisme chargé de l'accréditation. Un cours accrédité est reconnu comme conforme à ce syllabus et peut faire l'objet d'un examen ISTQB® dans le cadre de la formation.

Les directives d'accréditation pour ce syllabus sont définies dans le document ISTQB® CT-GenAI Accreditation Guidelines (Directives d'accréditation ISTQB® CT-GenAI).

0.8 Prise en compte des normes

Il existe des normes associées aux caractéristiques de qualité et aux tests logiciels, à savoir celles référencées dans le syllabus de niveau Fondation, comme celles de l'IEEE et de l'ISO. Ces références ont pour but de fournir un cadre ou une source d'informations supplémentaires si le lecteur le souhaite. Veuillez noter que les syllabi utilisent les documents standard comme référence. Les documents normatifs ne sont pas destinés à être examinés. Reportez-vous au chapitre 6 pour plus d'informations sur les normes.

0.9 Niveau de détail

Le niveau de détail de ce syllabus permet d'assurer la cohérence des cours et des examens à l'échelle internationale. Pour atteindre cet objectif, le syllabus comprend :

- Des objectifs d'apprentissage décrivant l'intention de la certification ISTQB® Testeur certifié Tester avec l'IA générative.

- Une liste de termes que les étudiants doivent être capables de retenir
- Des objectifs d'apprentissage pour chaque domaine de connaissances, décrivant les résultats cognitifs à atteindre.
- Une description des concepts clés, y compris des références à des sources telles que la littérature acceptée ou les normes.
- Une description de chaque objectif d'apprentissage pratique et de la pratique recommandée pour soutenir l'apprentissage.

Le contenu du syllabus ne décrit pas tout le domaine de connaissances du test avec l'IA générative ; il reflète le niveau de détail qui sera couvert dans les cours de formation ISTQB® Testeur certifié Tester avec l'IA générative. Il se concentre sur les concepts et les techniques de test qui peuvent s'appliquer à tous les projets logiciels lorsque l'IA générative est utilisée pour tester.

Le syllabus utilise la terminologie (c'est-à-dire le nom et la signification) des termes utilisés dans le domaine des tests logiciels et de l'assurance qualité conformément au glossaire ISTQB®.

0.10 Organisation du syllabus

Il y a 5 chapitres dont le contenu peut faire l'objet d'un examen. Le titre principal de chaque chapitre précise la durée de celui-ci ; le calendrier n'est pas indiqué en dessous du niveau du chapitre. Pour les cours de formation agréés, le syllabus exige un minimum de 13,6 heures d'enseignement, réparties entre les 5 chapitres comme suit :

- Chapitre 1 : 100 minutes - Introduction à l'IA générative pour les tests logiciels
  - Le testeur apprend les bases des grands modèles de langage (LLM – Large Language Model), notamment la tokenisation et les capacités multimodales.
  - Le testeur explore les applications de l'IA générative dans le domaine des tests logiciels, en distinguant les chatbots IA des outils de test basés sur les LLM, et en expérimentant la tokenisation, les fenêtres contextuelles et les prompts[1] multimodaux.
- Chapitre 2 : 365 minutes - Ingénierie du prompting pour des tests logiciels efficaces
  - Le testeur apprend à créer des prompts efficaces et structurés pour l'IA générative dans le cadre de tests logiciels.
  - Le testeur acquiert une expérience pratique des techniques d'ingénierie des prompts pour les tâches de test logiciel et les applique.
- Chapitre 3 : 160 minutes - Gérer les risques liés à l'IA générative dans les tests logiciels
  - Le testeur apprend à identifier et à atténuer les hallucinations, les erreurs de raisonnement et les biais lors des tests avec l'IA générative.

[1] NDT : l'équipe de traduction a souhaité conserver le terme anglais « prompt » et ses dérivés au lieu de « requête » ou « invite », tout d'abord parce qu'il est largement entré dans le vocabulaire courant ; mais aussi par souci de clarté, notamment dans l'emploi d'expressions composées telles que « zero-shot prompting » qui auraient requis des périphrases peu usitées ou non unanimement acceptées.

- Le testeur apprend à traiter les questions de confidentialité et de sécurité des données liées à l'IA générative dans les tests logiciels.
- Le testeur prend conscience de la consommation énergétique et de l'impact environnemental de l'IA générative dans les tests logiciels.
- Le testeur se familiarise avec les réglementations, les normes et les meilleures pratiques en matière d'IA pour une utilisation éthique, transparente et sécurisée de l'IA générative dans les tests logiciels.

- Chapitre 4 : 110 minutes - Infrastructure de test basée sur LLM pour tester des logiciels
  - Le testeur explore les architectures d'IA générative telles que Retrieval-Augmented Generation et les agents d'IA générative.
  - Le testeur apprend le processus de réglage fin des LLMs pour les tâches de test logiciel.
  - Le testeur apprend les concepts des opérations sur les grands modèles de langage (LLMOps) pour le déploiement et la gestion des LLMs dans les tests logiciels.
- Chapitre 5 : 80 minutes - Déploiement et intégration de l'IA générative dans les organisations de test
  - Le testeur se familiarise avec une feuille de route structurée pour l'intégration de l'IA générative dans les processus de test.
  - Le testeur se familiarise avec la transformation organisationnelle nécessaire à l'intégration de l'IA générative dans les processus de test.

1 Introduction à l'IA générative pour les tests logiciels – 100 minutes

Mots-clés

None

Termes de l'IA générative

chatbot IA, deep learning, embedding, feature, fenêtre contextuelle, grand modèle de langage (LLM), IA générative, IA symbolique, LLM adapté aux instructions, LLM de fondation, LLM de raisonnement, machine learning, modèle multimodal, tokenisation, Transformer, Transformer génératif pré-entraîné

Objectifs d'apprentissage et objectifs d'apprentissage pratique pour le chapitre 1 :

1.1 Fondements et concepts clés de l'IA générative

GenAI-1.1.1	(K1)	Rappeler différents types d'IA : IA symbolique, machine learning classique, deep learning et IA générative
GenAI-1.1.2	(K2)	Expliquer les bases de l'IA générative et des grands modèles de langage
HO-1.1.2	(H1)	Pratiquer la tokenisation et l'évaluation du nombre de tokens lorsque vous utilisez un LLM pour une tâche de test logiciel
GenAI-1.1.3	(K2)	Faire la différence entre LLM de base, LLM adapté aux instructions et LLM de raisonnement
GenAI-1.1.4	(K2)	Résumer les principes de base des modèles multimodaux de langage et des modèles de vision
HO-1.1.4	(H1)	Écrire et exécuter un prompt pour un LLM multimodal utilisant à la fois des entrées textuelles et des images pour une tâche de test logiciel

1.2 Tirer parti de l'IA générative dans les tests logiciels : principes généraux

GenAI-1.2.1	(K2)	Donner des exemples de capacités clés des LLMs pour les tâches de test
GenAI-1.2.2	(K2)	Comparer les modalités d'interaction lorsque vous utilisez l'IA générative pour tester des logiciels

1.1 Fondements et concepts clés de l'IA générative

L'intelligence artificielle générative (IA générative) est une branche de l'intelligence artificielle qui utilise de grands modèles pré-entraînés pour générer des résultats semblables à ceux d'un être humain, tels que du texte, des images ou du code. Les grands modèles de langage (LLM) sont des modèles d'IA générative pré-entraînés sur de grands ensembles de données textuelles, ce qui leur permet de déterminer le contexte et de produire des réponses pertinentes en fonction des requêtes des utilisateurs.

Les concepts clés comprennent la tokenisation (c'est-à-dire la division du texte en unités pour un traitement efficace), les fenêtres contextuelles (limitation de la quantité d'informations prises en compte à la fois pour maintenir la pertinence) et les modèles multimodaux (capables de traiter plusieurs types de données telles que du texte, des images et de l'audio pour des interactions riches).

Dans le domaine des tests logiciels, ces LLM peuvent prendre en charge des tâches telles que la revue et l'amélioration des critères d'acceptation, la génération de cas de test ou de scripts de test, l'identification des défauts potentiels, l'analyse des canevas de défauts, la génération de données de test synthétiques ou la prise en charge de la génération de documentation, et ce tout au long du processus de test.

1.1.1 Panorama de l'IA : IA symbolique, machine learning classique, deep learning et IA générative

L'intelligence artificielle (IA) est un vaste domaine qui englobe différents types de technologies, chacune avec sa propre façon de résoudre les problèmes, telles que l'IA symbolique, le machine learning classique, le deep learning et l'IA générative (entre autres technologies qui ne relèvent pas du périmètre de ce syllabus) :

- L'IA symbolique utilise un système basé sur des règles pour imiter la prise de décision humaine. Essentiellement, l'IA symbolique représente les connaissances à l'aide de symboles et de règles logiques.
- Le machine learning classique est une approche basée sur les données qui nécessite la préparation des données, la sélection des caractéristiques et l'entraînement des modèles. Elle peut être utilisée pour des tâches telles que la catégorisation des défauts et la prédiction des problèmes logiciels.
- Le deep learning utilise des structures de machine learning, appelées réseaux neuronaux, pour apprendre automatiquement des features à partir de données. Les modèles de deep learning peuvent trouver des motifs dans des ensembles de données très volumineux et complexes, tels que des images, des vidéos, des fichiers audio ou du texte, sans que les utilisateurs aient besoin de définir manuellement des features. Dans la pratique, cela peut toutefois nécessiter une intervention humaine pour des tâches telles que l'annotation des données, le réglage des modèles ou la validation des résultats.
- L'IA générative utilise des techniques de deep learning pour créer de nouveaux contenus (textes, images, code) en apprenant et en imitant les canevas de ses données d'entraînement. Des modèles tels que les LLM peuvent générer du texte, écrire du code et simuler un raisonnement ou une résolution de problèmes dans le périmètre de leur entraînement.

En résumé, le domaine de l'IA a évolué dans plusieurs directions, chacune présentant des avantages et des limites différents. Le principal avantage de l'utilisation de l'IA générative pour les tests logiciels réside dans le fait qu'elle utilise des modèles pré-entraînés qui peuvent être appliqués directement aux tâches de

test sans nécessiter de phase d'entraînement supplémentaire, bien que cela comporte certains risques (voir section 3.1).

1.1.2 Notions de base sur l'IA générative et les LLMs

Basés sur le modèle d'apprentissage profond génératif pré-entraîné Transformer, les LLMs sont entraînés sur de très grands ensembles de données, notamment des livres, des articles et des sites web. Les petits modèles de langage (SLM – Small Language Model) sont des modèles compacts avec moins de paramètres que les grands modèles de langage, conçus pour fournir des solutions d'IA générative légères et ciblées.

Les LLMs peuvent gérer les nuances linguistiques et générer un contenu cohérent. Deux concepts clés aident les LLMs à traiter et à générer du contenu : la tokenisation et l'encodage appelé « embedding ». La tokenisation et les embeddings convertissent le langage en une forme numérique que le modèle peut traiter efficacement.

- La tokenisation dans les modèles de langage est le processus qui consiste à décomposer un texte en unités plus petites appelées tokens. Les tokens peuvent être aussi petits qu'un caractère ou aussi grands qu'un sous-mot ou un mot. Lorsqu'un LLM traite une phrase, il commence par tokeniser l'entrée afin que chaque token puisse être compris individuellement, tout en conservant le contexte global.
- Les embeddings sont des représentations numériques des tokens qui encodent leurs relations sémantiques, syntaxiques et contextuelles dans un format adapté au traitement par les modèles d'IA générative. Chaque token est transformé en un vecteur dans un espace à haute dimension, capturant des informations nuancées sur sa signification et son utilisation. Les tokens ayant des significations ou des rôles contextuels similaires ont des embeddings qui sont positionnés à proximité les uns des autres dans cet espace. Cette proximité permet aux LLM de comprendre les relations entre les mots, de conserver le contexte et de générer des réponses cohérentes et adaptées au contexte.

Les LLMs utilisent une architecture de réseau neuronal connue sous le nom de modèle « Transformer ». Les modèles Transformer excellent dans les tâches linguistiques en traitant le contexte de longues séquences de texte et en apprenant comment les tokens sont liés les uns aux autres. Lors de l'inférence, les LLMs prédisent le token suivant dans une séquence, en tirant parti de ces relations apprises pour générer un texte cohérent et adapté au contexte. Le modèle Transformer peut être utilisé pour générer un nouveau texte statistiquement plausible, basé sur les données d'entraînement et la requête. Mais plausible ne signifie pas nécessairement correct.

Les LLMs présentent un comportement non déterministe principalement dû à la nature probabiliste de leurs mécanismes d'inférence et à leurs configurations d'hyperparamètres. Ce comportement aléatoire inhérent peut entraîner des variations dans les résultats, même lorsque la même entrée est fournie plusieurs fois.

Dans le domaine des LLMs, la fenêtre contextuelle fait référence à la quantité de texte au total, mesurée en tokens, que le modèle peut prendre en compte lors de la génération de réponses. Une fenêtre contextuelle plus grande permet au modèle de maintenir la cohérence sur des passages plus longs, par exemple lors de l'analyse de longs logs de test. Cependant, l'augmentation du nombre de tokens dans la fenêtre contextuelle augmente également la complexité de calcul et le temps de traitement nécessaires au modèle pour fonctionner efficacement.

Objectif d'apprentissage pratique HO-1.1.2 (H1): Pratiquer la tokenisation et l'évaluation du nombre de tokens

Cette activité pratique est conçue pour aider les candidats à acquérir une compréhension concrète de la tokenisation et de ses implications lorsqu'ils travaillent avec des LLMs. L'exercice est divisé en deux parties principales :

- Tokenisation : utilisation d'un tokeniseur pour décomposer un échantillon de texte en tokens individuels. Examen du résultat pour voir comment les mots, la ponctuation et les phrases sont représentés, et identification des canevas ou des différences dans la tokenisation.
- Évaluation du nombre de tokens : mesure du nombre de tokens générés à partir de divers textes d'entrée. Analyse de l'influence du nombre de tokens sur les performances du modèle, en particulier en relation avec les limites de la fenêtre contextuelle du modèle et des considérations d'efficience.

À la fin de cet exercice, les candidats seront en mesure de mieux anticiper comment différentes structures de texte et différentes longueurs d'entrée peuvent affecter les interactions avec les LLMs.

1.1.3 LLM de base, adapté aux instructions et de raisonnement

Les grands modèles de langage sont développés à travers des étapes d'entraînement et progressivement spécialisés afin d'améliorer leur efficacité dans un large éventail de tâches. Ces étapes donnent lieu à trois grandes catégories : LLM de fondation, LLM adapté aux instructions et LLM de raisonnement.

- LLM de fondation : Il s'agit de modèles polyvalents entraînés sur des ensembles de données vastes et diversifiés comprenant du texte, du code, des images et d'autres modalités. Leur pré-entraînement approfondi leur permet de prendre en charge diverses tâches dans des domaines tels que le traitement du langage naturel, la vision par ordinateur et la reconnaissance vocale. Bien que puissants et flexibles, les modèles de fondation nécessitent généralement une adaptation supplémentaire pour répondre aux exigences spécifiques des requêtes à traiter.
- LLM adapté aux instructions : Dérivés des modèles de fondation, les LLMs adaptés aux instructions sont affinés à l'aide d'ensembles de données qui associent des requêtes à des réponses attendues. Cette étape améliore leur alignement avec les instructions humaines, ce qui renforce leur utilisabilité dans les applications du monde réel. Le processus d'ajustement consiste à optimiser le respect des tâches, le suivi des instructions et la cohérence des réponses, améliorant ainsi la capacité du modèle à interpréter et à agir efficacement selon l'intention de l'utilisateur.
- LLM de raisonnement : Les modèles de raisonnement étendent les modèles adaptés à des instructions en mettant l'accent sur les capacités cognitives structurées telles que l'inférence logique, la résolution de problèmes en plusieurs étapes et le raisonnement en chaîne. Ces modèles sont entraînés ou affinés à l'aide de tâches soigneusement sélectionnées qui exigent une compréhension contextuelle, des étapes de raisonnement intermédiaires et la synthèse d'informations complexes. Ils sont donc mieux adaptés aux tâches à forte charge cognitive, notamment dans les domaines techniques.

Dans le contexte des applications d'IA générative pour les tests logiciels, on utilise à la fois des LLM adaptés aux instructions (parfois appelés « non raisonnants ») et des LLM de raisonnement. Le choix dépend de la complexité et des exigences en matière de raisonnement de la tâche de test spécifique à accomplir.

1.1.4 LLM multimodaux et modèles de vision

Les LLM multimodaux étendent le modèle traditionnel du Transformer afin de traiter plusieurs modalités de données, notamment le texte, les images, le son et la vidéo. Ces modèles sont entraînés sur des ensembles de données volumineux et variés qui leur permettent d'apprendre les relations entre différents types de données. Afin de gérer diverses modalités, la tokenisation est adaptée à chaque type de données. Par exemple, les images sont converties en embeddings à l'aide de modèles de vision avant d'être traitées dans le modèle Transformer.

Les modèles de vision, un sous-ensemble des LLM multimodaux, intègrent spécifiquement des informations visuelles et textuelles pour effectuer des tâches telles que la légende d'images, la réponse à des questions visuelles et l'analyse de la cohérence entre les entrées textuelles et visuelles.

