

UNIVERSITÉ LIBRE DES PAYS DES GRANDS LACS
FACULTÉ DE SCIENCES ET TECHNOLOGIES
APPLIQUÉES

DÉPARTEMENT DE GÉNIE ÉLECTRIQUE ET INFORMATIQUE



BP. 368 GOMA

www.ulpgl.net

**CONCEPTION ET RÉALISATION D'UN
SYSTÈME D'AIDE À LA CORRECTION
D'ÉPREUVES À CARACTÈRE THÉORIQUE
BASÉ SUR LES MODÈLES DE LANGAGE
LARGES (LLM) : CAS DE GPT-3.5 ET LLAMA-3**

Par : MUHIGIRI ASHUZA Albin

Travail présenté en vue de l'obtention du Diplôme
d'ingénieur civil en génie électrique et informatique

Option : Génie Informatique

Directeur : Prof. BARAKA MUSHAGE Olivier

Encadreur : Ir. David KRAMÉ KADURHA

ANNEE ACADEMIQUE 2022 - 2023

Epigraphe

« L'intelligence artificielle a le potentiel de révolutionner l'éducation en personnalisant l'apprentissage et en offrant aux enseignants des outils puissants pour évaluer et soutenir leurs élèves. »

Yann LeCun

Dédicace

Je dédie ce travail à mes parents : **MUHIGIRI Namulabaha David** et **NZIGIRE Nshombo Françoise.**

MUHIGIRI Ashuza

Remerciements

Nous remercions tout d'abord le Dieu Tout-Puissant de nous avoir accordé force, santé et protection tout au long de notre recherche. Ensuite, nous exprimons notre gratitude envers les corps administratif et académique de l'Université Libre des Pays des Grands Lacs pour leur encadrement moral et intellectuel de qualité.

Nos sincères remerciements vont à notre Directeur, le Professeur **BARAKA MUSHAGE Olivier**, et à notre Encadreur, Ir. **David KRAMÉ KADURHA**, pour leur accompagnement précieux dans la réalisation de ce travail.

Nous sommes également reconnaissants envers nos parents, MUHIGIRI NAMULABARHA David et NZIGIRE NSHOMBO Françoise, pour leur soutien moral, spirituel et matériel.

Nous tenons particulièrement à remercier notre frère, MUHIGIRI CIRUZA Alain, pour son assistance morale et ses conseils continus.

Nos amis, frères et sœurs, MUHIGIRI MWINJA Aline, MUHIGIRI AKUJWE Arnold, MUHIGIRI ARSENE Benjamin, MUHIGIRI ANSIMA Yve, méritent notre gratitude pour leur soutien moral constant. Un merci spécial à notre ami et cher oncle, John NSHOMBO, ainsi qu'à sa charmante épouse, Aline ALIBARIKI, pour leur assistance morale, spirituelle et matérielle tout au long de notre parcours.

Nous remercions enfin nos amis et camarades, dont en particulier Alexandre CHAMBU, MUGISHO CIRINDYE Frederick, MWENDELWA David, KAVIRA AMANI Esther, BAHATI NGANDU Marc, MUMBERE KINZUMWA Simple et tous ceux qui, de près ou de loin, nous ont soutenus et qui trouvent ici l'expression de notre profonde gratitude.

MUHIGIRI Ashuza

Résumé

À l'ère des avancées technologiques, l'automatisation des processus éducatifs devient une priorité. Ce mémoire explore la conception et la mise en œuvre d'un système de correction automatisée, **Bic Rouge**, basé sur des modèles de langage Large (LLM) de pointe comme GPT-3.5 Turbo et LLAMA-3. L'objectif principal était d'évaluer si l'intégration de ces modèles dans le processus de correction d'épreuves théoriques pouvait améliorer l'efficacité, la précision et la cohérence des évaluations.

À travers ce travail, nous avons comparé les performances de différents LLMs en termes de correction automatique et mesuré leur alignement avec les évaluations humaines. GPT-3.5 Turbo a démontré une meilleure performance, bien que certaines limitations subsistent en termes de cohérence et de nuance, particulièrement pour des réponses complexes.

Le système permet également aux enseignants de modifier les corrections générées et de fournir des retours personnalisés aux étudiants. Cependant, certains défis subsistent, tels que l'optimisation des prompts et l'utilisation de techniques avancées pour mieux appréhender la subjectivité des réponses et améliorer la précision des évaluations.

Les perspectives d'évolution incluent l'ajout de nouvelles fonctionnalités au système, l'intégration de méthodes comme le **fine-tuning**, le **Retrieval-Augmented Generation (RAG)**, ainsi que l'utilisation de modèles de vision pour assister les enseignants dans des contextes moins numérisés. L'objectif final est de rendre le système plus robuste et accessible, notamment par le développement d'une application mobile.

Mots-clés : *LLM, correction automatisée, intelligence artificielle, évaluation académique, fine-tuning, Azure AI, GPT, Llamma, RAG, Python, FastAPI, JavaScript, ReactJS, Tailwind.*

Abstract

In the age of technological advancements, automating educational processes has become a priority. This thesis explores the design and implementation of an automated grading system, **Bic Rouge**, based on cutting-edge Large Language Models (LLMs) such as GPT-3.5 Turbo and LLAMA-3. The main objective was to assess whether integrating these models into the grading process for theoretical exams could enhance the efficiency, accuracy, and consistency of evaluations.

The study compares the performance of different LLMs in automatic grading and their alignment with human evaluations. GPT-3.5 Turbo exhibited superior performance, though limitations in coherence and nuance, particularly with complex responses, were noted.

The system also enables teachers to modify the generated corrections and provide personalized feedback to students. However, several challenges remain, such as optimizing prompts and leveraging advanced techniques to better capture the subjectivity of responses and improve evaluation accuracy.

Future developments include the addition of new features to the system, the integration of methods such as **fine-tuning**, **Retrieval-Augmented Generation (RAG)**, as well as the use of vision models to support teachers in less digitized environments. The ultimate goal is to make the system more robust and accessible, particularly through the development of a mobile application.

Keywords: LLM, automated grading, artificial intelligence, academic evaluation, fine-tuning, RAG, GPT, LLAMA, Python, FastAPI, JavaScript, ReactJS, Tailwind.

Table des matières

Dédicace.....	ii
Remerciements.....	iii
Résumé.....	iv
Abstract	v
Table des matières	vi
Liste des abréviations.....	ix
Liste des tableaux.....	xi
Liste des figures	xii
0. Introduction générale	1
0.1. Contexte	1
0.2. Identification et formulation du problème	2
0.3. Questions de recherche	3
0.4. Formulation des hypothèses.....	3
0.5. Justification du choix du sujet et motivations.....	4
0.5.1. L'objectif général.....	5
0.6. Méthodologie et délimitation du travail.....	6
0.7. Subdivision du travail	6
Chapitre 1 Généralités sur la correction d'épreuves théoriques et les modèles de langage large (LLM).....	8
1.1 Introduction partielle.....	8
1.2 Évaluation des épreuves théoriques : Définition, enjeux et défis	8
1.2.1 Définition	8
1.2.2 Enjeux	9

1.2.3	Défis	10
1.3	Exploration des modèles de langage larges (LLM)	11
1.3.1	Présentation et définition	11
1.3.2	Fonctionnement.....	12
1.3.3	Architecture.....	19
1.3.4	Types des LLM [31]	26
1.3.5	Applications Pratiques des LLM	32
1.4	Conclusion partielle	35
	 Chapitre 2 Techniques de correction automatique d'épreuves, modèles GPT-3 et LLAMA-3, et conception du Système.....	36
2.1	Introduction.....	36
2.2	Techniques de correction automatique d'épreuves	36
2.2.1	Approches traditionnelles	36
2.2.2	Approches modernes.....	39
2.2.3	Systèmes fonctionnels utilisant ces modèles	40
2.3	Présentation des modèles GPT-3.5 et LLAMA-3.....	41
2.3.1	Architecture et fonctionnement de GPT-3.5 Turbo [15] [63].....	41
2.3.2	Architecture et fonctionnement de LLAMA-3 [21] [65] [66]	45
2.3.3	Comparaison des caractéristiques et performances des modèles LLAMA-3 (8B) et GPT-3.5 Turbo [68] [69].....	48
2.4	Conception de l'architecture globale de Bic Rouge	50
2.4.1	Spécifications du système	51
2.4.2	Présentation des Éléments du Système	54
2.4.3	Conclusion partielle	60
	 Chapitre 3 Conception détaillée, réalisation et tests.....	62
3.1	Introduction partielle.....	62
3.2	Conception détaillée.....	62
3.2.1	Présentation des cas d'utilisation	62

3.2.2	Diagramme de cas d'utilisation	63
3.2.3	Conception de la base des données	71
3.2.4	Conception de l'API	75
3.2.5	Présentation des interfaces réalisés	80
3.3	Évaluation des la correction automatique et du feedback.....	96
3.3.1	Évaluation intrinsèque [62].....	97
3.3.2	Évaluation extrinsèque.....	97
3.4	Description succincte de l'implémentation.....	98
3.4.1	API REST	98
3.4.2	Client Web	99
3.4.3	Techniques de réalisation de systèmes basés sur l'IA générative	99
3.5	Tests et interprétation des résultats	100
3.5.1	Datasets utilisés pour les tests.....	100
3.5.2	Tests	103
3.5.3	Résultats des tests	104
3.6	Interprétation des résultats	111
3.7	Justification des choix d'implémentation	114
3.8	Conclusion partielle	115
	Conclusion générale.....	116
	Bibliographie.....	118
	Annexe A	126
A.1	Exemples de données de test.....	126
A.1.1	Copies et compositions	126
A.1.2	Sorties des LLMs	126
A.2	Exemples de prompts utilisés et configuration du modèle	127
A.3	Documentation sur les endpoints disponibles de l'API.....	129

Liste des abréviations

- AGIEval** : Artificial General Intelligence Evaluation,
- API** : Application Programming Interface,
- ARC** : AI2 Reasoning Challenge,
- BD** : Base de Données,
- BLEU** : Bilingual Evaluation Understudy,
- Bic Rouge** : Système d'aide à la correction des épreuves théoriques développé dans cette étude,
- CLIP** : Contrastive Language-Image Pre-Training,
- CU** : Cas d'Utilisation,
- DNN** : Deep Neural Network,
- DOCX** : Document XML (Microsoft Word),
- DROP** : Discrete Reasoning Over Paragraphs,
- F1-Score** : F1 Score (métrique d'évaluation des modèles),
- FM** : Foundation Model,
- GENAI** : Generative Artificial Intelligence,
- GQA** : Grouped-Query Attention,
- GPT** : Generative Pre-trained Transformer,
- GPT-3** : Generative Pretrained Transformer 3,
- HMM** : Hidden Markov Model,
- IA** : Intelligence Artificielle,
- JSON** : JavaScript Object Notation,
- LLAMA** : Large Language Model Meta AI,
- LLAMA-2** : Large Language Model Meta AI, 2nd Version,
- LLAMA-3** : Large Language Model Meta AI, 3rd Version,
- LLM** : Large Language Model,
- MATH** : Mathematics Reasoning Benchmark,
- MMLU** : Massive Multitask Language Understanding,

NLP : Natural Language Processing,
PDF : Portable Document Format,
RAG : Retrieval-Augmented Generation,
RLHF : Reinforcement Learning from Human Feedback,
RNN : Recurrent Neural Network,
ROUGE : Recall-Oriented Understudy for Gisting Evaluation,
SFT : Supervised Fine-Tuning,
SQL : Structured Query Language,
SQLite : Structured Query Language Lite,
SQuAD : Stanford Question Answering Dataset,
SVM : Support Vector Machine,
SSL/TLS : Secure Sockets Layer / Transport Layer Security,
T5 : Text-to-Text Transfer Transformer,
TTFT : Time to First Token,
ULPGL: Université Libre des Pays des Grands-Lacs.

Liste des tableaux

Tableau 1 Versions Disponibles et Spécifications des Modèles GPT-3.5 d'Azure OpenAI [64]	43
Tableau 2 Spécifications et Données de Formation pour les Modèles LLaMA 3 [67]	46
Tableau 3 Performances des Modèles Pré-entraînés de Base LLaMA 3 et LLaMA 2 [67]	47
Tableau 4 Comparaison des caractéristiques et performances des modèles LLAMA-3 (8B) et GPT-3.5 Turbo	48
<i>Tableau 5 : Documentation du cas d'utilisation « s'authentifier »</i>	66
<i>Tableau 6 : Documentation du Cas d'utilisation « créer un compte »</i>	67
<i>Tableau 7 : Documentation du Cas d'utilisation « CréerCours »</i>	68
<i>Tableau 8 : Documentation du cas d'utilisation « IncriptionCours »</i>	69
<i>Tableau 9 : Documentation du cas d'utilisation « SoumettreCopie »</i>	69
<i>Tableau 10 : Documentation du Cas d'utilisation « ExaminerEvaluation »</i>	70
Tableau 11 Dictionnaire de données	73
Tableau 12 Résultat du test manuel ici de système	105
Tableau 13 Configuration des paramètres de model GPT3.5 Turbo et Llama 3 8B -instruct	105
Tableau 14 Résultat détaillé des métriques GPT3.5 Turbo	107
Tableau 15 Résultat détaillé des métriques Llama 3 8B – Instruct	109
Tableau 16 Résultats détaillés des métriques de GPT-3.5 Turbo après ajustements et améliorations	110
Tableau 17 Configuration des paramètres de model GPT3.5 Turbo	111

Liste des figures

Figure 1.1 Mécanisme opérationnel des LLM cas de traduction automatique	12
Figure 1.2 Schéma des Composants d'Encodage et de Décodage [18].....	13
Figure 1.3 OpenAI Tokenizer pour GPT-3.5 & GPT-4.....	14
Figure 1.4 Comment un réseau de neurones (NN) "comprend" une phrase	14
Figure 1.5 Mécanisme d'attention (module d'auto-attention) avec Tensor2Tensor [18].....	15
Figure 1.6 Complétion de texte contextuelle avec GPT-4 Omni sur la plateforme ChatGPT.	16
Figure 1.7 Processus de Traduction avec un Modèle <i>Transformer</i> [18].....	17
Figure 1.8 Processus d'Entraînement des LLM [21].....	18
Figure 1.9 Architecture générique des <i>transformers</i> [2].....	21
Figure 1.10 Vue éclatée d'un <i>transformer</i> [29]	25
Figure 1.11 Génération d'images par un prompt avec <i>DALL-E</i> [31]	27
Figure 1.12 Intégration de Texte (<i>Embeddings</i>) [31].....	29
Figure 1.13 Foundation Models (FMs) [21]	30
Figure 1.14 Une taxonomie des LLMs pour les applications éducatives avec des travaux représentatifs [46].	35
Figure 2.1 Architecture globale de notre système	54
Figure 2.2 Architecture du Module de Correction.....	56
Figure 2.3 Architecture du processus d'Évaluation	57
Figure 2.4 Ébauche d'interface des résultats d'évaluation	58
Figure 2.5 Ébauche d'interface de login	59
Figure 2.6 Ébauche d'interface des Cours	60
Figure 3.1 Diagramme des cas d'utilisation global	64
Figure 3.2 Diagramme des classes représentant la base des données.....	72
Figure 3.3 Page d'accueil.....	81
Figure 3.4 Page de login	81
Figure 3.5 Page d'enregistrement.....	82
Figure 3.6 Dashboard présentation des cours	82

Figure 3.7 Dashboard, Button de création du cours.....	84
Figure 3.8 Formulaire de création du cours	84
Figure 3.9 Dashboard détails cours pour l'enseignant.....	85
Figure 3.10 Onglet de création d'épreuves	85
Figure 3.11 Onglet de création d'épreuves, choix de l'épreuve	86
Figure 3.12 Formulaire de création de l'épreuve	86
Figure 3.13 Présentation de l'épreuve créer	87
Figure 3.14 Visualisation de l'épreuve après création.....	87
Figure 3.15 Formulaire de composition de l'épreuve	88
Figure 3.16 Présentation des participant du cours (Esseinant et étudiants)	88
Figure 3.17 Vu d'ensemble d'une épreuve dans un cours sans soumission	89
Figure 3.18 Vu d'ensemble d'une épreuve dans un cours avec soumission.....	89
Figure 3.19 Vérification et validation de cotes pour l'épreuve	90
Figure 3.20 Dashboard : Button pour rejoindre un cours	91
Figure 3.21 Formulaire pour joindre un cours	92
Figure 3.22 Dashboard : Affichage des cours.....	92
Figure 3.23 Dashboard : Affichage détails du cours.....	93
Figure 3.24 Dashboard détails cours avec épreuve créer.....	93
Figure 3.25 Formulaire pour répondre aux questions de l'épreuve	94
Figure 3.26 Soumission de l'épreuve.....	94
Figure 3.27 Historiques des épreuves	95
Figure 3.28 Epreuve corrigée avec feedback	95
Figure 3.29 Prompt utilisateur utilisée pour la génération de feedback pour l'évaluation.	103
Figure 3.30 Résultat du test manuel sur la platform Azure AI studio	106
Figure 3.31 Résultat de test du modèle GPT 3.5 Turbo (cohérence, fluidité, enracinement)	106
Figure 3.32 Résultat de test du modèle GPT 3.5 Turbo (pertinence, similarité)	107
Figure 3.33 Résultat de test du modèle Llama 3 8B-Instruct (cohérence, fluidité, enracinement)	108
Figure 3.34 Résultat de test du modèle Llama 3 8B-Instruct (pertinence, similarité)	108

Figure 3.35 Résultat de test du modèle GPT 3.5 Turbo après ajustement et amélioration (cohérence, fluidité, enracinement).....	110
Figure 3.36 Résultat de test du modèle GPT 3.5 Turbo après ajustement et amélioration (pertinence, similarité)	110

0. Introduction générale

0.1. Contexte

Au cours des dernières années, grâce à l'avènement des *Large Language Models* ou LLM, nous avons fait un bond gigantesque en avant dans notre quête de plusieurs décennies pour construire des machines intelligentes. Bien avant l'émergence des LLM des avancées significatives avaient été réalisées. Ces modèles, qui s'inspirent des mécanismes complexes du cerveau humain, ont marqué une évolution notable dans le domaine de l'intelligence artificielle. Ils ont permis le développement d'applications capables de générer du contenu textuel, visuel et de programmation qui non seulement ressemble à celui produit par les humains, mais qui est également plausible et techniquement avancé, ouvrant ainsi la voie à des avancées sans précédent dans la création assistée par ordinateur. Les entreprises du monde entier ont commencé à expérimenter la nouvelle technologie avec la conviction qu'elle pourrait transformer les médias, la finance, le droit et les services professionnels, ainsi que les services publics tels que l'éducation [1].

Les modèles de langage large (LLM) sont des systèmes d'intelligence artificielle basés principalement sur l'architecture des *Transformers*. Cette méthode de traitement des données utilise des mécanismes d'attention pour optimiser la compréhension contextuelle et la génération de texte, permettant aux LLM de saisir les nuances linguistiques et de produire des réponses cohérentes et pertinentes. Proposé par des chercheurs de Google en 2017 et présentée dans un article de recherche intitulé « *Attention Is All You Need* » [2], les Transformers représentent une avancée majeure en intelligence artificielle, permettant une analyse et une prédiction de données textuelles avec une précision remarquable. Cependant, il est important de noter que tous les LLM ne sont pas construits sur cette architecture ; bien que la majorité le soient en raison de leur efficacité prouvée, d'autres approches existent également dans le domaine en constante évolution de l'IA.

Les systèmes d'intelligence artificielle générative, tels que ChatGPT lancé fin novembre 2022, ont révolutionné la perception publique de l'IA. Ces technologies, capables d'imiter les fonctions cognitives humaines pour générer du texte, des images, des vidéos, de la musique et du code, ont connu une adoption sans précédent, marquant un regain d'intérêt pour l'IA. Utilisés quotidiennement par des millions de personnes, les GenAI (IA Générative) offrent un potentiel d'adaptation à des applications spécifiques impressionnantes. Leur capacité à traiter l'information et à produire des connaissances pourrait transformer l'éducation en reproduisant des processus de pensée complexes, essentiels à l'apprentissage [3] [4].

0.2. Identification et formulation du problème

L'évaluation des épreuves théoriques est essentielle dans le système éducatif, car elle permet d'évaluer l'assimilation des connaissances par les étudiants. Toutefois, la correction manuelle présente des défis notables. **Le manque de temps et de ressources**, exacerbé par le nombre croissant d'étudiants par classe et la charge de travail des enseignants, limite la capacité à corriger efficacement. De plus, **la subjectivité inhérente** à la correction manuelle peut compromettre la fiabilité et l'uniformité des évaluations. Cette **tâche, souvent chronophage et laborieuse**, peut également empêcher les enseignants de se consacrer à d'autres activités pédagogiques. Enfin, le **temps restreint alloué à la correction** peut entraver la capacité des enseignants à fournir des retours détaillés et personnalisés. **Face à ces enjeux, l'implémentation d'un système d'assistance à la correction d'épreuves théoriques devient impérative pour renforcer l'efficacité, la précision et la cohérence des évaluations.** Les modèles de langage large (LLM) semblent prometteurs pour automatiser certaines tâches de correction et offrir un soutien précieux aux enseignants, en leur permettant de fournir des retours constructifs et personnaliser aux étudiants.

Le système développé dans le cadre de cette étude, visant à pallier les problèmes susmentionnés, sera dénommé **Bic Rouge**.

0.3. Questions de recherche

Vu le problème que nous venons de présenter, une question se pose :

Comment concevoir un système d'aide à la correction d'épreuves théoriques basé sur les LLM qui puisse améliorer l'efficacité, la précision et la cohérence du processus d'évaluation ?

La précédente question nous amène aussi à nous demander ceci :

- Comment l'intégration des LLM dans un système d'aide à la correction d'épreuves théoriques peut-elle renforcer l'efficacité, la précision et la cohérence du processus d'évaluation, en comparaison avec les méthodes traditionnelles ?
- Comment intégrer les LLM dans un système d'aide à la correction d'épreuves théoriques qui soit convivial et efficace pour les utilisateurs ?
- Comment mesurer l'impact du système d'aide à la correction d'épreuves théoriques sur la qualité de l'évaluation et l'apprentissage des étudiants ?

0.4. Formulation des hypothèses

A la suite des questions que nous venons de soulever, nous postulons que :

- Un système d'aide à la correction d'épreuves théoriques basé sur les LLM pourrait améliorer l'efficacité, la précision et la cohérence du processus d'évaluation.
- L'intégration des LLM dans un système d'aide à la correction d'épreuves théoriques convivial et efficace pour les enseignants pourrait améliorer l'expérience de correction et la qualité des évaluations.
- L'utilisation de métriques d'évaluation appropriées permettrait de mesurer l'impact du système d'aide à la correction d'épreuves théoriques sur la précision, la fiabilité et la cohérence des évaluations.

0.5. Justification du choix du sujet et motivations

L'évaluation des épreuves théoriques est une composante essentielle du système éducatif, car elle permet de mesurer l'assimilation des connaissances par les étudiants. Toutefois, la correction manuelle présente des défis significatifs, soulignant l'importance d'un système d'assistance à la correction pour améliorer l'efficacité, la précision et la cohérence de l'évaluation.

L'émergence des modèles de langage large (LLM) représente une avancée prometteuse pour répondre à ces besoins en automatisant les tâches de correction et en offrant un soutien aux enseignants. Les LLM ont démontré des capacités exceptionnelles dans diverses applications de traitement du langage naturel, y compris la compréhension, la génération de texte et la traduction automatique. Ces compétences indiquent que les LLM pourraient être utilisés efficacement pour la correction d'épreuves théoriques, en identifiant les erreurs grammaticales et orthographiques, en évaluant la cohérence et la structure des textes, et en vérifiant la compréhension des concepts fondamentaux.

Pour répondre à ce besoin spécifique, **nous avons pour objectif de développer une application web d'assistance à la correction d'épreuves théoriques**. Cette initiative s'inscrit dans un champ de recherche en plein essor, à la croisée de l'intelligence artificielle, du traitement du langage naturel et de l'éducation. Plusieurs études se sont intéressées à ce sujet [3] [4] [5] [6] [7] [8].

Ce projet de recherche vise à enrichir le corpus de connaissances en examinant l'application des modèles de langage à grande échelle (LLM) dans le processus de correction des épreuves théoriques. L'étude approfondira la compréhension des potentialités et des contraintes des LLM, tout en déterminant les pratiques optimales pour leur intégration efficace dans les systèmes d'assistance à la correction.

En outre, cette recherche comparera les performances des LLM avec celles des méthodes traditionnelles de correction manuelle, et mesurera l'influence de l'assistance automatisée sur la précision, la fiabilité et l'uniformité des évaluations.

Sur le plan social, les découvertes de cette étude pourraient offrir des bénéfices considérables aux différents intervenants du milieu éducatif dont principalement :

- Pour les enseignants, le système pourrait significativement réduire leur charge de travail. Cela leur permettrait de se concentrer davantage sur des aspects essentiels de l'enseignement, tels que la préparation du contenu pédagogique pour les étudiants. De plus, ce système pourrait aider l'enseignant à fournir des retours constructifs et personnalisés aux étudiants, améliorant ainsi l'expérience d'apprentissage.
- Les étudiants, quant à eux, pourraient jouir d'évaluations plus justes et promptes, accompagnées de commentaires plus approfondis et adaptés.
- Enfin, les décideurs politiques pourraient utiliser ces données pour orienter les politiques et initiatives visant à accroître l'efficacité et l'équité dans le système éducatif.

0.5.1. L'objectif général

Cette recherche a pour objectif principal de concevoir et réaliser un système (une application web) d'aide à la correction d'épreuves théoriques basé sur les modèles de langage large (LLM) pour améliorer l'efficacité, la précision et la cohérence du processus d'évaluation dans le système éducatif.

0.5.2. Les objectifs spécifiques

Pour arriver à bout de notre projet nous comptons :

- Réaliser une revue de la littérature sur l'évaluation des épreuves théoriques, en identifiant les défis et les enjeux actuels.
- Présenter une synthèse des modèles de langage large (LLM), en mettant l'accent sur leurs architectures, leurs capacités et leurs applications.
- Présentez les différentes techniques applicables à la correction d'épreuves théoriques et les capacités spécifiques des modèles GPT-3 et LLAMA-3 dans ce contexte.
- Concevoir une application d'aide à la correction d'épreuves théoriques efficace, conviviale et intuitive.
- Évaluer les performances du système d'aide à la correction d'épreuves théoriques sur l'ensemble de données d'épreuves théoriques.
- Évaluer l'impact du système d'aide à la correction d'épreuves théoriques sur la précision, la fiabilité et la cohérence des évaluations.

