

M1 CNS PARCOURS SYSTÈMES AUTONOMIQUES
TRAVAIL D'ÉTUDE ET DE RECHERCHE

ALIGNEMENT DES LLMS AVEC LES
GRAPHES DE CONNAISSANCES POUR LA
GÉNÉRATION D'EXPLICATIONS
ADAPTÉES



Présenté par :

MOKHTARI DHIA ELHAK

dhiaelhakmokhtari@gmail.com

LAKHMI BILLAL HICHEM

lakhmibilel12@gmail.com

OUHENIA ANIS

anisouhenia.pro@gmail.com

Encadrant :

HAMIDI MASSINISSA

Table des matières

1	Introduction	4
1.1	Objectif du Travail d'étude et de recherche	4
2	État de l'art	4
2.1	Les grands modèles de langage (LLMs) dans l'éducation	4
2.2	Représentations mentales en sciences cognitives et éducatives . .	5
2.3	Graphes de connaissances	6
2.4	Techniques d'alignement des LLMs	6
3	Méthodologie	7
3.1	Construction du graphe de connaissances	7
3.2	Modélisation des connaissances de l'apprenant	8
3.3	Alignment du LLM pour la génération d'explications adaptées .	9
4	Implémentation	10
4.1	Technologies et outils	10
4.2	Architecture du système	11
4.3	Fonctionnalités implémentées	11
5	Expérimentations et résultats	13
5.1	Protocole expérimental simulé	13
5.2	Résultats observés et comparatifs	15
5.3	Analyse qualitative	16
6	Discussion et limitations	16
6.1	Points forts	16
6.2	Limitations identifiées	17
7	Perspectives	17
7.1	Synthèse des contributions	17
7.2	Perspectives futures	18
8	Conclusion	19
	Références	20
	Glossaire	22
	Annexes	24

Abstract

Les grands modèles de langage (LLMs) transforment radicalement les méthodologies pédagogiques grâce à leur capacité à générer des contenus éducatifs dynamiques. Toutefois, leur utilisation optimale en éducation se heurte à une limitation fondamentale : leur incapacité à s'adapter aux représentations mentales individuelles des apprenants et à intégrer leurs prérequis spécifiques. Ce travail propose une approche innovante combinant un graphe de connaissances* pondéré et un LLM* pour pallier ces lacunes.

Notre système articule deux composantes synergiques :

Un graphe de connaissances modélisant les relations hiérarchiques entre concepts pédagogiques, enrichi de poids reflétant les dépendances de prérequis.

Un mécanisme d'orientation du LLM exploitant cette structure pour générer des explications et des quiz adaptés au profil cognitif de l'étudiant.

L'originalité de l'approche réside dans l'alignement dynamique entre la génération de contenu et l'état des connaissances de l'apprenant . Le système identifie les concepts maîtrisés, détecte les lacunes prérequisées, et guide le LLM pour produire des réponses pédagogiquement cohérentes et progressives.

Les résultats préliminaires suggèrent une amélioration significative de la pertinence contextuelle des sorties du LLM, notamment dans la création d'explications ciblées et d'exercices d'auto-évaluation adaptatifs. Cette fusion entre raisonnement symbolique (graphe) et apprentissage statistique (LLM) ouvre des perspectives prometteuses pour l'éducation personnalisée à grande échelle.

1 Introduction

Les grands modèles de langage (LLMs*) révolutionnent actuellement de nombreux domaines, dont celui de l'éducation. Leur capacité à générer du contenu cohérent et pertinent ouvre de nouvelles perspectives pour la création de ressources pédagogiques personnalisées. Cependant, malgré leurs performances impressionnantes, ces modèles présentent encore des lacunes significatives dans leur capacité à s'adapter aux différences individuelles des apprenants. L'un des défis majeurs dans l'utilisation des LLMs pour l'éducation réside dans leur incapacité native à prendre en compte la représentation mentale unique de chaque apprenant. Cette représentation, forgée par les expériences personnelles et les connaissances préalablement acquises, conditionne fortement la façon dont un nouvel apprentissage sera assimilé. De plus, les LLMs ne sont pas intrinsèquement conçus pour identifier et s'appuyer sur les prérequis spécifiques à chaque apprenant, ce qui limite considérablement la pertinence des contenus générés.

1.1 Objectif du Travail d'étude et de recherche

Ce travail vise à concevoir un assistant pédagogique intelligent capable de générer automatiquement des explications et des quiz adaptés au niveau réel de chaque étudiant. Pour cela, nous combinons deux composantes principales :

- un graphe de connaissances pondéré représentant les relations de pré-requis entre concepts.
- un grand modèle de langage (LLM) capable de produire du contenu éducatif personnalisé.

L'objectif est d'exploiter la structure du graphe pour guider le LLM dans la génération d'explications ciblées, en tenant compte de ce que l'étudiant connaît déjà (KNOWN) et des prérequis manquants. Ce système cherche à améliorer l'apprentissage en rendant les réponses du LLM plus cohérentes, progressives et pédagogiquement pertinentes.

2 État de l'art

2.1 Les grands modèles de langage (LLMs) dans l'éducation

Les LLMs comme GPT-4, Claude ou Mistral ont démontré des capacités impressionnantes dans la génération de texte cohérent et contextuel. Dans le domaine éducatif, ces modèles sont déjà utilisés pour diverses applications :

- Génération automatisée de questions et quiz
- Création de contenus pédagogiques adaptés à différents niveaux

- Production d'explications et d'exemples pour illustrer des concepts complexes
- Assistance à la rédaction et correction de travaux d'étudiants

Cependant, comme le soulignent [2] et [4], ces applications présentent encore des limitations importantes quant à la personnalisation des contenus. Les LLMs génèrent généralement des réponses basées sur des patterns statistiques sans véritable compréhension du niveau de connaissance spécifique de l'apprenant.



FIGURE 1 – Liste des LLMS les plus utilisés en 2024 (Source : Tech Radar)

2.2 Représentations mentales en sciences cognitives et éducatives

Les sciences cognitives ont largement étudié la façon dont les connaissances sont organisées dans l'esprit humain. Plusieurs théories et modèles sont particulièrement pertinents pour notre étude :

- **Les cartes conceptuelles*** : Développées par Novak et Gowin [5], elles représentent graphiquement les relations entre les concepts et aident à visualiser la structure de connaissance d'un individu.
- **La théorie des schémas*** : Proposée initialement par Bartlett [6] puis développée par d'autres chercheurs, elle suggère que les connaissances sont organisées en structures cognitives appelées schémas, qui facilitent la compréhension et l'assimilation de nouvelles informations.
- **Les modèles mentaux*** : Johnson-Laird [7] décrit comment les individus construisent des représentations internes des phénomènes qu'ils observent, influençant leur raisonnement et leur apprentissage.

Ces théories mettent en évidence l'importance d'identifier et d'exploiter les structures de connaissances préexistantes chez l'apprenant pour faciliter l'acquisition de nouvelles connaissances.

2.3 Graphes de connaissances

Les graphes de connaissances constituent une approche prometteuse pour modéliser formellement les structures de connaissances. Ils permettent de représenter les concepts et leurs relations sous forme de graphes orientés [8]. Dans le contexte éducatif, plusieurs travaux ont exploré l'utilisation des graphes de connaissances :

- **Knowledge Forest*** [9] : Utilise des graphes pour représenter les structures de connaissances et guider la génération de contenu pédagogique.
- **Intelligent Tutoring Systems*** : Des systèmes comme AutoTutor [10] intègrent des représentations de connaissances pour adapter leurs interactions aux besoins spécifiques des apprenants.
- **Curriculum Learning*** : L'organisation de l'apprentissage selon une progression logique basée sur les prérequis conceptuels [11].

Ces approches démontrent le potentiel des graphes de connaissances pour structurer et personnaliser l'apprentissage, mais leur intégration avec les capacités des LLMs reste un domaine relativement inexploré.