Dans le domaine des tests logiciels, les LLM multimodaux, en particulier ceux enrichis d'un modèle de vision, offrent des opportunités considérables. Ils peuvent analyser les éléments visuels des applications, tels que les captures d'écran et les maquettes d'interface graphique, ainsi que les descriptions textuelles associées, telles que les rapports de défauts ou les User Stories. Cette capacité permet aux testeurs d'identifier les divergences entre les résultats attendus et les éléments visuels réels sur une capture d'écran. En outre, les LLM enrichis d'un modèle de vision peuvent générer des cas de test riches et réalistes qui intègrent à la fois des données textuelles et des aspects visuels, augmentant ainsi la couverture globale.

Objectif d'apprentissage pratique HO-1.1.4 (H1) : Revue et exécution d'un prompt donné adressant une tâche de test à l'aide d'un modèle LLM multimodal

Cet exercice consiste à examiner et à exécuter une requête donnée pour un LLM multimodal à l'aide d'entrées textuelles et d'images afin de résoudre une tâche de test en deux étapes :

- Revoir les entrées : revue du prompt et des données d'entrée (texte et image).
- Exécution du prompt et vérification du résultat : utilisation d'un LLM multimodal pour traiter à la fois l'image et le texte, puis vérification de la réponse du LLM.

Cet exercice montre comment s'utilisent les LLM multimodaux pour une tâche impliquant à la fois des entrées textuelles et visuelles, dans des cas d'usage de l'IA pour le test de logiciels, notamment en reconnaissant les avantages et les difficultés potentiels associés.

1.2 Tirer parti de l'IA générative dans les tests logiciels : principes généraux

L'IA générative offre des capacités transformatrices dans diverses activités de test. Les LLMs excellent dans le traitement du langage naturel et du code, la génération de textes et de codes cohérents, la réponse

à des questions, la synthèse d'informations, la traduction de langues et l'analyse d'images dans un contexte multimodal.

Les professionnels du test, quels que soit leurs rôles, peuvent tirer parti de l'IA générative de deux manières complémentaires : grâce à des chatbots IA générative qui fournissent des réponses instantanées aux prompts, et grâce à des applications basées sur les LLMs intégrés dans des outils de test.

1.2.1 Capacités clés des LLMs pour les tâches de test

Les LLMs peuvent interpréter les exigences, les spécifications, les captures d'écran, le code, les cas de test et les rapports de défauts, ce qui en fait des outils permettant de traiter et de clarifier les informations nécessaires tout au long du processus de test et de générer des éléments du testware. Voici quelques-unes des principales capacités des LLMs pertinentes pour les tests logiciels :

- **Analyse et amélioration des exigences** : les LLMs peuvent aider à analyser les exigences et d'autres éléments de la base de test en identifiant les ambiguïtés, les incohérences ou les informations manquantes. Ils peuvent générer des questions pertinentes pour aider à clarifier les exigences lors des discussions avec les parties prenantes.
- **Création de cas de test** : les LLMs peuvent aider à générer des cas de test et suggérer des objectifs de test basés sur les exigences du système, les User Stories ou tout autre élément de la base de test.
- **Génération d'oracles de test** : les LLMs peuvent aider à générer les résultats attendus.
- **Génération de données de test** : les LLMs peuvent générer des ensembles de données, définir des valeurs limites et créer différentes combinaisons de données de test.
- **Aide à l'automatisation des tests** : les LLMs peuvent aider à générer des scripts de test à partir de la description des cas de test et améliorer les scripts de test existants en suggérant des modifications et en identifiant les techniques de conception des tests appropriées.
- **Analyse des résultats des tests** : les LLMs peuvent aider à analyser les résultats des tests en créant des résumés et en classant les anomalies en fonction de leur sévérité et de leur priorité.
- **Création de testware** : les LLMs peuvent aider à créer divers documents, notamment des plans de test, des rapports de test et des rapports de défauts, et à les mettre à jour au fur et à mesure de l'évolution du projet.

Ces capacités montrent comment les LLMs peuvent avoir un impact sur divers aspects des tests logiciels tout au long du processus de test.

1.2.2 Chatbots IA et applications de test basées sur LLM pour les tests logiciels

Les chatbots IA et les applications de test basées sur LLM peuvent tous deux aider les testeurs, malgré leurs différences en termes de fonctionnalités, de flexibilité et d'approches d'intégration.

Les chatbots IA offrent une interface conversationnelle conviviale qui permet aux testeurs de communiquer directement avec les LLMs. Cette interaction en langage naturel permet aux testeurs d'entrer des questions, des commandes ou des prompts et d'obtenir des réponses immédiates et contextuelles. Grâce à des techniques telles que le chaînage de prompts, les testeurs peuvent affiner les résultats de manière

itérative, ce qui rend les chatbots particulièrement efficaces pour des tâches routinières, les tests exploratoires et même l'intégration de nouveaux testeurs en leur fournissant un accès rapide aux connaissances et aux pratiques en matière de test.

Ces chatbots IA sont particulièrement utiles dans les scénarios nécessitant un retour rapide, une clarification des concepts de test ou une exploration dynamique des exigences et des cas de test potentiels. Leur interface intuitive les rend accessibles même aux parties prenantes non techniques, élargissant ainsi la base d'utilisateurs potentiels et encourageant une adoption large.

Les applications de test basées sur LLM, en revanche, impliquent l'intégration des capacités LLM via des API afin d'effectuer des tâches de test bien définies et souvent automatisées. Ces applications offrent une plus grande personnalisation et une plus grande évolutivité, permettant aux organisations et aux fournisseurs d'outils d'intégrer l'IA générative dans l'outillage de test existants. Cela permet l'automatisation de tâches répétitives ou complexes, telles que la génération de cas de test, l'analyse des défauts ou la synthèse des données de test. Dans des implémentations plus avancées, les organisations peuvent créer des agents IA spécialement conçus pour remplir certaines fonctions de test (voir chapitre 4).

Quelle que soit la manière dont le testeur interagit avec les LLMs, que ce soit via des chatbots ou des applications intégrées basées sur LLM, la mise en œuvre réussie de l'IA générative dans les tests nécessite une ingénierie de requêtes efficace (voir chapitre 2). Des requêtes soigneusement conçues et des instructions claires et spécifiques sont essentielles pour garantir que les résultats générés par les LLM sont précis, pertinents et alignés sur les objectifs des tests. Cette pratique permet de maximiser la valeur tirée de l'IA générative et garantit un soutien cohérent et fiable pour un large éventail d'activités de test.

2 Ingénierie du prompting pour des tests logiciels efficaces – 365 minutes

Mots-clés

cas de test, condition de test, critères d'acceptation, données de test, conception de test, rapport de test, script de test

Termes de l'IA générative

enchaînement de prompts, few-shot prompting, ingénierie du prompting, méta-prompting, one-shot prompting, prompt, prompt système, prompt utilisateur, traitement du langage naturel, zero-shot prompting

Objectifs d'apprentissage et objectifs d'apprentissage pratique pour le chapitre 2 :

2.1 Développement efficace des prompts

GenAI-2.1.1	(K2)	Donner des exemples de la structure des prompts utilisées dans l'IA générative pour les tests logiciels
HO-2.1.1	(H0)	Observer plusieurs prompts donnés pour des tâches de test logiciel, en identifiant les composants du rôle, du contexte, de l'instruction, des données d'entrée, des contraintes et du format de sortie dans chacune d'eux
GenAI-2.1.2	(K2)	Différencier les principales techniques de prompting pour les tests logiciels
HO-2.1.2a	(H0)	Observer des démonstrations d'enchaînement de prompts, de few-shot prompting et de méta-prompting appliqués à des tâches de test logiciel
HO-2.1.2b	(H1)	Identifier les techniques d'ingénierie du prompting utilisées dans des exemples donnés
GenAI-2.1.3	(K2)	Distinguer les prompts système des prompts utilisateur

2.2 Application des techniques d'ingénierie du prompting aux tâches de test logiciel

GenAI-2.2.1	(K3)	Appliquer l'IA générative aux tâches d'analyse de test
HO-2.2.1a	(H2)	Pratiquer les prompts multimodaux pour générer des critères d'acceptation pour une User Story basée sur un maquette IHM
HO-2.2.1b	(H2)	Pratiquer l'enchaînement de prompts et la vérification humaine pour analyser progressivement une User Story donnée et affiner les critères d'acceptation
GenAI-2.2.2	(K3)	Appliquer l'IA générative à la conception des tests et à l'implémentation des tests
HO-2.2.2a	(H2)	Pratiquer la génération de cas de test fonctionnel à partir de User Stories avec l'IA générative en utilisant l'enchaînement de prompts, les prompts structurés et les méta-prompts
HO-2.2.2b	(H2)	Utiliser la technique du few-shot prompting pour générer des conditions de test et des cas de test de style Gherkin à partir de User Stories

HO-2.2.2c	(H2)	Utiliser l'enchaînement de prompts pour prioriser les cas de test dans une suite de tests donnée, en tenant compte de leurs priorités et dépendances spécifiques
GenAI-2.2.3	(K3)	Appliquer l'IA générative aux tests de régression automatisés
HO-2.2.3a	(H2)	Pratiquer le few-shot prompting pour créer et gérer des scripts de test dirigés par les mots-clés
HO-2.2.3b	(H2)	Pratiquer l'ingénierie du prompting structuré pour l'analyse des rapports de test dans le contexte des tests de régression
GenAI-2.2.4	(K3)	Appliquer l'IA générative aux tâches de contrôle et de suivi des tests
HO-2.2.4	(H0)	Observer les métriques de suivi des tests préparées par l'IA générative à partir des données de test
GenAI-2.2.5	(K3)	Sélectionner et appliquer les techniques de prompts appropriées à un contexte et à une tâche de test donnés
HO-2.2.5	(H1)	Sélectionner et appliquer des techniques de prompting adaptées au contexte pour une tâche de test donnée

2.3 Évaluer les résultats de l'IA générative et affiner les requêtes pour les tâches de test logiciel

GenAI-2.3.1	(K2)	Comprendre les métriques permettant d'évaluer les résultats de l'IA générative sur des tâches de test
HO-2.3.1	(H0)	Observer comment les métriques peuvent être utilisées pour évaluer le résultat de l'IA générative sur une tâche test
GenAI-2.3.2	(K2)	Donner des exemples de techniques permettant d'évaluer et de remanier de manière itérative les prompts.
HO-2.3.2	(H1)	Évaluer et optimiser un prompt pour une tâche de test donnée

2.1 Développement efficace des prompts

Une conception efficace des prompts facilite la réalisation précise et efficace des tâches de test logiciel par les outils d'IA générative, et que les testeurs obtiennent des résultats utiles des LLMs. Un prompt structuré comprend différents composants (voir section 2.1.1). Chacun de ces composants contribue à la clarté et à la précision d'un prompt qui communique efficacement les exigences et les attentes aux LLMs.

Diverses techniques d'ingénierie du prompting améliorent l'efficacité des prompts dans les tests logiciels. Des techniques telles que l'enchaînement de prompts, le few-shot prompting et le méta-prompting permettent de répondre à des sujets complexes en matière de tests (voir section 2.1.2).

La combinaison de prompts structurés (voir section 2.1.1) et des techniques de prompt majeures vise à obtenir de bons résultats lors de l'interrogation d'un LLM pour des tâches de test logiciel (voir section 2.1.3).

2.1.1 Structure des prompts pour l'IA générative dans les tests logiciels

Un prompt structuré pour le test logiciel comprend généralement six composants :

- **Rôle** : le rôle définit la perspective ou le persona que le modèle d'IA générative doit adopter lorsqu'il génère une réponse. La spécification du rôle aide le LLM à déterminer ses responsabilités et à adopter un ton ou une approche appropriés, par exemple en agissant comme un testeur, un test manager ou un ingénieur en automatisation des tests.
- **Contexte** : le contexte fournit les informations de base dont le modèle d'IA générative a besoin pour déterminer les conditions de test. Cela comprend des détails sur l'objet de test, la fonctionnalité spécifique à tester et toute information contextuelle pertinente.
- **Instructions** : les instructions sont des directives données à l'IA générative qui décrivent la tâche spécifique à effectuer. Des instructions claires, impératives et concises comprennent une description de la tâche et toutes les exigences pertinentes pour celle-ci.
- **Données d'entrée** : les données d'entrée comprennent toutes les informations nécessaires à l'exécution de la tâche, telles que les User Stories, les critères d'acceptation, les captures d'écran, le code, les cas de test existants ou les exemples de sortie. Fournir des données d'entrée détaillées et structurées aide le LLM à générer des résultats plus précis et plus adaptés au contexte.
- **Contraintes** : les contraintes décrivent les restrictions ou les considérations particulières auxquelles le LLM doit se conformer. Les contraintes aident à préciser comment les instructions doivent être appliquées aux données d'entrée.
- **Format de sortie** : les spécifications de sortie indiquent le format, la structure ou les caractéristiques attendus de la réponse. Cela aide à façonner la sortie du LLM.

Ces composants forment la structure de base du prompt. Cette structure doit être combinée avec l'implémentation de techniques de prompting (voir section 2.1.2), en fonction de la tâche à accomplir et du LLM à utiliser.

Objectif d'apprentissage pratique HO-2.1.1 (H0) : Observer et analyser les composants du prompt

Dans une démonstration, plusieurs prompts structurés sont testées sur un chatbot IA, chacune étant adaptée à des tâches spécifiques de test logiciel. Ces prompts suivent un format structuré composé de six composants clés : rôle, contexte, instruction, données d'entrée, contraintes et format de sortie. La démonstration vise à faciliter l'observation et l'analyse de ces prompts structurés, en mettant en évidence la manière dont chaque composant contribue à fournir des informations précises, pertinentes et exploitables à un LLM utilisé pour une tâche de test logiciel.

2.1.2 Principales techniques de prompting pour les tests logiciels

Ces dernières années, de nombreuses techniques de prompting LLM ont été proposées pour différents cas d'utilisation de l'IA générative (Schulhoff 2024). Parmi celles-ci, trois techniques de prompting principales sont couramment utilisées pour les tâches de test avec l'IA générative en conjonction avec la structure de prompt à 6 composants décrite ci-dessus (voir section 2.1.1) : l'enchaînement de prompts, le few-shot prompting et le méta-prompting.

- **L'enchaînement de prompts** consiste à diviser une tâche en une série d'étapes intermédiaires (plusieurs prompts). Le résultat de chaque étape est vérifié manuellement ou automatiquement et remanié avant de passer à l'étape suivante. Cette approche permet d'obtenir une plus grande précision, car chaque réponse informe le prompt suivant. L'enchaînement de prompts est particulièrement utile dans les processus de test où les tâches sont complexes et nécessitent une décomposition en sous-tâches et une vérification systématique des résultats intermédiaires du LLM. Il permet également des interactions dynamiques dans les processus de test.
- **Le few-shot prompting** consiste à fournir au LLM des exemples dans le prompt. Alors que le zero-shot prompting (sans exemple) s'appuie sur les connaissances préexistantes du modèle pour générer une réponse, le one-shot prompting fournit un exemple pour illustrer le résultat souhaité pour une entrée donnée. Les prompts avec la technique few-shot prompting contiennent plusieurs exemples (quelques-uns) afin de consolider davantage le comportement souhaité du modèle en matière de réponse.

  Cette technique aide à guider le modèle en fournissant une référence claire et en garantissant que les résultats sont cohérents et conformes aux attentes. Le few-shot prompting est particulièrement efficace pour les tâches où des exemples peuvent illustrer le comportement requis, permettant ainsi au modèle de généraliser efficacement et de produire des résultats fiables.
- **Le méta-prompting** exploite la capacité de l'IA à générer ou à remanier ses propres prompts. Dans un cycle itératif, le LLM peut générer des prompts qui peuvent être évalués et affinés par le testeur. Cette approche optimise la qualité des prompts en tirant parti des connaissances des LLMs en matière de prompts optimisés. Le méta-prompting est particulièrement utile lorsque l'efficience et l'optimisation des prompts sont essentielles, car elle réduit l'effort manuel nécessaire à la conception de prompts efficaces. Un autre avantage du méta-prompting est que si le testeur ne sait pas comment créer un prompt efficace, il peut collaborer avec le LLM pour la co-créer. Cela reflète une forme de jumelage avec l'outil d'IA générative, où le testeur et l'IA travaillent ensemble de manière interactive pour atteindre un objectif commun. Ce concept de binôme met en avant une nouvelle façon de collaborer avec les outils d'IA, améliorant à la fois la productivité et

l'apprentissage, non seulement dans l'ingénierie du prompting, mais aussi dans la programmation en binôme et les tests en binôme.

Ces techniques de prompt peuvent être utilisées efficacement en combinaison pour améliorer les résultats du LLM (voir section 2.2.5).

Objectif d'apprentissage pratique HO-2.1.2a (H0): Observation et discussion de l'enchaînement de prompts, du few-shot prompting et du méta-prompting dans des tâches de test logiciel

Les participants découvrent l'enchaînement de prompts, le few-shot prompting et le méta-prompting sur un chatbot IA, chacun appliqué à des tâches spécifiques de test logiciel. La démonstration vise à explorer et à discuter ces techniques de prompt dans le contexte des tests logiciels, en soulignant comment chaque technique contribue à la précision et à l'exhaustivité des résultats du LLM.