0.6. Méthodologie et délimitation du travail

Pour affiner le système, nous envisageons d'employer des méthodes analytiques, incluant des techniques expérimentales pour évaluer l'adéquation entre l'application développée et le problème identifié, ainsi que des techniques documentaires pour une compréhension approfondie des méthodes usuelles et des améliorations potentielles.

Ce projet se concentre sur la correction d'épreuves théoriques dans divers domaines académiques, tels que les sciences humaines, les sciences sociales, le français, l'anglais, l'économie, le droit, etc. Le système conçu facilitera la correction de d'épreuves rédigés en français ou en anglais.

Le mémoire exclura les domaines suivants :

- Création d'un système de correction pour les épreuves mathématiques ou scientifiques.
- Traduction automatique de documents.
- Analyse de sentiments ou d'opinions dans les textes.
- Élaboration d'un système de tutorat ou d'enseignement personnalisé.

0.7. Subdivision du travail

Excepté l'introduction et la conclusion générales, ce travail est structuré de la manière suivante :

1. Au premier chapitre, intitulé « **Généralités sur la correction d'épreuves théorique et les modèles de langage large (LLM)** », nous examinons en détail toute la théorie nécessaire à la compréhension de notre étude.
2. Le second chapitre, « **Techniques de correction automatique d'épreuves, modèles GPT-3.5 et LLAMA-3, et conception du Système** », détaille les aspects cruciaux de la correction automatique des épreuves qui sont essentiels à notre recherche et décrit

progressivement la conception du système d'aide à la correction d'épreuves à caractère théorique.

3. Enfin, le troisième chapitre, « **Conception finale, réalisation et tests** », conclut la conception et aborde les éléments clés de l'implémentation, en s'appuyant sur la conception préalablement établie, avant de présenter les résultats des tests effectués.

Chapitre 1

Généralités sur la correction d'épreuves théoriques et les modèles de langage large (LLM)

1.1 Introduction partielle

Dans ce chapitre, nous procéderons à une analyse concise de la littérature existante sur l'évaluation des épreuves théoriques. Cette exploration mettra en exergue les défis actuels et les questions émergentes dans ce domaine. Nous aborderons également les Modèles de Langage larges (LLM), en proposant une analyse de leur mécanisme opérationnel, de leur processus d'entraînement, de leur architecture et composants, ainsi que des exemples concrets et de leurs multiples applications dans des scénarios pratiques. L'objectif est de fournir une compréhension approfondie des modèles de langage larges (LLM) et d'examiner leur impact potentiel sur l'évaluation des épreuves ainsi que sur l'éducation en général.

1.2 Évaluation des épreuves théoriques : Définition, enjeux et défis

1.2.1 Définition

L'évaluation des épreuves théoriques constitue une démarche systématique et rigoureuse visant à quantifier et à qualifier les acquis académiques des étudiants dans une discipline déterminée. Ce processus s'articule autour de méthodes d'examen variées, incluant mais ne se limitant pas à des tests écrits et oraux, tels que les questionnaires à choix multiples, les essais argumentatifs, les exposés et les travaux de recherche. Cette évaluation, intrinsèque au parcours pédagogique, permet aux éducateurs d'apprécier les compétences des apprenants en compilant, en évaluant et en interprétant des données probantes relatives à leur rendement dans un cadre normé et surveillé [11] [12].

L'évaluation des épreuves théoriques vise à [11] [13]:

- Attester de la consolidation des connaissances acquises par les étudiants et d'évaluer leur degré de compétence concernant le matériel didactique d'un cours spécifique ou d'un curriculum.
- Recueillir et analyser des données sur les performances individuelles et collectives des apprenants, ce qui permet aux enseignants d'ajuster leurs stratégies pédagogiques et de focaliser leurs efforts là où c'est le plus nécessaire.
- Déetecter les apprenants éprouvant des difficultés et élaborer pour eux des stratégies d'accompagnement personnalisées ou des programmes de remise à niveau.
- Sélectionner les candidats qualifiés pour l'accès à des formations supérieures ou l'attribution de bourses académiques.
- Examiner et mesurer l'efficacité des programmes d'enseignement en place et procéder aux ajustements requis pour leur amélioration continue.

1.2.2 Enjeux

L'analyse systématique des épreuves théoriques constitue un élément fondamental du cadre éducatif, impliquant des conséquences significatives pour les apprenants, le corps professoral et les entités académiques. Cette évaluation permet non seulement de mesurer l'acquisition des connaissances par les étudiants, mais aussi de calibrer l'efficacité pédagogique des enseignants et la qualité des programmes d'études proposés par les institutions. Ainsi, elle sert de baromètre pour l'excellence académique et l'amélioration continue du processus éducatif [11]. Différentes parties prenantes bénéficient de cette analyse de plusieurs manières spécifiques [13]:

- Pour les étudiants, elle permet d'apprécier objectivement leur progression et de leur offrir un retour constructif sur leur processus d'apprentissage.
- Pour les enseignants, elle est un outil d'évaluation indispensable qui renseigne sur les performances individuelles et collectives des étudiants, permettant ainsi d'affiner les méthodes pédagogiques et d'intervenir de manière ciblée.

- Pour les institutions éducatives, l'évaluation des épreuves théoriques garantit la certification des compétences acquises par les étudiants et assure le maintien de la qualité des programmes d'enseignement.

1.2.3 Défis

Bien que l'évaluation traditionnelle des épreuves théoriques soit cruciale, elle comporte des défis significatifs qui nécessitent une analyse approfondie.

Les principaux sont [14] [13]:

- **Subjectivité et biais des évaluateurs** : Les évaluations sont susceptibles de fluctuer selon les critères personnels des évaluateurs, influencées par leur état émotionnel, leur expérience et leurs préjugés inhérents.
- **Incohérence et manque de fiabilité inter-évaluateurs** : Il existe une variabilité significative dans les notes attribuées par différents évaluateurs pour une même évaluation, remettant en question la fiabilité des résultats obtenus.
- **Limites dans l'évaluation de compétences spécifiques** : Les méthodes d'évaluation traditionnelles peuvent échouer à apprécier des compétences clés telles que la pensée critique, la créativité et la résolution de problèmes.
- **Surcharge de travail pour les enseignants** : La correction des évaluations représente une charge de travail conséquente, pouvant réduire le temps alloué à d'autres tâches pédagogiques essentielles

Les enjeux associés aux méthodes d'évaluation actuelles peuvent entraîner des conséquences préjudiciables pour les étudiants, les enseignants et les établissements d'enseignement. Par conséquent, il est impératif de développer des stratégies d'évaluation théorique révolutionnaires qui assurent une plus grande objectivité, cohérence et fiabilité, tout en réduisant la charge de travail des enseignants. Dans cette optique, l'application des Modèles de Langage larges (LLM) pour l'assistance à la correction des évaluations théoriques se présente comme une alternative

prometteuse, offrant la possibilité de relever ces défis et d'améliorer la qualité des évaluations académique.

1.3 Exploration des modèles de langage larges (LLM)

1.3.1 Présentation et définition

Les Modèles de Langage Large (*Large language model*, LLM) sont des modèles de traitement du langage naturel ou (*Natural Language Processing*, NLP) basés sur des réseaux de neurones profonds (*Deep neural network*, DNN), notamment les architectures de type *Transformer*. Introduits par Vaswani et al en 2017 dans l'article « Attention is all you need » [2]. Ils se distinguent par leur capacité à utiliser des mécanismes d'attention, qui leur permettent de traiter et de générer du texte de manière contextuellement pertinente, assurant ainsi une compréhension nuancée et une production de contenu textuel d'une cohérence remarquable. Cette approche révolutionnaire a ouvert la voie à des applications NLP plus sophistiquées, capables de comprendre et de répondre aux nuances complexes du langage humain [5] [16].

Les LLM, tels que GPT (*Generative Pre-trained Transformer*), sont entraînés sur d'immenses corpus de données textuelles, englobant une vaste étendue de l'internet, incluant des sources diversifiées telles que Wikipedia et Common Crawl. Ces modèles acquièrent la capacité de réaliser une multitude de tâches en s'appuyant sur un nombre restreint d'exemples spécifiques. Cette méthode d'apprentissage, qui nécessite une supervision minimale et intègre l'apprentissage par renforcement à partir de retours humains (*Reinforcement Learning from Human Feedback*, RLHF), démontre la polyvalence et l'efficacité des modèles dans l'acquisition de connaissances à partir de vastes quantités de données non structurées. Grâce à RLHF, ces modèles peuvent ajuster leurs réponses et comportements en fonction des évaluations humaines, améliorant ainsi continuellement leurs performances [16]. Ces modèles sont capables d'effectuer des tâches multiples sans être spécifiquement programmés pour chacune, grâce à leur entraînement non supervisé sur de vastes corpus textuels. Cela leur permet de générer des réponses pertinentes dans divers contextes, allant de la génération de texte à la traduction automatique et au résumé automatique [16].

Un autre exemple emblématique de cette technologie est le modèle BERT (*Bidirectional Encoder Representations from Transformers*), introduit en 2019, qui se distingue par l'utilisation de *Transformers* bidirectionnels. Cette approche améliore significativement la compréhension contextuelle des mots au sein des phrases, en analysant le contexte dans les deux sens, gauche et droite, ce qui augmente la précision et la pertinence des résultats obtenus dans les tâches de NLP. Les LLM trouvent des applications variées, allant de l'éducation au service client, en passant par la recherche scientifique et la création de contenu, démontrant ainsi leur capacité à transformer divers secteurs d'activité [16] [17].

Dans cette section, nous allons nous concentrer spécifiquement sur les LLM basés sur l'architecture *Transformer*, en raison de leur popularité et de leurs performances impressionnantes dans diverses tâches de NLP. Toutefois, nous introduirons également d'autres types d'architectures pour offrir une vue d'ensemble complète du paysage actuel des LLM.

1.3.2 Fonctionnement

Comme mentionné ci-haut, bien que la majorité des LLM fonctionnent grâce à une architecture de réseaux de neurones profonds appelée *Transformer*, ce n'est pas toujours le cas et ce n'est pas obligatoire. Il existe plusieurs autres architectures utilisées pour la création des LLM, telles que les réseaux récurrents (RNN), les réseaux de neurones convolutifs (CNN) et les modèles basés sur l'attention. En générale, le mécanisme opérationnel des LLM peut être décrit en trois étapes principales : **entrée du texte ou input (prompt)**, **traitement du texte par le LLM**, et **sortie du texte ou output (complétion)**, comme illustré à la figure suivante.



Figure 1.1 Mécanisme opérationnel des LLM cas de traduction automatique

1.3.2.1 Entrée du Texte ou Input (Prompt)

L'utilisateur fournit un texte initial, appelé **prompt**, qui peut être une phrase, une question, ou un fragment de texte. Ce prompt sert de point de départ pour le modèle.

1.3.2.2 Traitement du Texte par le LLM

L'architecture interne du modèle est composée de deux éléments principaux : un composant d'encodage et un composant de décodage, reliés par des connexions. Le composant d'encodage consiste en une série d'encodeurs ; bien que la **Figure 1.2** illustre six encodeurs superposés, ce chiffre n'est pas figé et d'autres configurations peuvent être explorées. De manière similaire, le composant de décodage est constitué d'une série de décodeurs correspondant en nombre aux encodeurs [18].

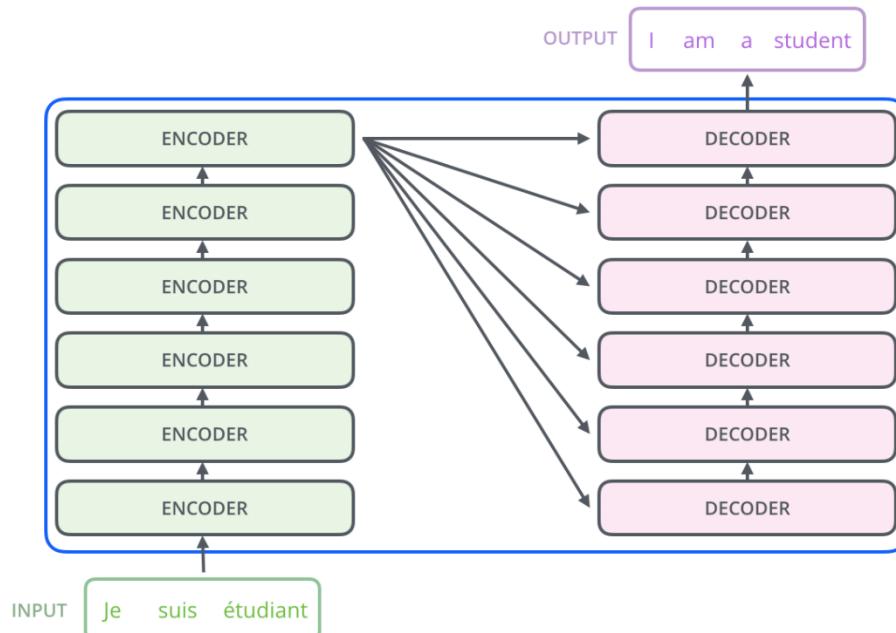


Figure 1.2 Schéma des Composants d'Encodage et de Décodage [18]

Une fois le prompt reçu, le LLM procède aux étapes suivantes :

1. **Encodage** : Le texte d'entrée est converti en une séquence de vecteurs de représentation à l'aide d'un encodeur, dont les détails seront exposés plus loin dans la section dédiée à l'architecture et aux composants. Chaque mot ou jeton (token) du texte est transformé en une représentation vectorielle qui capture ses caractéristiques contextuelles, comme illustré par la **Figure 1.3** et **Figure 1.4**.

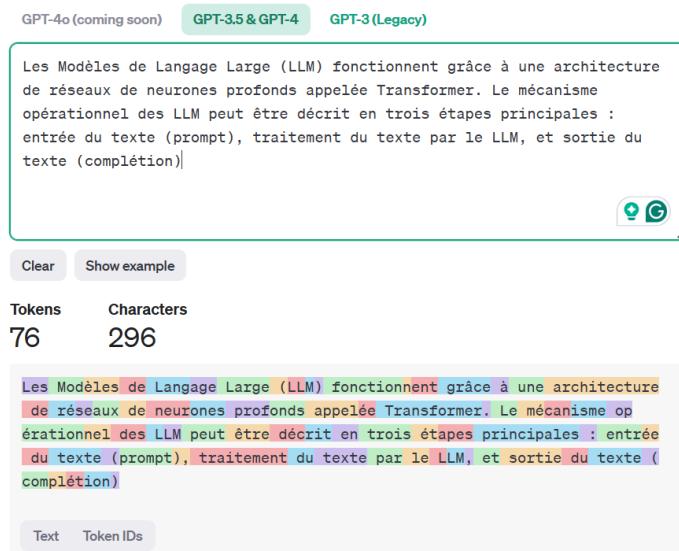


Figure 1.3 OpenAI Tokenizer pour GPT-3.5 & GPT-4

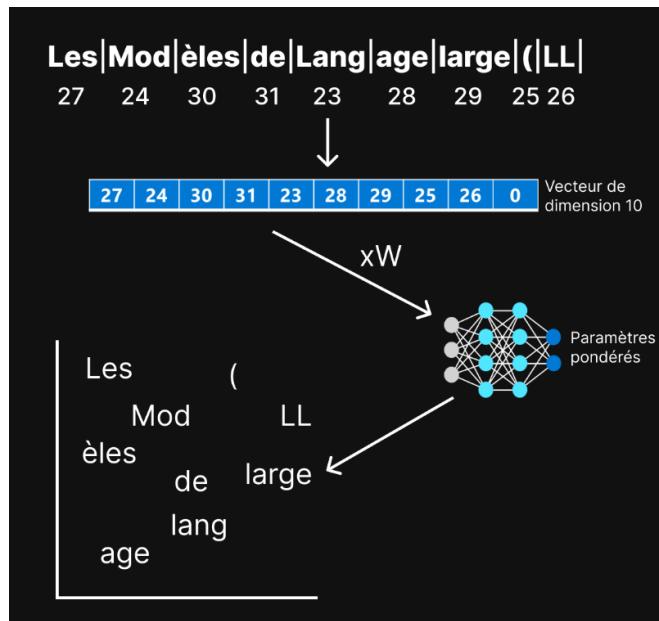


Figure 1.4 Comment un réseau de neurones (NN) "comprend" une phrase

2. **Mécanisme d'Attention** : Le mécanisme d'attention permet au modèle de pondérer l'importance de chaque mot du prompt en fonction de tous les autres mots de la séquence. Cela permet au modèle de comprendre le contexte global et les relations entre les mots, comme illustré à la **Figure 1.5** où pour le codage du mot « *it* », une partie du mécanisme d'attention se concentre sur « *The Animal* » [20].

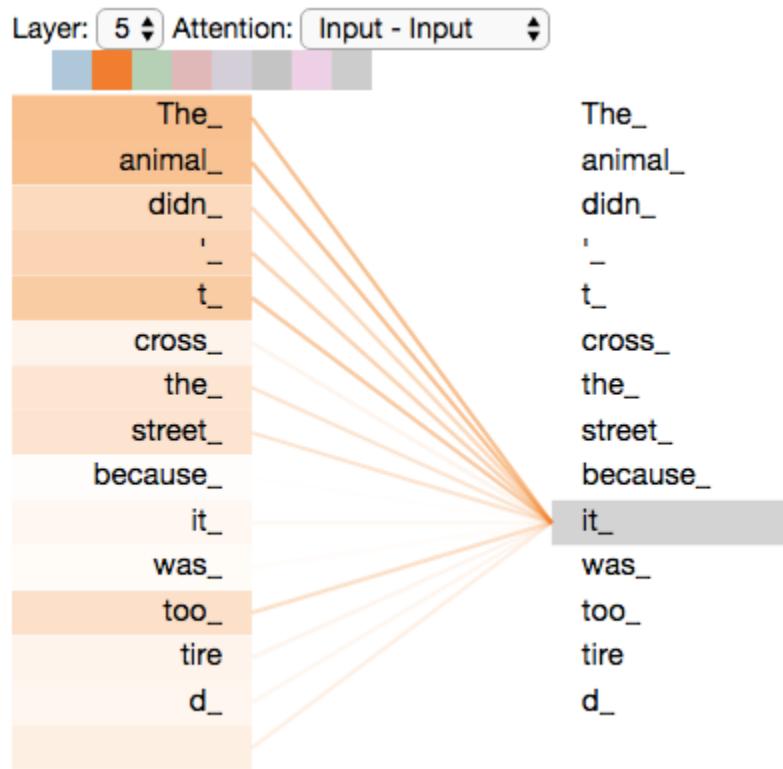


Figure 1.5 Mécanisme d'attention (module d'auto-attention) avec Tensor2Tensor [18]

3. **Décodage** : Le décodeur utilise les représentations contextuelles générées par l'encodeur pour produire une séquence de sortie. À chaque étape, le modèle prédit le mot suivant en tenant compte du prompt initial et des mots déjà générés [20].
4. **Génération de la Complétion** : Le modèle continue de générer des mots jusqu'à ce qu'il atteigne une condition d'arrêt pré définie, comme la fin d'une phrase ou un nombre maximal de tokens [20].

1.3.2.3 Sortie du Texte ou output (Compléction)

Le modèle génère une compléction de texte en parfaite adéquation avec le prompt initial, assurant une continuité contextuelle. Cette production peut revêtir diverses formes : une réponse à une interrogation, le développement d'une narration, une traduction illustrée à la manière de la **Figure 1.7**, ou toute autre variété de texte sollicitée par l'utilisateur.

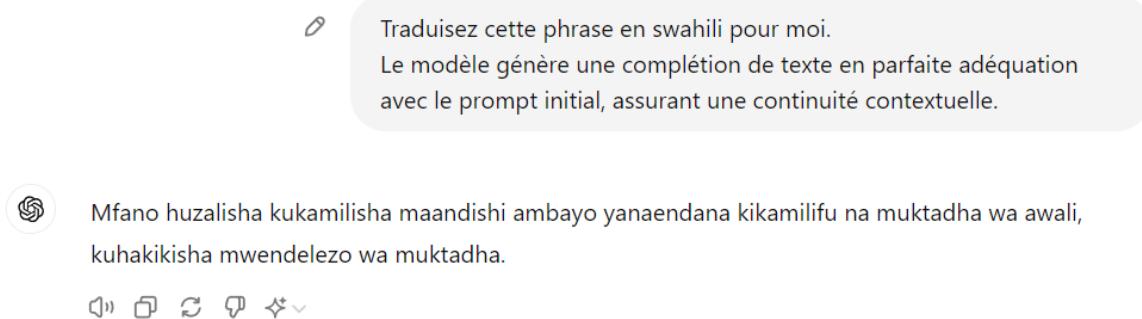


Figure 1.6 Compléction de texte contextuelle avec GPT-4 Omni sur la plateforme ChatGPT

En résumé, comme illustré par la **Figure 1.7**, le processus de génération de texte par un modèle de type *Transformer* se déroule de la manière suivante : le modèle reçoit en entrée une phrase en français "Je suis étudiant", et produit la traduction anglaise correspondante, "I am a student". Le processus commence par l'encodage des mots de la phrase d'entrée en vecteurs d'embedding, auxquels sont ajoutés des signaux temporels pour capturer la position des mots dans la phrase. Ces embeddings sont ensuite traités par une série d'encodeurs qui extraient les caractéristiques contextuelles.

Les résultats des encodeurs sont passés aux décodeurs, qui génèrent le texte de sortie mot par mot, en s'appuyant sur les *embeddings* temporels et les précédentes sorties du modèle. Finalement, une couche linéaire suivie d'une fonction *Softmax* convertit les représentations en probabilités pour chaque mot possible, permettant ainsi de sélectionner le mot le plus probable à chaque étape du décodage. Ce mécanisme assure une traduction précise et cohérente, démontrant l'efficacité des *Transformers* dans le traitement et la génération de texte.

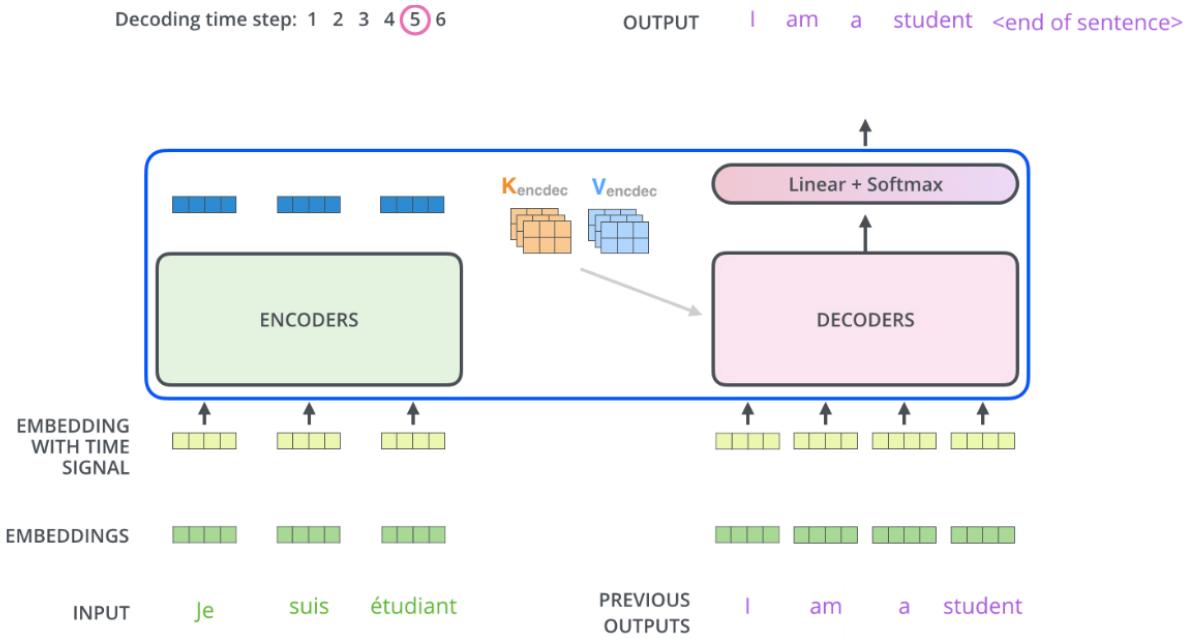


Figure 1.7 Processus de Traduction avec un Modèle *Transformer* [18]

1.3.2.4 Entraînement des LLMs

L'entraînement des LLMs est un processus complexe et intensif, qui s'appuie sur des corpus de données vastes et diversifiés pour enseigner aux modèles les structures linguistiques et les relations contextuelles. Les datasets couramment utilisés incluent *Wikipedia*, *Common Crawl*, *WebText*, *GitHub*, *Livres*, *arXiv*, *StackExchange*, et d'autres collections de textes disponibles publiquement. Ces ensembles de données fournissent une richesse de contenu couvrant une variété de domaines, ce qui permet aux modèles d'acquérir une compréhension large et nuancée du langage naturel. L'entraînement de ces modèles nécessite des ressources de calcul extrêmement importantes, souvent réalisées sur des infrastructures de superordinateurs ou des clusters de GPU, pour traiter les volumes massifs de données et ajuster les millions, voire les milliards, de paramètres des réseaux de neurones.

Les étapes générales de l'entraînement comprennent le **pré-entraînement**, où le modèle est initialement formé sur les vastes *datasets* pour apprendre les structures de base du langage, suivi du **fine-tuning**, où il est affiné sur des tâches spécifiques avec des *datasets* annotés plus petits et l'**évaluation**, où les performances du modèle sont mesurées à l'aide de métriques standardisées pour s'assurer de sa robustesse et de sa capacité de généralisation. Ce processus permet de développer des modèles capables de générer du texte, répondre à des questions, et effectuer diverses autres tâches de traitement du langage naturel avec une précision et une cohérence impressionnante, comme illustré dans la **Figure 1.8** [2] [16] [16] [21].