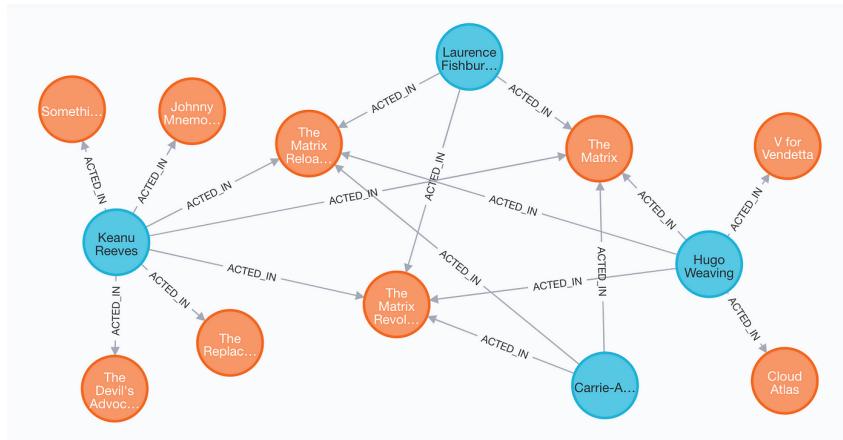


FIGURE 2 – Exemple d'un graphe de connaissances

2.4 Techniques d'alignement des LLMs

L'alignement des LLMs vise à faire correspondre leur comportement avec des objectifs ou contraintes spécifiques. Plusieurs techniques ont été développées récemment :

- **Prompt Engineering*** : Manipulation des instructions fournies au modèle pour orienter sa génération [12].
- **Fine-tuning*** : Ajustement des paramètres du modèle sur des données spécifiques au domaine ou à la tâche [13].
- **RLHF*** (Reinforcement Learning from Human Feedback) : Utilisation de feedback humain pour affiner le comportement du modèle [14].

- **Constitutional AI*** : Approche visant à aligner les modèles sur des principes ou contraintes spécifiques [15].

Dans le contexte éducatif, des travaux comme ceux de [16] ont commencé à explorer comment aligner les LLMs avec des objectifs pédagogiques, mais l'intégration avec des représentations individuelles des connaissances reste un défi majeur.

3 Méthodologie

Notre approche méthodologique s'articule autour de trois axes principaux : la construction d'un graphe de connaissances représentant les concepts du domaine et leurs relations, l'évaluation des connaissances de l'apprenant, et l'alignement du LLM pour générer des explications adaptées.

3.1 Construction du graphe de connaissances

Le graphe de connaissances constitue la structure fondamentale représentant le domaine d'apprentissage et les relations entre les concepts. Sa construction repose sur une approche semi-automatique combinant extraction automatique et validation humaine. Notre méthodologie s'appuie sur l'algorithme ACE* (AI-Assisted Construction of Educational Knowledge Graphs with Prerequisite Relations) [3], qui permet de générer des graphes orientés, pondérés et acycliques à partir de contenus pédagogiques textuels. Les étapes suivies sont les suivantes :

- **Extraction des concepts et des paires candidates** : Identification des concepts fondamentaux par analyse de curricula, manuels et ressources pédagogiques, suivie de la génération automatique de paires de concepts via ACE.
- **Évaluation sémantique des relations** : Chaque paire de concepts est évaluée selon deux scores :
 - **CSR*** (Concept Semantic Relatedness) : Mesure la proximité sémantique des concepts dans le texte.
 - **CER*** (Concept Embedding Relatedness) : Évalue leur proximité vectorielle dans un espace d'embedding*.
- **Filtrage et pondération des relations** : Les paires obtenant les scores les plus élevés sont sélectionnées comme relations de type *PRE-REQUISITE**, avec un poids compris entre 0 et 1 reflétant l'intensité du lien.

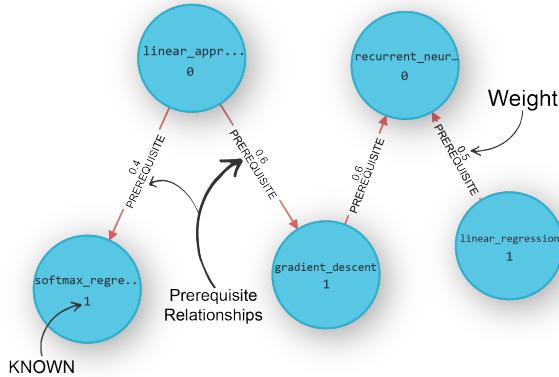


FIGURE 3 – Architecture du graphe de connaissances

- **Enrichissement par LLM** : Utilisation d'un LLM* pour suggérer des concepts ou relations supplémentaires, validés manuellement.
- **Validation et raffinement** : Vérification de la cohérence globale du graphe et de la pertinence des relations par des experts du domaine.

3.2 Modélisation des connaissances de l'apprenant

Pour adapter les explications au niveau de chaque apprenant, nous devons évaluer ses connaissances actuelles, en tenant compte de la force des relations entre les concepts dans le graphe de connaissances ainsi que de l'état de maîtrise de chaque concept. Chaque concept dans le graphe est marqué avec un flag **KNOWN***, qui peut avoir les valeurs suivantes :

- 0 : Le concept est inconnu.
- 1 : Le concept est maîtrisé.

En parallèle, les relations de prérequis entre les concepts sont associées à des poids compris entre 0 et 1, représentant la force de la dépendance entre les concepts. Ces poids permettent au système de prioriser les prérequis les plus importants dans le processus d'explication. Notre méthodologie comprend les étapes suivantes :

- **Évaluation diagnostique initiale** : Le système interroge le graphe de connaissances pour obtenir la liste des concepts inconnus (**KNOWN** = 0). Chaque concept est présenté à l'apprenant pour évaluer sa maîtrise. Si un concept est jugé maîtrisé (**KNOWN** = 1), il est mis à jour dans le graphe.
- **Annotation du graphe** : Le flag **KNOWN** est utilisé pour marquer le niveau de maîtrise de chaque concept (soit 0 pour inconnu, soit 1 pour maîtrisé). Lorsqu'un concept est maîtrisé, les concepts liés par des prérequis avec des poids élevés sont considérés comme plus urgents à abor-

der dans l'explication, tandis que ceux avec un poids faible peuvent être traités plus tard ou de manière optionnelle.

- **Mise à jour dynamique** : L'état de connaissance de l'apprenant est mis à jour à chaque interaction. Par exemple, un concept avec un flag `KNOWN` égal à 0 sera lié à des concepts ayant des prérequis avec des poids différents. Les prérequis avec un poids plus élevé seront prioritaires dans les explications, et le flag `KNOWN` des concepts sera ajusté en fonction des performances de l'apprenant.
- **Inférence de connaissances** : Le système utilise des règles d'inférence pour estimer la maîtrise des concepts non directement évalués. L'inférence prend en compte la valeur `KNOWN` des concepts liés et le poids des relations pour estimer si un concept non évalué est effectivement connu. Un concept avec un poids de relation élevé pourra être traité plus tôt, même s'il n'a pas encore été directement évalué.

3.3 Alignement du LLM pour la génération d'explications adaptées

L'objectif central est d'adapter les explications générées par le LLM au niveau de connaissance spécifique de l'apprenant. Notre approche d'alignement comprend :

- **Construction de prompts contextualisés** : Intégration des informations du graphe de connaissances et du profil de l'apprenant dans les instructions fournies au LLM.
- **Génération multi-étapes** : Utilisation d'une approche séquentielle où le LLM planifie d'abord l'explication en fonction des prérequis identifiés, puis génère le contenu correspondant.
- **Post-traitement basé sur les règles** : Application de règles de vérification pour s'assurer que l'explication n'utilise pas de concepts inconnus sans les expliquer.
- **Fine-tuning ciblé** : Adaptation du modèle sur un corpus d'explications annotées selon différents niveaux de connaissances préalables.

4 Implémentation

Notre implémentation repose sur une architecture modulaire intégrant plusieurs composants techniques. Cette section détaille les technologies utilisées et les choix d'implémentation.

4.1 Technologies et outils

Pour la réalisation de notre système, nous avons sélectionné les technologies suivantes :

- **Base de données de graphe Neo4j*** : Pour le stockage et la manipulation efficace du graphe de connaissances, permettant des requêtes complexes sur les relations entre concepts.
- **LLM local (Mistral)** : Déployé via Ollama* pour permettre une génération de contenu flexible et contrôlable, tout en limitant les coûts et les problèmes de confidentialité.
- **Framework Python** : Utilisation de bibliothèques comme LangChain* pour faciliter l'interaction avec le LLM et la construction de chaînes de traitement.
- **Interface utilisateur Chainlit*** : Création d'une interface conversationnelle permettant à l'apprenant d'interagir naturellement avec le système.
- **Intégration PyPDF2 et OCR** : Pour l'extraction de contenu à partir de documents PDF, permettant d'analyser les ressources pédagogiques et d'enrichir le graphe de connaissances.