Objectif d'apprentissage pratique HO-2.1.2b (H1): Identification de techniques d'ingénierie du prompting à partir d'exemples donnés

Les participants lisent une série d'exemples de prompts liés aux tests logiciels afin d'identifier les principales techniques de prompt utilisées. L'accent sera mis sur la reconnaissance de techniques telles que l'enchaînement de prompts, le few-shot prompting et le méta-prompting, tout en soulignant leurs caractéristiques distinctes et leurs applications pratiques.

Cette activité vise à approfondir la compréhension des participants sur la manière dont différentes techniques de prompting améliorent l'utilisation efficace de l'IA générative dans les tests logiciels.

2.1.3 Prompt système et prompt utilisateur

Les prompts système et les prompts utilisateur ont des objectifs différents dans les interactions avec les LLMs, chacun jouant un rôle distinct dans la formation de la conversation. Le prompt système est généralement défini par le développeur ou le testeur afin de guider le comportement global du LLM. Il n'est ni visible ni modifiable par l'utilisateur du chatbot dans la plupart des interfaces.

Un prompt système agit comme un ensemble de commandes prédéfinies qui définit le comportement, la personnalité et les paramètres opérationnels du LLM. Les paramètres opérationnels déterminent la manière dont le LLM répond, par exemple en utilisant un ton formel, en donnant des réponses concises, en respectant les règles spécifiques au domaine ou en évitant certains comportements. Le prompt système définit les règles pour l'ensemble de la conversation. Il peut contenir des parties d'un prompt structuré, telles que le rôle, le contexte et les contraintes.

Le prompt système reste constant tout au long de la session d'interaction et établit le cadre de base de la manière dont le LLM doit répondre. Par exemple, un prompt système peut dire : « Vous êtes un assistant spécialisé dans les tests logiciels. Répondez toujours clairement, utilisez un langage formel et concentrez-vous sur les pratiques recommandées par l'ISTQB®. Évitez les spéculations et citez les principes de test lorsque cela est pertinent. »

Le prompt utilisateur, quant à lui, représente l'entrée ou la question réelle de l'utilisateur du chatbot. Il change à chaque interaction et peut inclure des instructions spécifiques, des questions ou des tâches que l'utilisateur du chatbot souhaite voir traiter par le LLM. Contrairement au prompt système, les prompts

utilisateur sont directement visibles et constituent le contexte immédiat de chaque réponse. Par exemple, un prompt utilisateur pourrait être : « Enumère les principales différences entre les tests boîte noire et boîte blanche à l'aide d'exemples. »

En général, le prompt système est défini une seule fois au début de la conversation, puis des prompts utilisateur sont envoyés successivement pour chaque interaction. Le LLM génère des réponses en tenant compte à la fois du prompt système, qui reste inchangé, et du prompt utilisateur actuel. Pour une implémentation efficace, les prompts système doivent être clairs et précis quant au rôle du LLM et aux contraintes éventuelles. Ils peuvent également contenir des informations contextuelles et des instructions générales, par exemple concernant le résultat attendu.

Les prompts utilisateur doivent être ciblés et bien structurés, et inclure des instructions explicites ainsi que des informations contextuelles supplémentaires pertinentes et des instructions relatives au format de sortie.

2.2 Application des techniques d'ingénierie du prompting aux tâches de test logiciel

L'application des techniques d'ingénierie du prompting aux tests logiciels permet à l'IA générative de prendre en charge des tâches de test telles que l'analyse, la conception, l'automatisation, la hiérarchisation des cas de test, la détection des défauts, l'analyse de la couverture, ainsi que la surveillance et le contrôle des tests. En utilisant et en combinant des techniques telles que l'enchaînement de prompts, le few-shot prompting et le méta-prompting, les équipes peuvent adapter les prompts de l'IA aux objectifs spécifiques des tests, rendant ainsi les résultats plus précis, pertinents et efficaces. Des données d'entrée de haute qualité sont essentielles pour obtenir des résultats significatifs avec l'IA générative.

2.2.1 Analyse de test avec l'IA générative

L'IA générative peut prendre en charge les tâches d'analyse de test en générant et en hiérarchisant les conditions de test, en identifiant les défauts dans la base de test et en fournissant une analyse de couverture. Les données d'entrée comprennent les exigences, les User Stories, les spécifications techniques, les maquettes d'interface graphique et d'autres informations pertinentes. La sortie se compose de produits de travail d'analyse de test typiques, tels que des conditions de test hiérarchisées (par exemple, des critères d'acceptation).

Voici quelques tâches d'analyse de test typiques qui peuvent être prises en charge avec l'IA générative :

- **Identifier les défauts potentiels dans la base de test** : L'IA générative peut aider à analyser la base de test afin de détecter les incohérences, les ambiguïtés ou les informations incomplètes susceptibles d'entraîner des défauts. En comparant des motifs d'exigences similaires ou en appliquant les connaissances issues de rapports de défauts précédents, le LLM peut signaler les anomalies potentielles et suggérer des améliorations.
- **Générer des conditions de test en utilisant la base de test,** par exemple, pour les exigences/User Stories : les LLMs peuvent analyser les exigences et les User Stories afin de générer des conditions de test. Grâce au traitement du langage naturel, ils peuvent interpréter la signification des exigences et les décomposer en instructions mesurables et testables. Cela peut aider à traduire les exigences en conditions de test spécifiques.
- **Hiérarchiser les conditions de test en fonction du niveau de risque** : Grâce aux informations sur la probabilité du risque et l'impact du risque de défaillance pour chaque condition de test, un

LLM peut aider à hiérarchiser les efforts de test. En tenant compte d'aspects tels que la conformité réglementaire, les fonctionnalités destinées aux utilisateurs (par exemple, la fonctionnalité de connexion ou le traitement des paiements) et les données historiques sur les défauts, le LLM peut recommander des niveaux de priorité.

- **Aider à l'analyse de la couverture** : En cartographiant les exigences et les User Stories aux conditions de test, un LLM peut effectuer une analyse de couverture afin de déterminer si tous les aspects de la base de test sont couverts. Ceci est particulièrement utile pour les projets aux exigences complexes, où des lacunes dans la couverture peuvent entraîner des défauts manqués.
- **Suggérer des techniques de test** : L'IA générative peut suggérer des techniques de test pertinentes (par exemple, analyse des valeurs limites, partition d'équivalence) en fonction du type d'exigence ou de User Story testée. Cela peut aider les testeurs à appliquer les techniques de test les plus efficaces pour des conditions de test spécifiques.

La qualité et la pertinence des entrées fournies au LLM par rapport à la tâche à accomplir ont une incidence directe sur l'exactitude et la précision des résultats générés par le LLM.

Objectif d'apprentissage pratique 2.2.1a (H2): S'entraîner à créer des prompts multimodaux structurés afin de générer des critères d'acceptation pour une User Story basée sur une maquette d'interface graphique.

Il s'agit d'un exercice visant à s'entraîner à rédiger des prompts structurés à l'aide d'entrées multimodales (texte et image). L'objectif est de générer des critères d'acceptation de haute qualité (c'est-à-dire bien formés, clairs et complets) à partir d'une User Story et d'un wireframe d'interface graphique. D'autres éléments textuels peuvent être ajoutés pour fournir du contexte, tels que des contraintes sur les champs d'entrée ou des règles métier à appliquer au traitement des données.

Les résultats obtenus à partir du LLM sont comparés afin d'évaluer l'impact de différentes formulations de la prompt structuré (rôle, contexte, instruction, données d'entrée textuelles et image, contraintes et format de sortie) pour une tâche d'analyse de test.

Cet exercice permet d'acquérir une expérience pratique de l'importance d'une structuration des prompts, de la contribution d'instructions précises et de l'importance des données contextuelles textuelles et visuelles pour obtenir des résultats précis et pertinents à partir du LLM.

Objectif d'apprentissage pratique 2.2.1b (H2): Pratiquer l'enchaînement de prompts et la vérification humaine pour analyser progressivement une User Story donnée et affiner les critères d'acceptation.

Il s'agit d'un exercice visant à mettre en pratique l'enchaînement de prompts afin d'analyser une User Story donnée et de peaufiner les critères d'acceptation, en identifiant d'abord les ambiguïtés, puis en évaluant la testabilité et enfin en évaluant l'exhaustivité. Cet exercice encourage une approche étape par étape, en affinant l'analyse à chaque étape afin de s'assurer que les critères d'acceptation sont bien formulés et exploitables pour atteindre les objectifs du test. À chaque étape, les résultats fournis par le LLM sont vérifiés manuellement et corrigés, si nécessaire, soit en ajustant la sortie, soit par un processus d'enchaînement de prompts avec le LLM. De cette manière, l'étape suivante utilise un résultat propre de l'étape précédente pour aborder un autre aspect de l'amélioration des critères d'acceptation.

Cet exercice permet de découvrir de manière pratique les avantages de décomposer une tâche complexe en sous-tâches, avec vérification humaine des résultats de chaque étape.

2.2.2 Conception et implémentation des tests avec l'IA générative

Comme décrit dans [ISTQB_CTFL_SYL], la conception des tests implique l'élaboration et l'affinage des conditions de test, qui sont ensuite traduites en cas de test et autre testware. L'implémentation des tests implique la création ou l'acquisition du testware nécessaire pour exécuter les tests.

Les tests manuels et les scripts de test automatisés peuvent être créés, priorisés et organisés dans un calendrier d'exécution des tests avec le support de l'IA générative. L'IA générative peut aider de manière significative ce large groupe d'activités de test en facilitant la création et l'évaluation de divers testware, notamment des cas de test, des données de test, des scripts de test et des environnements de test.

Voici quelques tâches typiques de conception et d'implémentation de tests qui peuvent être prises en charge par l'IA générative:

- **Génération de cas de test** : Le traitement du langage naturel permet à l'IA générative de créer des versions préliminaires de cas de test basées sur des exigences fonctionnelles et non fonctionnelles. Lorsqu'il est prompté avec des informations appropriées, un LLM peut suggérer des préconditions et des entrées de test, des résultats attendus et des critères de couverture ; produisant ainsi des cas de test qui répondent à différents objectifs de test, de la vérification fonctionnelle de base aux tests de bout en bout complexes.

- **Synthèse des données de test** : L'IA générative peut créer des données de test synthétiques représentatives et respectueuses de la confidentialité des données, qui ressemblent aux données de production et qui couvrent des situations extrêmes et des conditions de test variées. Ces données de test synthétiques peuvent être utilisées pour des tests fonctionnels et non fonctionnels. Les données de test générées par l'IA peuvent être adaptées aux exigences des applications, simulant des scénarios réalistes, sans exposer d'informations sensibles.

- **Génération de scripts de test automatisés** : L'IA générative peut générer des procédures de test manuelles et des scripts de test automatisés à partir de cas de test structurés, en interprétant les étapes de test et en les traduisant en code compatible avec divers frameworks d'automatisation des tests. Ces scripts de test peuvent être mis à jour ou étendus en fonction des nouvelles exigences.

- **Planification et hiérarchisation de l'exécution des tests** : L'IA générative peut analyser les cas de test et leurs interdépendances, optimisant ainsi les calendriers d'exécution des tests en fonction de la priorité, des risques associés, de la disponibilité des ressources et des objectifs des tests.

Objectif d'apprentissage pratique 2.2.2a (H2) : Pratiquer la génération de cas de test fonctionnel à partir de User Stories avec l'IA en utilisant l'enchaînement de prompts, les prompts structurés et les méta-prompts

Cet exercice se concentre sur le développement de cas de test fonctionnels à partir de User Stories avec l'IA générative, en utilisant l'enchaînement de prompts, les prompts structurés et les techniques de méta-prompting, pour garantir une couverture complète. La première étape consiste à créer un prompt qui

demande à l'IA de générer des cas de test fonctionnels basés sur des critères d'acceptation donnés et suivant un format de sortie spécifique. La deuxième étape consiste à vérifier l'exhaustivité des cas de test générés. Ici, le prompt vérifie que chaque critère d'acceptation est couvert en demandant à l'IA de générer un tableau récapitulant la couverture. Enfin, la troisième étape consiste à créer un méta-prompt pour faciliter la création de procédures de test de bout en bout. Ce méta-prompt permet d'affiner le prompt afin de générer des tests de bout en bout complets, encourageant ainsi les améliorations itératives pour maximiser l'efficacité.

Cet exercice améliore la compréhension de l'utilisation des LLMs pour la génération de cas de test, la validation de la couverture et les tests de bout en bout.

Objectif d'apprentissage pratique 2.2.2b (H2) : Utiliser la technique few-shot prompting pour générer des cas de test de style Gherkin à partir de User Stories données.

Cet exercice consiste à utiliser le few-shot prompting pour générer des cas de test de style Gherkin à partir de User Stories données. En commençant par une revue d'exemples prédéfinis et de la syntaxe Gherkin, l'étape 1 consiste à sélectionner n exemples à inclure dans le prompt, chacun avec une User Story, des conditions de test et des cas de test attendus de type « étant donné – lorsque - alors » pour modéliser le résultat souhaité. Ce prompt est ensuite appliqué à une nouvelle User Story, générant des scénarios Gherkin qui reflètent les conditions de test d'origine. Si les résultats sont inexacts, le prompt ou les exemples doivent être remaniés.

Cet exercice permet d'acquérir de l'expérience dans l'application des techniques de few-shot prompting à des tâches réalistes de conception et d'implémentation de tests.

Objectif d'apprentissage pratique 2.2.2c (H2): Utiliser l'enchaînement de prompts pour hiérarchiser les cas de test au sein d'une suite de tests donnée, en tenant compte de leurs priorités et dépendances spécifiques

Cet exercice se concentre sur l'utilisation de l'IA générative pour améliorer la priorisation des cas de test dans une suite de tests donnée, avec une analyse des risques associés et des dépendances entre les cas de test. La session commence par un bref aperçu des différentes approches de test, telles que celles basées sur les risques, sur la couverture et sur les exigences, ainsi qu'une revue de la suite de tests donnée. Les participants réalisent ensuite la création de prompts afin de générer des plans de priorisation exploitables pour diverses stratégies de priorisation des tests. Les résultats du LLM basés sur le prompt et les données d'entrée fournies doivent être vérifiés manuellement afin de détecter toute erreur dans le raisonnement du LLM.

L'objectif de cet exercice est d'expérimenter l'IA générative sur des tâches de test qui nécessitent des capacités de raisonnement multicritères (ici, les différents risques et dépendances à prendre en compte pour la priorisation des cas de test).

2.2.3 Tests de régression automatisés avec l'IA générative

À mesure que chaque nouvelle itération ou release est terminée, le nombre de cas de test de régression à exécuter augmente généralement, ce qui en fait des candidats idéaux pour l'automatisation, en particulier dans les pipelines d'intégration continue / livraison continue (CI/CD), en raison de la fréquence élevée

d'exécution des tests. L'IA générative peut rationaliser ce processus en facilitant la création, la maintenance et l'optimisation des suites de tests de régression automatisés. En s'adaptant dynamiquement aux modifications du code et en effectuant des analyses d'impact, l'IA générative peut identifier les zones du logiciel les plus susceptibles d'être affectées par les modifications récentes, ce qui permet de concentrer les efforts de test de régression là où ils sont le plus nécessaires.

Voici quelques activités typiques de tests de régression automatisés et de reporting des tests qui peuvent être prises en charge par le prompting avec l'IA générative :

- **Implémentation de scripts de test automatisés basés sur des mots-clés** : Les LLMs peuvent être utilisés pour implémenter des scripts de test basés sur des frameworks d'automatisation des tests basés sur des mots-clés, où des mots-clés prédéfinis représentent des étapes de test courantes. L'IA générative peut cartographier ces mots-clés vers des cas de test spécifiques, générer des scripts de test et assister les testeurs et les ingénieurs en automatisation des tests dans leur travail.
- **Analyse d'impact et optimisation des tests** : L'IA générative peut être utilisée pour analyser les modifications apportées au code afin d'identifier les zones à haut risque, permettant ainsi de cibler les tests de régression là où ils sont le plus nécessaires.
- **Tests auto-réparateurs et adaptatifs** : L'IA générative peut être utilisée pour ajuster automatiquement les scripts de test afin de gérer les modifications mineures de l'interface utilisateur ou de l'API, évitant ainsi les défaillances inutiles dues à de petites modifications et garantissant la stabilité des suites de tests au fil du temps.
- **Reporting automatisé des tests et informations détaillées** : L'IA générative permet de générer des rapports de test détaillés et disponibles en temps opportun, avec des métriques de réussite, les défaillances et les informations clés, fournissant ainsi aux parties prenantes des tableaux de bord qui mettent en évidence les tendances des tests et offrent des informations prédictives sur les points de défaillance potentiels.
- **Amélioration des rapports de défauts et de l'analyse des causes racines** : L'IA générative peut prendre en charge la compilation automatique de rapports de défauts complets avec des logs de test, des captures d'écran et des données sur l'environnement de test.

Ces activités peuvent être appliquées à divers tests de régression fonctionnels et non fonctionnels. Cependant, les testeurs doivent être conscients que l'IA générative peut faire des erreurs. La sortie générée doit donc être soigneusement vérifiée, en fonction du risque associé (voir chapitre 3).