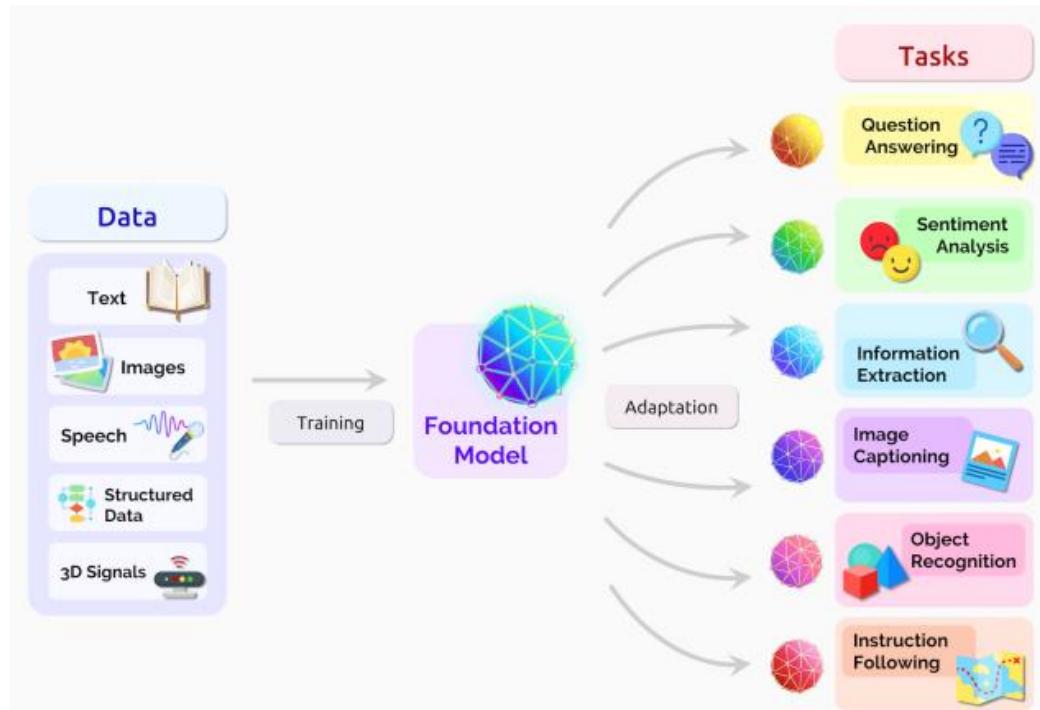


Figure 1.8 Processus d'Entraînement des LLM [21]

1.3.2.5 Pré-entraînement

Le pré-entraînement constitue la phase initiale essentielle dans le développement des LLM. Durant cette phase, le modèle est soumis à une large gamme de données, issues de sources diversifiées. Cette exposition vise principalement à inculquer au modèle les structures linguistiques de base et les relations contextuelles propres à la langue. Le modèle ainsi obtenu, souvent désigné sous le terme de « *Foundation model* » [21], traite d'importants volumes de

données, acquérant une compréhension fine des subtilités sémantiques et syntaxiques. Ce processus le prépare à des applications spécifiques, qui seront affinées lors d'étapes subséquentes de fine-tuning.

1.3.2.6 Fine-tuning

Après le pré-entraînement, le modèle subit une phase de fine-tuning, une étape clé du processus d'entraînement des LLMs. Cette phase utilise la technique de *transfer learning* [21], où les connaissances acquises lors du pré-entraînement sont transférées et adaptées pour des tâches spécifiques. Le modèle est ajusté avec des datasets plus petits et spécialisés, souvent annotés manuellement, pour améliorer sa performance sur des applications précises telles que la génération de texte, la classification de texte, ou la traduction, etc. Ce processus permet de raffiner le modèle, connu sous le nom de *foundation model* ou *Base LLM*, pour qu'il excelle dans des domaines particuliers, exploitant ainsi les capacités contextuelles et linguistiques développées lors de la phase initiale de pré-entraînement [4] [16] [22] [23].

1.3.2.7 Évaluation

L'évaluation des LLMs est une étape cruciale pour déterminer leur efficacité après les phases de pré-entraînement et de fine-tuning. Après avoir affiné le modèle pour des tâches spécifiques avec des *datasets* annotés, il est nécessaire de mesurer sa performance en utilisant diverses métriques standardisées. Les principales métriques d'évaluation incluent **la perplexité**, qui mesure la capacité du modèle à prédire un ensemble de données, **l'accuracy** pour évaluer la précision des tâches de classification, et le **BLEU (Bilingual Evaluation Understudy)** score pour évaluer la qualité des traductions automatiques. Ces évaluations sont effectuées sur des ensembles de test indépendants pour s'assurer que le modèle généralise bien et maintient une performance robuste sur des données nouvelles et non vues [24] [20] [25].

1.3.3 Architecture

L'entraînement des LLM est un processus complexe qui implique la formation des réseaux de neurones sur d'immenses corpus de données textuelles. Ce processus permet aux LLM de comprendre et générer du texte avec une haute précision et une cohérence contextuelle.

Différents types d'architectures de réseaux de neurones sont utilisées pour créer des LLM, chacune apportant des avantages uniques dans le traitement du langage naturel (NLP).

Parmi ces architectures, on trouve les ***Convolutional Attention Networks*** (CAN), qui combinent des réseaux neuronaux convolutifs (CNN) pour l'extraction de caractéristiques locales et des mécanismes d'attention pour la capture du contexte global. Les réseaux neuronaux récurrents (RNN), tels que les ***Long Short-Term Memory*** (LSTM) et les ***Gated Recurrent Units*** (GRU), sont également couramment utilisés pour leur capacité à gérer des séquences temporelles et à conserver l'information contextuelle sur de longues périodes. Les modèles plus récents incluent l'architecture ***Mamba*** [26] et des architectures hybrides comme ***Transformer-XL*** [27] et ***Transformer-Encoder-Decoder***, qui intègrent des composants de Transformers classiques pour améliorer la gestion du contexte à long terme et la génération de texte.

Cependant, comme mentionné au début de cette section, nous nous concentrerons principalement sur l'architecture Transformer, actuellement la plus populaire et la plus utilisée en raison de ses nombreux avantages, notamment sa scalabilité. Les Transformers ont révolutionné le domaine du traitement du langage naturel (NLP) grâce à leur capacité à traiter des séquences de données en parallèle et à capturer les dépendances à long terme de manière plus efficace que les architectures précédentes [16] [2].

1.3.3.1 Les transformers

Les modèles en question sont définis par une architecture générique structurée comme suit :

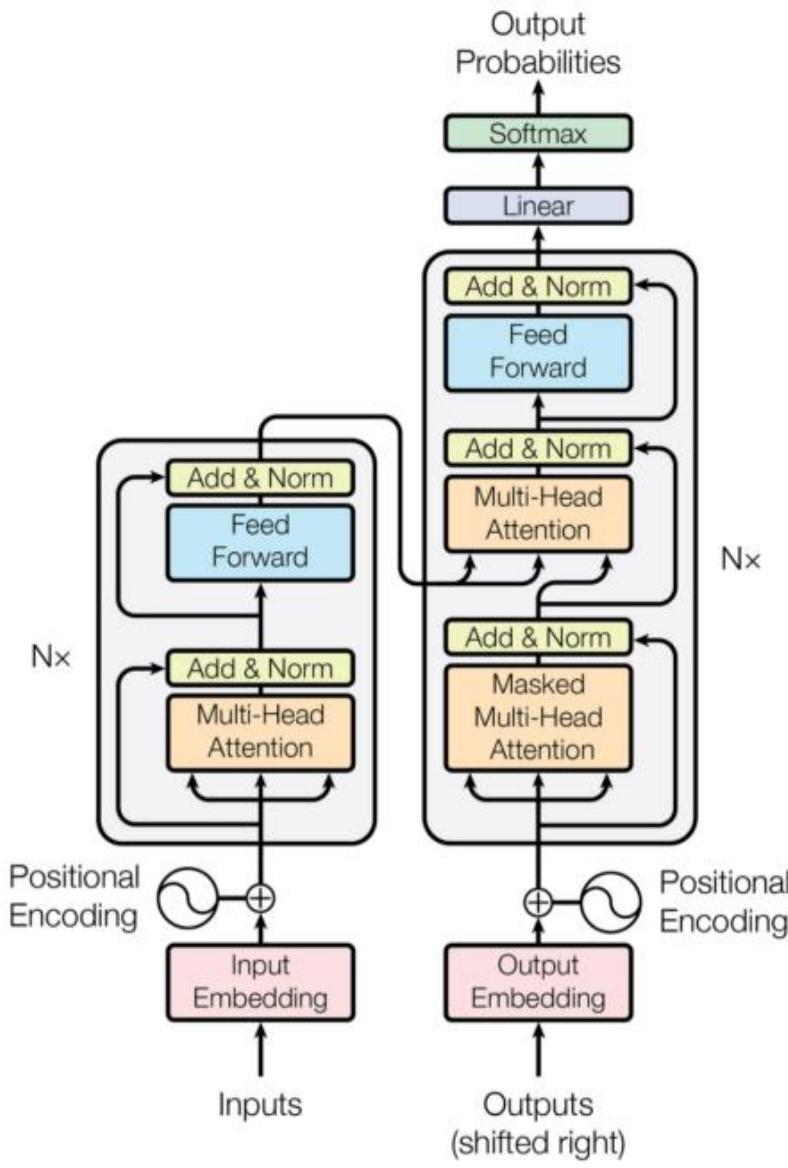


Figure 1.9 Architecture générique des *transformers* [2]

Les *transformers* sont constitués d'un encodeur et d'un décodeur, comme illustré sur la **Figure 1.9** (bien que certaines implémentations n'en utilisent qu'une partie selon la tâche) [28] [16].

Dans la suite de cette section, nous allons présenter une explication succincte de chacun des modules, présents sur la figure 1.9 en décrivant leurs fonctions et leur rôle selon l'ordre dans lequel les données traversent le modèle :

- 1) **Module d'embedding** : Nous savons que les données textuelles doivent être présentées au modèle sous forme numérique. Elles doivent donc être transformées avant de les passer aux parties suivantes. Néanmoins, vu que la représentation des entrées a un impact significatif sur les performances d'un modèle, cette représentation doit être bien choisie. Un choix intuitif, et qui s'avère être performant, est de tout faire pour que si deux termes ont des sens proches, ils aient aussi des représentations vectorielles proches. Cela est réalisé par différentes techniques que nous présenterons dans cette section, mais c'est là le rôle de la couche d'enchâssement (*embedding*) [29].
- 2) **L'encodage positionnel (positionnal encoding)** : Ce module ajoute l'information sur la position relative de chacun des éléments placés en entrée par rapport aux autres. Cela pallie le problème de perte d'information sur la position des mots quand on utilise un réseau non séquentiel comme les réseaux récurrents. Donc, la position de chaque terme de la séquence placée en entrée est encodée dans un vecteur puis ajoutée à l'encodage global du terme. L'un des encodages les plus utilisés est celui basé sur les fonctions trigonométriques tel qu'introduit dans [2], [29].
- 3) **Module d'auto-attention** [29]: La couche d'attention, présentée en première position dans la boîte de l'encodeur, est en fait une couche dite de self-attention car elle opère sur la même séquence d'entrée. L'opération est réalisée pour permettre au modèle d'avoir une représentation de l'importance des termes dans la séquence d'entrée, les uns par rapport aux autres.

Pour illustration, considérons la phrase suivante : *Walter est malade, il préfère se reposer.*

Dans cette phrase, l'un des constats qu'on peut faire est que, le nom "Walter" est beaucoup plus lié au prénom "il" qu'au verbe "préférer". C'est à l'établissement des tels liens dans les représentations que sert le module d'auto-attention ici présenté. Il est important que ce lien soit implicitement présent dans les représentations, pour que le traitement soit efficace comme illustré par la **Figure 1.5** lors de la présentation des

mécanismes d'attention dans la section 1.3.2. Donc cette couche est en fait un prolongement de celle d'*embedding*. Il s'agit ici d'un mécanisme basé sur le produit scalaire mis à l'échelle (*scaled dot-product*). En effet, très brièvement, l'idée du *scaled dot-product attention* consiste à opérer une recherche des termes sur lesquels focaliser l'attention de la même façon qu'on réalise la recherche de la signification d'un mot dans un dictionnaire. Supposons qu'on veuille avoir la signification d'un mot dont on ne connaît pas l'orthographe exacte. Pour retrouver ce dernier dans un dictionnaire, il suffit de rechercher le mot qui ressemble le plus à l'orthographe que nous estimons être la plus vraisemblable. Mathématiquement, cette recherche de similitude correspond à un produit scalaire. Similairement, le *scaled dot-product* consiste à générer trois éléments qui sont la clé ou *key* k, la valeur ou *value* v et la requête ou *query* q. La requête correspond au mot qu'on cherche (orthographié selon ce que nous pensons), la clé correspond au mot présent dans le dictionnaire et la valeur correspond à la signification associée. Si on supposait qu'il existe plusieurs termes du dictionnaire qui s'orthographient presque de la même façon que le mot qu'on cherche, on devra passer par une mesure de similarité avant de se décider sur le sens le plus probable. Cela correspond à réaliser le produit de tous les k par les q présents, puis à normaliser l'ensemble des résultats de manière à ce qu'ils représentent des mesures de probabilité, et finir par choisir le sens v le plus probable.

Pour aller plus vite, on implémente ce processus en considérant tous les k, q et v au même moment de manière à réaliser le calcul une fois pour toutes. Cela revient à regrouper tous les k, q et v dans des matrices K, Q et V. Ce qui donne la relation qui définit l'attention par produit scalaire mis à l'échelle [2] [29]:

$$\text{Attention}(Q, K, V) = \text{Softmax}\left(\frac{Q \cdot k^T}{\sqrt{d_k}}\right) \cdot V \quad \text{Équation 1.1}$$

Dans cette relation, $\text{Attention}(Q, K, V) = \text{Softmax}\left(\frac{Q \cdot k^T}{\sqrt{d_k}}\right) \cdot V$

Équation 1.1, le terme $\sqrt{d_k}$ permet de mettre à l'échelle le résultat du produit scalaire de Q par K, c'est-à-dire $Q \cdot k^T$. Il faut noter que d_k est la dimension d'une clé, et que cette normalisation permet d'améliorer les performances du modèle mais elle n'est pas la seule envisageable [29]. Il est aussi important de remarquer que la couche d'attention utilise trois termes pour arriver à bout du problème. Ces trois termes sont obtenus par une transformation linéaire dont les poids sont appris à travers un réseau de neurones simple. Il faut aussi noter que l'on utilise parallèlement plusieurs modules d'attention pour capture toutes les caractéristiques des séquences (on parle de *multi-head attention*). Pour une plus ample illustration, voir la **Figure 1.10** [29].

- 4) **Le module *feed-forward*** : Il s'agit en fait d'un réseau de neurones de propagation avant classique. Il permet de réaliser le traitement qui fait suite à l'attention [29].
- 5) **Couche d'attention encodeur-décodeur** : Il s'agit de la couche qui reçoit les données en provenance de l'encodeur. Il s'agit ici d'une couche d'attention et non d'autoattention comme c'était le cas pour la première couche de l'encodeur. En effet, contrairement à la couche de self-attention, pour laquelle tous les trois paramètres sont calculés à partir de la même séquence, la couche d'attention ici prend les clés K et valeurs V provenant de l'encodeur mais une requête Q provenant du décodeur. Une autre couche *feed-forward* suit celle-ci et a le même rôle que celle de l'encodeur [29].
- 6) **Module d'attention masquée** : Il s'agit de la première couche du décodeur. C'est aussi un module de self-attention auquel on ajoute le masquage. Ce module est dit masqué suite au fait que, comme le décodeur est un module de génération, on ne regarde que les termes précédemment générés, en masquant les termes qui seront probablement générés aux pas d'après. Cela est réalisé en rendant juste leurs probabilités nulles [29].

7) **Module linéaire final** : Il s'agit d'un réseau de neurones classique pour réaliser la déduction finale, le tout étant passé à la fin à travers une opération *softmax* qui permet de transformer les résultats en probabilité d'éléments générés (cela permet de choisir le terme le plus vraisemblable à générer comme sortie).

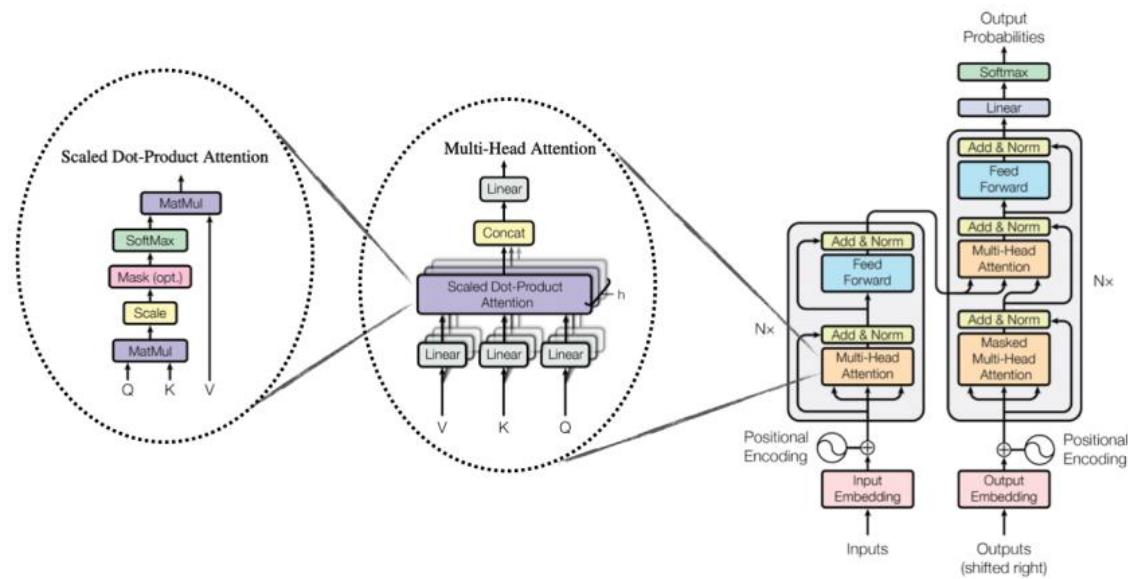


Figure 1.10 Vue éclatée d'un *transformer* [29]

1.3.3.2 Longueur du Contexte (*context length*)

La longueur du contexte, ou *context length*, est un aspect crucial dans les performances des LLMs. Elle désigne la capacité du modèle à prendre en compte un certain nombre de *tokens* (unités de texte) en input et lors de la génération de texte ou de la réponse à une requête. Par exemple, **GPT-3.5** et **GPT-4** peuvent gérer des contextes allant jusqu'à 2048 et 8192 *tokens* respectivement, ce qui leur permet de conserver une cohérence et une pertinence sur de longues séquences de texte. **Claude**, développé par *Anthropic*, peut traiter jusqu'à 100k *tokens*, offrant

une capacité exceptionnelle pour des tâches nécessitant un contexte étendu. De même, les modèles **LLAMA** de Meta AI ont des capacités de contextes variables en fonction de leurs configurations spécifiques. **Gemini de Google**, une autre avancée récente, se distingue également par sa capacité à gérer de grands contextes, facilitant des applications complexes et détaillées [16] [21] [30].

1.3.4 Types des LLM [31]

Les LLM peuvent être classés en fonction de plusieurs critères : leur architecture, les données d'entraînement utilisées et leur champ d'application. Ces modèles sont conçus pour diverses tâches, allant de la génération de texte à la reconnaissance vocale, en passant par la génération d'images et la multimodalité.

- **Reconnaissance audio et vocale (Audio and speech recognition)** : Pour la reconnaissance vocale, les modèles de type **Whisper** constituent un excellent choix en raison de leur polyvalence et de leur capacité à gérer la reconnaissance vocale multilingue. **Whisper** est entraîné sur une large gamme d'éléments audio, ce qui le rend performant pour la transcription et la traduction automatique de la parole. D'autres modèles similaires incluent **DeepSpeech** de Mozilla, qui utilise des réseaux de neurones profonds pour la reconnaissance vocale précise, et **Wav2Vec 2.0** [32] de Meta AI, connu pour sa capacité à apprendre à partir de données audios non annotées.
- **Génération d'images (Image generation)** : Pour la génération d'images, **DALL-E** et **Midjourney** sont parmi les options les plus connues. **DALL-E**, développé par OpenAI, utilise une version modifiée de **GPT-3** pour générer des images à partir de descriptions textuelles, comme illustré dans la **Figure 1.11. Midjourney**, un autre modèle puissant, permet de créer des images réalistes à partir de prompts textuels complexes. **Stable Diffusion** est également notable pour sa capacité à générer des images haute résolution à partir de textes, utilisant une approche de diffusion progressive pour affiner les images générées [33] [34].



Figure 1.11 Génération d'images par un prompt avec *DALL-E* [31]

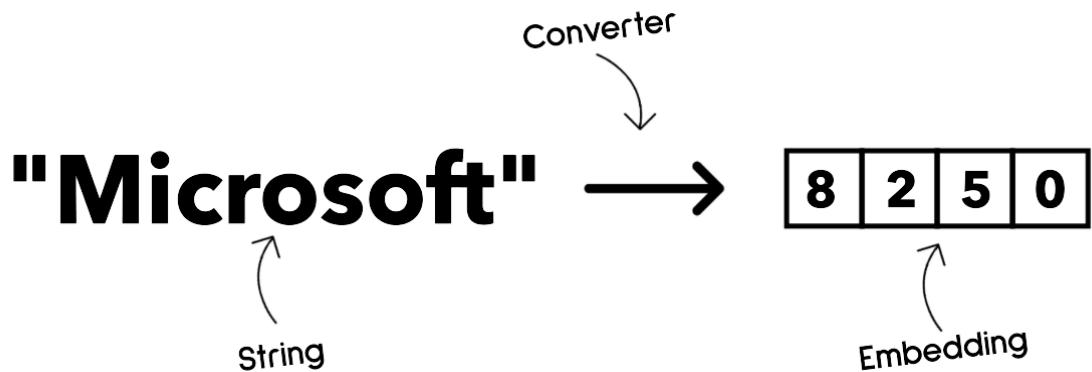
- **Génération de texte et de code (*Text and code generation*)** : La génération de texte est la fonction principale de nombreux LLM, avec des modèles tels que ***GPT-3.5***, ***GPT-4***, ***Gemini***, ***LLaMa***, et ***Claude***. ***GPT-3.5*** et ***GPT-4***, développés par OpenAI, sont connus pour leur capacité à générer du texte cohérent et pertinent sur de longues séquences. ***Gemini***, quant à lui, se distingue par sa scalabilité et ses performances élevées dans diverses tâches de NLP. D'autres modèles comme ***BLOOM*** [35], un modèle open-source développé par *BigScience*, et ***T5*** (*Text-to-Text Transfer Transformer*) [36] de Google, sont également largement utilisés pour la génération de texte. En plus de la génération de texte, certains modèles sont spécialement conçus pour la génération de code. Par exemple, ***StarCoder*** et ***StarCoderBase*** [37] sont des LLMs pour le code, entraînés sur des données sous licence permissive provenant de GitHub. Ces modèles sont optimisés pour comprendre et générer du code dans divers langages de programmation, facilitant ainsi les tâches de développement logiciel automatisé. De

même, **Code Llama** [38] est une famille de LLMs pour le code, basée sur **LLaMa 2** et **CodeParrot** [31].

Dans notre travail, nous nous sommes proposés d'utiliser LLaMa-2 et GPT-3.5, qui sont reconnus pour leur efficacité et leur précision dans la génération de texte (développés plus en détail dans le chapitre suivant). Ces modèles offrent des avantages uniques en termes de compréhension contextuelle et de génération de contenu pertinent, ce qui les rend particulièrement adaptés à notre application d'assistance à la correction d'épreuves théoriques.

- **Multimodalité (Multi-modality)** : Pour des applications nécessitant le traitement de multiples types de données en entrée et en sortie, des modèles comme **GPT-4 Turbo with Vision** et **GPT-4o** [28] sont particulièrement adaptés. Ces modèles combinent le traitement du langage naturel avec la compréhension visuelle, permettant des interactions riches et complexes via des interfaces multimodales. D'autres modèles multimodaux incluent **CLIP** (*Contrastive Language-Image Pre-Training*) [39] de OpenAI, qui associe des textes à des images pour diverses applications en vision par ordinateur et NLP.
- **Génération de son (Audio)** : Les modèles de génération audio ont considérablement évolué ces dernières années, offrant des capacités impressionnantes pour des applications variées telles que la synthèse vocale, la création musicale et la transformation audio. Parmi les modèles notables, nous trouvons **Tacotron 2** de Google, qui produit des synthèses vocales naturelles à partir de texte, et **WaveNet**, également de Google, qui génère des formes d'onde audio de haute qualité. Un outil plus récent de ElevenLabs, **ElevenLabs Prime Voice AI**, repousse les limites de la synthèse vocale en offrant des voix d'une fluidité et d'une expressivité remarquables, ce qui ouvre de nouvelles perspectives dans les domaines de l'assistance vocale, de la narration et des interfaces utilisateur [40] [41].

- **Intégration de Texte (*Embeddings*)** : Les modèles d'intégration de texte transforment le texte en une représentation numérique appelée "*embedding*". Cette représentation capture des informations sémantiques essentielles du texte d'entrée, permettant aux machines de mieux comprendre les relations entre les mots ou les phrases, (comme illustré dans la **Figure 1.12**). Les *embeddings* sont cruciaux pour l'efficacité des modèles de classification ou de clustering, qui fonctionnent mieux avec des données numériques. Ils jouent souvent un rôle important dans l'apprentissage par transfert (*transfer learning*), un processus qui consiste à entraîner un modèle sur une tâche



substitutive disposant d'une grande quantité de données, puis à réutiliser les poids du modèle pour d'autres tâches en aval. Quelques exemples clés de modèles d'intégration de texte incluent *OpenAI Embeddings*, *Jurassic-1 Jumbo*, *BLOOM* [35] de Hugging Face et BigScience, ainsi que *LaMDA* (*Language Model for Dialogue Applications*) [42]. Ces modèles ont été entraînés sur de vastes corpus de données textuelles et sont utilisés dans divers contextes pour améliorer la compréhension et la génération de texte [31].

Figure 1.12 Intégration de Texte (*Embeddings*) [31]

1.3.4.1 Foundation Models (Base model) Vs LLMs

Le terme "*Foundation model*" (modèle fondamental ou modèle de base) a été introduit par des chercheurs de Stanford pour souligner leur caractère central mais incomplet [21].