FIGURE 4 – Technologies utilisées

4.2 Architecture du système

Le système se compose de plusieurs modules qui gèrent les connaissances de l'apprenant, les prérequis et leur poids, tout en utilisant les flags **KNOWN** pour ajuster les interactions :

- **Module de gestion du graphe de connaissances** : Responsable de la création et mise à jour du graphe stocké dans Neo4j, en marquant chaque concept avec un flag **KNOWN** et en associant des prérequis avec des poids compris entre 0 et 1.
- **Module d'évaluation des connaissances** : Ce module génère des questions pour évaluer la maîtrise des concepts. En fonction des réponses de l'apprenant, le flag **KNOWN** est mis à jour et les prérequis avec des poids élevés sont priorisés dans les explications suivantes.
- **Module d'analyse de documents** : Après le téléchargement des PDF par l'apprenant, le système extrait les concepts mentionnés et les compare avec l'état du graphe de connaissances. Les flags **KNOWN** et les poids des prérequis guident la génération des explications.
- **Module d'alignement du LLM** : Le LLM utilise le profil actualisé de l'apprenant, incluant les flags **KNOWN** et les relations de prérequis pondérées, pour générer des explications contextualisées et adaptées au niveau de l'utilisateur.
- **Interface conversationnelle** : Permet à l'apprenant d'interagir avec le système. Elle présente des explications personnalisées et recueille des retours pour ajuster le flag **KNOWN** de chaque concept en fonction des interactions et des performances.

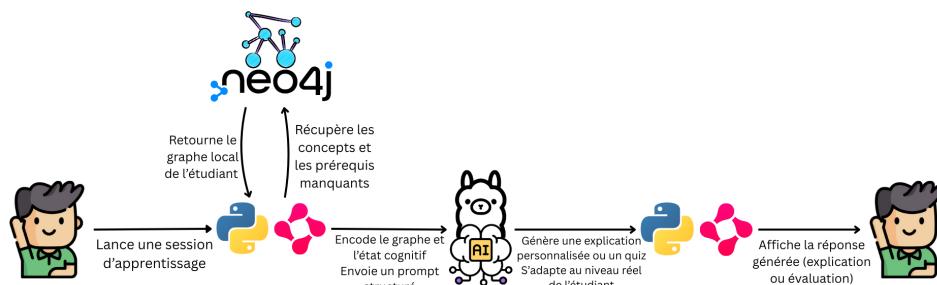


FIGURE 5 – Schéma global du système

4.3 Fonctionnalités implémentées

Notre système intègre plusieurs fonctionnalités concrètes permettant une adaptation dynamique des contenus générés à l'état cognitif de chaque apprenant :

- **Évaluation interactive des prérequis** : Le système interroge Neo4j pour identifier les concepts non encore maîtrisés (**KNOWN** = 0). L'ap-

nant est invité à répondre à des questions de type "connaissez-vous ce concept?", permettant de mettre à jour son graphe personnel.

- **Génération d'explications conditionnelles** : Lorsqu'un concept est inconnu, le système récupère ses prérequis pondérés dans le graphe et demande au LLM, via un prompt, d'expliquer d'abord ces prérequis avant de traiter le concept cible. Cela garantit une progression adaptée au profil de l'utilisateur.
- **Encodage textuel du graphe dans le prompt** : Les relations entre concepts extraits de Neo4j sont encodées sous forme de texte puis intégrées dans le prompt envoyé au LLM. Cette approche, inspirée de l'article *Talk Like a Graph**, améliore la qualité du raisonnement du modèle.
- **Few-shot prompting* simplifié** : Des exemples d'explications ou de quiz sont insérés dans certains prompts pour guider le LLM, améliorant ainsi la structuration et la clarté des contenus générés.

Pourquoi c'est utile ?

- Les LLMs peinent à raisonner sur des graphes complexes sans aide
 - Donner quelques exemples représentatifs améliore fortement leur performance, surtout pour des tâches comme l'identification des prérequis ou le raisonnement chaîné
 - L'effet est particulièrement positif lorsque les graphes sont encodés de façon claire et concise (ex. : encodage incident)[1]
- Des études montrent que combiner un bon encodage textuel du graphe avec du few-shot prompting augmente la précision de tâches de raisonnement jusqu'à +60% [1]
- **Génération automatique de quiz** : Après chaque explication, un quiz Vrai/Faux est automatiquement généré par le LLM* au format JSON. La validation de ce quiz permet de mettre à jour le statut du concept (KNOWN = 1) dans le graphe.
 - **Analyse de documents PDF** : L'apprenant peut charger un fichier PDF. Le système en extrait le contenu (texte ou via OCR), l'indexe avec Chroma, puis utilise un retriever pour poser des questions sur le contenu ou détecter les concepts abordés.
 - **Mise à jour du graphe basée sur les interactions** : Chaque réponse correcte à un quiz ou confirmation d'un concept connu déclenche une mise à jour dans Neo4j, modifiant dynamiquement le graphe de connaissances de l'apprenant.

5 Expérimentations et résultats

Bien que notre système soit encore en phase de développement, nous avons pu formuler des observations et comparer théoriquement les bénéfices attendus par rapport à un LLM standard. Cette section présente notre protocole expérimental envisagé, ainsi qu'une analyse comparative basée sur le fonctionnement interne du système et des cas simulés.

5.1 Protocole expérimental simulé

Pour évaluer notre approche, nous avons conçu un protocole expérimental simulé pouvant être réalisé avec un groupe d'apprenants dans un futur proche. Il s'articule autour des étapes suivantes :

- **Constitution du graphe de connaissances** : Construction d'un graphe orienté et pondéré dans un domaine ciblé (par exemple : machine learning), comprenant 35 concepts et plus de 200 relations de prérequis.

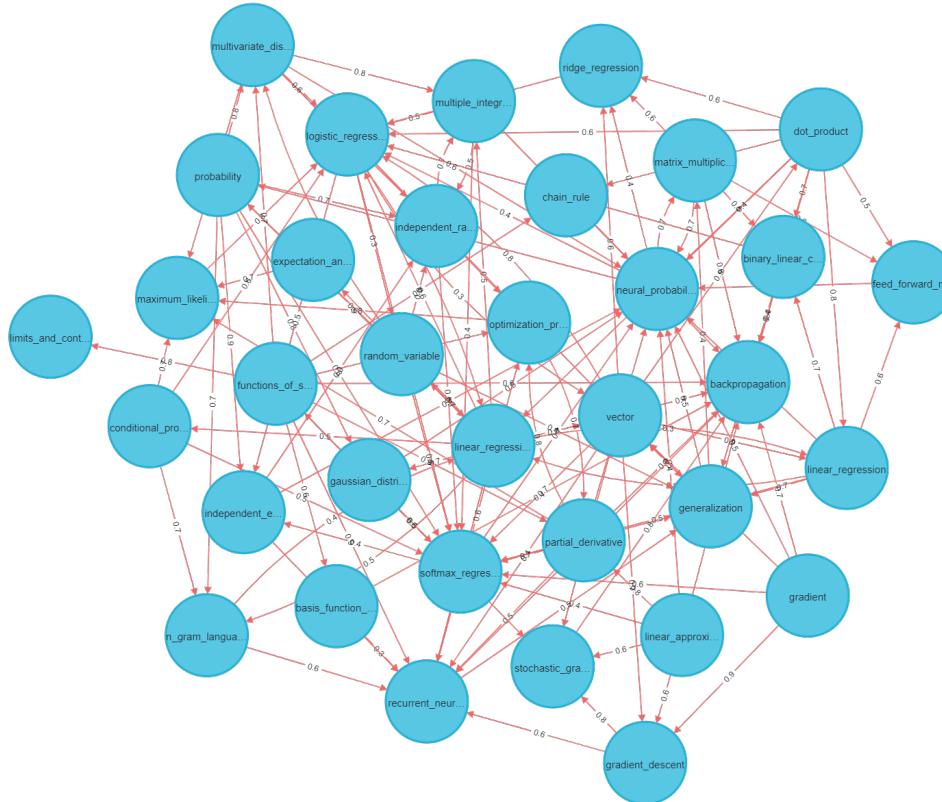


FIGURE 6 – Graph de prérequis de Machine Learning

- **Modélisation des apprenants** : Chaque étudiant est associé à une vue personnalisée du graphe où chaque concept possède un attribut KNOWN, mis à jour dynamiquement après un quiz ou une évaluation interactive.