En outre, l'IA générative peut faciliter les tests de régression automatisés de bout en bout basés sur l'interface graphique et l'API, chacun présentant ses propres défis et solutions. Les tests de l'interface graphique deviennent souvent instables en raison des changements récurrents apportés à l'interface utilisateur. L'IA générative peut adapter automatiquement les scripts de test pour gérer des changements tels que les localisateurs dynamiques et les interactions modifiées, réduisant ainsi le besoin d'intervention manuelle. Les tests de régression des API sont confrontés à des défis tels que la modification des formats de requête/réponse, des points de terminaison et de l'authentification. L'IA générative peut adapter automatiquement les scripts de test aux spécifications API en constante évolution et générer diverses données de test, ce qui permet de maintenir une couverture complète et de réduire le besoin de mises à jour manuelles.

Objectif d'apprentissage pratique 2.2.3a (H2): Pratiquer le few-shot prompting pour créer et gérer des scripts de test basés sur des mots-clés

Cet exercice se concentre sur le développement et l'automatisation de scripts de test pour une application web donnée à l'aide d'un framework d'automatisation des tests de l'interface graphique. L'exercice est structuré en deux sections principales : l'automatisation des tests et le débogage des scripts de test. La première partie de l'exercice fournit des conseils sur la création d'une documentation pour une bibliothèque de mots-clés, la génération de scripts de test initiaux, la validation de ces scripts de test par l'IA et l'extension de la couverture avec des scripts de test supplémentaires. La deuxième partie met l'accent sur le soutien au débogage, en utilisant des prompts système pour créer un assistant IA capable de vérifier et de corriger les scripts de test.

Cet exercice combine l'automatisation traditionnelle des tests avec la validation assistée par l'IA, démontrant comment le few-shot prompting peut être utilisé efficacement pour créer, maintenir et déboguer des scripts de test basés sur des mots-clés.

Objectif d'apprentissage pratique 2.2.3b (H2): Pratiquer la rédaction de prompts structurés pour l'analyse de rapports de test dans le contexte des tests de régression

Cet exercice illustre une approche méthodique de l'analyse des rapports de tests de régression, à l'aide de prompts structurés. Le processus commence par une analyse des résultats des tests fournis et une comparaison avec la spécification des tests. Il se poursuit ensuite par le regroupement des défauts similaires, la maintenance d'une liste d'anomalies connues et une vérification croisée des constatations. Chaque étape est liée à la suivante dans une conversation LLM s'enchaînant.

L'approche étape par étape montre comment des prompts structurés peuvent être utilisées pour transformer les résultats des tests de régression et les logs de test en informations exploitables, soutenant ainsi une analyse efficace des rapports de test dans le contexte des tests de régression.

2.2.4 Suivi des tests et contrôle des tests avec l'IA générative

Les tâches de suivi des tests nécessitent la récupération de grandes quantités de données (parfois non structurées), qui sont souvent déjà disponibles dans des outils de gestion des tests que l'IA générative peut aider à analyser et à synthétiser.

L'IA générative facilite un certain nombre de tâches de suivi et de contrôle des tests, notamment :

- **Suivi des tests et analyse des métriques :** L'IA générative peut faciliter l'automatisation du suivi des tests, ainsi que l'analyse des tendances afin de prévoir les risques potentiels et d'alerter les équipes en cas d'écart par rapport au plan. Les équipes restent ainsi informées et peuvent prendre les mesures nécessaires pour maintenir les normes de qualité.
- **Contrôle des activités de test :** L'IA générative peut faciliter le contrôle des tests en fournissant des informations permettant de redéfinir les priorités, d'ajuster le calendrier des tests et de réaffecter les ressources si nécessaire. Cela garantit que les tests restent flexibles et axés sur les domaines hautement prioritaires.
- **Informations sur la clôture des tests et apprentissage continu :** L'IA générative peut aider en générant des rapports de clôture de test, mettant en évidence les réussites et les leçons apprises.

Cela permet aux équipes d'affiner leurs stratégies de test et d'améliorer les processus de test futurs.

- **Visualisation et reporting améliorés des métriques de test :** L'IA générative peut aider à créer des tableaux de bord dynamiques et des résumés en langage naturel, garantissant ainsi que toutes les parties prenantes ont accès aux métriques pertinentes. Cette aide fournit les informations nécessaires pour prendre des décisions rapides et offre une vision claire de la progression des tests.

Objectif d'apprentissage pratique 2.2.4 (H0): Observer les métriques de suivi des tests préparées par l'IA à partir des données de test

Cette démonstration illustre comment l'IA générative peut aider les équipes de test en transformant les données de test en métriques de suivi des tests exploitables, facilitant ainsi la prise de décisions éclairées. À partir des données de test extraites des outils de test, un LLM les traite pour générer des métriques clés telles que la progression des tests, les tendances des défauts ou la couverture, mettant en évidence les risques potentiels. Ces métriques générées par l'IA peuvent ensuite être affichées sur un tableau de bord et résumées en langage naturel pour être facilement comprises par toutes les parties prenantes.

Cette démonstration illustre comment l'IA générative transforme les données de test en informations pratiques, aidant les équipes de test à suivre la progression des tests, à gérer la qualité et à s'adapter rapidement aux changements.

2.2.5 Choisir des techniques de prompting pour les tests logiciels

Le tableau suivant montre l'adéquation des trois techniques de prompting mentionnées à la section 2.1.2 en fonction des caractéristiques de la tâche de test.

Technique de prompting	**Cas d'utilisation recommandé**	**Principales caractéristiques et applications**
Enchaînement de prompts	Tâches complexes nécessitant une vérification humaine à chaque étape	Divise les tâches en étapes plus petites, utiles pour l'analyse des tests, la conception des tests et l'automatisation des tests, où chaque étape du test est vérifiée pour s'assurer de son exactitude.
Few-shot prompting	Tâches répétitives ou spécifiques/contraintes en matière de format de sortie	Fournit des exemples à l'IA générative pour la génération répétitive avec un canevas spécifique, par exemple dans un cas de test de style Gherkin (basé sur un scénario), des tests dirigés par les mots-clés ou le reporting des tests avec un format de sortie spécifique.
Méta-prompting	Tâches flexibles et dynamiques, utiles pour créer des prompts pour de nouvelles tâches	Description générale de l'objectif et de la tâche à accomplir, qui guide le LLM dans la création du prompt. Utile pour toutes sortes de tâches complexes

		telles que l'analyse de rapports de test et la détection d'anomalies.

Il est aussi possible d'utiliser plusieurs techniques pour un même cas d'utilisation. Par exemple, le méta-prompting peut être utilisé pour créer un prompt initial. Ce prompt généré peut contenir des exemples qui doivent être adaptés et peut être améliorés (few-shot prompting). Enfin, il peut être utile de diviser la tâche en sous-tâches plus petites afin de permettre la validation des étapes intermédiaires (enchaînement de prompts).

Objectif d'apprentissage pratique 2.2.5 (H1) : Sélectionner des techniques de prompting adaptées au contexte pour des tâches de test données

Cet exercice se concentre sur la sélection de techniques de prompting appropriées pour différentes tâches de test. Les participants se voient attribuer plusieurs tâches de test présentant différents niveaux de difficulté. Pour chaque tâche de test, les participants doivent évaluer la nature de la tâche (nécessite-t-elle de la précision ou une structure répétitive ?) et suggérer la ou les techniques de prompting les mieux adaptées au contexte afin de répondre aux besoins spécifiques de la tâche. Les choix sont discutés en groupe.

Cet exercice est conçu pour approfondir la compréhension de la manière dont différentes techniques de prompting peuvent être utilisées efficacement dans le cadre d'activités de test logiciel.

2.3 Évaluer les résultats de l'IA générative et affiner les prompts pour les tâches de test logiciel

L'évaluation des performances de l'IA générative dans le domaine des tests logiciels nécessite un ensemble clair de métriques permettant d'évaluer la qualité, la pertinence et l'efficacité des résultats générés (Li 2024). Ces métriques, qu'elles soient générales ou spécifiques à une tâche, contribuent à optimiser les prompts LLM.

2.3.1 Métriques pour évaluer les résultats de l'IA générative sur des tâches de test

Plusieurs métriques peuvent être utilisées pour évaluer la qualité et l'efficience des résultats de l'IA générative sur une tâche test :

Métrique	**Description**	**Exemple**
Exactitude	Mesure l'exactitude globale de la sortie générée par rapport à des cas de test rédigés par des experts, à des exigences ou à d'autres normes.	Le degré de couverture de toutes les exigences spécifiées par les cas de test générés.

Précision	Évalue l'exactitude de la sortie générée par rapport à un objectif spécifique.	Le degré auquel les cas de test générés identifient correctement les anomalies.
Rappel	Mesure la capacité d'un modèle à identifier toutes les instances pertinentes dans un ensemble de données.	Le degré auquel les cas de test générés couvrent la partition d'équivalence valide et invalide d'une classe de données.
Pertinence et adéquation contextuelle	Détermine si le résultat généré est applicable et approprié dans un contexte donné.	Le degré de cohérence des cas de test générés avec la base de test et l'intégration des exigences spécifiques au domaine.
Diversité	Garantit la prise en charge d'un large éventail d'entrées et de scénarios, évitant ainsi les répétitions.	Le degré auquel les cas de test générés couvrent divers comportements des utilisateurs et explorent les cas aux limites.
Taux de réussite des exécutions	Mesure la proportion de cas de test ou de scripts de test générés qui peuvent être exécutés avec succès.	Déterminer combien de scripts de test générés peuvent être exécutés sans erreurs de syntaxe ni problèmes de format de sortie dans un environnement de test qui fonctionne correctement.
Efficience temporelle	Évalue le temps gagné par rapport au test manuel.	Temps nécessaire à l'IA pour générer des cas de test par rapport au temps qu'il faudrait à un humain pour créer manuellement des tests équivalents.

En plus de ces métriques générales, des métriques spécifiques aux tâches peuvent être adaptées pour évaluer dans quelle mesure l'IA générative prend en charge des activités de test spécifiques.

Pour évaluer efficacement ces métriques, les testeurs peuvent effectuer des revues manuelles ou les automatiser, par exemple en comparant la sortie du LLM à une référence prédéfinie. Compte tenu de la nature non déterministe de l'IA générative, les métriques doivent être basées sur des données statistiquement pertinentes.

Objectif d'apprentissage pratique 2.3.1 (H0) : Observer comment les métriques peuvent être utilisées pour évaluer le résultat de l'IA générative sur une tâche de test

Au cours d'une démonstration sur une tâche de test donnée, des métriques adaptées à la tâche sont présentées pour évaluer les résultats de l'IA générative, ainsi que leur application concrète aux résultats obtenus avec un LLM sur cette tâche de test.

Cette démonstration illustre l'importance des métriques d'évaluation pour garantir la fiabilité des résultats de l'IA générative dans le domaine des tests logiciels.

2.3.2 Techniques d'évaluation et d'affinage itératif des prompts

Sur la base des métriques présentées ci-dessus, des techniques spécifiques d'évaluation et d'affinage sont utilisées pour améliorer les résultats de l'IA :

- **Modification itérative des prompts :** Commencer par un prompt de base et le modifier de manière itérative en fonction des résultats observés, en ajoutant progressivement plus de contexte ou en ajustant la formulation (par exemple en ce qui concerne la terminologie) afin d'améliorer la spécificité et la pertinence.
- **Tests A/B des prompts :** Création de plusieurs versions de prompts et évaluation pour détecter celui qui produit les meilleurs résultats en fonction de métriques prédéfinies. Cette approche permet de déterminer la formulation ou la structure de prompt qui produit les résultats les plus précis et les plus pertinents.
- **Analyse des résultats :** Examen des résultats générés par l'IA afin de détecter les inexactitudes ou les incohérences, par exemple par rapport à la base de test. La compréhension des types d'erreurs et d'incohérences peut aider à affiner les prompts afin d'éviter des défauts similaires lors des itérations suivantes.
- **Intégration des retours des utilisateurs :** Recueillir les commentaires des testeurs sur l'utilité et la clarté des résultats générés, par exemple en ce qui concerne le niveau de détail des tests générés. Analyser leurs commentaires et les utiliser pour affiner les prompts afin de mieux répondre aux besoins réels en matière de test.
- **Ajustement de la longueur et de la spécificité des prompts :** Essais de différentes longueurs de prompts et différents niveaux de détail. Parfois, l'ajout de contexte supplémentaire peut améliorer la qualité de la réponse. Dans d'autres cas, des prompts plus courts peuvent donner lieu à de meilleurs résultats.

En utilisant ces techniques, les équipes de test peuvent organiser des sessions d'évaluation et d'optimisation des prompts afin d'assurer l'amélioration continue du prompting de l'IA générative. Le partage des pratiques au sein de l'équipe de test ou de l'organisation de test permet non seulement de normaliser les techniques de prompting et de maintenir une qualité constante, mais aussi de promouvoir une culture d'apprentissage et d'amélioration itérative. Cette approche collaborative contribue à l'évolution des méthodologies de test de l'IA générative en permettant aux équipes de test : de s'appuyer sur des connaissances collectives, d'éviter les erreurs répétées et d'affiner plus efficacement leur utilisation des outils d'IA générative au fil du temps, par exemple en partageant des bibliothèques de prompts.

Objectif d'apprentissage pratique 2.3.2 (H1) : Évaluer et optimiser un prompt pour une tâche de test donnée

Cet exercice se concentre sur l'application de techniques d'optimisation des prompts à une tâche de test donnée. Les participants commenceront par un prompt initial et le remanieront de manière itérative afin d'améliorer les résultats générés par l'IA. Ils utiliseront des techniques telles que les tests A/B et la vérification humaine pour évaluer et améliorer la qualité des prompts. L'objectif est de permettre aux participants de découvrir comment le remaniement itératif conduit à une génération de cas de test plus efficace et plus pertinente dans le contexte.

À la fin de l'exercice, les participants auront effectué plusieurs itérations de remaniement des prompts et évalué chaque itération à l'aide des métriques discutées afin d'améliorer la qualité des résultats de l'IA.

3 Gérer les risques liés à l'IA générative dans les tests logiciels – 160 minutes

Mots-clés

confidentialité des données, sécurité, vulnérabilité

Termes de l'IA générative

Biais, erreur de raisonnement, hallucination, température

Objectifs d'apprentissage et objectifs d'apprentissage pratique pour le chapitre 3 :

3.1 Hallucinations, erreurs de raisonnement et biais

GenAI-3.1.1	(K1)	Rappeler les définitions des hallucinations, des erreurs de raisonnement et des biais dans les systèmes d'IA générative
GenAI-3.1.2	(K3)	Analyser les hallucinations, les erreurs de raisonnement et les biais dans les résultats des LLMs
HO-3.1.2a	(H1)	Expérimenter les hallucinations lors de tests avec l'IA générative
HO-3.1.2b	(H1)	Expérimenter les erreurs de raisonnement lors de tests avec l'IA générative
GenAI-3.1.3	(K2)	Résumer les techniques d'atténuation des hallucinations, des erreurs de raisonnement et des biais de l'IA générative dans les tâches de test logiciel
GenAI-3.1.4	(K1)	Rappeler les techniques d'atténuation pour le comportement non déterministe des LLMs

3.2 Confidentialité des données et risques de sécurité liés à l'IA générative dans les tests logiciels

GenAI-3.2.1	(K2)	Expliquer les principaux risques liés à la confidentialité des données et à la sécurité associés à l'utilisation de l'IA générative dans les tests logiciels
GenAI-3.2.2	(K2)	Donner des exemples de confidentialité des données et de vulnérabilités liées à l'utilisation de l'IA générative dans les tests logiciels
GenAI-3.2.3	(K2)	Résumer les stratégies d'atténuation visant à protéger la confidentialité des données et à renforcer la sécurité dans l'IA générative pour les tests logiciels
HO-3.2.3	(H0)	Reconnaître les risques liés à la confidentialité et à la sécurité des données dans une étude de cas de test avec l'IA générative

3.3 Consommation énergétique et impact environnemental de l'IA générative pour les tests logiciels

GenAI-3.3.1	(K2)	Expliquer l'impact des caractéristiques des tâches et de l'utilisation des modèles sur la consommation énergétique de l'IA générative dans les tests logiciels
HO-3.3.1	(H1)	Utiliser un simulateur pour calculer l'énergie et les émissions de CO_2 pour des tâches de test données avec l'IA générative

3.4 Règlements, normes et cadres de bonnes pratiques en matière d'IA

GenAI-3.4.1 (K1) Rappeler des exemples de réglementations, de normes et de cadres de bonnes pratiques en matière d'IA pertinents pour l'IA générative pour les tests logiciels

3.1 Hallucinations, erreurs de raisonnement et biais

Les systèmes d'IA générative, en particulier les LLMs, sont sujets à certains défauts, notamment des hallucinations, des erreurs de raisonnement et des biais. Ces défauts réduisent la qualité des résultats de l'IA générative lors des tâches de test, ce qui se traduit par la génération de testware qui ne répond pas aux attentes des testeurs. Ces hallucinations, erreurs de raisonnement et biais doivent être identifiés par les testeurs dans les résultats des LLMs, et des mesures doivent être prises pour atténuer ces risques.

Le comportement non déterministe des LLMs (voir section 1.1.2) rend difficile la correction de ce type de défauts ; ceux-ci peuvent sembler être corrigés pour une sortie générée par un LLM, mais réapparaître dans une autre conversation avec ce même LLM.