Cette terminologie désigne un modèle d'intelligence artificielle répondant à plusieurs critères clés, comme illustré dans la

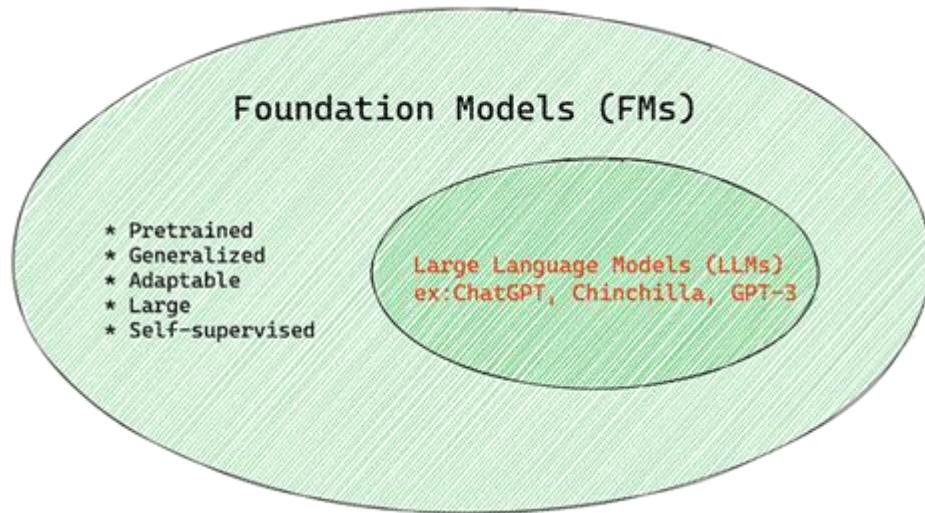


Figure 1.13.

Figure 1.13 Foundation Models (FMs) [21]

- **Apprentissage non supervisé ou auto-supervisé** : Ces modèles sont entraînés sur des données multimodales non étiquetées, ce qui signifie que le processus d'apprentissage ne nécessite pas d'annotation ou d'étiquetage humain des données.
- **Grande taille et complexité** : Il s'agit de modèles de grande envergure basés sur des réseaux de neurones profonds, souvent entraînés sur des milliards de paramètres. Leur vaste capacité permet une compréhension et une génération de texte très précises.
- **Fonction de base pour d'autres modèles** : Ces modèles sont conçus pour servir de base à d'autres modèles. Ils constituent un point de départ pour la construction de modèles plus spécifiques par le biais d'un ajustement fin (fine-tuning).

Par exemple, la première version de ChatGPT a été construite en utilisant un modèle fondamental appelé GPT-3.5. OpenAI a utilisé des données spécifiques aux conversations pour créer une version ajustée de GPT-3.5 spécialisée dans les interactions conversationnelles,

comme les chatbots. Ainsi, GPT-3.5 sert de foundation model, tandis que ChatGPT représente un LLM dérivé de ce modèle fondamental, comme expliqué dans la section sur l'entraînement [31].

1.3.4.2 Model Open-Source vs Model propriétaire [21]

- **Modèles Open Source** : Les modèles open source offrent une accessibilité universelle et une liberté d'utilisation, souvent mis à disposition par les créateurs ou la communauté scientifique. Leur nature transparente et flexible permet une inspection et une adaptation approfondies, facilitant la personnalisation pour diverses applications des LLM. Cependant, ils peuvent présenter des limites, notamment en termes d'optimisation pour la production et de financement, ce qui peut impacter la maintenance et l'actualisation. Des exemples notables incluent **Alpaca** [43], **Bloom**, **LLaMA**, **Phi-3** [44], etc.
- **Modèles Propriétaires** : Les modèles propriétaires comme le mot l'indique, sont de modélés détenus exclusivement par des entreprises, souvent inaccessibles au grand public. Développés par des géants technologiques disposant de ressources conséquentes, ces modèles sont optimisés pour des performances élevées en production, spécialement pour des tâches ciblées. Cependant, leur nature fermée implique que les utilisateurs ne peuvent ni les inspecter, ni les modifier, ce qui peut entraîner des coûts supplémentaires liés à l'abonnement ou à l'utilisation. De plus, le contrôle des données d'entraînement échappe aux utilisateurs, qui doivent se fier aux entreprises pour la confidentialité et l'utilisation éthique de l'IA. Parmi les exemples notables, on trouve les modèles d'OpenAI tels que **GPT-3.5 et GPT-4**, **Google Bard** et **Claude 2**.

1.3.4.3 Scaling laws [17]

Les lois d'échelle, ou "*scaling laws*", sont essentielles pour comprendre et optimiser les performances des LLMs. Elles révèlent que la performance de ces modèles est principalement déterminée par trois facteurs : **la taille du modèle** (le nombre de paramètres sans compter les

embeddings), **la taille du jeu de données** (la quantité de données d'entraînement) et **le volume de calcul alloué à l'entraînement** (soit la quantité totale d'opérations effectuées durant l'apprentissage). Ce dernier point est important car il impacte directement la convergence du modèle : plus le volume de calcul est élevé, plus le modèle apprend efficacement à chaque itération.

En d'autres termes, on peut améliorer un modèle en augmentant le nombre de paramètres, en entraînant sur plus de données, ou en effectuant plus d'itérations d'entraînement ce qui permet certes d'améliorer la performance du modèle, mais cela peut aussi entraîner des risques de surapprentissage (overfitting) et accentuer les biais. Les relations entre ces facteurs et la performance suivent généralement une loi de puissance. Les relations entre la perte, le nombre de paramètres **N**, la taille du dataset **D**, et le budget de calcul **C** ont été quantifiées, montrant comment la perte varie en fonction de ces facteurs. Il a été constaté que la perte **L** diminue en fonction de ces facteurs : $L(N) \propto N^{-\alpha N}$, $L(D) \propto D^{-\alpha D}$, $L(C) \propto C^{-\alpha C}$ [45]. Ces relations permettent de prédire la perte en modifiant l'un de ces facteurs tout en maintenant les autres constants. Par exemple, pour le modèle GPT-3 avec 96 couches et une dimensionnalité de 12288, le nombre de paramètres atteint environ 175 milliards. Comprendre ces lois d'échelle est crucial pour planifier l'entraînement de modèles visant une performance particulière, car elles permettent de prévoir les besoins en données et en calcul pour atteindre des performances optimales. Cependant, la loi d'échelle selon Kaplan n'est pas le seul principe directeur ; récemment d'autres cadres, tels que le *Chinchilla Scaling Laws* [45], qui s'avère beaucoup plus intéressants dans la compréhension et l'optimisation de la performance des LLMs.

1.3.5 Applications Pratiques des LLM

Les applications pratiques des LLM sont vastes et variées, couvrant des domaines tels que l'éducation, le service client, la santé, la recherche scientifique, et le marketing. Voici quelques exemples concrets [46]:

- **Éducation** : Les LLM peuvent faciliter la correction automatique des devoirs et la génération de matériel pédagogique, offrant ainsi un soutien précieux aux enseignants

et une personnalisation accrue de l'apprentissage pour les étudiants. Grâce à des systèmes comme ChatGPT et Copilot, qui représentent une avancée significative dans le domaine de l'éducation, ces outils peuvent fournir des explications détaillées sur des sujets complexes et générer des quiz ou des exercices d'entraînement. L'exemple le plus fascinant, **GPT-4o**, où le “o” signifie “*omni*”, introduit le 13 mai cette année par OpenAI est l'un de model multimodaux (*Multi-modality*) comme introduit ci-haut, intègre des capacités pour gérer le texte, la parole et la vidéo, marquant un bond significatif dans l'applicabilité de l'IA dans divers domaines, notamment l'éducation. Dans des environnements d'apprentissage de plus en plus numériques et diversifiés, les capacités multimodales de GPT-4o sont sur le point de redéfinir la technologie éducative. Dans le domaine de l'éducation, GPT-4o pourrait servir de tuteur personnalisé, s'adaptant au style d'apprentissage de chaque étudiant et fournissant des instructions sur mesure. Cela rendrait l'éducation plus accessible et efficace, en particulier pour les étudiants ayant besoin de soutien supplémentaire ou ayant des besoins d'apprentissage uniques [46]. Toutefois, il est crucial de noter que l'utilisation excessive de ces LLM peut avoir des effets négatifs sur les élèves ou les étudiants qui n'ont pas encore acquis suffisamment de connaissances de base, car cela pourrait court-circuiter leur processus d'apprentissage authentique. La **Figure 1.14** illustre une taxonomie des LLMs pour les applications éducatives avec des travaux représentatifs, mettant en lumière leur potentiel et les diverses façons dont ils peuvent être utilisés dans le domaine de l'éducation [46].

- **Service Client** : L'automatisation des réponses aux requêtes clients est une application courante des LLM. Ils améliorent l'efficacité du service client en fournissant des réponses rapides et précises, tout en réduisant les coûts opérationnels [47].
- **Santé** : Les LLM révolutionnent le domaine médical avec leurs applications variées. Ils sont essentiels pour l'analyse et la segmentation d'images médicales, l'examen des variants génétiques, la prédiction de la structure des protéines et l'interprétation des données cliniques. Ces outils avancés facilitent le diagnostic des maladies, la création

de plans de traitement sur mesure et l'analyse de volumes massifs de données biomédicales pour déceler des tendances ou anomalies. Parmi les exemples marquants, *DeepChem* [48], lancé par Google AI en 2020, détecte les signes de cancer du sein avec une précision rivalisant avec celle des radiologues, tandis qu'*AlphaFold 2* [49], dévoilé en 2021, prédit la structure des protéines avec une exactitude sans précédent, promettant des progrès significatifs dans la recherche pharmaceutique chercheurs [50] [51].

- **Recherche Scientifique** : Les LLM assistent dans la rédaction d'articles scientifiques, l'extraction d'informations pertinentes à partir de grandes bases de données, et la génération d'hypothèses pour guider les recherches futures. Leur capacité à synthétiser et à analyser de grandes quantités d'informations est particulièrement utile pour les chercheurs [52] [46].
- **Marketing et Média** : Dans le domaine du marketing, les LLM sont utilisés pour créer du contenu attrayant pour les campagnes publicitaires, les articles de blog, et autres supports médiatiques. Ils peuvent générer des textes personnalisés pour différents segments de marché et optimiser les messages pour un impact maximal [48].

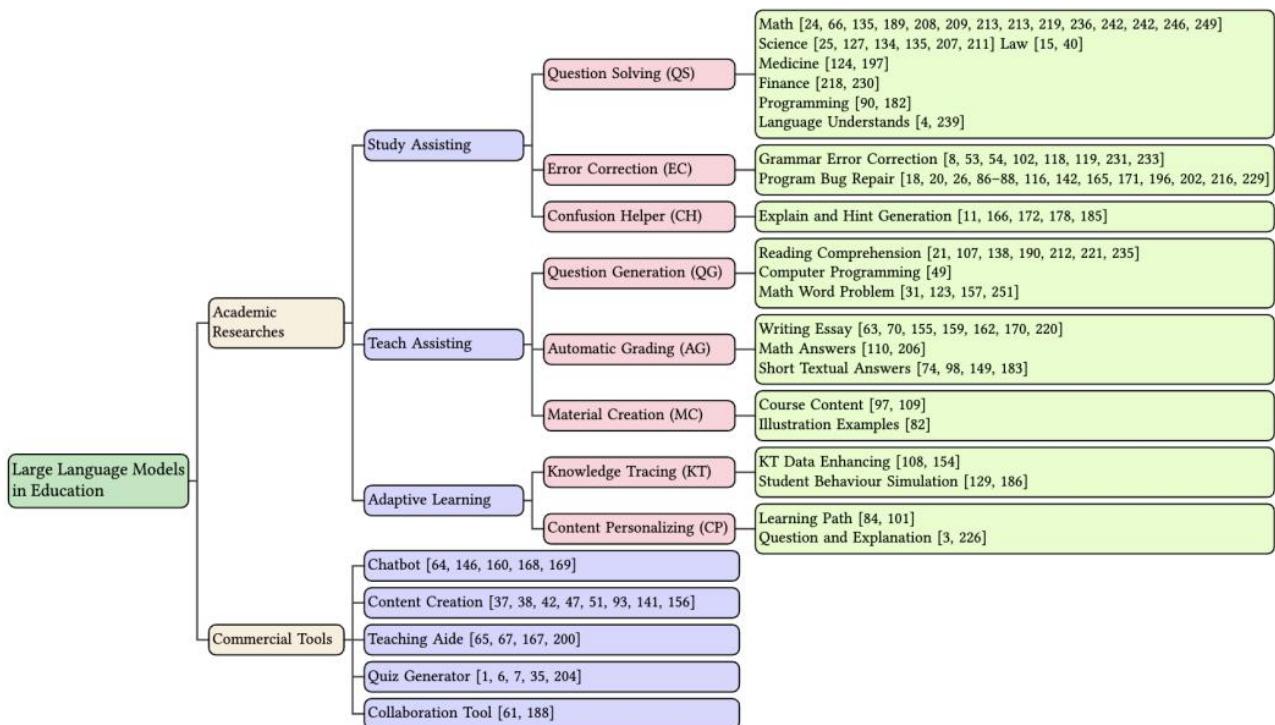


Figure 1.14 Une taxonomie des LLMs pour les applications éducatives avec des travaux représentatifs [46].

Cette figure illustre une taxonomie des LLMs appliqués à l'éducation, en les divisant en quatre grandes catégories : *Assistance aux études*, *Assistance à l'enseignement*, *Apprentissage adaptatif*, et *Outils commerciaux*. Dans la catégorie *Assistance aux études*, on retrouve des sous-catégories telles que la résolution de questions, la correction d'erreurs, et l'aide à la génération d'explications, avec des références à des études pertinentes. De même, l'assistance à l'enseignement inclut la génération de questions, la correction automatique et la création de matériel pédagogique. L'apprentissage adaptatif est principalement représenté par la traçabilité des connaissances (Knowledge Tracing) et la personnalisation du contenu, permettant une approche plus individualisée pour les apprenants. Enfin, les outils commerciaux tels que les chatbots et les générateurs de quiz mettent en lumière l'application concrète des LLMs dans des solutions pratiques disponibles sur le marché. Cette taxonomie montre comment les LLMs peuvent être utilisés à la fois pour soutenir les étudiants et les enseignants, tout en favorisant un apprentissage plus personnalisé grâce à l'adaptation des contenus et des chemins d'apprentissage.

1.4 Conclusion partielle

Ce chapitre a présenté un panorama des défis et questions émergentes liés à l'évaluation des épreuves théoriques, avec une exploration approfondie des Modèles de Langage Larges (LLM), notamment les *Transformers*. Nous avons examiné leur fonctionnement, processus d'entraînement, et architecture, ainsi que les distinctions entre les *Foundation Models* et les LLM, et entre les modèles open-source et propriétaires. Nous avons également illustré les applications pratiques des LLM dans des domaines variés comme l'éducation, le service client, la santé, la recherche scientifique et le marketing.

Chapitre 2 Techniques de correction automatique d'épreuves, modèles GPT-3.5 et LLAMA-3, et conception du Système

2.1 Introduction

Dans ce chapitre, nous allons explorer les techniques de correction d'épreuves théoriques, un aspect essentiel de l'évaluation académique. Nous commencerons par examiner les méthodes traditionnelles et modernes d'automatisation de ce processus, en présentant leurs avantages et leurs limites. Ensuite, nous détaillerons les modèles que nous avons choisis pour notre projet, à savoir GPT-3.5 Turbo et LLAMA-3, deux des dernières avancées en intelligence artificielle générative. Nous analyserons leur architecture, leur fonctionnement et leurs performances, tout en mettant en avant les innovations qu'ils apportent. Enfin, nous proposerons une conception simplifiée de l'architecture globale de notre système, nommé ***Bic Rouge***, en précisant le rôle de chaque composant.

2.2 Techniques de correction automatique d'épreuves

2.2.1 Approches traditionnelles

Les techniques de correction automatique d'épreuves ont évolué au fil des années. Les premières méthodes incluaient l'utilisation de règles définies par des experts, les systèmes à base de règles (*Rule-Based Systems*), et les algorithmes de correspondance de modèles. Ces approches, bien que robustes dans certains contextes, souffraient souvent d'un manque de flexibilité et d'adaptabilité face à la diversité des réponses possibles [54] [55] [56].

2.2.1.1 Systèmes à base de règles (*Rule-Based Systems*)

Les premières méthodes de correction automatique utilisaient des systèmes à base de règles, où des experts définissaient explicitement les règles grammaticales et syntaxiques à suivre. Ces systèmes fonctionnaient sur le principe de l'analyse syntaxique, comparant les réponses des étudiants à un ensemble préétabli de règles pour détecter les erreurs. Par exemple, un système à base de règles pourrait vérifier la concordance des temps verbaux ou la position correcte des adjectifs dans une phrase [54] [55].

❖ **Avantages [54]:**

- Précision élevée dans des contextes spécifiques où les règles sont bien définies.
- Facilité d'interprétation des erreurs et des corrections, car les règles sont explicites et compréhensibles.

❖ **Limitations [54]:**

- Manque de flexibilité face à la diversité des réponses possibles et aux variations linguistiques.
- Dépendance aux experts linguistiques pour la création et la maintenance des règles.
- Difficulté d'adaptation aux nouvelles langues ou domaines spécifiques sans réécrire des règles complexes.

2.2.1.2 Algorithmes de mesure de similarité [57] [18] [58]

Une approche basée sur la mesure de similarité est couramment utilisée pour évaluer automatiquement les réponses des étudiants, sans entraîner de modèles spécifiques. Ces méthodes reposent sur la comparaison des segments de texte des réponses des étudiants avec des réponses de référence, à l'aide de mesures telles que les n-grammes.

❖ **Exemples d'algorithmes utilisés [20] :**

- Algorithmes de **distance de Levenshtein** pour mesurer les différences entre deux chaînes de caractères.
- Algorithmes de **similarité cosinus** pour comparer la similarité sémantique entre deux segments de texte.
- **La mesure BLEU** : Initialement conçue pour évaluer les traductions automatiques, peut être adaptée pour l'évaluation des réponses libres en comparant les n-grammes des réponses étudiantes à des réponses de référence fournies par les enseignants.
- **La mesure ROUGE** : Également basée sur les n-grammes, est utilisée pour comparer la similarité entre deux textes en termes de co-occurrence de sous-séquences de mots.

❖ **Avantages :**

- Simplicité de mise en œuvre dans des contextes où des modèles de réponses correctes sont disponibles.
- Rapidité de traitement pour des textes courts et bien structurés.

❖ **Limitations :**

- Performance limitée pour les textes longs et complexes.
- Incapacité à comprendre le contexte au-delà des simples correspondances de texte.
- Bruit élevé en cas de diversité linguistique et stylistique dans les réponses des étudiants.

2.2.2 Approches modernes

Avec les avancées en apprentissage automatique et en traitement du langage naturel (NLP), de nouvelles méthodes plus efficaces ont émergés pour la correction automatique d'épreuves. Ces approches modernes incluent l'utilisation de modèles statistiques, d'apprentissage supervisé et d'apprentissage profond [20] [18].

2.2.2.1 Modèles statistiques

Les modèles statistiques ont été parmi les premières approches à dépasser les systèmes à base de règles. Ils utilisaient des méthodes probabilistiques pour identifier et corriger les erreurs dans les textes [58].

❖ **Exemples de techniques** [18]:

- **Modèles de Markov cachés (HMM)** pour la reconnaissance des séquences de mots.
- **Modèles de langage n-grammes** pour estimer la probabilité de séquences de mots.

❖ **Avantages :**

- Capacité à capturer les régularités linguistiques sans règles explicites.
- Adaptabilité à de nouvelles données par réentraînement.

❖ **Limitations :**

- Complexité croissante avec l'augmentation de la longueur des séquences de mots.
- Difficulté à capturer les dépendances à long terme dans les textes.

2.2.2.2 Apprentissage profond [20] [18]

L'apprentissage profond, basé sur les réseaux de neurones profonds (DNN), a révolutionné la correction automatique d'épreuves en introduisant des modèles capables de comprendre le contexte de manière plus sophistiquée.

❖ **Exemples de modèles :**

- Réseaux de neurones récurrents (RNN) pour la modélisation des séquences de texte.
- Transformers pour la capture des dépendances à long terme dans les textes.

❖ **Avantages :**

- Capacité à capturer des relations complexes et des dépendances contextuelles dans les textes.
- Excellentes performances sur des tâches de correction contextuelles et grammaticales.

❖ **Limitations :**

- Exigences computationnelles élevées pour l'entraînement et l'inférence.
- Nécessité de grandes quantités de données pour un entraînement efficace.

2.2.3 Systèmes fonctionnels utilisant ces modèles

Plusieurs systèmes fonctionnels ont été développés en utilisant ces modèles avancés, démontrant leur efficacité dans la correction automatique d'épreuves

Exemples de systèmes :

- **ExamWizard** : Utilise des techniques de NLP et de machine learning pour corriger automatiquement les épreuves théoriques et fournir des feedbacks détaillés [59].

- **Gradescope (Turnitin)** : Emploie des algorithmes avancés de correction automatique pour évaluer les réponses des étudiants et détecter le plagiat [60].
- **Grammarly** : Utilise des techniques de machine learning et des modèles de langage pour fournir des corrections grammaticales et stylistiques [61].

Après avoir exploré ces méthodes et mis en lumière leurs avantages et leurs limites, nous allons maintenant nous concentrer sur les modèles spécifiques que nous avons sélectionnés pour notre travail.

2.3 Présentation des modèles GPT-3.5 et LLAMA-3

Comme annoncé ci-dessus dans le chapitre précédent, pour la réalisation de notre projet, nous nous focalisons sur deux des modèles de langage large : **GPT-3.5** et **LLAMA-3**. Ces modèles, développés respectivement par *OpenAI* et *Meta*, se distinguent par leurs architectures sophistiquées et leurs performances remarquables sur une variété de tâches de traitement du langage naturel. GPT-3.5, et en particulier sa version Turbo, est reconnu pour son efficacité et sa capacité à générer du langage naturel de haute qualité. LLAMA-3, avec ses variantes optimisées pour le dialogue, offre des résultats exceptionnels en termes de compréhension et de génération de texte [16] [21]. L'évaluation de ces modèles repose sur des métriques de performance rigoureuses telles que MMLU (*Multitask Multidomain Language Understanding*), ARC (*AI2 Reasoning Challenge*), *HellaSwag* et *TruthfulQA*, afin de garantir leur fiabilité et leur utilité [62]. Cette partie présente en détail les architectures, les performances, ainsi que les aspects de latence et de coût de ces modèles, fournissant une vue d'ensemble claire et concise qui guidera leur application dans notre système.

2.3.1 Architecture et fonctionnement de GPT-3.5 Turbo [15] [63]

La série des modèles GPT (Generative Pre-trained Transformers), développée par OpenAI, représente une avancée significative dans le domaine du traitement du langage naturel (NLP). La famille de modèles GPT, allant de GPT-1 à GPT-4 actuellement, a vu plusieurs itérations

et améliorations au fil du temps, chaque génération apportant des raffinements en termes de performance, de capacité de traitement et de génération du langage naturel.

Les modèles de la série GPT-3.5 incluent **code-davinci-002**, **text-davinci-002**, **text-davinci-003**, et **gpt-3.5-turbo**. GPT-3, avec son modèle de base **davinci** qui comporte 175 milliards de paramètres, est reconnu pour être un générateur de texte très performant. OpenAI a suivi deux voies de mise à niveau pour davinci : le fine-tuning supervisé pour créer **InstructGPT** et **text-davinci-001**, et la formation spécialisée pour le code pour créer **Codex** et **code-cushman-001**.

En 2022, OpenAI a publié **code-davinci-002** pour les tâches de génération de code, qui est devenu le modèle de base pour la série GPT-3.5. OpenAI a ensuite utilisé le fine-tuning supervisé pour créer **text-davinci-002** et a introduit la stratégie de *Reinforcement Learning from Human Feedback* (RLHF) pour créer **text-davinci-003**, améliorant ainsi sa capacité à comprendre les instructions et à générer du texte. Basé sur **text-davinci-003**, GPT-3.5 Turbo a été optimisé pour les interactions de type chat, se révélant être le modèle GPT-3.5 le plus performant tout en étant plus économique que **text-davinci-003**.

2.3.1.1 Architecture

GPT-3.5 Turbo utilise l'architecture **Transformer**, introduite plus haut dans la section **1.3.3**. Cette architecture est constituée de blocs d'encodage et de décodage interconnectés par des mécanismes d'attention multi-têtes et des couches *feed-forward*. Ces éléments permettent au modèle de capturer des dépendances complexes entre les mots dans un texte, facilitant ainsi la génération de réponses cohérentes et pertinentes.

À l'instar de tous les modèles basés sur l'architecture des transformateurs, GPT-3.5 Turbo intègre des mécanismes clés tels que l'Attention Multi-Têtes et les couches Feed-Forward. Ce modèle a été spécialement optimisé pour les interactions de type conversationnel, avec un accent particulier sur la fluidité et le naturel des dialogues, ce qui lui permet de surpasser ses prédecesseurs dans les tâches liées aux échanges verbaux.

2.3.1.2 Capacités et Performances [63]

GPT-3.5 Turbo a montré des performances remarquables dans divers benchmarks de traitement du langage naturel (NLP). Selon Zhang et al, GPT-3 peut acquérir des connaissances linguistiques complexes et reconnaître des informations sémantiques dans des contextes continus. Yang et al, ont exploré le potentiel de ChatGPT, qui inclut GPT-3.5 Turbo, dans des tâches telles que le résumé de texte basé sur les aspects et la traduction automatique. En outre, Qin et al, ont analysé les capacités de ChatGPT en zéro-shot à travers sept catégories de tâches représentatives.

Cependant, il est aussi important de reconnaître les limitations des modèles GPT-3.5. Kocón et al, ont comparé ChatGPT avec d'autres modèles de pointe sur 25 tâches différentes en NLP, révélant certains biais et limites du modèle. Chen et al, ont également mené des tests de robustesse sur GPT-3.5, explorant ses vulnérabilités potentielles et les défis associés à son utilisation dans des applications réelles.

2.3.1.3 Versions Disponibles et Spécifications

Le tableau **Tableau 1** présente un récapitulatif des différentes versions et spécifications des modèles GPT-3.5 disponibles via **Azure OpenAI**, incluant la disponibilité géographique, la longueur maximale de séquence en tokens, ainsi que les données de formation correspondantes. **Azure OpenAI** est un service cloud fourni par Microsoft Azure qui permet d'accéder aux modèles avancés de traitement du langage naturel développés par OpenAI, tels que GPT-3.5, LLAMA et d'autres.