— Interface :

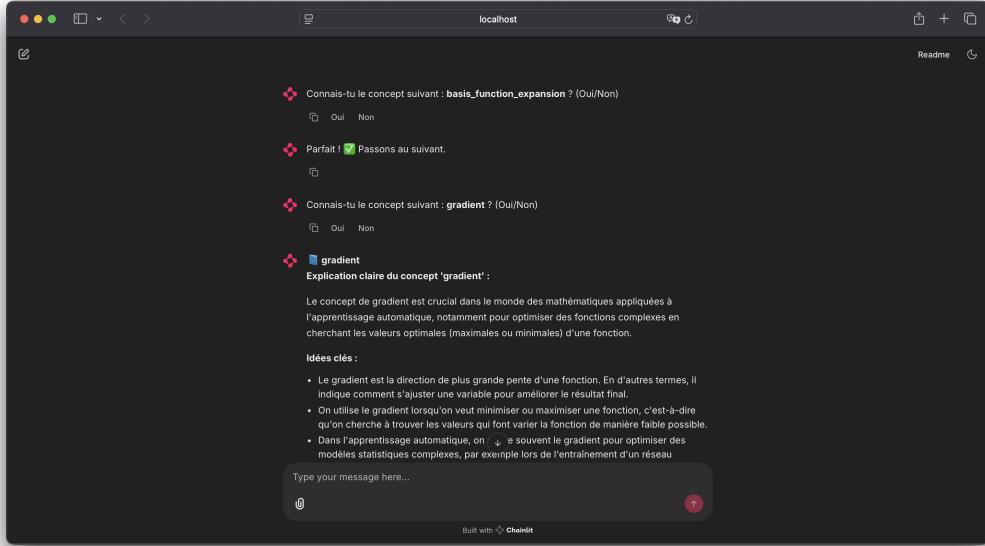


FIGURE 7 – Quiz diagnostique initial, et l'explication des prérequis inconnus

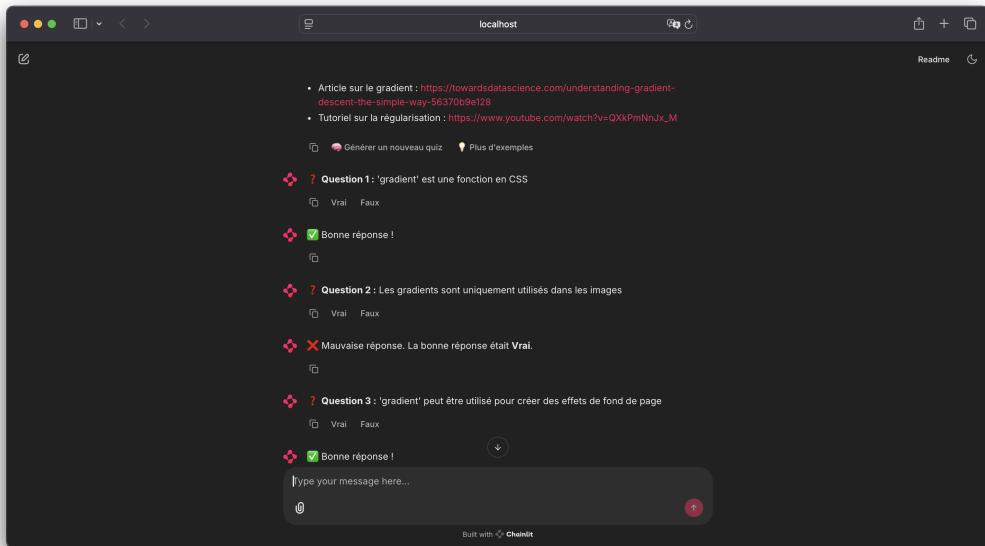


FIGURE 8 – quiz test après l'explication adaptée

— Comparaison des approches :

- **LLM standard** : Génération d'explications à partir d'un texte ou d'une question, sans tenir compte du niveau réel de l'étudiant.
- **Notre système aligné** : Utilisation du graphe de prérequis pour guider dynamiquement le LLM*, en intégrant uniquement les concepts maîtrisés et les prérequis nécessaires dans le prompt.

— Éléments évalués :

- Cohérence des explications par rapport au niveau réel de l'étudiant.
- Progressivité dans l'introduction des concepts.
- Réduction des erreurs liées à des prérequis manquants.

5.2 Résultats observés et comparatifs

Les différences de comportement entre un LLM* standard et notre approche orientée graphe ont été observées sur plusieurs aspects clés :

Critère	LLM standard	Notre système (KG + LLM)
Prise en compte du niveau étudiant	Aucune	Oui (via KNOWN et prérequis)
Structuration des explications	Variable, parfois hors de portée	Guidée par les prérequis
Cohérence des chaînes logiques	Parfois incohérente	Contrôlée via le graphe orienté
Progressivité	Aléatoire	Ajustée dynamiquement
Personnalisation	Générique	Personnalisée par graphe individuel

TABLE 1 – Comparatif entre un LLM standard et notre système aligné
De plus, notre système est capable :

- D'identifier les prérequis manquants d'un concept demandé.
 - De générer un quiz ciblé sur ces lacunes.
 - D'expliquer d'abord les prérequis avant de traiter le concept cible.
- Ces capacités ne sont pas possibles avec un LLM* utilisé seul.

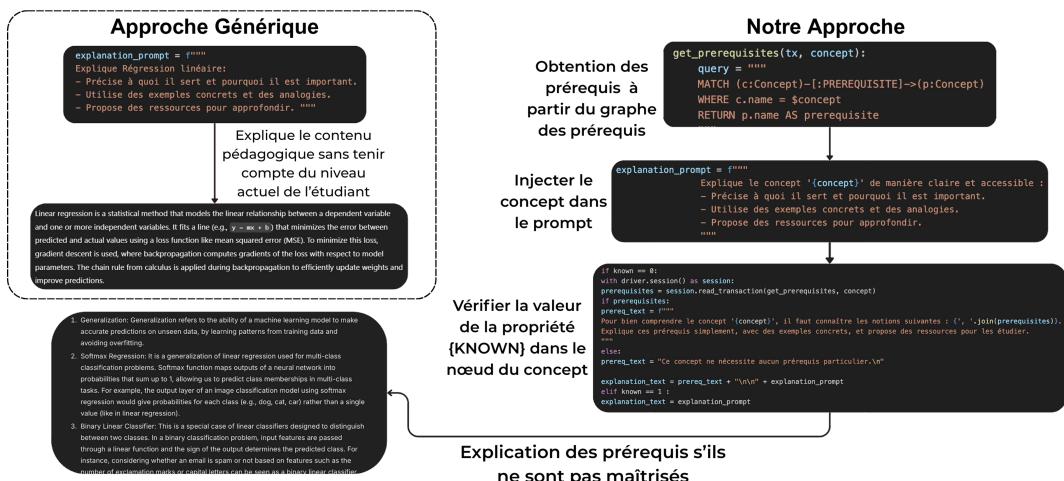


FIGURE 9 – Approche d'un LLM Standard VS Notre Approche

5.3 Analyse qualitative

Les retours d'utilisation sur des cas simulés et interactifs ont permis d'identifier plusieurs bénéfices notables :

- **Meilleure clarté des explications** : L'introduction préalable des pré-requis réduit les confusions et facilite la compréhension des concepts complexes.
- **Sentiment de progression logique** : Les apprenants perçoivent une montée en compétence structurée et cohérente.
- **Pertinence des contenus générés** : Le LLM, guidé par le graphe, évite d'aborder des notions non maîtrisées par l'étudiant, ce qui limite les hallucinations et les erreurs de raisonnement.

Toutefois, plusieurs limites ont également été relevées :

- **Qualité des prompts sensible à l'encodage du graphe** : Une mauvaise structuration textuelle des relations peut dégrader la pertinence des explications.
- **Présence de redondances** : Si le modèle est trop fortement guidé, des répétitions peuvent apparaître dans les explications successives.
- **Calibration du score de confiance (KNOWN) perfectible** : Les seuils actuels nécessitent des ajustements pour mieux refléter la maîtrise réelle de l'apprenant.