3.1.1 Hallucinations, erreurs de raisonnement et biais dans l'IA générative

Les **hallucinations** se produisent lorsqu'un LLM génère une sortie qui se révèle factuellement incorrecte ou non pertinente pour une tâche donnée. Dans les tests logiciels, les hallucinations produites par les LLMs peuvent se manifester sous forme de cas de test fictifs ou non pertinents, de la génération de scripts de test incorrects ou non fonctionnels, ou de la suggestion de cas de test qui vérifient des critères d'acceptation inexistants. Cela peut induire les testeurs en erreur et compromettre la validité des résultats des tests.

Les **erreurs de raisonnement** se produisent lorsque les LLMs interprètent mal les structures logiques, telles que les relations de cause à effet, la logique conditionnelle ou les processus de résolution de problèmes étape par étape ; ce qui conduit à des conclusions incorrectes. Contrairement aux humains, les LLMs ne disposent pas d'un véritable raisonnement logique et s'appuient sur la reconnaissance de motifs, ce qui peut conduire à des erreurs logiques lors de l'exécution de tâches telles que le raisonnement mathématique (Mirzadeh 2024). La planification des tests et la hiérarchisation des cas de test sont des exemples de tâches de test qui nécessitent un raisonnement logique et où les LLMs peuvent commettre des erreurs de raisonnement.

Les **biais** des LLMs (Gallegos 2024) proviennent des données sur lesquelles le modèle a été entraîné. Ces biais peuvent conduire à des résultats qui favorisent certains types d'informations, d'approches ou d'hypothèses. Par exemple, les LLMs entraînés principalement sur des données en anglais peuvent sous-représenter les perspectives non anglophones. Dans les tests logiciels, les biais peuvent influencer les réponses des LLMs, par exemple lors de la génération de données de test ou du remaniement des critères d'acceptation pour les cas de test.

Les hallucinations, les erreurs de raisonnement et les biais dans les résultats de l'IA générative résultent de la nature des données d'entraînement et des limites inhérentes au modèle Transformer (voir chapitre 1). Reconnaître et surmonter ces problèmes permet d'améliorer la qualité des résultats de l'IA générative dans les processus de test.

3.1.2 Identifier les hallucinations, les erreurs de raisonnement et les biais dans les résultats des LLMs

L'intégration efficace des systèmes d'IA générative dans les tests logiciels nécessite la capacité de détecter les hallucinations, les erreurs de raisonnement et les biais dans les résultats fournis par les LLMs. Selon le type de problème, différentes approches de détection peuvent être appliquées. Voici les approches courantes qui sont appliquées au moyen d'une revue humaine ou d'une combinaison de revue humaine et de vérification automatisée :

Détection des hallucinations :

- **Vérification croisée** : comparer les résultats générés par l'IA avec la documentation existante, les exigences et le comportement connu du système. Des outils automatisés peuvent aider à recouper les résultats avec des sources de données établies afin de signaler les divergences.
- **Consultation d'experts dans le domaine** : faites appel à des experts pour valider l'exactitude du contenu généré. Leur expertise est essentielle pour saisir les nuances que les systèmes automatisés pourraient négliger.
- **Contrôles de cohérence** : vérifier que les résultats générés sont cohérents entre eux et avec les informations connues. Les systèmes automatisés peuvent aider à identifier les motifs d'erreurs et à signaler les incohérences.

Détection d'erreurs de raisonnement :

- **Validation logique** : Évaluer le flux logique (par exemple, la cohérence, la cohésion et le raisonnement structuré dans le texte généré) du contenu généré par l'IA afin d'en vérifier la cohérence et l'exactitude au moyen de cycles de revue. Des outils automatisés peuvent être utiles, mais les cas complexes peuvent nécessiter un jugement humain.
- **Test des résultats** : par exemple, exécution des cas de test ou des scripts de test générés sur les objets de test afin de vérifier les résultats du test. Cette opération peut être partiellement ou entièrement automatisée, selon le type de testware généré.

Détection de biais :

- Revoir comment les testware générés, tels que les données de test synthétiques, sont représentés de manière équitable et précise par rapport à la stratégie de test.

Évaluer les biais liés aux types de tests, tels que la sous-représentation des tests non fonctionnels dans les résultats générés par le LLM. L'implémentation effective de ces méthodes de détection dépendra du niveau de risque estimé d'hallucinations, d'erreurs de raisonnement ou de biais dans la tâche de test effectuée avec l'IA générative.

Objectif d'apprentissage pratique 3.1.2a (H1) : Expérimenter les hallucinations de l'IA générative liées à une tâche de test logiciel

Cet exercice se concentre sur l'expérimentation d'exemples d'hallucinations de l'IA générative en relation avec l'ensemble des connaissances en matière de test logiciel. Les participants confrontent au moins deux LLMs à une situation dans laquelle les LLMs inventent des éléments non pertinents, par exemple en ajoutant des critères sans rapport qui n'existent pas dans les données contextuelles fournies. Différents prompts sont testés afin d'examiner leur influence sur les hallucinations.

Cet exercice permet de mieux comprendre comment identifier les hallucinations de l'IA générative lors des tests logiciels.

Objectif d'apprentissage pratique 3.1.2b (H1) : Expérimenter les erreurs de raisonnement de l'IA générative dans une tâche de planification des tests

Cet exercice vise à présenter un exemple d'erreur de raisonnement de l'IA générative. Il s'agit d'un exemple de problème à résoudre dans le domaine de la planification des tests, tel que l'estimation de l'effort de test et la hiérarchisation des cas de test (voir [ISTQB_CTFL] - Chapitre 5). L'exercice est conçu avec une certaine complexité des données d'entrée, ce qui nécessite des compétences en résolution de problèmes et met en évidence les limites des LLMs pour cet objectif. Le résultat du LLM sera comparé au résultat exact qui devrait être obtenu. Trois types de LLM différents seront essayés (LLM, SLM et modèle de raisonnement), et des variations du prompt seront utilisées pour tenter d'améliorer les résultats.

Cet exercice permet de mieux comprendre comment identifier les erreurs de raisonnement de l'IA générative dans les tâches de test logiciel qui nécessitent des compétences en résolution de problèmes logiques.

3.1.3 Techniques d'atténuation des hallucinations, des erreurs de raisonnement et des biais de l'IA générative dans les tâches de test logiciel

Afin de minimiser les résultats non souhaités de l'IA générative dans les tests logiciels, plusieurs stratégies peuvent être employées pour réduire les hallucinations, les erreurs de raisonnement et les biais. Ces problèmes sont plus susceptibles de se produire lorsque les prompts ne sont pas correctement conçus (voir chapitre 2) ou lorsque les données d'entrée contextuelles pertinentes font défaut pour une tâche de test donnée. Les techniques clés pour atténuer les risques associés aux hallucinations, aux erreurs de raisonnement et aux biais de l'IA comprennent :

- **Fournir un contexte complet** : veiller à ce que le prompt contienne toutes les informations pertinentes (voir section 2.1.1), en offrant un contexte complet pour guider l'IA dans la production de résultats précis.
- **Diviser les prompts en segments gérables** : diviser les prompts complexes en étapes plus petites à l'aide de techniques d'enchaînement de prompts (voir section 2.1.2), en vérifiant systématiquement chaque sortie avant de passer à la suivante. Cette approche étape par étape peut aider à la détection d`erreurs de raisonnement dès le début du processus de génération.
- **Utiliser des formats de données clairs et interprétables** : éviter les formats qui peuvent être ambigus ou difficiles à interpréter pour l'IA générative. Des formats structurés et simples aident le modèle à se concentrer sur les aspects essentiels de la tâche.
- **Sélectionner le modèle d'IA générative approprié à la tâche** : utiliser un LLM spécialement entraîné pour la tâche à accomplir (voir section 5.1.3).

- **Comparer les résultats entre les modèles** : lorsque cela est approprié, évaluer le prompt avec plusieurs LLMs et comparer les résultats permet de détecter les erreurs et de sélectionner les résultats les plus fiables.

Le chapitre 4 présente deux techniques complémentaires permettant d'améliorer les résultats du LLM : la génération augmentée par la recherche (Retrieval-Augmented Generation) et le réglage fin (Fine-Tuning).

3.1.4 Atténuation du comportement non déterministe des LLMs

Le comportement non déterministe inhérent aux LLMs (Shuyin 2023) peut entraîner des variations dans les résultats, même lorsque la même entrée est fournie. Cela résulte des processus d'échantillonnage probabilistes utilisés lors de l'inférence. Par conséquent, il peut être difficile d'obtenir des résultats cohérents et reproductibles lors de l'utilisation des LLMs, en particulier pour les résultats longs, ce qui augmente le risque de variabilité.

Bien qu'il soit impossible de garantir une reproductibilité totale, certaines stratégies peuvent contribuer à réduire la variabilité :

- **Ajuster les paramètres de température du LLM** : abaisser la température pendant la génération de réponses (inférence) réduit la distribution de probabilité, ce qui diminue le caractère aléatoire et permet d'obtenir des résultats plus cohérents. Cependant, cela limite également la créativité et la diversité des réponses, rendant les résultats plus répétitifs ou trop déterministes.
- **Définition de graines aléatoires** (NDT : random seeds) : certaines implémentations LLM permettent de définir une valeur de graine pour le générateur de nombres aléatoires, garantissant ainsi l'utilisation de la même séquence pseudo-aléatoire (c'est-à-dire des valeurs aléatoires déterministes), ce qui améliore la reproductibilité.

Réduire le risque d'hallucinations et d'erreurs de raisonnement dans les résultats du LLM implique de remédier à ce comportement non déterministe, par exemple en automatisant certains aspects de la vérification des résultats de façon à mettre en place un processus d'évaluation structuré et cohérent.

3.2 Confidentialité des données et risques de sécurité liés à l'IA générative dans les tests logiciels

L'IA générative dans les tests introduit des risques liés à la confidentialité et à la sécurité des données en raison du traitement d'informations sensibles et des vulnérabilités potentielles de l'infrastructure de test alimentée par LLM. Une protection robuste des données est essentielle pour prévenir les violations, les accès non autorisés et l'exposition de données confidentielles.

3.2.1 Confidentialité des données et risques de sécurité liés à l'utilisation de l'IA générative

L'IA générative peut traiter de grandes quantités de données susceptibles de contenir des informations sensibles ou personnelles. Cela soulève les questions suivantes sur la confidentialité des données :

- **Exposition involontaire des données** : les modèles d'IA générative peuvent générer des résultats qui révèlent accidentellement des informations sensibles.

- **Manque de contrôle sur l'utilisation des données** : les outils d'IA générative peuvent stocker et traiter des données sensibles sans le consentement explicite ou le contrôle de l'utilisateur. Cela peut entraîner une utilisation abusive ou un accès non autorisé.
- **Risques liés à la conformité** : l'utilisation d'outils d'IA générative sans respecter les réglementations en matière de protection des données, telles que le règlement général sur la protection des données (RGPD, règlement (UE) 2016/679), pourrait entraîner des litiges juridiques.

De plus, des risques de sécurité spécifiques apparaissent lors des tests avec l'IA générative, tels que :

- Les infrastructures de test basées sur les LLMs peuvent être vulnérables aux attaques de sécurité, telles que les violations de données ou les accès non autorisés.
- Les acteurs malveillants peuvent exploiter les vulnérabilités des LLMs, comme les attaques manipulatrices (voir section 3.2.2), pour modifier leur comportement ou extraire des informations sensibles.
- Les attaquants peuvent introduire intentionnellement des données d'entrée malveillantes afin de tromper les LLMs et compromettre leur précision ou leur sécurité.

3.2.2 Confidentialité des données et vulnérabilités dans l'IA générative pour les processus et outils de tests

Le tableau suivant donne quelques exemples de vecteurs d'attaque dans les processus et outils de test de l'IA générative.

Vecteur d'attaque	Description	Exemple
Exfiltration de données	Prompts conçus pour extraire des données confidentielles d'entraînement.	Dépasser la fenêtre contextuelle du LLM avec de longs prompts afin de surcharger la mémoire de l'IA pourrait l'amener à révéler des extraits aléatoires de ses données d'entraînement et potentiellement exposer des informations sensibles.
Manipulation des prompts	Introduction de données qui perturbent les résultats de l'IA.	Images qui induisent l'IA dans un contexte différent, provoquant ainsi des hallucinations, par exemple sur les critères d'acceptation.
Contamination des données	Manipulation des données d'entraînement.	Fournir de fausses évaluations lors de la notation des résultats d'un rapport de test généré par l'IA.
Génération de code malveillant	Manipulation d'un LLM pour générer des portes dérobées (par exemple, des appels de commandes externes) pendant l'utilisation.	Génération de code pour ouvrir un canal de communication avec une adresse IP malveillante spécifique.

3.2.3 Stratégies d'atténuation pour protéger la confidentialité des données et renforcer la sécurité lors des tests avec l'IA générative

À mesure que l'IA générative se généralise, et compte tenu des risques inhérents, des réglementations et des normes apparaissent pour les atténuer (voir section 3.4.1).

Les réglementations en matière de protection des données telles que le RGPD ne restreignent pas explicitement les applications de l'IA générative, mais prévoient des garanties susceptibles de limiter ce qui peut être fait, notamment en ce qui concerne la légalité et les restrictions relatives aux finalités de la collecte, du traitement et du stockage des données.

Pour atténuer ces risques, les organisations doivent mettre en œuvre des mesures robustes de confidentialité des données, notamment :

- **Minimisation des données** : éviter le traitement des données sensibles sauf si la loi l'autorise et n'utiliser que la quantité nécessaire de données non sensibles lors des tests avec l'IA afin de réduire les risques liés à la confidentialité des données.
- **Anonymisation et pseudonymisation des données** : masquage ou remplacement des informations sensibles par des données non identifiables.
- **Stockage et transmission sécurisés des données** : implémentation d'un cryptage puissant et de contrôles d'accès.
- **Formation du personnel** : les organisations doivent mettre en place des programmes et des politiques de formation clairs afin de garantir une utilisation responsable des outils d'IA générative, de promouvoir des pratiques éthiques et d'atténuer les risques potentiels.

Des stratégies d'atténuation supplémentaires peuvent être envisagées lors de l'implémentation de l'IA générative pour les tests :

- **Revue systématique des résultats générés** : l'évaluation humaine est essentielle pour garantir la qualité et la précision des tâches de test basées sur l'IA générative.
- **Évaluation par comparaison avec un autre LLM** : cette méthode consiste à utiliser plusieurs LLMs sur une tâche donnée afin d'évaluer les résultats en comparant leurs réponses.
- **Choix d'un environnement sécurisé et opérationnel** : en fonction du niveau de confidentialité requis, les organisations peuvent opter pour différentes solutions sécurisées : utilisation d'une offre commerciale sécurisée proposée par un fournisseur LLM, exploitation du LLM dans un cloud sécurisé ou installation du LLM dans l'infrastructure de l'organisation.
- **Audits de sécurité et évaluations des vulnérabilités réguliers** : identification et correction des faiblesses des systèmes d'IA générative.
- **Rester informé des meilleures pratiques en matière de sécurité** : se tenir au courant des dernières directives et technologies en matière de sécurité.

Ces stratégies sont souvent complémentaires et leur combinaison est nécessaire pour garantir la sécurité des données lors de l'utilisation de l'IA générative. Il est fortement recommandé d'impliquer des ingénieurs en sécurité expérimentés, des conseillers juridiques, le directeur technique (CTO) ou le responsable de la sécurité des systèmes d'information (CISO), s'ils sont présents dans l'organisation.

Objectif d'apprentissage pratique 3.2.3 (H0) : Reconnaître les risques liés à la confidentialité et à la sécurité des données dans une étude de cas donnée sur tester avec l'IA générative

Cette démonstration illustre comment des risques liés à la confidentialité et à la sécurité des données peuvent survenir lors de l'utilisation de l'IA générative dans les tests logiciels. Les participants exploreront des études de cas afin d'identifier les menaces potentielles, telles que les vulnérabilités des modèles, l'accès non autorisé aux données ou l'utilisation malveillante des résultats générés. Ils exploreront des stratégies d'atténuation, notamment le traitement sécurisé des données, les contrôles d'accès robustes et les pratiques de surveillance de l'IA, tout en réfléchissant aux implications éthiques et pratiques.

À la fin, les participants comprendront les principes de confidentialité des données et apprendront à reconnaître et à traiter les risques de sécurité dans les conditions de test de l'IA générative.

3.3 Consommation énergétique et impact environnemental de l'IA générative dans les tests logiciels

Des études telles que (Luccioni 2024a) montrent que l'entrainement et l'utilisation des LLMs nécessitent la mise en œuvre intensive d'un grand nombre de ressources informatiques spécialisées. Les LLMs sont mis à disposition sous forme de services Web, et leur utilisation augmente la charge sur les appareils, les réseaux et les centres de données, ce qui entraîne une consommation d'énergie plus élevée.

3.3.1 L'impact de l'utilisation de l'IA générative sur la consommation d'énergie et les émissions de CO2

L'impact environnemental de l'IA générative ne doit pas être sous-estimé, car la consommation d'énergie augmente fortement avec l'utilisation. La complexité de la tâche et les ressources informatiques nécessaires influencent la consommation d'énergie. Par exemple, la génération d'une seule image à l'aide d'un modèle d'IA puissant peut consommer autant d'énergie que la recharge complète d'un smartphone, tandis que la génération de texte ne consomme qu'un faible pourcentage de la charge d'un smartphone (Heikkilä 2023).