Tableau 1 Versions Disponibles et Spécifications des Modèles GPT-3.5 d'Azure OpenAI [64]

ID du Modèle	Disponibilité Géographique	Longueur de Séquence Max (tokens)	Données de Formation (jusqu'à)
gpt-35-turbo (0301)	East US, France Central, South	4,096	Sep 2021

	Central US, UK South, West Europe		
gpt-35-turbo (0613)	Australia East, Canada East, East US, East US 2, France Central, Japan East, North Central US, Sweden Central, Switzerland North, UK South	4,096	Sep 2021
gpt-35-turbo-16k (0613)	Australia East, Canada East, East US, East US 2, France Central, Japan East, North Central US, Sweden Central, Switzerland North, UK South	16,384	Sep 2021
gpt-35-turbo-instruct (0914)	East US, Sweden Central	4,097	Sep 2021
gpt-35-turbo (1106)	Australia East, Canada East, France Central, South India, Sweden Central, UK South, West US	Input: 16,385 Output: 4,096	Sep 2021

Parmi les différentes versions disponibles, nous avons choisi d'utiliser gpt-3.5-turbo pour notre système, en raison de sa capacité optimisée à traiter des instructions et à générer des réponses contextuellement appropriées et sa disponibilité.

2.3.2 Architecture et fonctionnement de LLAMA-3 [21] [65] [66]

La famille des modèles LLaMA de Meta est devenue l'une des séries de LLMs open-source les plus puissantes. Ces modèles ont été entraînés sur des billions de tokens provenant exclusivement de jeux de données disponibles publiquement, évitant ainsi l'utilisation de données propriétaires et inaccessibles. L'objectif principal de cette initiative est de démocratiser l'accès et l'étude des LLMs.

En avril de cette année (2024), Meta a publié la famille de modèles LLaMA 3 après le succès retentissant de la famille de modèles LLaMA 2 l'année précédente. Cette nouvelle série comprend des modèles génératifs pré-entraînés et ajustés pour les instructions, disponibles en tailles de 8 milliards (8B) et 70 milliards (70B) de paramètres. Les modèles LLaMA 3 sont disponibles en versions pré-entraînées et optimisées pour les instructions, permettant une adaptation flexible selon les besoins spécifiques des utilisateurs. Ils surpassent de nombreux modèles de chat open source sur des benchmarks industriels courants. Conçus pour traiter exclusivement du texte en entrée et en sortie, ces modèles se concentrent sur la génération et la compréhension textuelles de haute qualité.

2.3.2.1 Architecture

L'architecture de LLaMA 3 est celle d'un modèle de langage auto-régressif utilisant une architecture de *Transformers* optimisée. Les versions ajustées emploient le *supervised fine-tuning (SFT)* et le RLHF pour s'aligner sur les préférences humaines en termes d'utilité et de sécurité. Les deux versions utilisent l'attention par groupes de requêtes (*Grouped-Query Attention*, GQA) pour améliorer l'évolutivité de l'inférence.

Le **Tableau 2***Tableau 6*, présente les spécifications et les données d'entraînement pour les modèles LLaMA 3 de 8M et 70M de paramètres. Il décrit les principales caractéristiques de chaque modèle, incluant le type de données utilisées pour l'entraînement (un mélange de

données disponibles publiquement), le nombre de paramètres en milliards, la longueur du contexte en tokens, ainsi que des informations supplémentaires telles que la présence de GQA (Oui), le nombre total de tokens (15T+), et la limite de connaissance fixée à mars 2023. Ces informations offrent une vue d'ensemble des capacités des modèles LLaMA 3 en termes de taille, d'architecture et de connaissances, importantes pour évaluer leur performance sur des tâches complexes.

Tableau 2 Spécifications et Données de Formation pour les Modèles LLaMA 3 [67]

Modèle	Données d'entraînement	Paramètres en milliards	Longueur du contexte en tokens	GQA	Nombre de tokens	Limite de connaissance
LLaMA 3 8M	Mélange de données en ligne disponibles publiquement.	8M	8000	Oui	15T+	Mars 2023
LLaMA 3 70M	Mélange de données en ligne disponibles publiquement	70M	8000	Oui	15T+	Mars 2023

2.3.2.2 Capacités et Performances

Grâce au pré-entraînement intensif, les modèles LLAMA3 ont atteint des performances de pointe dans une large gamme de tâches, établissant la famille LLaMA comme l'une des meilleures LLM open-source disponibles pour diverses applications et scénarios de déploiement.

- **Modèles pré-entraînés de base**

Le tableau **Tableau 3** présente une comparaison des performances des modèles base pour Llama3 8B, Llama 2 13B, Llama 3 70B, et Llama 2 70B sur divers benchmarks, illustrant leurs capacités respectives en matière de compréhension générale, de raisonnement basé sur la connaissance, et de compréhension de lecture.

Tableau 3 Performances des Modèles Pré-entraînés de Base LLaMA 3 et LLaMA 2 [67]

Général	MMLU (5-shot)	66.6	45.7	53.8	79.5	69.7
	AGIEval English (3-5 shot)	45.9	28.8	38.7	63.0	54.8
	CommonSenseQA (7-shot)	72.6	57.6	67.6	83.8	78.7
	Winogrande (5-shot)	76.1	73.3	75.4	83.1	81.8
	BIG-Bench Hard (3-shot, CoT)	61.1	38.1	47.0	81.3	65.7
	ARC-Challenge (25-shot)	78.6	53.7	67.6	93.0	85.3
Knowledge reasoning	TriviaQA-Wiki (5-shot)	76.5	72.2	79.6	89.7	87.5
	SQuAD (1-shot)	76.4	72.2	72.1	85.6	82.6
Reading comprehension	QuAC (1-shot, F1)	44.4	39.6	44.9	51.1	49.4
	BoolQ (0-shot)	75.7	65.5	66.9	79.0	73.1

	DROP (3-shot, F1)	58.4	37.9	49.8	79.7	70.2
--	-------------------	------	------	------	------	------

2.3.2.3 Données d'Entraînement, Matériel et Logiciels

LLAMA 3 a été pré-entraîné sur plus de 15 trillions de *tokens* issus de sources publiques. Les données de fine-tuning comprennent des ensembles de données d'instructions publiques ainsi que plus de 10 millions d'exemples annotés par des humains. Il est important de noter qu'aucune des données de pré-entraînement ou de fine-tuning n'inclut des informations provenant des utilisateurs de Meta. La collecte des données de pré-entraînement s'est arrêtée en mars 2023 pour les deux versions du modèle.

Pour conclure cette partie, nous présentons ci-dessous dans le **Tableau 4** un récapitulatif des principales caractéristiques et performances de nos deux modèles.

2.3.3 Comparaison des caractéristiques et performances des modèles

LLAMA-3 (8B) et GPT-3.5 Turbo [68] [69]

Tableau 4 Comparaison des caractéristiques et performances des modèles LLAMA-3 (8B) et GPT-3.5 Turbo

Caractéristiques	LLAMA-3 (8B)	GPT-3.5 Turbo
Développeur	Meta	OpenAI
Fenêtre de contexte	8 000 tokens	4097 tokens
Date de sortie	18 avril 2024	28 novembre 2022
Limite de connaissance	Décembre 2023	Mars 2023
Open Source	Oui	Non
Performance		
MMLU (5-shot)	68.4	70.0

HellaSwag (10-shot)	Non disponible	85
MATH (0-shot)	51.9	43.1
<hr/>		
Coût d'entrée	0,17 \$ par million de tokens	1,5 \$ par million de tokens
Coût de sortie	0,20 \$ par million de tokens	3 \$ par million de tokens
Latence (TTFT)	0.32s	0.37s
Débit (tokens/s)	~1250	84

Principales différences

- **Fenêtre de contexte** : LLAMA-3 a une fenêtre de contexte plus large (8 000 tokens) comparée à GPT-3.5 Turbo (4097 tokens).
- **Performance** : GPT-3.5 Turbo surpassé LLAMA-3 dans le benchmark MMLU (68.4 contre 70.0).
- **Coûts** : LLAMA-3 est beaucoup moins cher car il est open source, tandis que GPT-3.5 a un coût relativement élevé.

Ces informations montrent que bien que LLAMA-3 ait des performances compétitives, GPT-3.5 Turbo offre des avantages dans certains benchmarks. Pour la réalisation de notre travail, nous avons accédé au modèle GPT-3.5 Turbo via l'API mise à disposition par *OpenAI*¹ ainsi

¹ <https://openai.com/>

que via le service *Azure OpenAI*² de *Microsoft Azure*. Pour LLaMA, nous avons utilisé la plateforme *Together.ai*³ et *GitHub Models*⁴, où nous sommes testeurs bêta du service.

2.4 Conception de l'architecture globale de Bic Rouge

Il existe un large éventail de méthodes pour le développement des systèmes informatiques, mais en règle générale, toutes suivent les étapes suivantes [29]:

- 1) **Spécifications** : Définir avec précision les fonctionnalités et les objectifs du système.
- 2) **Conception et mise en œuvre** : Concevoir l'architecture du système et le réaliser conformément aux spécifications.
- 3) **Validation** : Tester le système pour vérifier s'il répond aux objectifs définis dans les spécifications.
- 4) **Évolution** : Inclure toutes les activités post-livraison, telles que la versionnage et la maintenance.

Nous n'adopterons pas une méthode de conception particulière ici. Toutefois, pour maintenir une approche méthodique, nous nous inspirerons des quatre étapes classiquement suivies lors de la conception des systèmes informatiques. Dans ce deuxième chapitre, nous présentons uniquement les spécifications du système et une ébauche de conception avec une vue d'ensemble de l'architecture globale. Les détails complémentaires seront abordés dans le chapitre suivant.

² <https://azure.microsoft.com/en-us/products/ai-services/openai-service>

³ <https://www.together.ai/>

⁴ <https://docs.github.com/en/github-models>

2.4.1 Spécifications du système

2.4.1.1 Objectifs

Le système vise à aider les enseignants à corriger les épreuves à caractère théoriques de manière plus efficace et objective, en utilisant GPT-3 et LLAMA-3.

Les objectifs principaux incluent :

- Réduction du temps nécessaire pour corriger les épreuves.
- Amélioration de la cohérence et de l'objectivité des corrections.
- Fourniture de feedback constructif aux étudiants.

2.4.1.2 Fonctionnalités pour les Enseignants

- **Création de Compte** : Permet aux enseignants de s'inscrire et de créer des comptes en fournissant leurs informations et leur affiliation.
- **Création de Cours** : Les enseignants peuvent créer des cours et générer des codes uniques pour chaque cours.
- **Création d'Évaluations** : Capacité de créer des évaluations avec des questions, grilles de correction, et critères d'évaluation.
- **Examen des Évaluations** : Possibilité de revoir, accepter, rejeter ou modifier les notes et commentaires générés par le LLM.
- **Statistiques de Performance** : Consultation des statistiques de performance des étudiants.

2.4.1.3 Fonctionnalités pour les Étudiants

- **Création de Compte** : Inscription des étudiants avec sélection de leur établissement.

- **Inscription aux Cours :** Rejoindre des cours et des épreuves via des codes fournis par les enseignants.
- **Soumission d'Évaluations :** Soumission de leurs travaux peut se faire soit en répondant directement aux questions sur la plateforme ou en uploadant leurs travaux dans différents formats (PDF, Word).
- **Examen des Notes et Commentaires :** Accès aux notes et commentaires générés par le LLM.

2.4.1.4 Exigences Fonctionnelles

1) Soumission d'Évaluations :

- Les étudiants doivent pouvoir soumettre leurs travaux soit en répondant directement aux questions sur la plateforme, soit en téléchargeant leurs travaux dans différents formats (PDF, Word).

2) Examen des Notes et Commentaires :

- Les étudiants doivent avoir accès aux notes et commentaires générés par le LLM pour chaque évaluation soumise.

3) Création de Compte :

- Les enseignants et les étudiants doivent pouvoir créer des comptes en fournissant les informations nécessaires.

4) Création et Gestion de Cours :

- Les enseignants doivent pouvoir créer, modifier, et supprimer des cours ainsi que des épreuves, et générer des codes de cours uniques.

5) Statistiques de Performance :

- Les enseignants doivent pouvoir consulter les statistiques de performance des étudiants pour chaque cours.

2.4.1.5 Exigences Non Fonctionnelles

1) Compatibilité des Formats :

- Le système doit supporter les fichiers .pdf, .docx, et .txt pour la soumission des évaluations.

2) Sécurité des Données :

- Mise en place d'un processus d'authentification sécurisé pour protéger l'accès aux comptes des utilisateurs.
- Utilisation de protocoles de sécurité (SSL/TLS) pour la transmission des données.

3) Performance :

- Traitement efficace et rapide d'épreuves soumis pour correction par le LLM.
- Capacité de gérer un grand nombre d'utilisateurs simultanément sans dégradation des performances.

4) Fiabilité et Disponibilité :

- Le système doit être disponible en permanence (24/7) avec un temps d'arrêt minimal.
- Mise en place de mécanismes de sauvegarde et de récupération des données pour éviter toute perte de données.

5) Scalabilité :

- Le système doit pouvoir évoluer pour gérer une augmentation du nombre d'utilisateurs et de soumissions sans compromettre les performances.

6) Interface Utilisateur :

- L'interface utilisateur doit être intuitive et facile à utiliser pour les enseignants et les étudiants.
- Support multi-plateforme pour une accessibilité depuis différents dispositifs (ordinateurs, tablettes, smartphones).

2.4.2 Présentation des Éléments du Système

L'architecture globale de notre système est organisée selon une architecture classique à trois tiers. Elle se présentera comme sur la **Figure 2.1**:

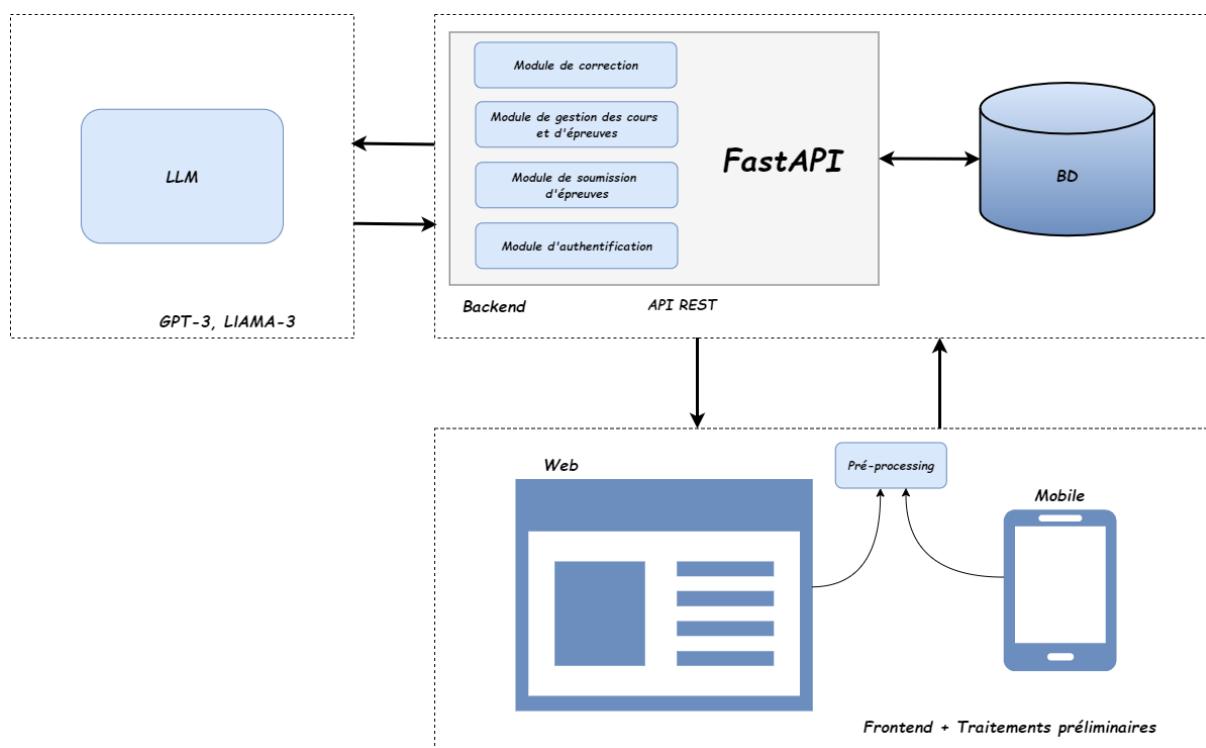


Figure 2.1 Architecture globale de notre système

L'architecture du système est une architecture 3-tiers classique, constituée des trois couches suivantes :

1) Tier de Présentation :

- **Interface Utilisateur** : Accessible via des navigateurs web ou des applications mobiles, cette interface permet aux enseignants et aux étudiants d'interagir avec le système. Les fonctionnalités incluent la création de comptes, la soumission des évaluations, l'examen des notes et des commentaires, ainsi que la gestion des cours.
- **Module de Pré-processing** : Ce module se charge de l'acquisition des données soumises par les étudiants dans divers formats (PDF, DOCX, etc.). Il extrait les données des documents fournis et les convertit en un format plus léger (JSON) avant de les envoyer à l'API. Cela assure une manipulation plus efficace des données et garantit la qualité dès la notation et du feedback de l'LLM.

2) Tier Applicatif (Middle-Tier) :

Cette couche est gérée par une API REST (Representational State Transfer) basée sur FastAPI, qui orchestre les différents modules fonctionnels du système. Elle comprend les modules suivants :

- **Module de correction** : Utilise les modèles LLM (GPT-3, LLAMA-3) pour évaluer les épreuves et générer des commentaires.
- **Module de gestion des cours et des épreuves** : Permet la création et la gestion des cours et des évaluations par les enseignants.
- **Module de soumission d'épreuves** : Gère la soumission des travaux des étudiants, acceptant différents formats de fichiers tels que .pdf, .docx, et .txt.
- **Module d'authentification** : Assure la sécurité des accès au système en gérant l'inscription et l'authentification des utilisateurs.

L'interface de communication avec les LLMs, potentiellement via des technologies comme *Langchain*, s'intègre également dans cette couche pour l'interaction avec les LLMs.

3) Tier de Données :

Cette couche est représentée par la base de données (BD), qui stocke toutes les informations relatives aux utilisateurs, cours, épreuves, soumissions, notes et commentaires. Elle garantit la persistance et l'intégrité des données.

2.4.2.1 Architecture du Module de Correction

L'architecture du module de correction peut être divisée en trois étapes principales : le prétraitement des textes, l'analyse par les LLMs, et l'agrégation et le stockage des résultats. Comme illustrer par la **Figure 2.2** :

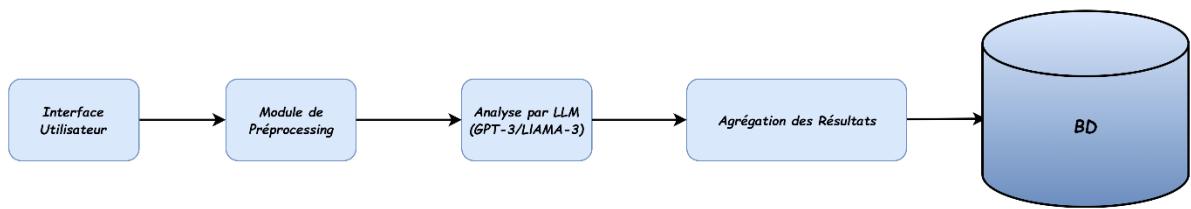


Figure 2.2 Architecture du Module de Correction

Comme l'illustre la **Figure 2.2**, Le module de correction de notre système suit une architecture composée de quatre étapes principales : l'interface utilisateur, le prétraitement, l'analyse par les LLMs (GPT-3 ou LLAMA-3), et l'agrégation des résultats avant le stockage dans une base de données.

- **Interface Utilisateur** : Les enseignants soumettent leurs grilles de correction et les étudiants leurs réponses via cette interface. Cette interface permet la transmission fluide des informations vers le module de traitement.
- **Prétraitement des Textes** : Dans cette étape, les textes soumis sont transformés en prompts structurés pour être traités par les modèles de langage. Le texte est segmenté en unités de traitement (phrases, mots) à travers un processus de tokenisation. Cette étape inclut aussi le nettoyage des données, en supprimant les caractères non pertinents, pour garantir une analyse optimale.

- **Analyse par les LLMs** : Une fois le prétraitement achevé, le prompt est envoyé au modèle de langage choisi (GPT-3 ou LLAMA-3) pour analyse. Le modèle génère automatiquement des notes et des commentaires en se basant sur les grilles de correction des enseignants. Il fournit également un retour détaillé pour chaque question et une appréciation globale.
- **Agrégation et Stockage des Résultats** : Les notes et commentaires produits par les LLMs sont ensuite agrégés pour former une évaluation globale de chaque étudiant. Ces résultats sont stockés dans une base de données (BD), où ils peuvent être consultés ultérieurement par les enseignants et les étudiants.

2.4.2.2 Architecture du processus d'Évaluation



Figure 2.3 Architecture du processus d'Évaluation

La **Figure 2.3** présente les étapes clés du processus d'évaluation de notre système.

- Entrée : La première étape regroupe les éléments nécessaires pour l'évaluation, à savoir la composition des épreuves, les réponses des étudiants, et les critères de correction fournis par les enseignants.
- Analyse LLM : Le modèle de langage (LLM) analyse les informations fournies, en comparant les réponses des étudiants avec les critères de correction définis par les enseignants.
- Sortie : Le LLM produit une épreuve corrigée, comprenant une note et un feedback détaillé pour chaque étudiant.

2.4.2.3 Présentation des interfaces

Le système propose deux interfaces principales : une pour les enseignants et une pour les étudiants. Chacun comprenant plusieurs fonctionnalités essentielles. En voici donc quelques ébauches :



Figure 2.4 Ébauche d'interface des résultats d'évaluation



Figure 2.5 Ébauche d'interface de login

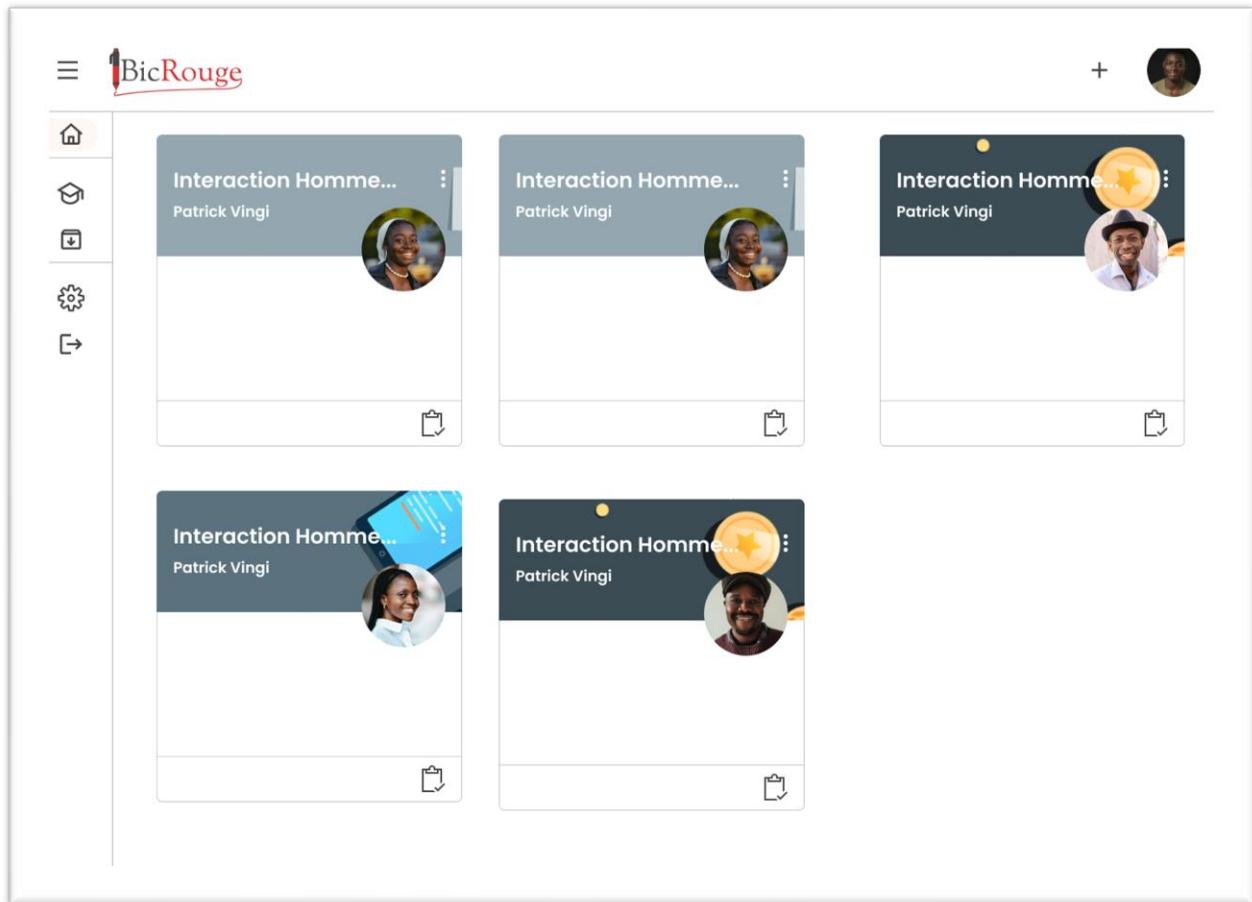


Figure 2.6 Ébauche d'interface des Cours

Ces interfaces donnent un aperçu global sur les autres fonctionnalités du système pour.

2.5 Conclusion partielle

Dans cette partie, nous venons de présenter les techniques modernes et traditionnelles utilisées pour la correction automatique des épreuves théoriques, en mettant en lumière les avantages et les limitations de chaque méthode. Nous avons ensuite détaillé les modèles GPT-3.5 Turbo et LLAMA-3, qui constituent le cœur technologique de notre projet, Bic Rouge. Une attention particulière a été portée à l'examen de leur architecture, de leur fonctionnement et de leurs performances spécifiques, en soulignant les innovations qu'ils apportent dans le domaine de l'intelligence artificielle générative. Enfin, nous avons conçu l'architecture globale de notre

système, en expliquant le rôle et le fonctionnement de chaque module, y compris la gestion des cours et des évaluations, le traitement des soumissions des étudiants, l'analyse et la correction par les modèles LLM, ainsi que l'agrégation et le stockage des résultats. Cette conception fournit une base solide pour le développement et la mise en œuvre du système Bic Rouge, visant à améliorer l'efficacité et l'objectivité de la correction des épreuves théoriques dans le contexte académique.