6 Discussion et limitations

L'analyse des résultats obtenus nous permet d'identifier les forces et les limites de notre approche, ainsi que des pistes d'amélioration pour les travaux futurs.

6.1 Points forts

Notre système présente plusieurs avantages notables :

- **Personnalisation effective** : L'alignement du LLM avec le graphe de connaissances permet une adaptation précise des explications au profil individuel de l'apprenant.
- **Évolutivité du modèle apprenant** : La mise à jour dynamique du profil de connaissances permet au système de s'adapter à la progression de l'apprenant.
- **Transparence du processus** : L'utilisation explicite du graphe de connaissances rend le processus d'adaptation plus compréhensible et contrôlable que les approches boîtes noires.
- **Indépendance relative du domaine** : Bien que notre prototype ait été testé dans un domaine spécifique, l'architecture proposée est trans-

posable à d'autres domaines d'apprentissage.

6.2 Limitations identifiées

Malgré ces avantages, notre approche présente plusieurs limitations qu'il convient de reconnaître :

- **Coût de construction du graphe** : La création initiale d'un graphe de connaissances de qualité reste un processus exigeant en temps et en expertise.
- **Précision de l'évaluation des connaissances** : L'évaluation automatique des connaissances de l'apprenant reste imparfaite, avec des risques de faux positifs et de faux négatifs.
- **Dépendance aux capacités du LLM** : La qualité des explications générées dépend fondamentalement des capacités du modèle de langage sous-jacent.
- **Gestion des styles d'apprentissage** : Notre approche actuelle se concentre principalement sur les connaissances préalables, sans prendre en compte d'autres facteurs comme les styles d'apprentissage préférentiels.
- **Passage à l'échelle** : L'application de notre approche à des domaines très vastes ou interdisciplinaires pose des défis en termes de complétude et de cohérence du graphe de connaissances.

7 Perspectives

7.1 Synthèse des contributions

Ce travail a exploré l'alignement des grands modèles de langage (LLMs) avec les représentations mentales des apprenants grâce à l'utilisation de graphes de connaissances. Nos principales contributions sont :

- Une méthodologie pour la construction et l'exploitation de graphes de connaissances représentant à la fois le domaine d'apprentissage et le profil de l'apprenant.
- Une approche d'alignement des LLMs permettant de générer des explications adaptées au niveau de connaissance spécifique de chaque apprenant.
- Un prototype fonctionnel démontrant la faisabilité et l'efficacité de cette approche dans un contexte éducatif.
- Des résultats expérimentaux suggérant une amélioration significative de l'efficacité de l'apprentissage grâce à la personnalisation des explications.

Ces contributions ouvrent la voie à des systèmes d'apprentissage plus in-

telligents et personnalisés, capables de s'adapter aux besoins spécifiques de chaque apprenant.

7.2 Perspectives futures

Plusieurs directions prometteuses pourraient être explorées pour étendre et améliorer ce travail :

- **Automatisation de la construction du graphe** : Développement de méthodes plus automatisées pour extraire des graphes de connaissances à partir de ressources pédagogiques existantes.
- **Intégration de dimensions affectives** : Prise en compte non seulement des connaissances, mais aussi des facteurs motivationnels et émotionnels dans l'adaptation des explications.
- **Approches multimodales** : Extension du système pour générer des explications combinant texte, visualisations et autres modalités adaptées aux préférences de l'apprenant.
- **Évaluation à long terme** : Études longitudinales pour évaluer l'impact de notre approche sur l'apprentissage à long terme et le transfert de connaissances.
- **Co-construction du graphe** : Développement d'approches permettant aux apprenants de contribuer à l'enrichissement du graphe de connaissances, reflétant leur propre compréhension du domaine.

8 Conclusion

En conclusion, l’alignement des LLMs avec les graphes de connaissances représente une approche prometteuse pour personnaliser l’apprentissage à l’ère de l’intelligence artificielle. Notre travail constitue une première étape dans cette direction, avec de nombreuses opportunités d’amélioration et d’extension dans des travaux futurs.

Références

- [1] Bahare Fatemi, Jonathan Halcrow, Bryan Perozzi (2023). *Talk like a Graph : Encoding Graphs for Large Language Models*. arXiv preprint arXiv :2310.04560v1
- [2] Zhao, W., Eger, S., Gurevych, I. (2023). *A Survey on Large Language Models for Education*. arXiv preprint arXiv :2312.10530.
- [3] Aytekin, M. C., Saygin, Y. (2024). *ACE : AI-Assisted Construction of Educational Knowledge Graphs with Prerequisite Relations*. *Journal of Educational Data Mining*, 16(2), 85–114. <https://doi.org/10.5281/zenodo.1425089>
- [4] Kasneci, E., Sessler, K., Küchemann, S., Bannert, M., Dementieva, D., Fischer, F., Gasser, U., Groh, G., Günemann, S., Hüllermeier, E., et al. (2023). *ChatGPT for good ? On opportunities and challenges of large language models for education*. Learning and Individual Differences, 103, 102274.
- [5] Novak, J. D., & Gowin, D. B. (1984). *Learning how to learn*. Cambridge University Press.
- [6] Bartlett, F. C. (1932). *Remembering : A study in experimental and social psychology*. Cambridge University Press.
- [7] Johnson-Laird, P. N. (1983). *Mental models : Towards a cognitive science of language, inference, and consciousness*. Harvard University Press.
- [8] Hogan, A., Blomqvist, E., Cochez, M., d'Amato, C., Melo, G. D., Gutierrez, C., Kirrane, S., Gayo, J. E. L., Navigli, R., Neumaier, S., et al. (2021). *Knowledge graphs*. ACM Computing Surveys, 54(4), 1-37.
- [9] Ye, P., Han, X., Lin, B., Liu, Q., Tong, Z., Wei, F. (2023). *Leveraging Knowledge Graphs for Zero-Shot LLM-Based Learning*. Proceedings of the AAAI Conference on Artificial Intelligence, 37(8), 11124-11132.
- [10] Graesser, A. C., Chipman, P., Haynes, B. C., & Olney, A. (2005). *Auto-Tutor : An intelligent tutoring system with mixed-initiative dialogue*. IEEE Transactions on Education, 48(4), 612-618.
- [11] Yu, W., Luo, M., Zhou, P., Si, C., Zhou, Y., Wang, X., Wu, J., Chen, X. (2020). *Curriculum Learning : A Survey*. Machine Learning, 1-40.
- [12] Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E., Le, Q., Zhou, D. (2022). *Chain of thought prompting elicits reasoning in large language models*. Advances in Neural Information Processing Systems, 35, 24824-24837.

- [13] Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., et al. (2022). *Training language models to follow instructions with human feedback*. Advances in Neural Information Processing Systems, 35, 27730-27744.
- [14] Christiano, P. F., Leike, J., Brown, T., Martic, M., Legg, S., & Amodei, D. (2017). *Deep reinforcement learning from human preferences*. Advances in Neural Information Processing Systems, 30.
- [15] Bai, Y., Kadavath, S., Kundu, S., Askell, A., Kernion, J., Jones, A., Chen, A., Goldie, A., Mirhoseini, A., McKinnon, C., et al. (2022). *Constitutional AI : Harmlessness from AI Feedback*. arXiv preprint arXiv :2212.08073.
- [16] Kumar, A., Kalwani, N., Li, J., Gupta, S., West, P. (2023). *Supporting Education with Large Language Models : Opportunities and Alignment Challenges*. arXiv preprint arXiv :2308.05591.
- [17] Chen, X., Liu, Q., Li, S., Du, Y., Yu, W., Wu, J. (2023). *Adaptive Curriculum Learning with Large Language Models for Educational Data Mining*. arXiv preprint arXiv :2310.04560.
- [18] Sun, H., Zhang, Y., Wang, S., Tan, M. (2025). *Educational Knowledge Graph Enhanced Large Language Models*. arXiv preprint arXiv :2503.13340.
- [19] Sharath, S., Pardos, Z., Chuang, I. (2024). *Embedding Knowledge Graphs in Educational Data Mining*. Journal of Educational Data Mining (JEDM), 16(1).
- [20] Lee, S., Ryu, J., Choi, J., Hwang, S. (2022). *Prompt Engineering for Educational Applications of Large Language Models*. arXiv preprint arXiv :2210.12228.