Même s'il est difficile d'obtenir des données précises sur l'impact environnemental de l'IA générative (Luccioni 2024b), il est clair que ces opérations énergivores contribuent collectivement à des émissions de CO_2 importantes (Berthelot 2024). Si une seule recherche ou une seule tâche de génération de texte peut sembler négligeable, leur effet cumulé sur des millions d'utilisateurs à travers le monde entraîne une pression environnementale considérable.

L'adoption de bonnes pratiques, telles que la limitation des interactions inutiles avec les modèles, est essentielle pour atténuer les risques environnementaux posés par l'IA générative.

Objectif d'apprentissage pratique 3.3.1 (H1) : Utiliser un simulateur pour calculer l'énergie et les émissions de CO_2 pour des tâches de test données avec l'IA générative.

Cet exercice se concentre sur l'évaluation de la consommation d'énergie et des émissions de CO_2 associées à diverses tâches d'IA générative dans le domaine des tests logiciels. Les participants utiliseront des simulations pour calculer ces métriques et examiner comment les différentes caractéristiques des tâches et l'utilisation des LLMs affectent l'impact environnemental.

En observant comment différents facteurs influencent la consommation d'énergie et les émissions, les participants comprennent les principaux facteurs de consommation d'énergie avec les LLMs.

3.4 Réglementations, normes et cadres de bonnes pratiques en matière d'IA

L'IA générative transforme les tests logiciels en aidant les testeurs dans diverses tâches de test (voir chapitre 2). Cependant, ces opportunités s'accompagnent également de risques importants, tels que les erreurs de raisonnement, la confidentialité des données, les vulnérabilités et les impacts environnementaux (voir sections 3.1, 3.2 et 3.3). Pour faire face à ces risques, il convient de tenir compte des réglementations générales, des normes et des cadres de bonnes pratiques en matière d'IA.

3.4.1 Réglementations, normes et cadres relatifs à l'IA générative dans le domaine des tests logiciels

Le tableau ci-dessous donne une vue d'ensemble des principales directives relatives à l'utilisation de l'IA générative dans les tests logiciels :

Nom / Type	Description	Application dans les tests logiciels
ISO/IEC 42001:2023 Technologies de l'information – Intelligence artificielle – Système de management Type: Norme	Spécifie les exigences relatives à la gestion des systèmes d'IA au sein d'une organisation.	Garantit que l'IA générative utilisée lors des tests respecte les pratiques recommandées, favorisant ainsi la cohérence et la fiabilité.
ISO/IEC 23053:2022 Cadre pour les systèmes d'intelligence artificielle (IA) utilisant le machine learning Type: Norme	Fournit un cadre pour les processus du cycle de vie de l'IA, en mettant l'accent sur la sûreté et la transparence.	Fournit un cadre pour la qualité des données, la transparence et la sûreté lors de l'utilisation de l'IA générative pour tester.
EU AI Act	Établit un cadre juridique traitant des risques liés à l'IA,	Exige la conformité en matière de transparence, de responsabilité et

Type: Réglementation	classant les applications par niveau de risque. Source: (AI Act 2024)	d'atténuation des biais pour l'IA générative utilisée dans les tests.
NIST AI Risk Management Framework (US) Type: Framework	Propose des recommandations pour la gestion des risques liés à l'IA, en mettant l'accent sur l'équité, la transparence et la sécurité. Source: (NIST AI RMF 1.0)	Garantit l'équité et atténue les risques liés à l'IA générative, afin d'éviter les résultats de test biaisés.

À mesure que les technologies d'IA et leur cadre réglementaire continuent d'évoluer, il est impératif pour les organismes de test de se tenir informés des développements en matière de réglementations, de normes, de lois nationales et de cadres de bonnes pratiques, tels que ceux présentés dans ce tableau.

4 Infrastructure de test basée sur LLM pour les tests logiciels – 110 minutes

Mots-clés

infrastructure de test

Termes de l'IA générative

agent basé sur LLM, , base de données vectorielle, exploitation de grands modèles de langage (LLMOps), fine-tuning, Retrieval-Augmented Generation (RAG)

Objectifs d'apprentissage et objectifs d'apprentissage pratique pour le chapitre 4 :

4.1 Approches architecturales pour les infrastructures de test basées sur LLM

GenAI-4.1.1	(K2)	Expliquer les principaux composants architecturaux et concepts de l'infrastructure de test basée sur LLM
GenAI-4.1.2	(K2)	Résumer les principes de la technologie Retrieval-Augmented Generation
HO-4.1.2	(H1)	Expérimentation de la technologie Retrieval-Augmented Generation sur une tâche de test donnée
GenAI-4.1.3	(K2)	Expliquer le rôle et l'application des agents basés sur LLM dans l'automatisation des processus de test
HO-4.1.3	(H0)	Observation d'un agent basé sur LLM pour automatiser une tâche de test répétitive

4.2 Fine-Tuning et LLMOps : opérationnalisation de l'IA générative pour les tests logiciels

GenAI-4.2.1	(K2)	Expliquer le fine-tuning des modèles de langage pour des tâches de test spécifiques
HO-4.2.1	(H0)	Observer un exemple de processus de fine-tuning pour une tâche de test et un modèle de langage donnés
GenAI-4.2.2	(K2)	Expliquer les LLMOps et leur rôle dans le déploiement et la gestion des LLMs pour les tâches de test

4.1 Approches architecturales pour les infrastructures de test basées sur LLM

Les chatbots IA et les outils de test basés sur les LLMs sont deux types d'infrastructures de test utilisant les LLM (voir section 1.2.2).

Au-delà de l'architecture de base d'une infrastructure de test basée sur LLM (voir section 4.1.1), le Retrieval-Augmented Generation (voir section 4.1.2) et les architectures d'agents basés sur LLM (voir section 4.1.3) étendent les fonctionnalités et l'utilité des LLM dans les tests logiciels.

4.1.1 Composants architecturaux clés et concepts de l'infrastructure de test basée LLM

Une infrastructure de test basée LLM fait référence à un système qui intègre un LLM dans le processus de test logiciel afin d'améliorer l'automatisation, le raisonnement et la prise de décision. Contrairement à un chatbot IA traditionnel, qui se concentre principalement sur les interactions conversationnelles, un outil de test basé sur LLM est conçu pour prendre en charge les tests logiciels en traitant les prompts liés aux tests, par exemple en analysant les exigences, en générant des cas de test et en évaluant les résultats.

L'architecture type d'une infrastructure de test basée sur LLM suit une conception à plusieurs composants qui facilite une interaction sécurisée et efficace avec le LLM. L'architecture se compose de composants front-end et back-end, ainsi que de sources de données externes et d'un LLM intégré :

- Le front-end sert d'interface utilisateur où les testeurs interagissent avec le système en saisissant des requêtes ou des commandes.
- Le back-end traite les entrées utilisateur et gère les fonctions critiques telles que l'authentification, la récupération des données, la préparation des prompts et l'interaction avec le LLM.
- Le LLM, qui peut être hébergé en tant que service tiers (accessible via une API) ou en tant que modèle interne personnalisé, génère des réponses basées sur des prompts structurés.

Cette architecture va au-delà du modèle client-serveur traditionnel en intégrant des composants de traitement intelligents, tels que les LLMs et les back-ends multi-sources :

1. Le LLM n'est pas seulement un serveur, mais un composant de traitement intelligent qui interprète et raisonne en fonction des produits testés.
2. Contrairement aux chatbots basés sur des règles qui suivent des réponses scriptées, une infrastructure de test basée sur LLM génère des informations de test de manière dynamique à partir du contexte, tel que les exigences, le code ou les résultats des tests.
3. Le back-end intègre plusieurs sources de données, telles que :
    - Des bases de données relationnelles (pour les données structurées utilisées dans les tests, telles que les cas de test).
    - Des bases de données vectorielles (pour la recherche sémantique de contenus connexes à l'aide d'embeddings ; voir section 4.1.2).

4. Le back-end améliore les résultats bruts du LLM grâce à un post-traitement, garantissant ainsi que ses réponses correspondent aux conditions de test du processus de test avant de les présenter au front-end.

4.1.2 Retrieval-Augmented Generation

La technologie Retrieval-Augmented Generation (RAG) améliore les LLMs en intégrant des sources de données supplémentaires dans leur processus de génération de réponses (Zhao 2024), augmentant ainsi la pertinence et la précision de leurs résultats.

Le RAG combine des systèmes de recherche avec des modèles de langage pour générer des réponses adaptées au contexte. Lors du prétraitement, les documents volumineux sont divisés en petits morceaux (par exemple, de 256 à 512 tokens) afin de garantir une recherche ciblée et la compatibilité avec la fenêtre contextuelle du modèle. Chaque fragment est nettoyé, traité et encodé dans un vecteur de haute dimension (embedding) à l'aide de modèles pré-entraînés. Ces embeddings, qui peuvent être stockés dans des bases de données vectorielles, permettent une recherche efficace basée sur la similarité au moment de l'exécution (inférence). Un prompt utilisateur est encodé, les fragments pertinents sont récupérés sur la base de leur similarité sémantique, puis utilisés comme contexte pour le modèle de langage afin de générer une réponse fondée.

Une réponse pertinente est essentiellement une sortie générée par le modèle de langage qui est profondément enracinée dans des informations pertinentes, précises et adaptées au contexte, recueillies pendant le processus de recherche. Elle garantit que la réponse n'est pas seulement basée sur l'entraînement préexistant du modèle, mais également enrichie par des données précises pertinentes pour le prompt. Cette synergie entre la recherche et la génération améliore la précision et la pertinence des réponses, les rendant plus fiables et plus informatives pour l'utilisateur.

Dans la phase de traitement des prompts utilisateur, un système RAG fonctionne selon un processus en deux étapes :

1. Récupération : à partir d'un prompt utilisateur, le système récupère les informations pertinentes dans les bases de données vectorielles créées précédemment. Cette récupération repose généralement sur la similarité sémantique entre les embeddings du prompt et ceux des morceaux.
2. Génération : les informations récupérées sont ensuite transmises au LLM, qui génère une réponse combinant ses connaissances existantes et les données nouvellement acquises, ce qui permet d'obtenir un résultat plus précis et plus adapté au contexte.

Le RAG dans les tests logiciels permet à l'infrastructure de test basée sur LLM d'accéder aux sources de données de l'entreprise, telles que les bases de données, la documentation et les référentiels, afin de récupérer des informations contextuelles en temps réel, garantissant ainsi que les tâches de test, telles que l'analyse ou la conception des tests, sont alignées sur les dernières spécifications, exigences et données de test existantes.

Objectif d'apprentissage pratique 4.1.2 (H1) : Expérimenter la technologie Retrieval-Augmented Generation pour une tâche de test donnée

Cet exercice pratique se concentre sur l'application des techniques RAG à une tâche de test donnée. Les participants expérimenteront un système RAG en y intégrant des documents et observeront comment il génère des réponses plus ou moins précises à partir d'informations complexes. Les

participants compareront les résultats du LLM avec et sans RAG pour la tâche de test donnée. Cet exercice vise à identifier les forces et les limites du système RAG dans le traitement de différents types de tâches de test.

En examinant les données récupérées et les résultats générés, les participants comprendront mieux le rôle du RAG dans l'amélioration des processus de test basés sur LLM.

4.1.3 Le rôle des agents basés sur LLM dans l'automatisation des processus de test

Les agents basés LLM (Wang 2024) sont des applications d'IA générative spécialisées, alimentées par des LLMs et conçues pour le traitement semi-autonome ou autonome de tâches définies. À la base, ces agents s'appuient sur les LLMs pour la compréhension et la génération du langage naturel, complétées par la possibilité de traiter des instructions, de récupérer du contexte et de mener des actions intelligentes.

Contrairement aux chatbots IA traditionnels qui se concentrent uniquement sur les interactions question-réponse, les agents basés sur LLM peuvent effectuer des tâches ou « agir » en invoquant un ensemble prédéfini de fonctions, communément appelées « outils ». Cette capacité leur permet d'interagir avec des systèmes externes et de les manipuler, ce qui les rend très polyvalents dans l'exécution des tâches. Le degré d'autonomie des agents basés sur LLM peut varier :

- Les agents autonomes fonctionnent indépendamment, exécutant des tâches avec une intervention humaine minimale à l'aide de règles prédéfinies, d'un apprentissage par renforcement et de boucles de rétroaction adaptatives.
- Les agents semi-autonomes exécutent des tâches sous la supervision périodique d'un humain afin de garantir que le résultat correspond aux objectifs définis par l'utilisateur.

Les architectures multi-agents impliquent un système collaboratif dans lequel plusieurs agents, chacun ayant des rôles spécialisés, communiquent et se coordonnent pour résoudre des problèmes complexes plus efficacement qu'un seul agent. Cet effort coordonné entre plusieurs agents IA est appelé « orchestration ».

Dans les processus de test, les agents basés sur LLM peuvent automatiser les tâches de test en émulant le raisonnement et la prise de décision humains. Cependant, ces agents souffrent des mêmes problèmes d'hallucinations, d'erreurs de raisonnement et de biais observés lors de l'utilisation des LLMs (voir section 3.1). Ces agents peuvent produire des résultats incorrects ou trompeurs, ce qui peut affaiblir la fiabilité des processus de test automatisés. Ces risques peuvent être atténués en mettant en œuvre des procédures de vérification automatisées pour les résultats des agents ou en utilisant des agents semi-autonomes pour les tâches critiques.

Objectif d'apprentissage pratique 4.1.3 (H0) : Observer comment un agent basé LLM aide à automatiser une tâche de test répétitive

La démonstration se concentre sur une tâche de test effectuée par un agent basé sur LLM. Les données d'entrée passées à l'agent, son comportement et les résultats de ses actions seront présentés, afin d'illustrer les différents aspects de l'intégration de solutions basées sur des agents dans un processus de test.

Cette démonstration montre un exemple concret d'un agent basé LLM dans le contexte d'une tâche test.

4.2 Fine-tuning et LLMOps : opérationnalisation de l'IA générative pour les tests logiciels

Deux pratiques clés pour mettre en place et gérer une infrastructure de test basée sur LLM pour les tests comprennent le fine-tuning des LLMs et la gestion du pipeline opérationnel via LLMOps (Mailach 2024).

4.2.1 Fine-tuning des LLMs pour les tâches de test

Le fine-tuning adapte un modèle de langage pré-entraîné, tel qu'un LLM ou un petit modèle de langage (SLM, voir section 1.1.2), afin qu'il puisse effectuer des tâches spécifiques ou pour l'ajuster à des domaines particuliers (Parthasarathy 2024). Cela implique de poursuivre l'entraînement du modèle sur un ensemble de données ciblé, lui permettant ainsi d'acquérir des connaissances et des nuances spécifiques au domaine. Le fine-tuning améliore les performances du modèle pour des applications spécialisées, le rendant plus précis et plus pertinent pour le cas d'utilisation prévu.

Dans la pratique, le fine-tuning est approprié pour doter les LLMs génériques de capacités de raisonnement spécialisées pertinentes pour un domaine spécifique ou pour adopter un vocabulaire propre à ce domaine. Le fine-tuning peut également être appliqué à des modèles plus petits, appelés SLMs, qui sont moins gourmands en ressources. En affinant un SLM (fine-tuning), il est possible d'atteindre des niveaux de performance plus élevés pour des tâches spécifiques sans requérir la même charge informatique pour les LLMs. Cette comparaison met en évidence la flexibilité et l'efficience de l'utilisation des LLMs et des SLMs en fonction des exigences spécifiques de la tâche.

Par exemple, dans le domaine des tests logiciels, le fine-tuning peut permettre à un LLM ou à un SLM de générer des cas de test à partir de User Stories dans un format de sortie spécifique au contexte de l'organisation. En entraînant le modèle sur les User Stories de l'organisation et les cas de test correspondants, le modèle s'aligne sur le processus de test et la terminologie spécifiques à l'organisation.

Le fine-tuning d'un modèle d'IA générative pour les tests logiciels présente plusieurs défis :

- Éviter les résultats biaisés ou inexacts tout en veillant à utiliser des ensembles de données d'entraînement de haute qualité et spécifiques à la tâche.
- Atténuer le surajustement (le modèle devient trop spécialisé dans les données d'entraînement, ce qui nuit à ses performances sur des données nouvelles et inconnues) afin de maintenir la généralisation dans différents scénarios.
- Remédier à l'opacité (manque de transparence dans la manière dont un LLM prend ses décisions ou produit ses résultats) dans le raisonnement du modèle, car l'opacité complique le débogage et la validation.
- Gérer les ressources informatiques significatives requises pour le processus de fine-tuning (pour les LLMs).

Objectif d'apprentissage pratique 4.2.1 (H0) : Observer un exemple de processus de fine-tuning pour une tâche de test et un LLMSLM donnés

Cette démonstration montre les différentes étapes nécessaires au fine-tuning d'un LLM pour une tâche de test donnée. Elle commence par la sélection d'un LLM ou SLM approprié. Ensuite, un ensemble de données adapté à la tâche de test donnée est présenté. Puis, une solution type pour le processus de fine-tuning est présentée (par exemple, un framework de machine learning). Enfin, un prompt est envoyé au modèle affiné et la qualité de la sortie générée est discutée.

Cette démonstration du processus de fine-tuning LLM/SLM pour une tâche de test montre plusieurs aspects clés de ce processus et aborde en particulier la qualité des données d'entraînement.