Chapitre 3 Conception détaillée, réalisation et tests

3.1 Introduction

Dans les chapitres précédents, nous avons esquissé tout l'environnement du système que nous avons prévu élaborer à travers ce travail. A présent, nous allons présenter avec précision les diverses parties de notre système. Dans ce chapitre, nous comptons en premier lieu finaliser la conception entamée au chapitre précédent. En second lieu, nous allons décrire les diverses méthodes utilisées pour évaluer notre système, nous allons décrire l'implémentation de notre système, nous allons tester les modèles dont nous avons fait usage et nous allons justifier les choix d'implémentation que nous avons faits. Les tests en particulier seront faits de manière à pouvoir donner une justification cohérente des choix d'implémentation que nous avons faits et à approuver, par la même occasion, les résultats obtenus. Cela est donc l'objet de ce chapitre et nous estimons qu'il nous permettra de présenter les points techniques saillants de notre système.

3.2 Conception détaillée

Comme nous l'avons annoncé au précédent chapitre, dans cette partie nous allons finaliser la conception entamée. Cela consiste principalement en la conception de la base des données, de l'API et des interfaces. Plus précisément, il s'agira de conception et présentation.

3.2.1 Présentation des cas d'utilisation

L'objectif fondamental de cette section est d'identifier les principaux cas d'utilisation. Dans cette partie nous nous intéressons à la réalisation des diagrammes des cas d'utilisation.

Ces diagrammes décrivent précisément les besoins du client final et spécifient le comportement attendu par l'application à développer. Généralement un diagramme de cas d'utilisation modélise un service rendu par l'application.

Avant de faire l'étude des cas d'utilisation nous commençons par la recherche des acteurs qui régissent notre champ d'étude.

Dans notre système, nous avons défini trois acteurs dont, deux acteurs primaires et un acteur secondaire. Ces trois acteurs sont :

- **Enseignant** : un enseignant est un utilisateur inscrit dans le système en tant qu'enseignant et qui peut créer des cours, créer des épreuves, évaluer et approuver la correction du modèle dans ses épreuves.
- **Etudiant** : un étudiant représente un utilisateur qui peut s'enregistrer dans le système, joindre un cours et participer aux différentes épreuves des cours, recevoir la note et le feedback sur chaque épreuve.
- **LLM** : C'est le modèle de langage large qui a comme rôle de noter et générer un feedback constructif pour chaque étudiant et dans chaque évaluation.

3.2.2 Diagramme de cas d'utilisation

Le diagramme de cas d'utilisation représente la structure des grandes fonctionnalités nécessaires aux utilisateurs du système. C'est le premier diagramme du modèle UML, celui où s'assure la relation entre l'utilisateur et les objets que le système met en œuvre. Le rôle des diagrammes de cas d'utilisation est de recueillir, d'analyser et d'organiser les besoins, et de recenser les grandes fonctionnalités du système [68] [69]. Le diagramme de cas d'utilisation donne une vision globale du comportement fonctionnel du système. Un cas d'utilisation représente le dialogue entre l'acteur et le système de manière abstraite. Dans un diagramme de cas d'utilisation, les utilisateurs, appelés acteurs, interagissent avec les cas d'utilisation (*use case*).

3.2.2.1 Diagramme de cas d'utilisation global

La figure suivante représente les besoins fonctionnels de nos deux acteurs.

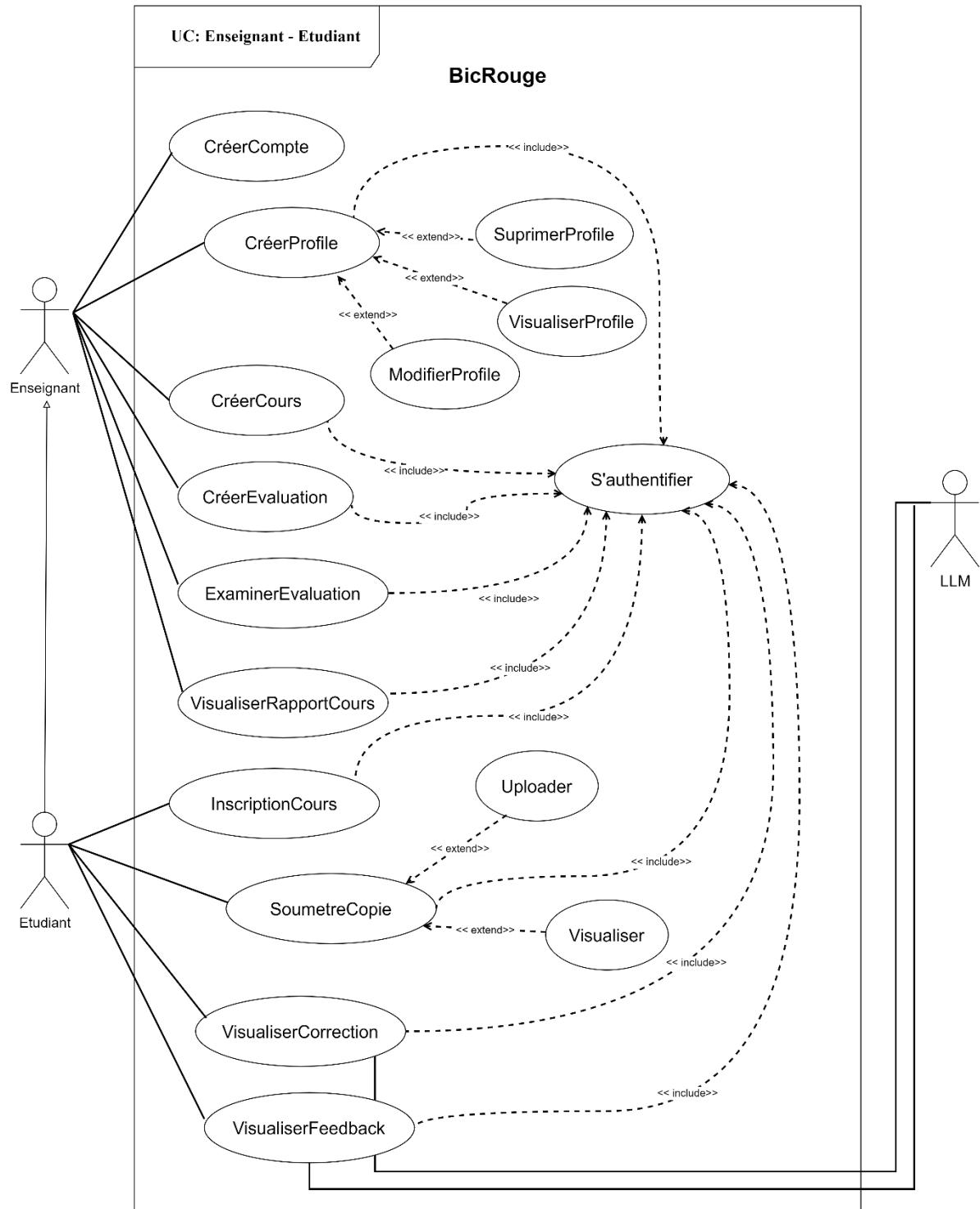


Figure 3.1 Diagramme des cas d'utilisation global

Comme on peut le remarquer sur la Error! Reference source not found. Ce diagramme modélise les interactions des acteurs présents dans le système.

Cas d'utilisation de l'enseignant :

- **CréerCompte** : Permet à l'enseignant de créer un nouveau compte pour accéder au système.
- **S'authentifier** : L'enseignant doit s'authentifier pour pouvoir accéder à son profil et gérer les cours. Ce cas d'utilisation est inclus dans plusieurs autres actions comme **ModifierProfile**, **CréerCours**, et **CréerÉvaluation**, car l'authentification est nécessaire avant d'effectuer ces actions.
- **GérerProfile** : Permet à l'enseignant de gérer son profil en incluant les actions suivantes :
 - **VisualiserProfile** : L'enseignant peut consulter les informations de son profil.
 - **ModifierProfile** : L'enseignant peut modifier son profil (inclus également **S'authentifier**).
 - **SupprimerProfile** : L'enseignant peut supprimer son profil.
- **CréerCours** : L'enseignant peut créer un cours.
- **CréerÉvaluation** : Une fois un cours créé, l'enseignant peut créer une évaluation (inclus également **CréerCours** et **S'authentifier**).
- **ExaminerÉvaluation** : Permet à l'enseignant d'examiner les évaluations soumises par les étudiants, les notes et le feedback du modèle.
- **VisualiserRapportCours** : L'enseignant peut accéder à un rapport des cours.

Cas d'utilisation spécifiques à l'étudiant :

- **InscriptionCours** : L'étudiant peut s'inscrire à des cours en utilisant des informations ou des codes fournis par les enseignants.

- **SoumettreCopie** : Une fonctionnalité clé permettant aux étudiants de soumettre leurs travaux ou copies pour correction. Cette action comprend également les extensions suivantes :
 - **Uploader** : L'étudiant peut télécharger un fichier pour soumission.
 - **Visualiser** : Pour bien visualiser et vérifier le formulaire de question avant de soumettre.
2. **VisualiserCorrection** : Une fois la correction terminée, l'étudiant peut consulter les notes et commentaires donnés par le modèle ou l'enseignant sur ses évaluations.
 3. **VoirFeedback** : L'étudiant peut accéder à un historique de toutes les corrections précédentes, facilitant la révision des notes et feedback reçus au fil du temps.

3.2.2.2 Documentation des différents cas d'utilisation

Tout acteur doit pouvoir s'authentifier afin d'accéder à son espace privé. Avant d'entrer ses identifiant il doit choisir son rôle (Enseignant ou Etudiant).

Tableau 5 : Documentation du cas d'utilisation « s'authentifier »

CU : S'authentifier
ID : 1
Description brève : Connexion au système
Acteurs primaires : Tous les acteurs
Acteurs secondaires : Aucun
Pré conditions : L'acteur s'est déjà connecté
Enchaînement principal : <ol style="list-style-type: none"> 1. Le CU commence quand l'utilisateur ouvre la page de connexion et entre son login et son mot de passe. 2. Il clique enfin sur le bouton de connexion
Post condition : Acteur connecté

Enchaînement alternatif :

1. **Erreur de login ou de mot de passe**
Le système vide les cases complétées et invite l'acteur à corriger son mot de passe et/ou son login.
2. **Erreur d'authentification (pas de connexion)**
Le système invite l'acteur à vérifier sa connexion.
3. **Erreur de rôle**
Le système invite l'acteur à vérifier son rôle.

3.2.2.2.1 Documentation du Cas d'utilisation « Créer un compte »

L'acteur doit pouvoir créer un compte dans le système afin d'obtenir les paramètres de connexion.

Tableau 6 : Documentation du Cas d'utilisation « créer un compte »

CU : Créer un compte
ID : 2
Description brève : Crédation du compte.
Acteurs primaires : Tous les acteurs
Acteurs secondaires : Aucun
Pré conditions : L'utilisateur doit accéder au système
Enchaînement principal : <ol style="list-style-type: none">1. L'acteur demande le formulaire d'inscription,2. Le système affiche le formulaire d'inscription,3. L'acteur complète et valide le formulaire d'inscription,4. Le système vérifie les données saisies par l'acteur5. L'acteur click sur le bouton créer un compte
Post condition : Un compte nouveau d'étudiant est créé

Enchaînement alternatif :

6. **Données saisies sont correctes,**

Le système redirige l'acteur vers la page de connexion.

7. **Données saisies sont incorrectes**

Le système redirige l'acteur vers son Dashboard et affiche un message de succès.

3.2.2.2.2 Documentation du Cas d'utilisation « CréerCours »

Une fois connecté L'enseignant doit pouvoir créer des cours et partager le code avec ses étudiants pour qu'ils puissent joindre le cours et répondre aux épreuves.

Tableau 7 : Documentation du Cas d'utilisation « CréerCours »

CU : CréerCours
ID : 3
Description brève : Crédit du cours
Acteurs primaires : Enseignant
Acteurs secondaires : Aucun.
Pré conditions : Acteur connecté (« authentifié »)
Enchaînement principal : <ol style="list-style-type: none">1. Après connexion, l'acteur est dirigé sur sa page d'accueil où se trouve une barre de navigation.2. Le CU commence quand il appuie sur « Créer cours » ouvert après avoir cliqué sur l'icône +.3. L'acteur voit alors un formulaire d'enregistrement du cours et est invité à, remplir le formulaire.
Post condition : Modifications enregistrées dans la BD
Enchaînement alternatif : <ol style="list-style-type: none">1. L'acteur oublier de remplir un champ obligatoire : Le système invite l'acteur à remplir le champ.

3.2.2.2.3 Documentation du cas d'utilisation « InscriptionCours »

Tableau 8 : Documentation du cas d'utilisation « InscriptionCours »

CU : InscriptionCours
ID : 4
Description brève : Pour répondre aux épreuves l'acteur doit joindre un cours et ainsi accéder à toutes les épreuves créer pour ce cours.
Acteurs primaires : Etudiant
Acteurs secondaires : Aucun
Pré conditions : Authentification (acteur connecté au système)
Enchaînement principal : <ol style="list-style-type: none"> 1. Après connexion, une un menu et une barre de navigation s'affiche 2. Dans la barre de navigation, il y a une icône « + » 3. Le CU démarre si l'acteur appui sur l'icône « + ». 4. Un bouton s'ouvre sous forme d'un popup 5. Une fois clique sur le bouton « Joindre un cours », un formulaire s'ouvre et l'acteur peut entrer le code du cours.
Post condition : Cours Ajouter à la page.
Enchaînement alternatif : <ol style="list-style-type: none"> 1. Code du cours non valide : Le système invite l'acteur à entre un code valide.

3.2.2.2.4 Documentation du cas d'utilisation « SoumettreCopie »

Tableau 9 : Documentation du cas d'utilisation « SoumettreCopie »

CU : SoumettreCopie
ID : 5
Description brève : Une fois connecter l'acteur peut accéder à ses courses et ainsi il peut voir toutes les épreuves qui lui sont assignées dans chaque cours et répondre aux questions soit sur la plateforme en remplissant le formulaire ou en uploadant un fichier.
Acteurs primaires : Etudiant
Acteurs secondaires : Aucun
Pré conditions : Authentification (acteur connecté au système)

Enchaînement principal :
1. Après connexion, une un menu et une barre de navigation ainsi qu'une liste cliquable de cours dans lesquels l'acteur est inscrit s'affiche.
2. Le CU démarre si l'acteur cliquer sur un cours.
3. L'acteur et diriger sur une page ayant une liste des épreuves créer pour ce cours.
4. Une fois clique sur l'une de épreuves, une page ayant les questions pour l'épreuve s'ouvrir et l'acteur peut répondre aux questions. Ou un formulaire ayant un espace pour uploader le fichier s'afficher.
Post condition : Epreuve soumis avec succès
Enchaînement alternatif :
1. L'échéance de l'épreuve et dépasser : Le système bloque le bouton « Soumettre » et indique l'acteur que l'échéance est finie.

3.2.2.2.5 Documentation du Cas d'utilisation « ExaminerEvaluation »

Une fois connecter L'enseignant doit pouvoir créer des cours et partager le code avec ses étudiants pour qu'ils puissent joindre le cours et répondre aux épreuves.

Tableau 10 : Documentation du Cas d'utilisation « ExaminerEvaluation »

CU : ExaminerEvaluation
ID : 6
Description brève : Création des paiement
Acteurs primaires : Enseignant
Acteurs secondaires : Aucun.
Pré conditions : Acteur connecté (« authentifié »)
Enchaînement principal :
4. Après connexion, l'acteur est dirigé sur sa page d'accueil où se trouve une barre de navigation et un menu.
5. Le CU commence quand il appuie sur « Notes » dans le menu.
6. L'acteur est dirigé sur une page, où s'affiche les informations de chaque étudiant et sa cote pour l'épreuve dans un Tableau.
7. S'il clique sur une ligne de son choix, les informations sur cette épreuve pour cet étudiant s'affiche ainsi il peut voir les réponses, la cote pour

chaque question, la cote totale et le feedback donner par le model. Ou le fichier envoyer par l'étudiant.

Post condition : Visualisation des côtes, des Feedback et modification si nécessaire.

Enchaînement alternatif : Aucun

3.2.3 Conception de la base des données

La conception d'une base des données passe en général par les étapes qui suivent [58] :

- L'analyse des besoins en stockage ;
- Le regroupement des données repérées comme utiles en tables selon leur affinité ;
- La spécification des clés primaires et l'analyse des relations entre tables ;
- La normalisation de la base des données.

Étant donné ce que nous avons précisé au chapitre précédent à la section **2.4.1** comme spécifications du système, et ensuite ce que nous avons mentionné comme rôles principaux de la base des données dans notre système, il se dégage les besoins suivants en termes de stockage :

- Il nous sera utile de stocker principalement les informations sur les cours, la composition de l'enseignants (question, réponse, critère de cotation), les réponses de l'étudiant, les cotes et le feedback de l'LLM pour chaque évaluation.
- Nous allons également stocker les informations d'authentification de utilisateurs (enseignant et étudiants), notamment l'email, les noms, l'institution de l'enseignant mais également les mots de passe.

Des considérations précédentes nous avons mise en place le diagramme des classes suivant :

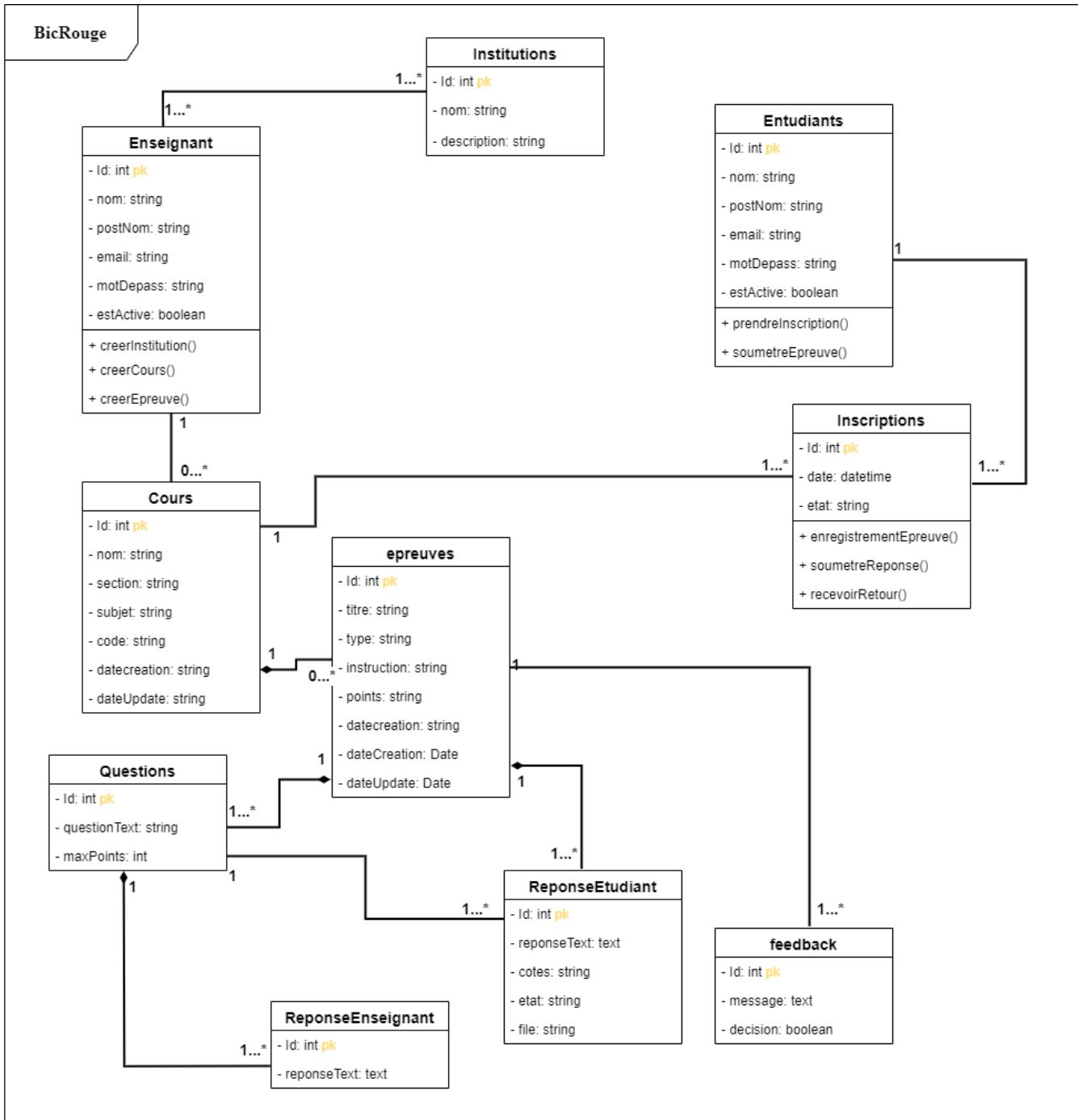


Figure 3.2 Diagramme des classes représentant la base des données

Pour mieux comprendre la modélisation de la base de données représentée à la **Figure 3.2**, voici par la présentation des données dans un dictionnaire de données (voir Tableau 9)

Tableau 11 Dictionnaire de données

Table	Codification	Libellé
Enseignant	Id	Identification de l'enseignant.
	nom	Nom de l'enseignant.
	postNom	Post-nom de l'enseignant.
	email	Adresse électronique de l'enseignant.
	motDepasse	Mot de passe de l'enseignant (cet attribut enregistra le Hash du mot de passe).
	estActiver	Attribut vérifier l'activité de l'enseignant sur la platform.
Etudiant	Id	Identifiant de l'étudiant.
	nom	Nom de l'étudiant.
	postNom	Post-nom de l'étudiant
	email	Adresse électronique de l'étudiant
	motDepasse	Mot de passe de l'enseignant (cet attribut enregistra le Hash du mot de passe).
	estActiver	Attribut vérifier l'activité de l'enseignant sur la platform.
Cours	Id	Identifiant du cours
	nom	Nom du cours
	section	Promotion ou classe dans laquelle le cours est dispensé.
	domaine	Domaine du cours (Langues, science, ...)

	dateCreation	La date de création du cours
	dateUpdate	La date de mise à jour du cours.
Epreuves	Id	Identifiant de l'épreuve.
	title	Intitulé de l'épreuve.
	type	Type d'épreuve.
	instruction	Instruction de l'épreuve.
	points	La note de l'épreuve.
	dateCreation	Date de création de l'épreuve.
	dateUpdate	Date de mise à jour de l'épreuve.
Institutions	Id	Identifiant de l'institution
	nom	Nom de l'institution
	description	Description de l'institution.
Questions	Id	Identifiant de la question
	questionText	L'intitulé de la question.
	maxPoints	Note de la question.
ReponseEnseignant	Id	L'identifiant de la réponse de l'enseignant.
	reponseText	L'intitulé de la réponse de l'enseignant.
ReponseEtudiant	Id	L'identifiant de la réponse de l'étudiant.
	reponseText	L'intitulé de la réponse de l'étudiant.
	cote	Note de l'étudiant sur l'évaluation.

	Commentaire	Commentaire du model sur la réponse de l'étudiant.
	etat	Estat de la correction approuver ou non approuver.
	file	Fichier de l'épreuve.
Feedback	Id	Id du feedback (commentaire global du model sur l'évaluation)
	message	Commentaire du model
Inscription	Id	Identifiant de l'inscription de l'étudiant au cours
	date	Date de l'inscription
	etat	Estat de l'inscription.

3.2.4 Conception de l'API

La création d'une interface de programmation d'applications (API) repose sur une réflexion approfondie concernant les services à offrir via cette API. Dans notre cas, nous visons à développer une plateforme qui permettrait à l'enseignant de créer un cours, de partager le code du cours avec les étudiants, de concevoir des épreuves au sein de ce cours, et de valider les notes générées par le modèle de langage (LLM). Par ailleurs, l'API doit également offrir aux étudiants la possibilité de rejoindre un cours, de consulter et répondre aux questions des épreuves proposées, ainsi que de recevoir automatiquement une correction avec les notes et feedbacks associés à leurs réponses après la soumission de l'épreuve. Pour cela, nous avons prévu 2 familles d'end-points :

- 1) End-points réservés à la notation et l'envoie de feedback ;
- 2) End-points réservés à tout ce qui concerne les vues de l'API (ses interfaces devant permettre l'authentification, l'enregistrement, la gestion des cours la gestion d'épreuves, les questions, les réponses de l'étudiants et l'enrôlement) ;

End-points réservés à la notation et l'envoie de feedback

Pour la notation et l'envoi des feedbacks, nous avons défini quatre principaux end-points. Le premier permet l'envoi de la correction de l'enseignant, ainsi que des critères de correction et des réponses de l'étudiant. Le deuxième end-point est dédié à la gestion des notes et des feedbacks pour une épreuve donnée. Le troisième résume les performances globales de l'étudiant dans un cours, incluant la cote totale, la moyenne. Enfin, le quatrième end-point permet la mise à jour des notes de l'étudiant de manière dynamique.

On a l'end-point de notation et envoie des feedbacks donnés par :

POST `/api/get_grading/` Get Grading Endpoint



Celui de la sélection de notes et feedback and une épreuve donnée :

GET `/api/grading_feedback/{student_id}/{assignment_id}` Get Grading Feedback



Celui du résumé des performances d'étudiant dans un cours :

GET `/api/courses/{course_id}/all-assignments` Get All Course Assignments And Students



Et celui de la mise à jour de cotes de l'étudiant :

PUT `/api/assignments/{assignment_id}/students/{student_id}/questions/{question_id}` Update Student Grade



End-points réservés à tout ce qui concerne les vues de l'API

Tout d'abord, le point d'entrée est donné par l'end-point :

GET `/api/home` Root



Ensuite, les end-points réservé à l'authentification :

- L'authentification de l'enseignant :

POST /api/teacher/token Login

POST /api/teacher/refresh Refresh Token Endpoint

GET /api/teacher/me Read Users Me

Elle comprend un end-point pour la connexion (**login**), un autre pour actualiser le jeton d'authentification (**refresh token**), et un dernier pour obtenir les informations de l'utilisateur courant.

- L'authentification de l'étudiant :

POST /api/student/token Login

POST /api/student/refresh Refresh Token Endpoint

GET /api/student/me Read Users Me

Ensuite, les end-points réservé aux enregistrements :

- L'enregistrement de l'enseignant :

POST /api/teachers/register Create Teacher

GET /api/teachers/institutions Get Institutions

Elle comprend un end-point pour l'inscription d'un enseignant (**register**), ainsi qu'un autre pour obtenir la liste des institutions (**institutions**).