Glossaire

LLM* *Large Language Model* - Modèle de langage de grande taille, système d'IA entraîné sur d'immenses corpus de textes capables de générer du contenu linguistique cohérent et contextuel.

Graphe de connaissances* Structure de données qui représente les connaissances sous forme de graphe orienté où les nœuds sont des concepts et les arêtes des relations sémantiques entre ces concepts.

Prompt Engineering* Technique consistant à formuler et structurer précisément les instructions données à un LLM pour optimiser ses réponses selon des objectifs spécifiques.

Fine-tuning* Processus d'ajustement des paramètres d'un modèle pré-entraîné sur un ensemble de données spécifiques à une tâche pour améliorer ses performances sur cette tâche.

RLHF* *Reinforcement Learning from Human Feedback* - Apprentissage par renforcement à partir de feedback humain, méthode pour affiner le comportement d'un modèle en utilisant des évaluations humaines.

Constitutional AI* Approche visant à encadrer le comportement des modèles d'IA par un ensemble de principes prédéfinis, similaires à une constitution.

Cartes conceptuelles* Représentations graphiques organisées montrant les relations entre différents concepts, utilisées comme outils pédagogiques.

Théorie des schémas* Théorie cognitive suggérant que la connaissance est organisée en unités structurées (schémas) qui facilitent la compréhension et l'assimilation de nouvelles informations.

Modèles mentaux* Représentations internes simplifiées que les individus construisent pour comprendre et interagir avec le monde extérieur.

Curriculum Learning* Stratégie d'apprentissage qui consiste à organiser l'acquisition des connaissances selon une progression de difficulté croissante.

ACE* *AI-Assisted Construction of Educational Knowledge Graphs with Prerequisite Relations* - Algorithme d'assistance à la construction de graphes de connaissances éducatifs avec relations de prérequis.

CSR* *Concept Semantic Relatedness* - Mesure quantifiant la proximité sémantique entre deux concepts dans un texte.

CER* *Concept Embedding Relatedness* - Évaluation de la proximité de deux concepts dans un espace vectoriel d'embeddings.

PREREQUISITE* Relation dans un graphe de connaissances indiquant qu'un concept doit être maîtrisé avant d'aborder un autre concept.

KNOWN* Attribut binaire dans un graphe de connaissances indiquant si un concept est maîtrisé (1) ou non (0) par un apprenant.

Few-shot prompting* Technique consistant à inclure quelques exemples dans les instructions données à un LLM pour guider sa génération.

Intelligent Tutoring Systems* Systèmes informatiques conçus pour fournir un enseignement personnalisé sans intervention humaine directe.

Knowledge Forest* Approche utilisant des graphes pour représenter et exploiter des structures de connaissances dans un contexte éducatif.

Embedding* Représentation vectorielle d'un mot, phrase ou concept dans un espace multidimensionnel, capturant ses caractéristiques sémantiques.

Chain of thought* Technique de prompting encourageant un LLM à décomposer son raisonnement en étapes intermédiaires explicites.

Neo4j* Système de gestion de base de données orientée graphe permettant de stocker et interroger efficacement des données structurées en graphe.

LangChain* Framework Python facilitant le développement d'applications basées sur des LLMs en permettant de créer des chaînes de traitement.

Chainlit* Bibliothèque Python pour créer rapidement des interfaces conversationnelles basées sur des LLMs.

Ollama* Outil permettant d'exécuter localement des modèles de langage open-source.

Talk Like a Graph* Approche consistant à encoder textuellement les informations d'un graphe pour les intégrer dans un prompt de LLM.