4.2.2 LLMOps lors du déploiement et de la gestion des LLMs pour les tests logiciels

LLMOps, ou Large Language Model Operations, désigne l'ensemble des pratiques, outils et processus conçus pour rationaliser le développement, le déploiement et la maintenance des LLMs dans les environnements de production (Sinha 2024).

L'utilisation de l'IA générative dans les processus de test d'une organisation peut se faire de différentes manières, qui influenceront les décisions à prendre en matière de LLMOps. Voici trois approches possibles :

- **Utilisation d'un chatbot IA** : les principales considérations pour cette approche comprennent la gestion de la confidentialité des données et des risques de sécurité tout en optimisant les coûts. Les organisations peuvent utiliser des plateformes LLM-as-a-Service, si les garanties nécessaires sont fournies, ou déployer une infrastructure interne utilisant des LLMs sous licence open source pour un meilleur contrôle. Un audit rigoureux des garanties des fournisseurs ou des capacités internes est essentiel pour atténuer les risques liés à la confidentialité et à la sécurité des données (voir section 3.2) et pour garantir l'efficience opérationnelle.
- **Utilisation d'un outil de test doté de capacités d'IA générative** : cette approche soulève des considérations similaires à celles des chatbots IA, telles que la confidentialité des données, la sécurité et les coûts opérationnels. En outre, les organisations doivent évaluer la sécurité des données et les garanties de performance offertes par le fournisseur de l'outil de test. Ces outils de test complètent généralement les processus de test existants, qui nécessitent une analyse coûts-avantages et une évaluation des risques approfondies.
- **Développement en interne d'un outil de test basé sur l'IA générative** : cette approche met l'accent sur le contrôle global de la confidentialité des données et des risques de sécurité, ainsi que sur la planification minutieuse des coûts d'exploitation de l'IA, tels que les ressources informatiques, le stockage des données et la formation du personnel. Les organisations doivent également mettre en place des processus structurés pour valider et maintenir les développements spécifiques à l'IA générative. Le développement de solutions en interne nécessite une expertise dans l'implémentation et le déploiement d'une infrastructure de test basée sur LLM.

Ces approches ne s'excluent pas mutuellement, car une organisation peut utiliser un chatbot IA pour certaines tâches tout en développant des outils personnalisés pour d'autres. Elles peuvent donc être mises en œuvre simultanément en fonction des activités de test spécifiques concernées.

En outre, elles peuvent intégrer des technologies supplémentaires, telles que le RAG et le fine-tuning des LLMs/SLMs, afin d'améliorer l'efficacité et l'adaptabilité des processus de test avec l'IA générative.

5 Déploiement et intégration de l'IA générative dans les organisations de test – 80 minutes

Mots-clés

Aucun

Termes de l'IA générative

IA fantôme

Objectifs d'apprentissage et objectifs d'apprentissage pratique pour le chapitre 5 :

5.1 Feuille de route pour l'adoption de l'IA générative dans les tests logiciels

GenAI-5.1.1	(K1)	Rappeler les risques liés à l'IA fantôme
GenAI-5.1.2	(K2)	Expliquer les aspects clés à prendre en compte lors de la définition d'une stratégie d'IA générative pour les tests logiciels
GenAI-5.1.3	(K2)	Résumer les critères clés pour sélectionner les LLMs/SLMs pour les tâches de test logiciel dans un contexte donné
HO-5.1.3	(H1)	Estimer les coûts récurrents liés à l'utilisation de l'IA générative pour une tâche de test donnée
GenAI-5.1.4	(K1)	Rappeler les phases clés de l'adoption de l'IA générative dans une organisation test

5.2 Gérer le changement lors de l'adoption de l'IA générative pour les tests logiciels

GenAI-5.2.1	(K2)	Expliquer les compétences et les connaissances essentielles requises pour que les testeurs puissent travailler efficacement avec l'IA générative dans les processus de test
GenAI-5.2.2	(K1)	Rappeler les stratégies visant à développer les compétences en IA au sein des équipes de test afin de soutenir l'adoption de l'IA générative dans les activités de test
GenAI-5.2.3	(K1)	Reconnaître comment les processus de test et les responsabilités évoluent au sein d'une organisation de test lors de l'adoption de l'IA générative

5.1 Feuille de route pour l'adoption de l'IA générative dans les tests logiciels

Une stratégie de test avec l'IA générative doit prendre soigneusement en compte des aspects clés tels que les objectifs de test à atteindre, la sélection appropriée du LLM, les problèmes liés aux données d'entrée utilisées pour les prompts et la conformité aux normes et réglementations en matière d'IA. Sur la base de cette stratégie, l'organisation peut établir une feuille de route et suivre les progrès de l'intégration de l'IA générative dans les processus de test.

5.1.1 Risques liés à l'IA fantôme

L'IA fantôme peut entraîner des risques en matière de sécurité, de conformité et de confidentialité des données:

- Faiblesses en matière de sécurité de l'information et de confidentialité des données : les outils d'IA personnels peuvent présenter des lacunes en matière de sécurité, ce qui peut entraîner des violations de données.
- Conformité et questions réglementaires : L'utilisation d'outils d'IA non approuvés peut entraîner la non-conformité aux normes et réglementations de l'industrie (voir section 3.4.1), ce qui peut avoir des conséquences juridiques.
- Propriété intellectuelle vague : l'utilisation d'outils d'IA assortis d'accords de licence peu clairs peut exposer les utilisateurs de LLM à des litiges en matière de propriété intellectuelle, en particulier si des données protégées par le droit d'auteur sont traitées sans autorisation appropriée.

Une stratégie et des étapes pour l'intégration et le déploiement de l'IA générative peuvent aider les organisations à tester et à éviter les risques liés à l'IA fantôme.

5.1.2 Aspects clés d'une stratégie d'IA générative dans le domaine des tests logiciels

Pour implémenter avec succès une stratégie d'IA générative dans les tests, les organisations doivent examiner attentivement plusieurs facteurs clés afin de garantir une intégration fluide et des résultats optimaux.
Cela commence par la définition d'objectifs de test mesurables pour l'IA générative, tels que l'augmentation de la productivité des tests, le raccourcissement des cycles de test et l'amélioration de la qualité des tests. Le choix des LLMs appropriés est essentiel (voir section 5.1.3) et doit être aligné sur ces objectifs de test, tout en garantissant la compatibilité avec l'infrastructure de test existante et en répondant aux exigences de mise à l'échelle du système.

La qualité des données joue un rôle essentiel, car l'efficacité des tests basés sur LLM dépend de données d'entrée précises et pertinentes, protégées par des procédures de sécurité robustes. Il est donc essentiel de maintenir une qualité élevée des données d'entrée afin d'obtenir des résultats fiables.

Des programmes de formation complets devraient être proposés afin de garantir que les équipes de test disposent des compétences techniques et éthiques nécessaires pour utiliser efficacement les outils d'IA générative. Outre la formation, des métriques spécifiques devraient être collectées afin de mesurer l'efficacité des résultats de l'IA générative (voir section 2.3.1).

Pour garantir la conformité aux normes réglementaires et le respect des directives éthiques, les organisations doivent établir des directives pour l'utilisation de l'IA générative, y compris des règles relatives à l'utilisation des données sensibles, des obligations de transparence (par exemple, ce qui a été généré à l'aide de l'IA générative) et des contrôles de qualité avec revue des testware générés.

5.1.3 Sélectionner des LLMs/SLMs pour des tâches de test logiciel

Il existe une large gamme de LLMs/SLMs, chacun avec des capacités fonctionnelles différentes (par exemple, entrée multimodale, capacités de raisonnement), des caractéristiques techniques (par exemple, taille de la fenêtre contextuelle) et des types de licence (par exemple, commerciale ou open source). Bien qu'il existe de nombreux benchmarks pour évaluer les LLMs/SLMs pour des tâches telles que le traitement du langage naturel, la génération de code ou l'analyse d'images ; seuls quelques-uns sont spécifiquement axés sur les tâches de test logiciel (Wenhan 2024). Par conséquent, la sélection des LLMs/SLMs pour les tâches de test nécessite un examen attentif de plusieurs critères clés :

- **Performances du modèle** : évaluer les performances du modèle pour les tâches de test ciblées par rapport aux benchmarks de l'organisation, à l'aide de métriques telles que celles présentées dans la section 2.3.1.
- **Potentiel de fine-tuning** : évaluer s'il est possible et utile d'affiner le modèle de langage (LLM ou SLM) à l'aide de données spécifiques au domaine afin d'améliorer les performances pour un cas d'utilisation donné, en augmentant la précision et la pertinence dans des contextes spécialisés.
- **Coût récurrent** : tenir compte des coûts récurrents liés à l'utilisation du LLM/SLM, y compris les coûts de licence et les dépenses opérationnelles, afin de s'assurer qu'ils correspondent au budget de l'organisation pour les tâches de test ciblées.
- **Communauté et assistance** : choisir des modèles bénéficiant d'une communauté active et d'une documentation détaillée pour faciliter l'implémentation et le dépannage.

En évaluant soigneusement ces critères, les organismes de test peuvent sélectionner un ou plusieurs LLMs/SLMs qui répondent à leurs besoins spécifiques et à leurs contraintes organisationnelles.

Objectif d'apprentissage pratique 5.1.3 (H1) : Estimation des coûts récurrents liés à l'utilisation de l'IA générative pour une tâche de test donnée

Cet exercice se concentre sur l'estimation des coûts récurrents liés à l'utilisation de l'IA générative pour une tâche de test spécifique, sur la base de diverses hypothèses. Ces hypothèses incluent des facteurs tels que le nombre de tokens dans les données d'entrée et de sortie, les prompts utilisés et la fréquence de la tâche. Les modèles de tarification de plusieurs fournisseurs de LLM/SLM seront explorés et comparés, y compris au moins une solution commerciale et un modèle sous licence open source.

Cet exercice permet de calculer et d'expérimenter les coûts récurrents de l'IA générative à l'aide d'hypothèses concrètes, ce qui aide à comprendre les implications financières des différentes approches et des différents fournisseurs.

5.1.4 Phases d'adoption de l'IA générative dans les tests logiciels

L'adoption de l'IA générative au sein d'une organisation de test implique trois phases clés :

1. **Découverte** : La première phase se concentre sur la sensibilisation et le renforcement des capacités. Les activités comprennent la formation des équipes de test aux concepts de l'IA générative, l'accès aux LLMs/SLMs et l'expérimentation de cas d'utilisation initiaux afin de familiariser les testeurs avec l'IA générative et de renforcer leur confiance.
2. **Initiation et définition de l'utilisation** : une fois les bases acquises, la deuxième phase consiste à identifier et à hiérarchiser les cas d'utilisation pratiques de l'IA générative dans les tests logiciels. Cette phase comprend l'évaluation d'une infrastructure de test basée sur LLM, le développement d'une expertise et la mise en adéquation avec les besoins de l'organisation (voir [ISTQB_CTFL_SYL] section 6).
3. **Utilisation et itération** : à ce stade avancé, les organisations intègrent pleinement l'IA générative dans leurs processus de test. Un suivi continu de la progression de l'IA générative pour les tests logiciels et les outils associés est mis en place, ainsi que la mesure et la gestion de la transformation afin de garantir des avantages durables et l'évolutivité et le passage à l'échelle.

Ces phases peuvent se dérouler en parallèle pour différents cas d'utilisation. Par exemple, l'analyse des rapports de test peut être plus avancée dans la feuille de route alors que l'automatisation des tests en est encore à ses débuts.

Il est également important de reconnaître et de traiter rapidement les préoccupations telles que la crainte de perdre son emploi, qui peuvent avoir un impact sur l'adoption et le moral de l'équipe.

5.2 Gérer le changement lors de l'adoption de l'IA générative pour les tests logiciels

La mise en œuvre réussie de l'IA générative dans une organisation de test nécessite une approche structurée des processus de gestion du changement. Les aspects clés comprennent le développement des compétences essentielles en IA générative et l'évolution des rôles traditionnels du test afin d'intégrer les processus de test basés sur l'IA. La transformation implique à la fois des compétences techniques et des aspects organisationnels.

5.2.1 Compétences et connaissances essentielles pour tester avec l'IA générative

L'intégration réussie de l'IA générative dans les tests nécessite la maîtrise des techniques d'ingénierie du prompting, la compréhension des fenêtres contextuelles des modèles et le développement de méthodes de revue des tests. Les testeurs doivent combiner leur expertise du domaine et des tests avec des compétences en IA, pour évaluer les tests basés sur le LLM dans des tâches telles que la génération de cas de test, l'analyse des rapports de défauts et la génération de données de test.

Les compétences clés comprennent l'évaluation des capacités des LLMs, la compréhension des techniques d'amélioration des prompts et l'évaluation des testware générés par l'IA. Les connaissances essentielles incluent la compréhension des risques inhérents à l'IA générative, ainsi que la connaissance des stratégies d'atténuation courantes. Les testeurs doivent comprendre les implications, en matière de sécurité des données, du partage de logiciels de test avec les LLMs ; mettre en

œuvre une expurgation appropriée des données (suppression ou masquage des informations sensibles, personnelles ou confidentielles) et suivre les pratiques d'ingénierie du prompting préservant la confidentialité des données.

Les considérations environnementales comprennent l'optimisation de la sélection des modèles et des canevas d'utilisation afin de réduire la charge informatique ; la sélection de modèles adaptés aux tâches de test et l'équilibre entre les avantages de l'automatisation de l'IA générative et son impact sur les coûts et la consommation d'énergie.

5.2.2 Développer des capacités d'IA générative au sein des équipes de test

Une approche pratique est essentielle pour former stratégiquement les équipes de test à l'IA générative. Cela comprend la pratique avec divers LLMs/SLMs, le suivi de chemins d'apprentissage structurés et le développement progressif d'un savoir-faire grâce au partage au sein de l'organisation. La formation est axée sur le développement de compétences pratiques grâce à des exercices guidés, l'apprentissage entre pairs et l'intégration progressive de l'IA dans les tâches de test quotidiennes.

Les membres de l'équipe de test passent de la maîtrise de la création de prompts de base à l'utilisation de techniques plus ciblées, telles que les prompts spécifiques aux tests. Un canevas de prompt est un template / gabarit réutilisable pour créer des prompts efficaces, afin de guider l'IA générative vers des résultats cohérents et fiables.

Des communautés de pratique internes soutiennent le partage continu des connaissances, avec des réunions régulières pour mettre en avant les applications réussies de l'IA générative, discuter des défis et affiner les meilleures pratiques. Ces communautés favorisent l'amélioration continue en partageant des bibliothèques de canevas de prompts et en documentant les leçons tirées de l'IA générative pour l'implémentation de tests dans différents projets et domaines.

5.2.3 Évolution des processus de test dans les organisations de test basées sur l'IA

L'intégration de l'IA générative transforme les processus de test traditionnels des testeurs et des responsables de tests, au sein des organisations de test.

Les testeurs évoluent de spécialistes de la conception et de l'exécution des tests, à des spécialistes des tests assistés par l'IA ; combinant leur expertise en techniques de test avec des compétences pour guider et vérifier les testware générés par l'IA. Leurs tâches de test s'étendent à la revue de l'ensemble des résultats basés sur l'IA, au remaniement des prompts et à la maintenance des bibliothèques de prompts spécifiques aux tests.

Les responsabilités des responsables des tests évoluent afin d'inclure le développement d'une stratégie de test basée sur l'IA, la gestion des risques basée sur l'IA, ainsi que le suivi et le contrôle des processus de test basés sur l'IA. Les responsables des tests s'attachent à équilibrer les capacités humaines et celles de l'IA, à établir des cadres de gouvernance de l'IA pour les cas d'utilisation et à veiller à ce que leurs équipes de test conservent à la fois leurs compétences traditionnelles en matière de test et leurs connaissances en matière d'IA. Les test managers ne se contenteront pas de diriger les testeurs humains, mais devront également coordonner les agents de test basés sur l'IA générative, ce qui nécessitera de nouvelles compétences en matière de management pour superviser des équipes hybrides composées d'humains et d'outils d'IA générative.

6 Références

Normes

- **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)

Documents ISTQB®

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

Référence au glossaire

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

Livres

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

Articles

- (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, and 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, and 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).

Pages 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. Accessed 24 Nov. 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. Retrieved from 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. Retrieved from https://www.nature.com/articles/d41586-024-00478-x

(Google Dev Glossary 2024) Google Developers. (n.d.). Machine learning glossary: Generative AI. Retrieved November 24, 2024, from 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. Accessed 24 Nov. 2024.

Les références précédentes renvoient à des informations disponibles sur Internet et ailleurs. Ces références ont été vérifiées au moment de la publication du présent syllabus, l'ISTQB® ne peut être tenu responsable si elles ne sont plus disponibles.

7 Annexe A – Objectifs d'apprentissage/Niveau de connaissance

Les objectifs d'apprentissage spécifiques à ce syllabus sont indiqués au début de chaque chapitre. Chaque sujet du syllabus sera examiné en fonction de l'objectif d'apprentissage qui lui est associé.

Les objectifs d'apprentissage commencent par un verbe d'action correspondant à son niveau cognitif, comme indiqué ci-dessous.

Niveau 1 : Se souvenir (K1)

Le candidat se souviendra, reconnaîtra et se rappellera d'un terme ou d'un concept.

Verbes d'action : identifier, rappeler, se souvenir, reconnaître.