- Celui réservé à l'enregistrement de l'étudiant :

POST /api/students/register Create Student

Par après on a les end-points réservé à la gestion des cours :

GET	/api/courses Fetch All Courses		
POST	/api/courses Create New Course		
GET	/api/courses/{course_id} Fetch Course		
DELETE	/api/courses/{course_id} Delete Existing Course		
PUT	/api/courses/{course_id} Update Existing Course		
GET	/api/courses/{course_id}/teacher-and-students Get Course Teacher And Students		

Ils sont constitués d'un end-point pour récupérer tous les cours (**fetch all courses**), un autre pour créer un nouveau cours (**create new course**), et d'autres pour récupérer un cours spécifique (**fetch course**), mettre à jour (**update existing course**), ou supprimer un cours (**delete existing course**). Enfin, un end-point est disponible pour obtenir les informations sur l'enseignant et les étudiants d'un cours spécifique (**get course teacher and students**).

Puis ceux réservés à la génération des épreuves :

POST	/api/assignments Create New Assignment			
GET	/api/courses/{course_id}/assignments Fetch All Assignments			
GET	/api/assignments/{assignment_id} Fetch Assignment			
DELETE	/api/assignments/{assignment_id} Delete Existing Assignment			
PUT	/api/assignments/{assignment_id} Update Existing Assignment			
GET	/api/students/{student_id}/courses/{course_id}/assignments Fetch Student Assignments			

Voici leur description :

- Un end-point pour créer un nouveau devoir (**create new assignment**).
- Un autre pour récupérer tous les devoirs d'un cours spécifique (**fetch all assignments**).
- Un end-point pour obtenir les détails d'un devoir spécifique (**fetch assignment**).

- Un autre pour supprimer un devoir existant (**delete existing assignment**).
- Un end-point pour mettre à jour les informations d'un devoir (**update existing assignment**).
- Enfin, un end-point est disponible pour récupérer les devoirs d'un étudiant spécifique dans un cours donné (**fetch student assignments**).

Puis vient ceux réservés à la génération des questions :

POST	/api/questions Create Questions Endpoint		
GET	/api/questions/{assignment_id} Get Questions Assignment Endpoint		
PUT	/api/questions/{question_id} Update Question		

Ils sont constitués d'un end-point pour créer une nouvelle question (**create new question**), un autre pour récupérer toutes les questions associées à un devoir donné (**fetch assignment questions**), et un autre pour mettre à jour une question spécifique (**update existing question**). Ces end-points permettent de gérer l'ajout, la récupération et la modification des questions d'un devoir dans l'application.

Puis ceux réservés à la génération des réponses de l'étudiants :

POST	/api/student_responses/ Create Student Responses Endpoint		
GET	/api/student_responses/assignment/{assignment_id}/student/{student_id}/question/{question_id} Get Student Responses By Assignment Student And Question Endpoint		

Ils sont constitués d'un end-point pour créer une nouvelle réponse de l'étudiant (**create student response**) et un autre pour récupérer les réponses d'un étudiant pour une question spécifique dans une épreuve donnés (**fetch student response by assignment, student, and question**).

Finalement, on a un end-point pour la gestion de enregistrent des étudiants :

POST	/api/enrollments/ Create Enrollment Route	
DELETE	/api/enrollments/ Delete Enrollment Route	
GET	/api/students/{student_id}/enrollments Get Enrollments By Student Route	
GET	/api/students/{student_id}/courses Get Courses By Student Route	

Ils sont constitués d'un end-point pour créer un nouvel enrôlement au cours (**create enrollement**), un autre pour effacer l'enrôlement (**delete enrollement**), un autre pour la sélection d'un enrôlement a un cours pour un étudiant donné, et le dernier end-point pour sélectionner les cours dans lesquels l'étudiant est enrôlé.

3.2.5 Présentation des interfaces réalisés

Étant donné l'API mise sur pieds, nous avons également implémenté un client web devant s'en servir pour en illustrer le fonctionnement. Pour cela, nous avons voulu exploiter toutes les possibilités de l'API en se basant sur les spécifications du système énumérées dans le chapitre précédent ainsi que les objectifs de ce travail. Le développement de ce client a été réalisé en utilisant **React.js**, une bibliothèque JavaScript permettant de créer des interfaces utilisateur dynamiques et modulaires, facilitant ainsi la gestion de l'état et le rendu efficace des composants. De plus, **Tailwind CSS** a été utilisé pour la mise en forme et le style des interfaces. Tailwind est un framework CSS qui permet de concevoir rapidement des interfaces élégantes et responsives grâce à son approche utilitaire, réduisant ainsi le besoin de CSS personnalisé. Ainsi, sur la base des ébauches d'interfaces qui ont été présentées à la section **ci-dessus**, nous avons adopté ce qui suit comme interfaces du client de notre système.

Comme mentionné précédemment dans la section **3.2.1**, notre système comprend deux types d'utilisateurs : l'enseignant et l'étudiant. Les interfaces décrties us sont conçues pour suivre le flux de travail spécifique à chaque utilisateur, en fonction de leurs rôles respectifs.



Figure 3.3 Page d'accueil

La **Figure 3.3** présente la page d'accueil du système.

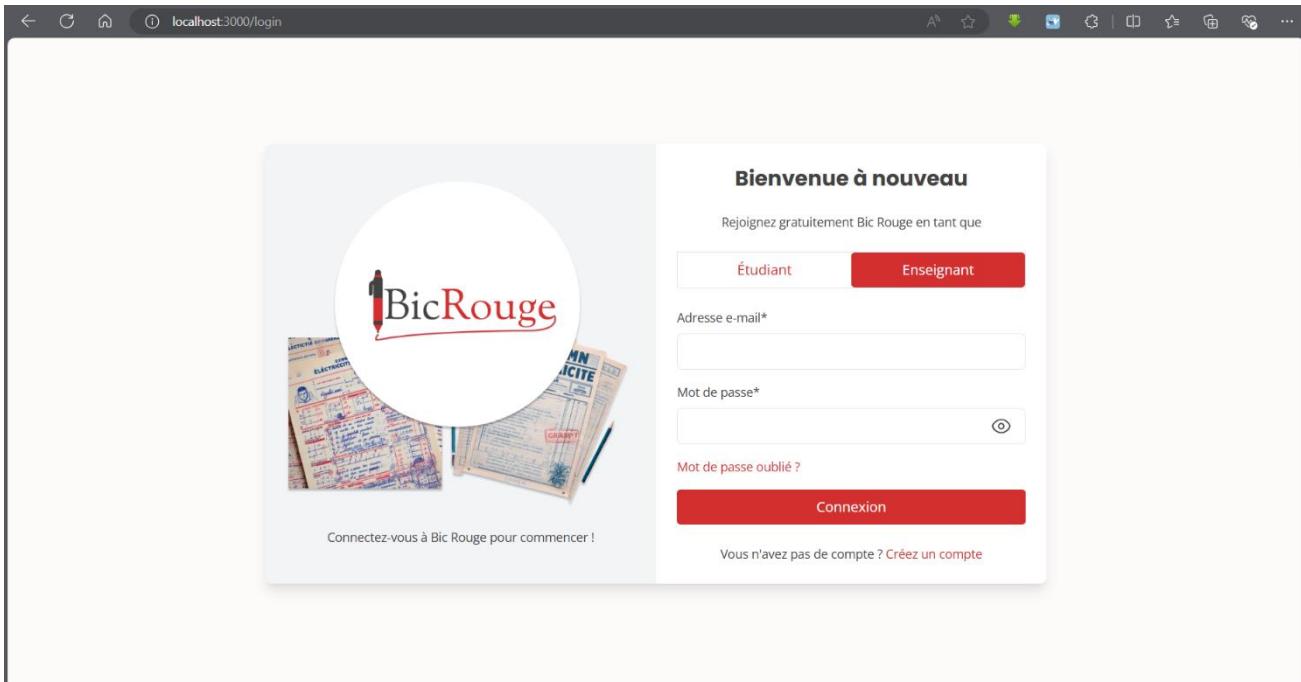


Figure 3.4 Page de login

Sur la **Figure 3.4** montre la page de connexion du système.

localhost:3000/signup?role=teacher

Créer un compte

Rejoignez gratuitement Bic Rouge en tant que

Étudiant Enseignant

Adresse e-mail*

Nom* Postnom*

Prénom

Sélectionnez votre institution*

Université Catholique de Bukavu (UCB)

Institution introuvable ? [Créez-la !](#)

Mot de passe*

Les mots de passe doivent comporter au moins 8 caractères et inclure une combinaison de lettres, de chiffres et d'autres caractères spéciaux.

[inscription](#)

Vous avez déjà un compte ? [Connexion](#)

Figure 3.5 Page d'enregistrement

La **Figure 3.5** présente la page d'enregistrement du système.

localhost:3000/student/dashboard

BicRouge

- Home
- Cours
- Epreuves
- Cours archivés
- Paramètres
- Déconnexion

Français L1 N'Kumbi

Initiation à la... L1 Mbuyi

Physique généra... L1 Mbuyi

Statistiques L1 Mbuyi

VIH et Santé pu... L1 N'Kumbi

Figure 3.6 Dashboard présentation des cours

La **Figure 3.6** présente le tableau de bord du système avec les cartes des différents cours.

1. Flux de travail de l'enseignant

Après s'être enregistré et connecté au système, l'enseignant est redirigé vers son tableau de bord, comme montré à la **Figure 3.7**. Sur la gauche, un menu latéral présente les options "**Home ou Accueil**", "**Cours**", "**Épreuves**", "**Cours archivés**", "**Paramètres**" et "**Déconnexion**". La partie principale du tableau de bord affiche trois cartes de cours, car l'enseignant a déjà créé trois cours.

Le flux de travail commence lorsque l'enseignant clique sur l'icône "+" dans la barre de navigation, ce qui ouvre une fenêtre contextuelle avec un bouton "**Créer cours**". En cliquant sur ce bouton, un formulaire apparaît (**Figure 3.8**) pour la création d'un cours. Une fois le cours créé, une nouvelle carte de cours s'ajoute au tableau de bord. En cliquant sur cette carte, une page s'ouvre avec un sous-menu récapitulant toutes les tâches que l'enseignant peut effectuer.

Le premier onglet, "**Flux**", présente un résumé du cours, y compris le titre, le code et les épreuves en cours (**Figure 3.9**). En cliquant sur l'onglet "**Tâches**", une page permettant de créer et de visualiser toutes les épreuves créées s'ouvre (**Figure 3.10**). En cliquant sur le bouton "**Créer**", un pop-up s'ouvre avec plusieurs types d'épreuves à choisir (interrogation, examen, devoir, etc.) comme montré à la **Figure 3.11**. Après avoir sélectionné le type d'épreuve, un formulaire apparaît pour remplir les détails de l'épreuve (**Figure 3.12**). Une fois l'épreuve créée, elle s'affiche sur la page, accompagnée d'un message de confirmation (**Figure 3.13**).

En cliquant sur la carte de l'épreuve créée, la carte s'expand, exposant plus d'information sur l'épreuve comme la date de création, l'échéance, et le nombre de soumissions (**Figure 3.14**).

En sélectionnant "**Composer**", un autre pop-up s'ouvre pour composer les questions et réponses de l'épreuve (**Figure 3.15**). Après avoir rempli le formulaire, l'enseignant peut enregistrer, et la fenêtre se ferme.

Sous l'onglet "**Apprenants**" (**Figure 3.16**), l'enseignant peut voir tous les étudiants inscrits au cours. Enfin, dans l'onglet "**Notes**" (**Figure 3.27**), l'enseignant accède à une vue d'ensemble des notes pour chaque épreuve et pour chaque étudiant, avec des options pour modifier et valider les corrections de l'LLM, si nécessaire (**Figure 3.19**).

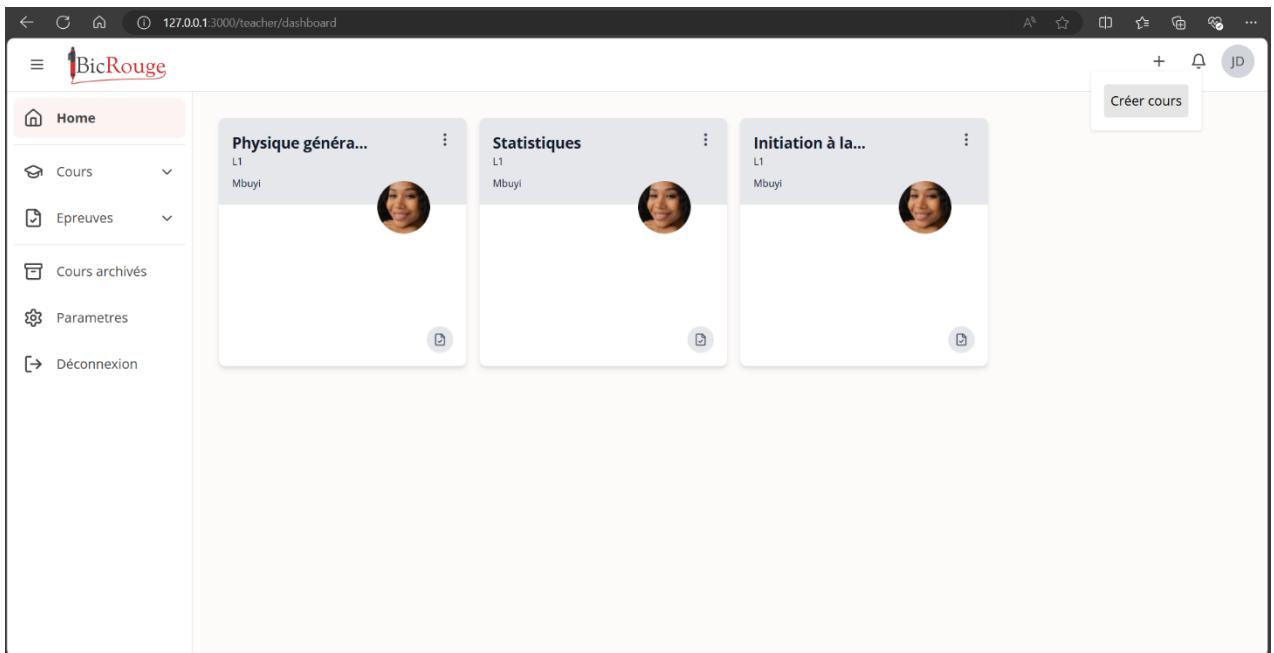


Figure 3.7 Dashboard, Button de création du cours

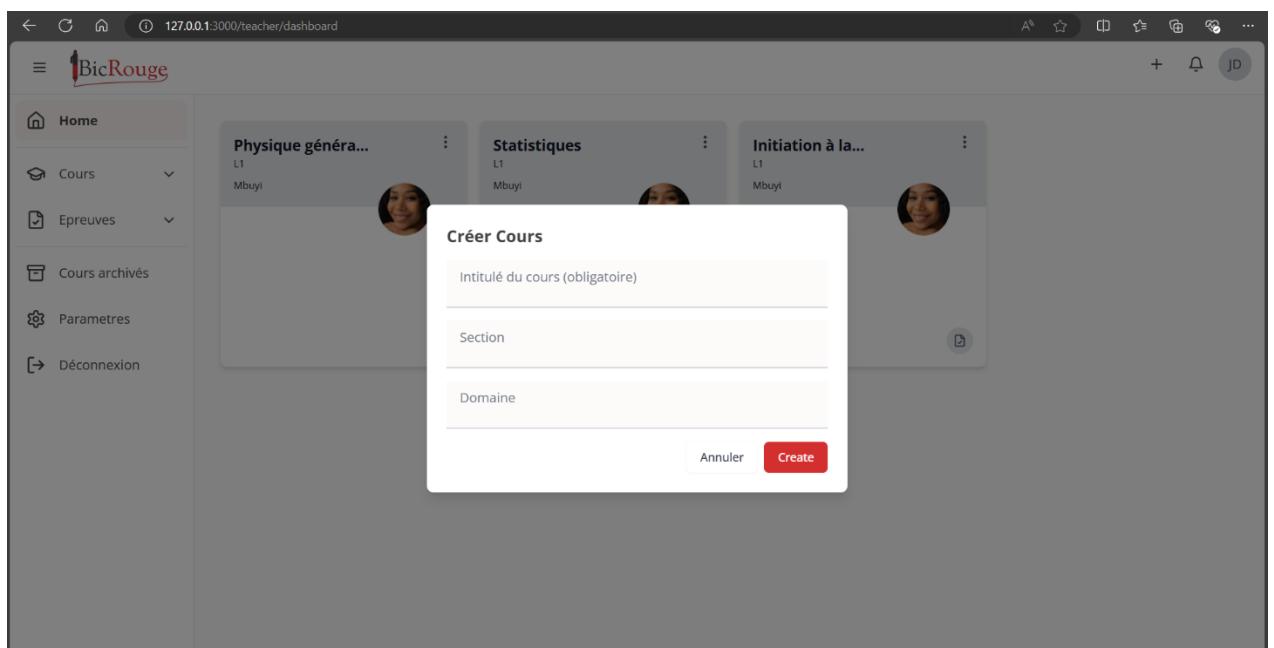


Figure 3.8 Formulaire de création du cours

The screenshot shows the BicRouge teacher dashboard for a course titled "Physique générale" (General Physics) by Mbuyi. The dashboard has a sidebar on the left with links for Home, Cours, Epreuves, Cours archivés, Paramètres, and Déconnexion. The main content area is divided into sections: "Flux", "Tâche", "Apprenants", and "Notes". The "Flux" section displays a large image of a laptop and two smartphones. Below it, there's a box for the course code "LISJ4RO" and another for "Annonces à propos du cours". The "Tâche" section is currently selected and shows a red button "+ Créer". The message "Vous n'avez aucun épreuve créée" (You have not created any exam yet) is displayed.

Figure 3.9 Dashboard détails cours pour l'enseignant

This screenshot shows the same BicRouge teacher dashboard as Figure 3.9, but with the "Tâche" tab selected. The main content area features a prominent red button labeled "+ Créer". Below the button, a message states "Vous n'avez aucun épreuve créée" (You have not created any exam yet).

Figure 3.10 Onglet de création d'épreuves

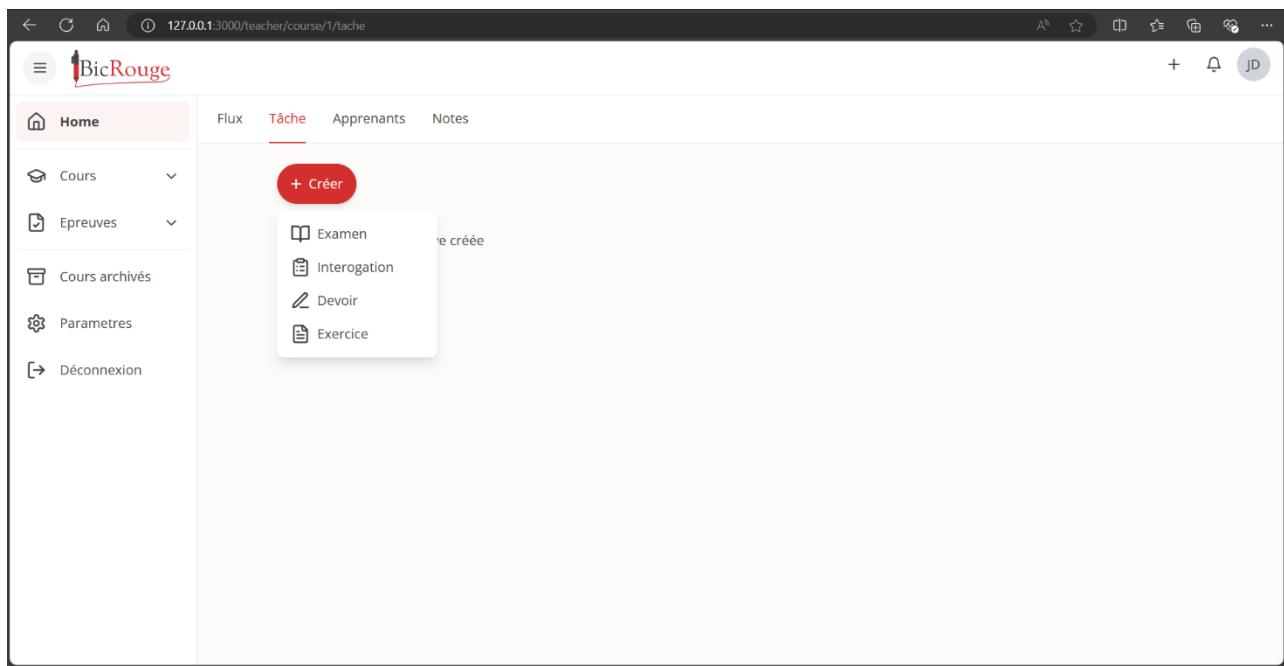


Figure 3.11 Onglet de création d'épreuves, choix de l'épreuve

A screenshot of a 'Create new task' form. The left side features a rich text editor with a placeholder text 'Interrogation de rattrapage d'électricité générale' and a toolbar with various text styling options. Below the editor is an 'Attacher' section with three buttons: 'Composer' (with a plus sign icon), 'Lien' (with a link icon), and 'Télécharger' (with an upward arrow icon). The right side of the form contains several input fields: 'For' (set to 'Physique générale'), 'Points' (set to 10), 'Échéance' (set to 'Selectionner une date' with a date picker showing '09/08/2024 03:47 AM'), and 'Type' (set to 'Interrogation'). At the top right of the form is a red 'Assigner' button.

Figure 3.12 Formulaire de création de l'épreuve

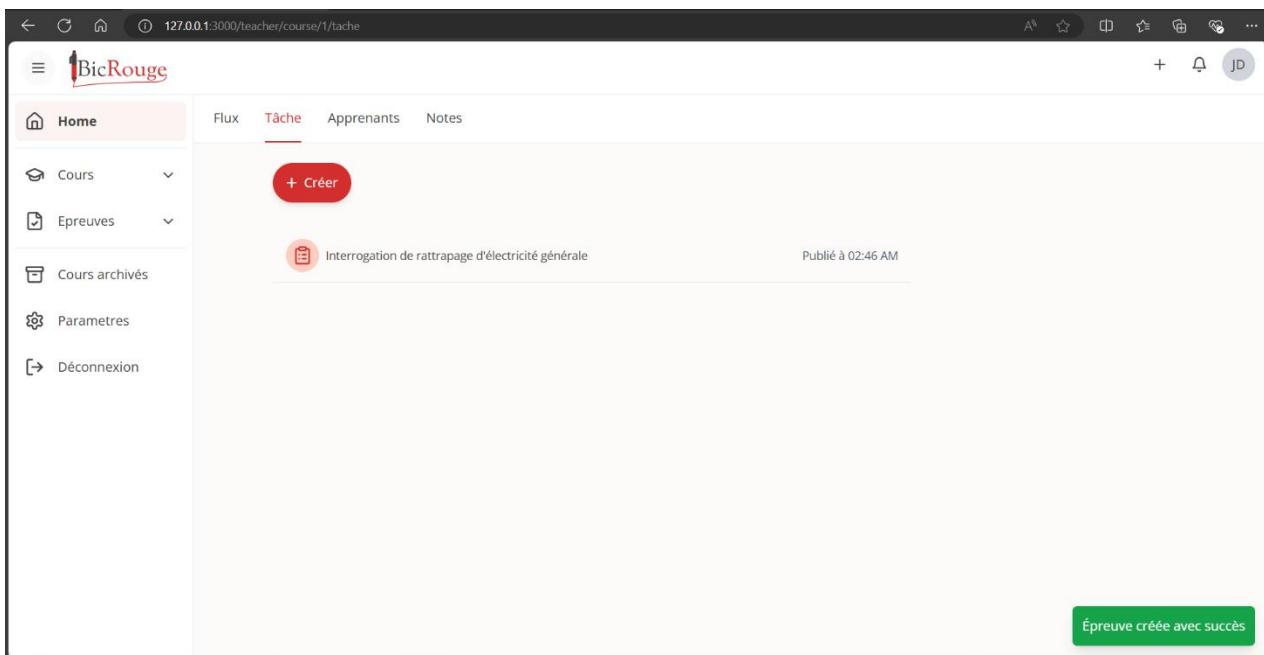


Figure 3.13 Présentation de l'épreuve créer

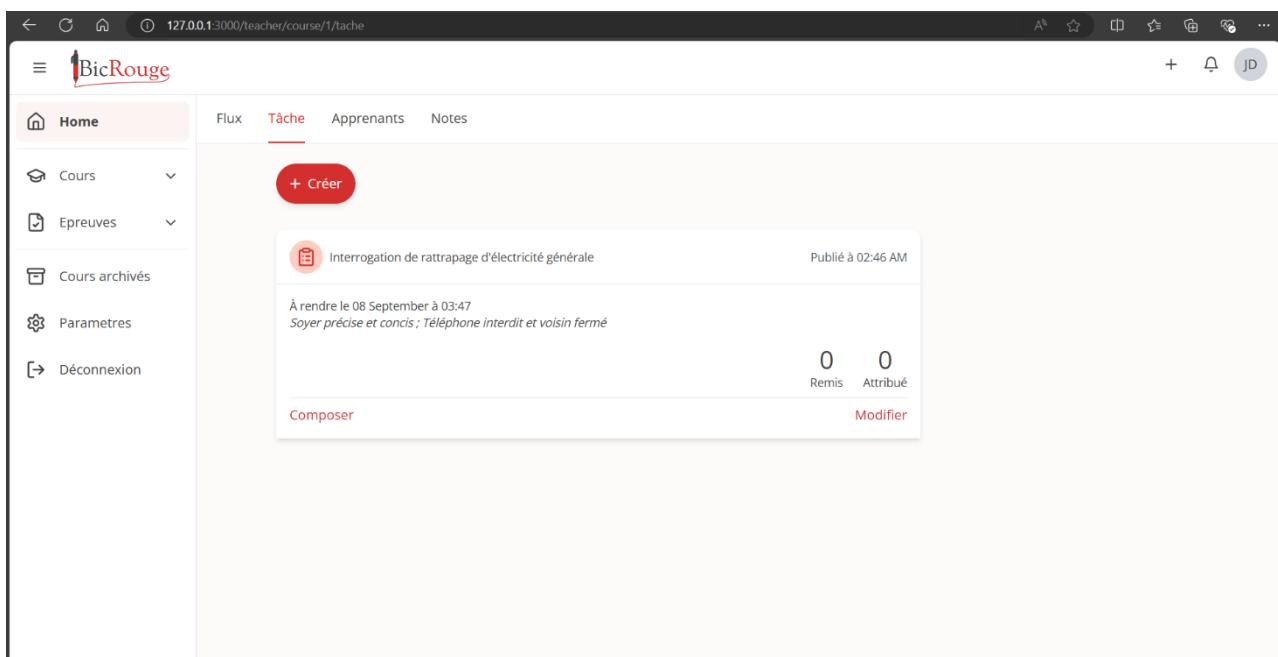


Figure 3.14 Visualisation de l'épreuve après création

The screenshot shows a web application for creating assignments. At the top, there's a header with a back arrow, a refresh button, and a URL bar showing '127.0.0.1:3000/teacher/courses/1/assignments/1'. On the right side of the header are several small icons. Below the header, a red banner displays the title 'Interrogation de rattrapage d'électricité générale' and a subtitle 'Soyer précise et concis ; Téléphone interdit et voisin fermé'. In the center, there's a question box for 'Question 1' with the text: 'Par rapport à leurs constitutions, quelle est la différence entre un conducteur et un isolant?'. Below it is a text input field containing the same question. A second question box for 'Question 2' is partially visible. At the bottom left is a red button labeled '+ Ajouter une question'.