Annexes

Diagramme de séquence

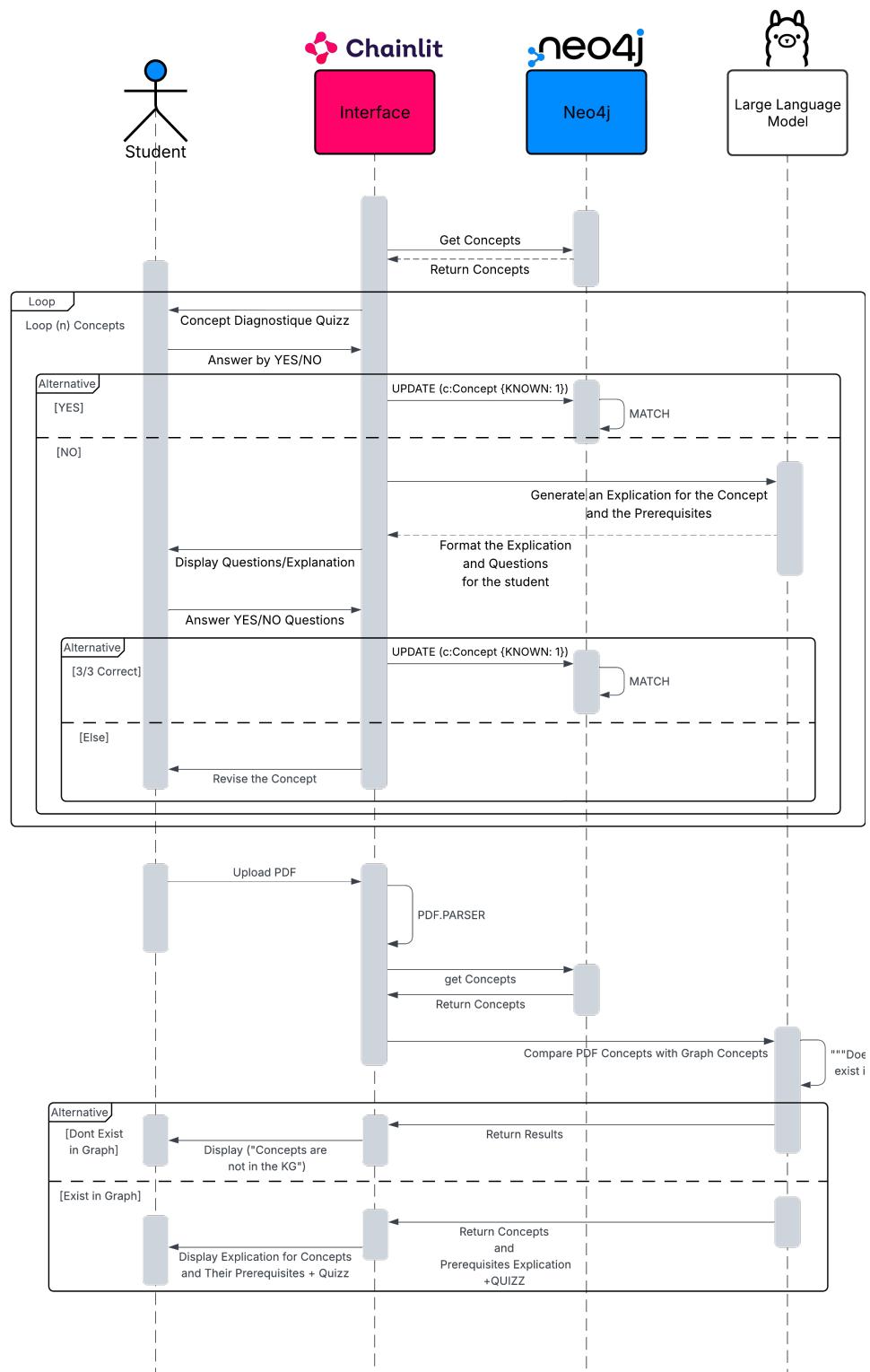


FIGURE 10 – Diagramme de séquence de notre système

Code Python du système

```
1
2 //fonction lecture PDF
3 def extract_text_from_pdf(file_path):
4     """
5         Extrait le texte du fichier PDF. Si le PDF contient uniquement des
6             ↳ images, l'OCR est utilisé pour extraire le texte des images.
7         Si le PDF contient à la fois du texte et des images, le texte est
8             ↳ extrait et l'OCR est appliqué aux pages sans texte.
9
10    :param file_path: Chemin vers le fichier PDF.
11    :return: Le texte extrait du PDF.
12    """
13
14    pdf = PyPDF2.PdfReader(file_path)
15    pdf_text = ""
16    images = convert_from_path(file_path)
17    ocr_text = ""
18
19    for page_number, page in enumerate(pdf.pages):
20        # Tenter d'extraire le texte
21        text = page.extract_text()
22        if text:
23            pdf_text += text
24        else:
25            # Si aucun texte n'est extrait, appliquer l'OCR sur
26            # l'image de cette page
27            print(f"Pas de texte trouvé sur la page {page_number + 1},
28                  ↳ tentative d'OCR.")
29            img = images[page_number]
30            ocr_text += pytesseract.image_to_string(img)
31
32    full_text = pdf_text + "\n" + ocr_text
33
34    # Si le texte extrait est vide, retournez une indication
35    if not full_text.strip():
36        return None # Indique que le PDF ne contient pas de texte
37
38    return full_text
39
40
41    # -----
42    # Fonctions Graph
43
44    # -----
```

```

37
38 def get_prerequisites(tx, concept):
39     query = """
40     MATCH (c:Concept)-[:PREREQUISITE]->(p:Concept)
41     WHERE c.name = $concept
42     RETURN p.name AS prerequisite
43     """
44     result = tx.run(query, concept=concept)
45     return [record["prerequisite"] for record in result]
46
47 def get_all_concepts(tx):
48     query = """
49     MATCH (c:Concept)
50     WHERE c.name IS NOT NULL
51     RETURN c.name AS concept
52     ORDER BY rand()
53     """
54     result = tx.run(query)
55     return [record["concept"] for record in result]
56
57 # -----
58 # Chat start
59 # -----
60 @cl.on_chat_start
61 async def on_chat_start():
62     with driver.session() as session:
63         concepts = session.read_transaction(get_random_concepts)
64
65     if not concepts:
66         await cl.Message("Aucun concept trouvé dans la base de
67             ↳ données.").send()
68         await handle_pdf_upload()
69     return
70
71     cl.user_session.set("concepts", concepts)
72     cl.user_session.set("current_index", 0)
73
74     await ask_concept_question()
75
76 # -----
77 # Étape 1 : Demande initiale

```

```

77  # -----
78
79  async def ask_concept_question():
80      concepts = cl.user_session.get("concepts")
81      index = cl.user_session.get("current_index")
82
83      if index >= len(concepts):
84          await cl.Message(" Tu as terminé tous les concepts ! Place
85                          ↳ maintenant à l'analyse du PDF.").send()
86          await handle_pdf_upload()
87
88      return
89
90      concept = concepts[index]
91      await cl.Message(
92          content=f"Connais-tu le concept suivant : **{concept}** ?
93          ↳ (Oui/Non)",
94          actions=[
95              cl.Action(name="yes", label="Oui", payload={"known":
96                  ↳ True}),
97              cl.Action(name="no", label="Non", payload={"known":
98                  ↳ False})
99          ]
100     ).send()
101
102     # -----
103     # Si le concept est connu
104     # -----
105
106     @cl.action_callback("yes")
107     async def handle_known_concept(action):
108         # Récupérer le concept actuel
109         concepts = cl.user_session.get("concepts")
110         index = cl.user_session.get("current_index")
111         concept = concepts[index]
112
113         # Mise à jour de la propriété 'known' à 1 dans Neo4j
114         try:
115             with driver.session() as session:
116                 session.run("""
117                     MATCH (c:Concept)
118                     WHERE c.name = $concept
119                     SET c.known = 1

```

```

114         """ , concept=concept)
115             await cl.Message("Parfait ! Passons au suivant.").send()
116     except Exception as e:
117         await cl.Message(f" Erreur lors de la mise à jour du concept :
118                         ↳ {e}").send()
119
120     # Passer au concept suivant
121     cl.user_session.set("current_index",
122                         ↳ cl.user_session.get("current_index") + 1)
123
124     # -----
125     # Si inconnu → Explication + Quiz
126     # -----
127     @cl.action_callback("no")
128     async def handle_unknown_concept(action):
129         concepts = cl.user_session.get("concepts")
130         index = cl.user_session.get("current_index")
131         concept = concepts[index]
132
133         with driver.session() as session:
134             prerequisites = session.read_transaction(get_prerequisites,
135                                             ↳ concept)
136
137         prereq_text = f"""
138                         Pour bien comprendre le concept '{concept}' ,
139                         ↳ il est important de se concentrer
140                         ↳ uniquement sur les prérequis les plus
141                         ↳ pertinents et directement liés à ce
142                         ↳ concept. Parmi les notions suivantes : {',
143                         ↳ '.join(prerequisites)} , identifie
144                         ↳ uniquement celles qui sont étroitement
145                         ↳ liées à '{concept}' .
146
147             Explique uniquement ces prérequis de manière
148             ↳ simple, avec des exemples concrets, et
149             ↳ propose des ressources utiles pour les
150             ↳ étudier."""
151
152         concept_text = f"""

```

```

141     Explique ensuite le concept '{concept}' de manière claire et
142         ↳ accessible :
143             - Décris son but, son utilité et dans quel contexte il est
144                 ↳ important.
145             - Utilise des exemples concrets et des analogies pour faciliter la
146                 ↳ compréhension.
147             - Suggère quelques ressources (articles, tutoriels, vidéos, etc.)
148                 ↳ pour aller plus loin et mieux maîtriser ce concept.
149             """
150
151
152     quiz_prompt = (
153         f"Génère 3 questions Vrai/Faux avec les bonnes réponses, au
154             ↳ format JSON comme ceci :\n"
155         f"[{{'question': '...', 'answer': 'Vrai'}}, ...] sur le
156             ↳ concept '{concept}'."
157     )
158
159     try:
160         explanation = await cl.make_async(llm.invoke)(prereq_text +
161             ↳ "\n\n" + concept_text)
162         quiz_json = await cl.make_async(llm.invoke)(quiz_prompt)
163
164         # Vérification du format du JSON
165         try:
166             quiz_data = json.loads(quiz_json.strip())  # Utilisation
167                 ↳ de json.loads() pour éviter les problèmes de sécurité
168             if not isinstance(quiz_data, list) or not
169                 ↳ all(isinstance(q, dict) and 'question' in q and
170                     ↳ 'answer' in q for q in quiz_data):
171                 raise ValueError("Le format du quiz généré est
172                     ↳ incorrect.")
173         except (json.JSONDecodeError, ValueError) as e:
174             await cl.Message(f" Erreur dans le format du quiz généré :
175                 ↳ {str(e)}").send()
176             cl.user_session.set("current_index",
177                 ↳ cl.user_session.get("current_index") + 1)
178             await ask_concept_question()
179
180         return
181
182     except Exception as e:

```

```

168         await cl.Message(f" Erreur pendant la génération :
169             ↪  {e}").send()
170         cl.user_session.set("current_index",
171             ↪  cl.user_session.get("current_index") + 1)
172         await ask_concept_question()
173
174     return
175
176     # Afficher explication
177     await cl.Message(f" **Explication de
178         ↪  {concept}**\n\n{explanation.strip()}").send()
179
180     # Sauvegarde du quiz et gestion de la session
181     cl.user_session.set("current_quiz", quiz_data)
182     cl.user_session.set("quiz_index", 0)
183     cl.user_session.set("quiz_score", 0)
184
185     await send_quiz_question()
186
187     # -----
188     # Envoyer une question du quiz
189     # -----
190
191     async def send_quiz_question():
192
193         quiz = cl.user_session.get("current_quiz")
194         index = cl.user_session.get("quiz_index")
195
196
197         if index >= len(quiz):
198             score = cl.user_session.get("quiz_score")
199             total = len(quiz)
200
201             result_msg = f" Tu as bien compris ! Score : {score}/{total}"
202             ↪  if score == total else f" Tu as eu {score}/{total}.
203             ↪  Continue de réviser !"
204
205             await cl.Message(result_msg).send()
206
207             # Si toutes les questions sont répondues correctement, mettre
208             # à jour le concept comme "connu"
209             if score == total:
210                 concepts = cl.user_session.get("concepts")
211                 current_index = cl.user_session.get("current_index")
212                 if current_index < len(concepts):
213                     concept = concepts[current_index]