Exemples
Rappeler les concepts de la pyramide des tests.
Reconnaître les objectifs habituels des tests.

Niveau 2 : Comprendre (K2)

Le candidat peut sélectionner les raisons ou les explications des affirmations liées au sujet, et peut résumer, comparer, classer et donner des exemples pour le concept de test.

Verbes d'action : classer, comparer, différencier, distinguer, exemplifier, expliquer, donner des exemples, interpréter, résumer.

Exemples	**Notes**
Classifier les outils de test en fonction de leur objectif et des activités de test qu'ils soutiennent.	
Comparer les différents niveaux de test.	Peut être utilisé pour rechercher des similitudes, des différences ou les deux.
Différencier le test du débogage.	Recherche les différences entre les concepts.
Distinguer les risques projet et les risques produit.	Permet de classer séparément deux (ou plusieurs) concepts.
Expliquer l'impact du contexte sur le processus de test.	

Exemples	Notes
Donner des exemples expliquant pourquoi il est nécessaire de tester.	
Déduire la cause racine des défauts à partir d'un profil de défaillances donné.	
Résumer les activités du processus de revue des produits d'activités.	

Niveau 3 : Appliquer (K3)

Le candidat peut exécuter une procédure lorsqu'il est confronté à une tâche familière, ou sélectionner la procédure correcte et l'appliquer dans un contexte donné.

Verbes d'action : appliquer, mettre en œuvre, préparer, utiliser

Exemples	Notes
Appliquer l'analyse des valeurs limites pour dériver des cas de test à partir d'exigences données.	Doit faire référence à une procédure, une technique, un processus, etc.
Implémenter des méthodes de collecte de métriques pour soutenir les exigences techniques et de management.	
Préparer des tests de facilité d'installation pour les applications mobiles.	
Utiliser la traçabilité pour surveiller l'avancement des tests afin de vous assurer qu'ils sont complets et cohérents avec les objectifs du test, la stratégie de test et le plan de test.	Peut être utilisé dans un LO qui souhaite que le candidat soit capable d'utiliser une technique ou une procédure. Similaire à « appliquer ».

Référence

(Pour les niveaux cognitifs des objectifs d'apprentissage)

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 Annexe B - Matrice de traçabilité des objectifs métiers avec les objectifs d'apprentissage

Cette section présente la traçabilité entre les objectifs métier et les objectifs d'apprentissage de la certification Testeur certifié Tester avec l'IA générative. Les objectifs d'apprentissage pratique ne sont pas mentionnés dans ce tableau, car chaque HO est associé à un seul LO. La traçabilité entre un HO et un BO s'effectue via le LO auquel le HO est associé.

Objectifs métier : Testeur certifié Tester avec l'IA générative			BO1	BO2	BO3	BO4	BO5
GenAI-BO1	Comprendre les concepts fondamentaux, les capacités et les limites de l'IA générative		8				
GenAI-BO2	Développer des compétences pratiques pour requêter de grands modèles de langage pour les tests logiciels			10			
GenAI-BO3	Mieux comprendre les risques et les mesures d'atténuation liés à l'utilisation de l'IA générative pour tester des logiciels				11		
GenAI-BO4	Mieux comprendre les applications des solutions d'IA générative pour tester les logiciels					19	
GenAI-BO5	Contribuer efficacement à la définition et à la mise en œuvre d'une stratégie et d'une feuille de route en matière d'IA générative pour les tests logiciels au sein d'une organisation						13

Objectifs métier : Testeur certifié Tester avec l'IA générative			BO1	BO2	BO3	BO4	BO5
LO	Objectifs d'apprentissage	Niveau K					
1	Introduction à l'IA générative pour les tests logiciels						
1.1	Fondements et concepts clés de l'IA générative						
GenAI-1.1.1	Rappeler différents types d'IA : IA symbolique machine learning classique, deep learning et IA générative	K1	X				
GenAI-1.1.2	Expliquer les bases de l'IA générative et des grands modèles de langage	K2	X				
GenAI-1.1.3	Faire la différence entre LLM de base, LLM adapté aux instructions et LLM de raisonnement	K2	X				
GenAI-1.1.4	Résumer les principes de base des modèles multimodaux de langage et des modèles de vision	K2	X				
1.2	Tirer parti de l'IA générative dans les tests logiciels : principes généraux						
GenAI-1.2.1	Donner des exemples de capacités clés des LLMs pour les tâches de test	K2	X			X	
GenAI-1.2.2	Comparer les modalités d'interaction lorsque vous utilisez l'IA générative pour tester des logiciels	K2	X			X	

Objectifs métier : Testeur certifié Tester avec l'IA générative			BO1	BO2	BO3	BO4	BO5
2	Ingénierie du prompting pour les tests logiciels						
2.1	Développement efficace des prompts						
GenAI-2.1.1	Donner des exemples de la structure des prompts utilisées dans l'IA générative pour les tests logiciels	K2		X			
GenAI-2.1.2	Différencier les principales techniques de prompting pour les tests logiciels	K2		X			
GenAI-2.1.3	Distinguer les prompts système des prompts utilisateur	K2		X			
2.2	Application des techniques d'ingénierie du prompting aux tâches de test logiciel						
GenAI-2.2.1	Appliquer l'IA générative aux tâches d'analyse de test	K3		X			
GenAI-2.2.2	Appliquer l'IA générative à la conception des tests et à l'implémentation des tests	K3		X			
GenAI-2.2.3	Appliquer l'IA générative aux tests de régression automatisés	K3		X			
GenAI-2.2.4	Appliquer l'IA générative aux tâches de contrôle et de suivi des tests	K3		X			

Objectifs métier : Testeur certifié Tester avec l'IA générative			BO1	BO2	BO3	BO4	BO5
GenAI-2.2.5	Sélectionner et appliquer les techniques de prompt appropriées à un contexte et à une tâche de test donnés	K3		X		X	
2.3	Évaluer les résultats de l'IA générative et affiner les prompts pour les tâches de test logiciel						
GenAI-2.3.1	Comprendre les métriques permettant d'évaluer les résultats de l'IA générative sur des tâches de test	K2		X	X	X	
GenAI-2.3.2	Donner des exemples de techniques permettant d'évaluer et de remanier de manière itérative les prompts	K2		X	X	X	
3	Gérer les risques liés à l'IA générative dans les tests logiciels						
3.1	Hallucinations, erreurs de raisonnement et biais						
GenAI-3.1.1	Rappeler les définitions des hallucinations, des erreurs de raisonnement et des biais dans les systèmes d'IA générative	K1	X		X	X	
GenAI-3.1.2	Analyser les hallucinations, les erreurs de raisonnement et les biais dans les résultats des LLMs	K3			X	X	
GenAI-3.1.3	Résumer les techniques d'atténuation des hallucinations, des erreurs de raisonnement et des biais de l'IA générative dans les tâches de test logiciel	K2			X	X	

Objectifs métier : Testeur certifié Tester avec l'IA générative			BO1	BO2	BO3	BO4	BO5
GenAI-3.1.4	Rappeler les techniques d'atténuation pour le comportement non déterministe des LLMs	K1	X		X	X	
3.2	Risques liés à la confidentialité et à la sécurité des données dans le domaine de l'IA générative pour les tests logiciels						
GenAI-3.2.1	Expliquer les principaux risques liés à la confidentialité des données et à la sécurité associés à l'utilisation de l'IA générative dans les tests logiciels	K2			X	X	
GenAI-3.2.2	Donner des exemples de confidentialité des données et de vulnérabilités liées à l'utilisation de l'IA générative dans les tests logiciels	K2			X	X	
GenAI-3.2.3	Résumer les stratégies d'atténuation visant à protéger la confidentialité des données et à renforcer la sécurité dans l'IA générative pour les tests logiciels	K2			X	X	
3.3	Consommation énergétique et impact environnemental de l'IA générative pour les tests logiciels						
GenAI-3.3.1	Expliquer l'impact des caractéristiques des tâches et de l'utilisation des modèles sur la consommation énergétique de l'IA générative dans les tests logiciels	K2			X	X	
3.4	Règlements, normes et cadres de bonnes pratiques en matière d'IA						
GenAI-3.4.1	Rappeler des exemples de réglementations, de normes et de cadres de bonnes pratiques en matière d'IA pertinents pour l'IA générative pour les tests logiciels	K1			X	X	X

Objectifs métier : Testeur certifié Tester avec l'IA générative			BO1	BO2	BO3	BO4	BO5
4	Infrastructure de test basée sur LLM pour les tests logiciels						
4.1	Approches architecturales pour les infrastructures de test basées sur LLM						
GenAI-4.1.1	Expliquer les principaux composants architecturaux et concepts de l'infrastructure de test basée sur LLM	K2				X	X
GenAI-4.1.2	Résumer les principes de la technologie Retrieval-Augmented Generation	K2				X	X
GenAI-4.1.3	Expliquer le rôle et l'application des agents basés sur LLM dans l'automatisation des processus de test	K2				X	X
4.2	Fine-Tuning et LLMOps : opérationnalisation de l'IA générative pour les tests logiciels						
GenAI-4.2.1	Expliquer le fine-tuning des modèles de langage pour des tâches de test spécifiques	K2				X	X
GenAI-4.2.2	Expliquer les LLMOps et leur rôle dans le déploiement et la gestion des LLMs pour les tâches de test	K2				X	X
5	Déploiement et intégration de l'IA générative dans les organisations de test						
5.1	Feuille de route pour l'adoption de l'IA générative dans les tests logiciels						

Objectifs métier : Testeur certifié Tester avec l'IA générative			BO1	BO2	BO3	BO4	BO5
GenAI-5.1.1	Rappeler les risques liés à l'IA fantôme	K1					X
GenAI-5.1.2	Expliquer les aspects clés à prendre en compte lors de la définition d'une stratégie d'IA générative pour les tests logiciels	K2					X
GenAI-5.1.3	Résumer les critères clés pour sélectionner les LLMs/SLMs pour les tâches de test logiciel dans un contexte donné	K2					X
GenAI-5.1.4	Rappeler les phases clés de l'adoption de l'IA générative dans une organisation test	K1					X
5.2	Gérer le changement lors de l'adoption de l'IA générative pour les tests logiciels						
GenAI-5.2.1	Expliquer les compétences et les connaissances essentielles requises pour que les testeurs puissent travailler efficacement avec l'IA générative dans les processus de test	K2					X
GenAI-5.2.2	Rappeler les stratégies visant à développer les compétences en IA au sein des équipes de test afin de soutenir l'adoption de l'IA générative dans les activités de test	K1					X
GenAI-5.2.3	Reconnaître comment les processus de test et les responsabilités évoluent au sein d'une organisation de test lors de l'adoption de l'IA générative	K1					X

9 Annexe C – Notes de livraison ("Release Notes")

Cette version est la V1.0. Aucune note de livraison pour cette première version.

10 Annexe D – Termes de l'IA générative

Terme en français	Terme en anglais	Définition
Chatbot IA	AI chatbot	Agent conversationnel qui utilise un LLM pour traiter les prompts et générer des réponses textuelles semblables à celles d'un humain, permettant ainsi une communication interactive avec les utilisateurs.
Fenêtre contextuelle	Context window	Étendue du texte, mesurée en tokens, qu'un modèle de langage prend en compte lors de la génération de réponses, influençant la pertinence et la cohérence de ses résultats.
Deep learning	Deep learning	Apprentissage automatique utilisant des réseaux neuronaux à plusieurs couches.
Embedding	Embedding	Technique utilisée pour représenter des tokens sous forme de vecteurs denses dans un espace continu, apprise pendant l'entraînement afin de capturer les relations sémantiques, syntaxiques et contextuelles.
Feature	Feature	Attribut individuel mesurable des données d'entrée utilisé pour l'entraînement par un algorithme ML et pour la prédiction par un modèle ML.
Few-shot prompting	Few-shot prompting	Technique consistant à fournir à un modèle quelques exemples dans le prompt afin de le guider dans la génération de réponses appropriées.
Fine-tuning	Fine-tuning	Processus d'apprentissage supervisé utilisant un ensemble de données d'exemples étiquetés pour mettre à jour les poids du LLM et les adapter à des tâches ou des domaines spécifiques.
LLM de base	Foundation LLM	Modèles polyvalents pré-entraînés sur un large éventail de données textuelles, capables de prédire le mot suivant sur la base de canevas linguistiques appris
IA générative	Generative AI (GenAI)	Type de système d'intelligence artificielle qui utilise des modèles de machine learning pour générer (de nouveaux) contenus intellectuels qui ressemblent à des contenus créés par l'homme.
Transformer génératif pré-entraîné	Generative pre-trained transformer (GPT)	Type de modèle de deep learning basé sur l'algorithme Transformer, pré-entraîné sur de vastes quantités de données textuelles afin de comprendre et de générer des textes semblables à ceux rédigés par des humains.

Hallucination	Hallucination	Informations incorrectes créées par un système basé LLM.
LLM adapté aux instructions	Instruction-tuned LLM	Un LLM de base entraîné à suivre des instructions, souvent renforcé par des commentaires afin d'encourager les bonnes réponses.
Grand modèle de langage LLM)	Large Language Model (LLM)	Programme informatique qui utilise de très grandes collections de données linguistiques afin de comprendre et de produire du texte d'une manière similaire à celle des humains.
Agent basé sur LLM	LLM-powered agent	Application qui intègre le raisonnement, la prise de décision et la mémoire du LLM, à l'aide d'outils permettant d'effectuer des tâches.
LLMOps	LLMOps	Pratiques et outils axés sur le déploiement, la surveillance et la maintenance des LLMs dans des environnements de production.
Machine Learning (ML)	Machine Learning (ML)	Processus utilisant des techniques informatiques pour permettre aux systèmes d'apprendre à partir de données ou d'expériences (ISO/IEC TR 29119-11).
Méta-prompting	Meta prompting	Élaboration d'instructions de niveau supérieur qui génèrent des prompts spécifiques pour explorer ou automatiser des capacités.
Modèle multimodal	Multimodal model	Modèles d'IA générative capables de traiter et de générer du contenu à partir de plusieurs types de données, telles que du texte, des images et de l'audio.
Traitement du langage naturel (NLP)	Natural Language Processing (NLP)	Traitement par des ordinateurs de données encodées en langage naturel afin de récupérer des informations et de représenter des connaissances.
One-shot prompting	One-shot prompting	Technique de prompting où le prompt contient un exemple pour guider la réponse du LLM.
Prompt	Prompt	Entrée en langage naturel fournie pour obtenir une réponse spécifique dans l'IA générative et les grands modèles de langage.
Enchaînement de prompts	Prompt chaining	Technique de prompting qui consiste à utiliser la sortie d'un prompt comme entrée pour un autre, créant ainsi une séquence de prompts.
Ingénierie du prompting	Prompt engineering	Processus de conception et de remaniement des prompts d'entrée afin de guider des LLMs vers la production des sorties souhaitées.

LLM de raisonnement	Reasoning LLM	Un LLM s'appuyant sur des modèles adaptés aux instructions en affinant leur capacité à émuler des processus de raisonnement humains.
Retrieval-Augmented Generation (RAG)	Retrieval-augmented generation	Technique combinant les capacités des LLMs avec un système de recherche permettant de récupérer des données pertinentes afin de générer des réponses précises et adaptées au contexte.
IA fantôme	Shadow AI	Utilisation d'outils ou de systèmes basés sur l'IA générative au sein d'une organisation sans autorisation, ni supervision officielles.
Petit modèle de langage (SLM)	Small Language Model (SLM)	Modèles de langage intentionnellement conçus et entraînés pour être petits, offrant un équilibre entre efficience et compréhension du langage spécifique à une tâche.
IA symbolique	Symbolic AI	Approche de l'IA qui utilise des symboles, des règles et des connaissances structurées pour modéliser un raisonnement.
Prompt système	System prompt	Ensemble d'instructions prédéfinies, généralement cachées aux utilisateurs du chatbot IA, qui établissent de manière cohérente le contexte, le ton et les limites des réponses d'une LLM et guident son comportement tout au long des interactions.
Température	Temperature	Paramètre qui contrôle le caractère aléatoire ou la créativité des résultats d'un LLM.
Tokenisation	Tokenization	Processus consistant à décomposer un texte en unités plus petites afin qu'il puisse être traité par des modèles de langage.
Transformer	Transformer	Architecture de modèle de deep learning qui utilise des mécanismes d'auto-attention pour capturer les dépendances à longue portée dans les séquences d'entrée.
Prompt utilisateur	User prompt	Prompt saisi par un utilisateur dans un grand modèle de langage qui oriente la réponse du modèle afin qu'il accomplisse des tâches spécifiques ou fournisse les informations souhaitées.
Base de données vectorielle	Vector database	Base de données optimisée pour le stockage et l'interrogation de représentations vectorielles de données à haute dimension.
Modèle de vision	Vision-language model	Système d'IA générative qui traite conjointement des données visuelles et textuelles afin d'effectuer des tâches en reliant et en générant du contenu entre les deux modalités.

Zero-shot prompting	Zero-shot prompting	Technique de prompting dans laquelle le prompt ne contient aucun exemple et s'appuie sur les connaissances préexistantes du modèle pour générer une réponse.

11 Annexe E – Marques déposées

ISTQB® est une marque déposée de l'International Software Testing Qualifications Board.

12 Index

Tous les termes relatifs aux tests sont définis dans le glossaire ISTQB® (http://glossary.istqb.org/).