```

```

203     try:
204         with driver.session() as session:
205             session.run("""
206                 MATCH (c:Concept)
207                 WHERE c.name = $concept
208                 SET c.known = 1
209                 """ , concept=concept)
210             await cl.Message(f" Le concept '{concept}' est
211                             maintenant marqué comme connu.").send()
212         except Exception as e:
213             await cl.Message(f" Erreur lors de la mise à jour
214                             du concept : {e}").send()
215
216         # Passer au concept suivant
217         cl.user_session.set("current_index",
218             ↪ cl.user_session.get("current_index") + 1)
219
220         # Vérifier si tous les concepts ont été vus
221         concepts = cl.user_session.get("concepts")
222         current_index = cl.user_session.get("current_index")
223
224         if current_index >= len(concepts):
225             await cl.Message(" Tu as terminé tous les concepts ! Place
226                             maintenant à l'analyse du PDF.").send()
227             await handle_pdf_upload() # Lance l'analyse du PDF
228         else:
229             await ask_concept_question()
230
231     return
232
233     question = quiz[index]["question"]
234     await cl.Message(
235         content=f" **Question {index+1} :** {question}",
236         actions=[
237             cl.Action(name="true", label="Vrai", payload={"response":
238                 ↪ "Vrai"}),
239             cl.Action(name="false", label="Faux", payload={"response":
240                 ↪ "Faux"})
241         ]
242     ).send()
243
244     # -----

```

```

238 # Réponse au quiz
239 # -----
240 @cl.action_callback("true")
241 async def answer_true(action):
242     await handle_quiz_response("Vrai")
243
244 @cl.action_callback("false")
245 async def answer_false(action):
246     await handle_quiz_response("Faux")
247
248 async def handle_quiz_response(user_answer):
249     quiz = cl.user_session.get("current_quiz")
250     index = cl.user_session.get("quiz_index")
251     correct = quiz[index]["answer"]
252
253     if user_answer.lower() == correct.lower():
254         await cl.Message(" Bonne réponse !").send()
255         cl.user_session.set("quiz_score",
256             ↪ cl.user_session.get("quiz_score") + 1)
257     else:
258         await cl.Message(f" Mauvaise réponse. La bonne réponse était
259             ↪ **{correct}**.").send()
260
261     cl.user_session.set("quiz_index", index + 1)
262     await send_quiz_question()
263
264 # -----
265 # Lancer l'analyse du PDF après la fin des quiz
266 # -----
267
268 async def handle_pdf_upload():
269     files = None
270
271     # Demande d'upload de fichier
272     while files is None:
273         files = await cl.AskFileMessage(
274             content=" Veuillez uploader un fichier PDF pour commencer
275                 ↪ !",
276             accept=["application/pdf"],
277             max_size_mb=100,
278             timeout=180
279         ).send()

```

```

276
277     file = files[0]
278     pdf_text = extract_text_from_pdf(file.path)
279
280     if not pdf_text:
281         await cl.Message(content=" Le PDF téléchargé ne contient pas
282             ↵ de texte exploitable. Veuillez essayer avec un autre
283             ↵ fichier.").send()
284
285     return
286
287
288     # Découpage du texte
289     text_splitter = RecursiveCharacterTextSplitter(chunk_size=1200,
290             ↵ chunk_overlap=50)
291     texts = text_splitter.split_text(pdf_text)
292     metadatas = [{"source": f"i-pl"} for i in range(len(texts))]
293
294     # Embedding & indexation
295     embeddings = OllamaEmbeddings(model="nomic-embed-text")
296     docsearch = await cl.make_async(Chroma.from_texts)(texts,
297             ↵ embeddings, metadatas=metadatas)
298
299     # Mémoire de conversation
300     memory = ConversationBufferMemory(
301         memory_key="chat_history",
302         output_key="answer",
303         chat_memory=ChatMessageHistory(),
304         return_messages=True,
305     )
306
307     # Création de la chaîne de QA
308     chain = ConversationalRetrievalChain.from_llm(
309         llm=llm,
310         chain_type="stuff",
311         retriever=docsearch.as_retriever(),
312         memory=memory,
313         return_source_documents=True
314     )
315     cl.user_session.set("chain", chain)
316
317     # Message d'attente pour l'analyse du PDF

```

```

312     wait_message = await cl.Message(" Analyse du PDF en
313         ↪ cours...").send()
314
315     # Résumé du contenu
316     summary_prompt = f"""
317         Fais un résumé structuré et concis du texte suivant, en mettant en
318             ↪ valeur les concepts clés et idées principales :
319             {pdf_text}
320             """
321
322     summary = await cl.make_async(llm.invoke)(summary_prompt)
323
324     # Récupération de tous les concepts
325     with driver.session() as session:
326         concepts = session.read_transaction(get_all_concepts)
327         cl.user_session.set("concepts", concepts)
328
329     matched_concepts = set()
330
331     # Détection des concepts mentionnés dans le résumé
332     for concept in concepts:
333         prompt = f"""
334             Analyse ce résumé et détermine s'il mentionne ou traite du
335                 ↪ concept \'{concept}\'.
336             Réponds uniquement par 'Oui' ou 'Non'.
337
338             Résumé :
339             {summary}
340             """
341
342     try:
343         response = await cl.make_async(llm.invoke)(prompt)
344         if "oui" in response.strip().lower():
345             matched_concepts.add(concept)
346
347             with driver.session() as session:
348                 result = session.run("MATCH (c:Concept) WHERE
349                     ↪ c.name = $concept RETURN c.known AS known",
350                     ↪ concept=concept)
351                 known = result.single()["known"]
352
353             explanation_prompt = f"""

```

```

347     Explique le concept '{concept}' de manière claire et
348         ↳ accessible :
349             - Précise à quoi il sert et pourquoi il est important.
350             - Utilise des exemples concrets et des analogies.
351             - Propose des ressources pour approfondir.
352             """
353
354     if known == 0:
355         with driver.session() as session:
356             prerequisites =
357                 ↳ session.read_transaction(get_prerequisites,
358                     ↳ concept)
359
360             if prerequisites:
361                 prereq_text = f"""
362                     Pour bien comprendre le concept '{concept}',
363                         ↳ il est important de se concentrer
364                         ↳ uniquement sur les prérequis les plus
365                         ↳ pertinents et directement liés à ce
366                         ↳ concept. Parmi les notions suivantes : {',
367                         ↳ '.join(prerequisites)}}, identifie
368                         ↳ uniquement celles qui sont étroitement
369                         ↳ liées à '{concept}'.
370
371                     Explique uniquement ces prérequis de manière
372                         ↳ simple, avec des exemples concrets, et
373                         ↳ propose des ressources utiles pour les
374                         ↳ étudier."""
375
376             else:
377                 prereq_text = "Ce concept ne nécessite aucun
378                     ↳ prérequis particulier.\n"
379
380             explanation_text = prereq_text + "\n\n" +
381                 ↳ explanation_prompt
382
383             else:
384                 explanation_text = explanation_prompt
385
386
387             explanation = await
388                 ↳ cl.make_async(llm.invoke)(explanation_text)
389             await cl.Message(f" **Explication de
390                 ↳ {concept}**\n\n{explanation.strip()}").send()

```

```

371
372     except Exception as e:
373         print(f" Erreur lors de la vérification du concept
374             ↪ '{concept}' : {e}")
375
376     await wait_message.remove()
377
378     if matched_concepts:
379         cl.user_session.set("matched_concepts", matched_concepts)
380
381         # Génération du quiz
382         quiz_prompt = f"""
383             Génère 5 questions de quiz à choix multiples (QCM), chacune
384             ↪ portant sur un des concepts suivants : {',
385             ↪ '.join(matched_concepts)}.
386
387             Pour chaque question :
388                 - Propose une bonne réponse et trois distracteurs plausibles.
389                 - N'indique PAS la bonne réponse.
390                 - Utilise un format clair comme :
391
392                 **Question 1 :** Quel est le rôle de XYZ ?
393
394                 A. Réponse plausible
395                 B. Réponse plausible
396                 C. Réponse plausible
397                 D. Réponse plausible
398
399             Adapte la langue du quiz à celle des concepts si besoin.
400
401             """
402
403         quiz_output = await cl.make_async(llm.invoke)(quiz_prompt)
404         await cl.Message(f" **Quiz basé sur les concepts détectés
405             ↪ :**\n\n{quiz_output.strip()}").send()
406
407     else:
408
409         await cl.Message(" Aucun concept détecté dans ce
410             ↪ document.").send()
411
412     cl.user_session.set("full_pdf_text", pdf_text)
413     await cl.Message(" Le PDF a été traité. Vous pouvez maintenant
414             ↪ poser vos questions !").send()

```