Figure 3.15 Formulaire de composition de l'épreuve

The screenshot shows a web application for managing course participants. The top navigation bar includes a back arrow, a refresh button, and a URL bar showing '127.0.0.1:3000/teacher/course/1/apprenants'. The main menu on the left has items: Home (selected), Cours, Epreuves, Cours archivés, Paramètres, and Déconnexion. The 'Apprenants' tab is active. The interface is divided into two sections: 'Enseignant' (Teacher) and 'Etudiants' (Students). Under 'Enseignant', there's a profile for 'MK Mbuyi Kalala'. Under 'Etudiants', there's a list of 6 students: Amani Mwamba, Kinshasa Bakari, Tshepo D'Silva, Bakari Tshibanda, and Mwanza Mukendi, each with a small circular icon and a checkbox next to their name.

Figure 3.16 Présentation des participant du cours (Esseignant et étudiants)

The screenshot shows a web browser window for the BicRouge application at the URL 127.0.0.1:3000/teacher/course/1/notes. The left sidebar contains navigation links: Home, Cours, Epreuves, Cours archivés, Paramètres, and Déconnexion. The main content area has tabs for Flux, Tâche, Apprenants, and Notes, with Notes selected. A section titled "Apprenants" displays a single record: "8 sept. Interrogation de rattrapage d'électricité générale sur 10". Below it, a note states "Moyenne du cours" and "Aucune épreuve soumise jusqu'à présent".

Figure 3.17 Vu d'ensemble d'une épreuve dans un cours sans soumission

The screenshot shows the same BicRouge application interface as Figure 3.17, but with multiple submissions listed under the "Apprenants" section. The table includes columns for student initials, name, email, and score. The scores are all 5.0.

Apprenant	Moyenne
AM Amani Mwamba amani.mwamba@example.com	9
KB Kinshasa Bakari kinshasa.bakari@example.com	5
TD Tshepo D'Silva tshepo.dsilva@example.com	5
BT Bakari Tshibanda bakari.tshibanda@example.com	5
MM Mwanza Mukendi mwanza.mukendi@example.com	5

Figure 3.18 Vu d'ensemble d'une épreuve dans un cours avec soumission

Figure 3.19 Vérification et validation de cotes pour l'épreuve

2. Flux de travail de l'étudiant

Comme pour l'enseignant dans la **Figure 3.7**, le flux de travail de l'étudiant commence après l'enregistrement et la connexion au système sur le tableau de bord, où il peut voir les cartes des cours qu'il a déjà rejoints. Ici, il peut cliquer sur l'icône "+" dans la barre de navigation, et un pop-up s'ouvre, indiquant "**Joindre un cours**" (**Figure 3.20**). En cliquant, un autre pop-up s'ouvre, affichant un formulaire où l'étudiant peut entrer le code partagé par l'enseignant pour rejoindre le cours (**Figure 3.21**). En cliquant sur "**Joindre**", le pop-up se ferme et la carte du cours rejoint se charge sur le tableau de bord (**Figure 3.22**).

En cliquant sur la carte du cours, une nouvelle page se charge (**Figure 3.23**), avec une sous-barre de navigation comportant les onglets "**Flux**" (l'onglet courant), "**Notes**" et "**Apprenants**". Dans l'onglet "**Flux**", l'étudiant peut voir un résumé des informations sur le cours, y compris les épreuves en cours, le titre du cours et, s'il y a une épreuve en cours, le nom et la date limite de l'épreuve. Toutes les épreuves du cours sont listées en bas de la carte "**Annonce à propos du cours**" (**Figure 3.24**). En cliquant sur une carte d'épreuve, un pop-up en plein écran s'ouvre, affichant un formulaire avec toutes les questions et les champs pour répondre aux questions

(Figure 3.25). Après avoir répondu aux questions et cliqué sur le bouton "Envoyer", la correction est traitée (Figure 3.26). Si les notes sont déjà disponibles, l'étudiant est redirigé vers l'onglet "Notes", où il peut consulter la correction de toutes les épreuves (Figure 3.27). En cliquant sur la carte de l'épreuve souhaitée, un pop-up s'ouvre et l'étudiant peut voir ses notes ainsi que le feedback pour cette épreuve (Figure 3.28).

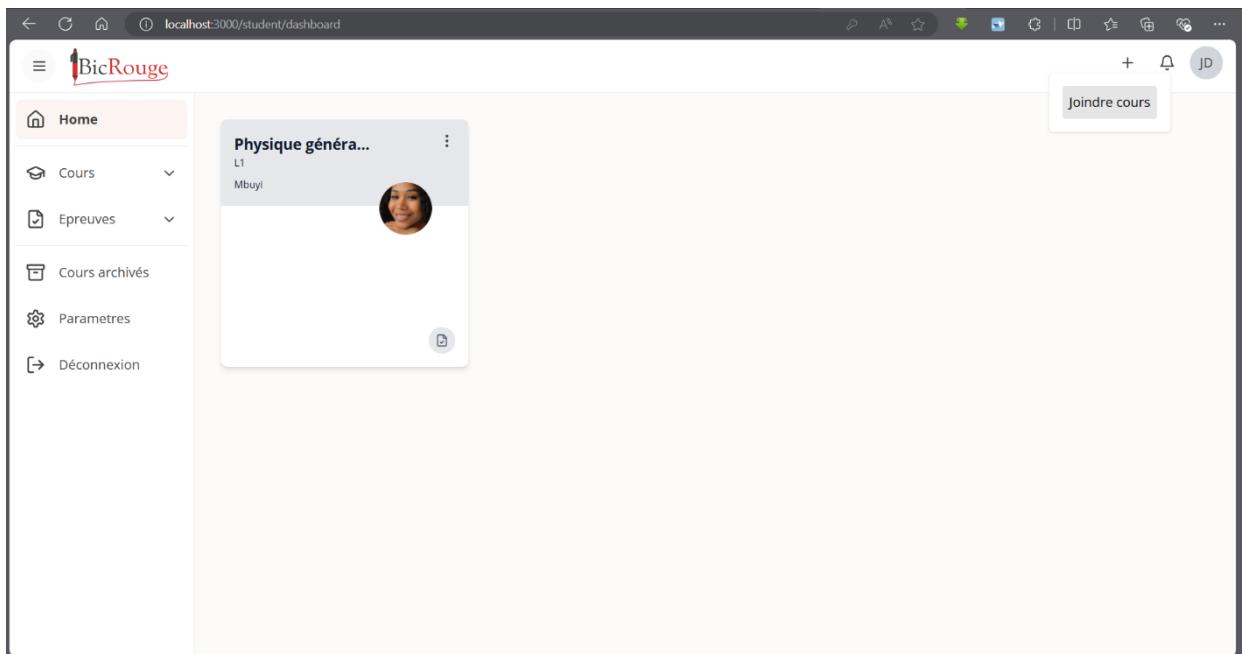


Figure 3.20 Dashboard : Button pour rejoindre un cours

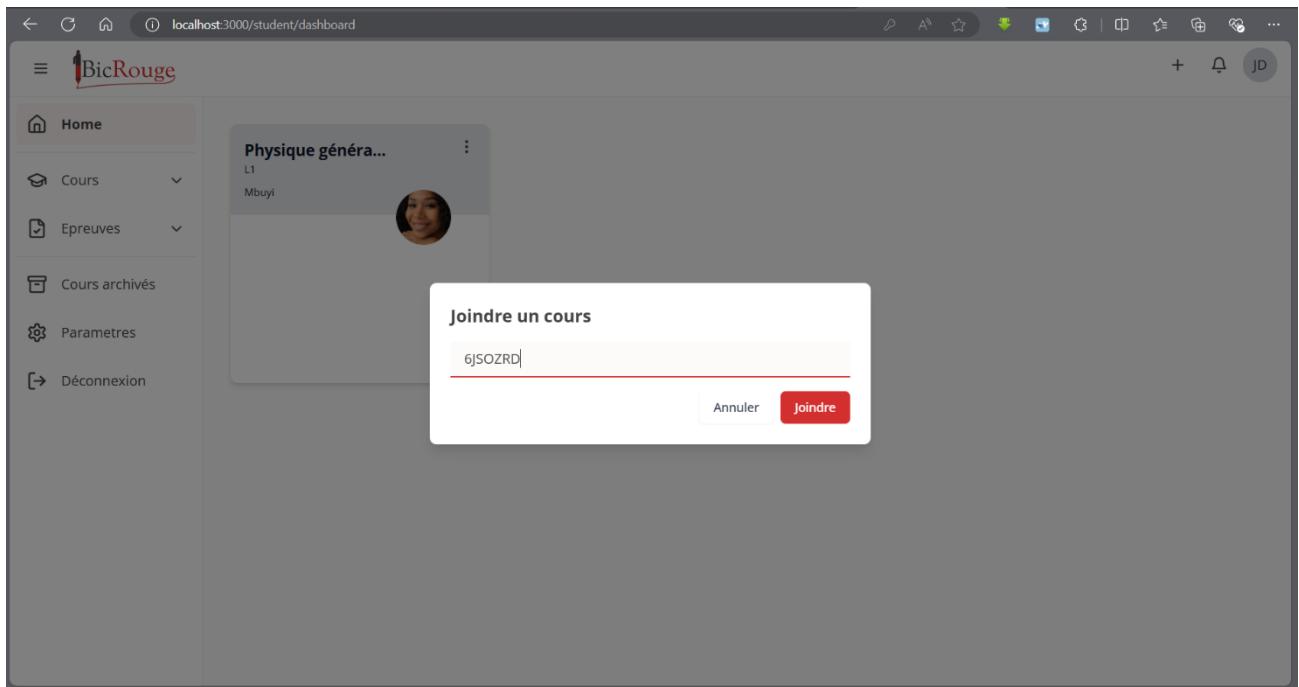


Figure 3.21 Formulaire pour joindre un cours

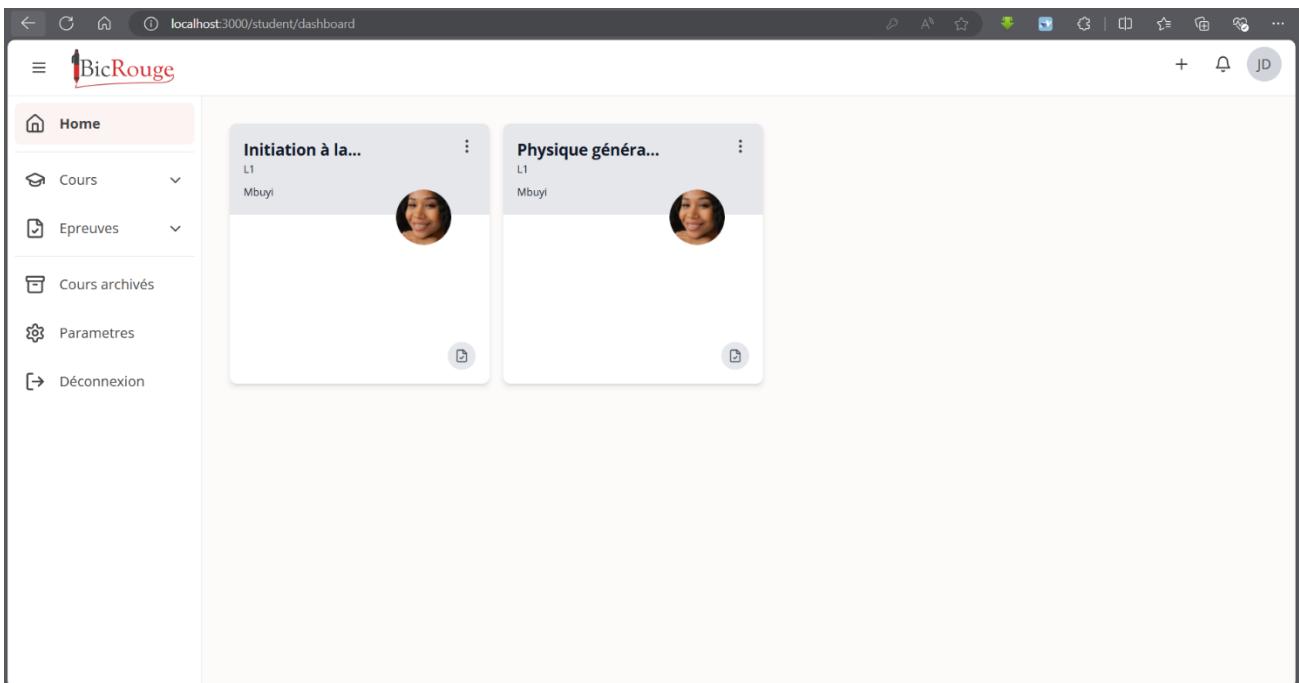


Figure 3.22 Dashboard : Affichage des cours

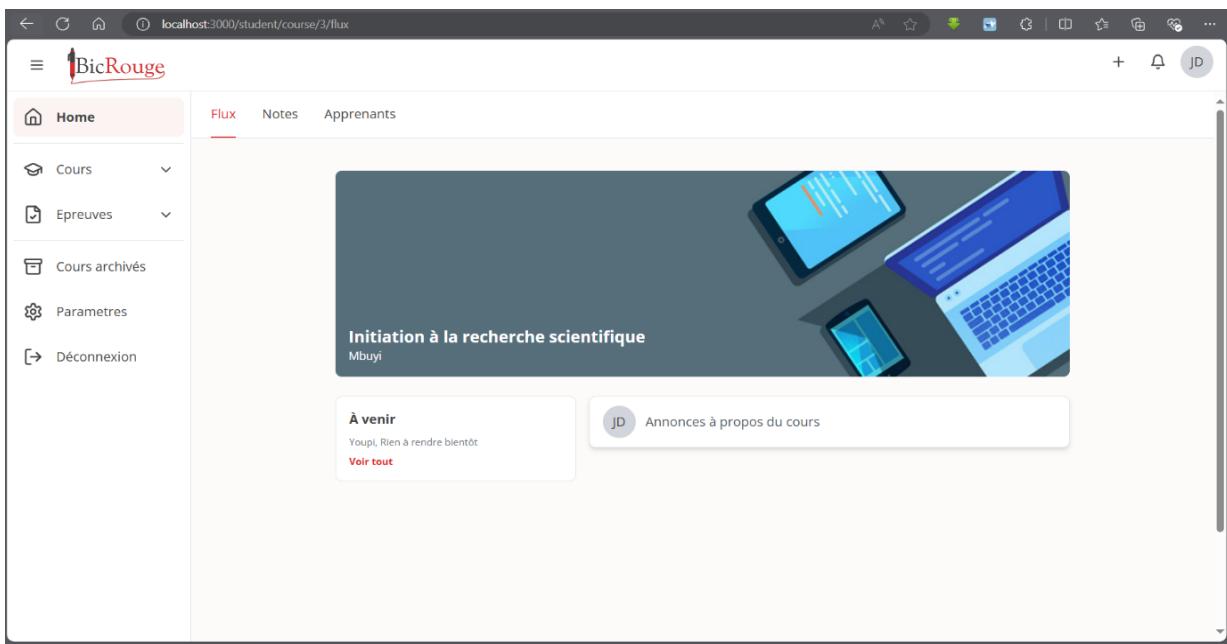


Figure 3.23 Dashboard : Affichage détails du cours

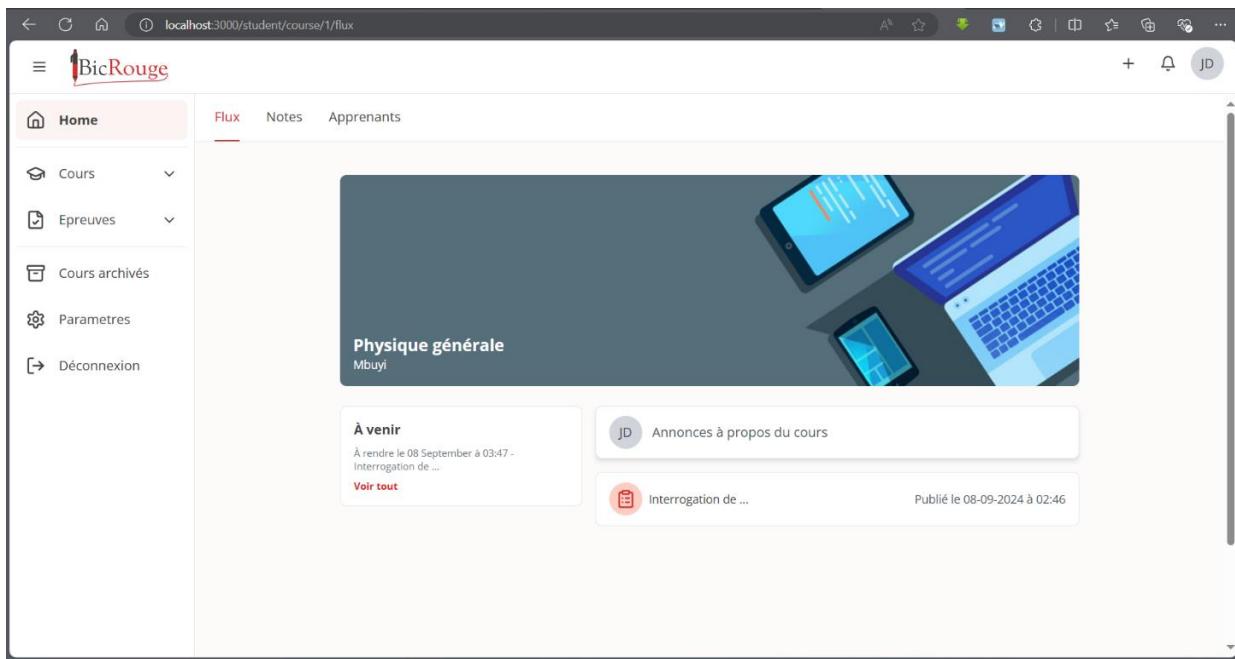


Figure 3.24 Dashboard détails cours avec épreuve créer

The screenshot shows a web browser window with the URL `localhost:3000/student/courses/1/assignments/1`. The page title is "Répondez aux Questions". At the top right are "Annuler" and "Soumettre" buttons. The main content area contains three questions:

- Interrogation de ratrappage d'électricité générale**: Soyer précise et concis ; Téléphone interdit et voisin fermé. /2
- Question 1**: Par rapport à leurs constitutions, quelle est la différence entre un conducteur et un isolant? /2
- Reponse:**
- Question 2**: Enoncez le Théorème de Thévenin? /2
- Reponse:**
- Question 3**: Quel est le rôle d'un générateur /2
- Reponse:**

Figure 3.25 Formulaire pour répondre aux questions de l'épreuve

The screenshot shows a web browser window with the URL `localhost:3000/student/courses/1/assignments/1`. The page title is "Répondez aux Questions". At the top right are "Annuler" and a refresh button. The main content area displays submitted answers for three questions:

- Question 3**: Quel est le rôle d'un générateur
Reponse: Le rôle d'un générateur c'est pour la production de l'énergie électrique afin d'être utilisé dans des récepteurs ou autres dispositifs électriques ou électroniques. /2
- Question 4**: Qu'est-ce qu'un circuit en série?
Reponse: Un circuit en série est un circuit électrique dans lequel les composants sont connectés de telle sorte que le courant suit un seul chemin. /2
- Question 5**: Qu'est-ce que la loi d'Ohm?
Reponse: La loi d'Ohm énonce que la tension aux bornes d'un conducteur est proportionnelle au courant qui le traverse, et la constante de proportionnalité est la résistance du conducteur. /2

Figure 3.26 Soumission de l'épreuve

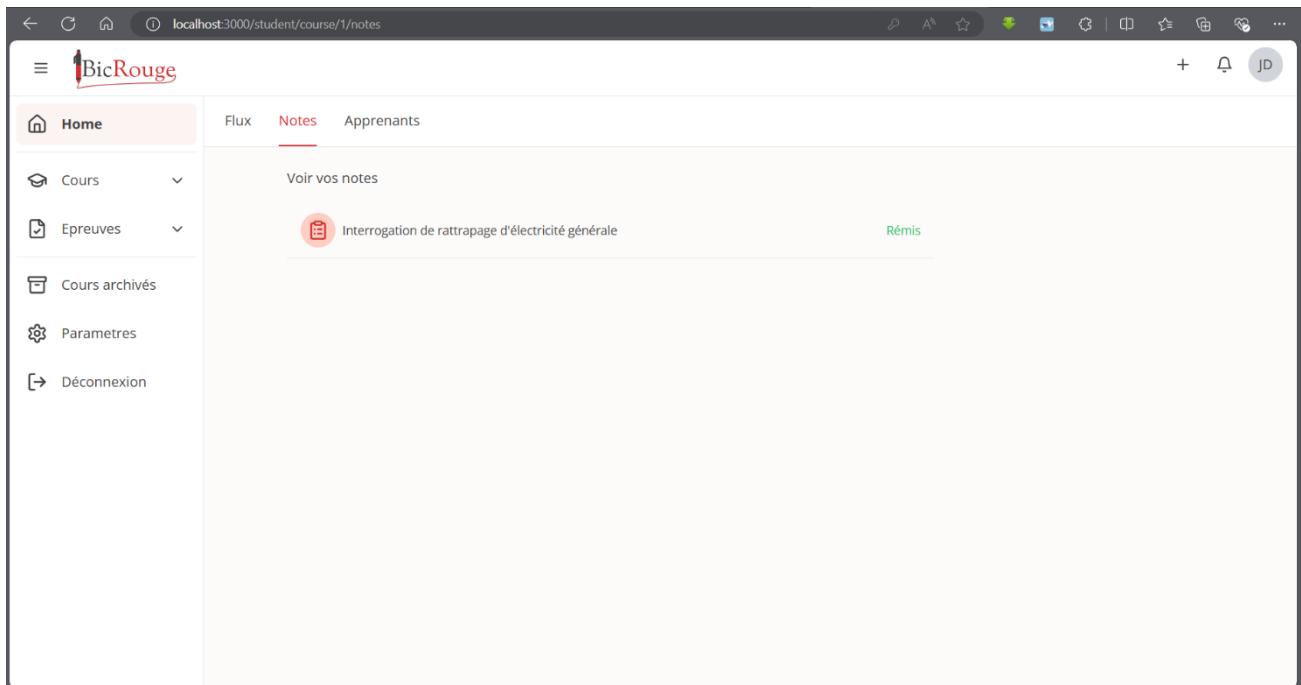


Figure 3.27 Historiques des épreuves

A screenshot of a web-based application window titled 'Mes Notes'. The top navigation bar includes icons for back, forward, search, and refresh, along with a URL 'localhost:3000/student/courses/1/assignments/1/notes'. The main header 'Mes Notes' has a red underline. The page displays an assignment titled 'Interrogation de rattrapage d'électricité générale'. It includes a 'Conseil' section with the text: 'Tu as une bonne compréhension générale des concepts, mais il y a des points spécifiques à améliorer dans tes réponses. Travaille sur la précision et l'ajout de détails pour renforcer tes réponses.' To the right, a box shows 'Échéance: 7 mai 2022, 7:00 AM' and 'Note provisoire: 9/10'. The assignment is divided into two sections: 'Question 1' (score 1.5/2) and 'Question 2' (score 2/2). Each section contains a 'Reponse:' field with a detailed feedback comment.

Figure 3.28 Epreuve corrigée avec feedback

Les autres fonctionnalités peuvent être explorées en utilisant le système.

Bien que notre système présente des similitudes visuelles avec Google Classroom, ses objectifs sont fondamentalement différents. Google Classroom est conçu pour la gestion des classes et ne propose l'automatisation des corrections que pour les questions à choix multiples via d'autres outils de la suite Google. En revanche, notre plateforme se concentre spécifiquement sur l'automatisation de la correction des épreuves théoriques, y compris la génération de retours constructifs pour les questions ouvertes. Elle peut également intégrer d'autres types de questions pour une évaluation plus complète.

3.3 Évaluation de la correction automatique et du feedback

L'évaluation de la correction automatique des épreuves et du feedback généré par des modèles de langage tels que Llama3 et GPT-3.5 Turbo repose sur une analyse à plusieurs niveaux. Ces modèles ont déjà été largement évalués à travers différents benchmarks comme MMLU (*Multitask Multidomain Language Understanding*), ou AGIEval (*Artificial General Intelligence Evaluation*.) [70], ce qui constitue leur évaluation intrinsèque, c'est-à-dire une évaluation directe de leurs performances sur des tâches spécifiques. Cependant, dans notre cas d'utilisation, nous allons nous concentrer davantage sur l'évaluation extrinsèque des modèles. Cela implique d'examiner comment ces systèmes se comportent dans des situations réelles, notamment en aidant à la correction d'épreuves et à la génération de feedback pédagogique. Cette approche permet de comparer la qualité des corrections générées par l'IA avec celles des enseignants, en tenant compte des critères d'évaluation comme la pertinence, la cohérence et la fluidité des réponses.

3.3.1 Évaluation intrinsèque [62].

L'évaluation intrinsèque se base sur l'utilisation de benchmarks spécifiques, tels que MMLU (Multitask Multidomain Language Understanding), AGIEval, et TriviaQA [70], qui permettent de mesurer la performance des modèles sur des tâches de compréhension et de raisonnement général. Ces benchmarks fournissent un cadre de référence objectif pour comparer la capacité des modèles à répondre à des questions ou résoudre des problèmes complexes, comme mentionné dans le chapitre précédent.

Les métriques comme la précision (Accuracy) et le F1-Score sont couramment utilisées pour évaluer leur performance sur ces benchmarks. Ces mesures offrent un cadre objectif pour juger de la qualité des réponses générées dans des contextes où il existe une réponse correcte prédefinie, comme la correction d'examens ou les tâches de compréhension de texte. La précision mesure le pourcentage de réponses correctes parmi celles générées, tandis que le F1-Score combine la précision (correctitude des réponses) et le rappel (capacité à trouver toutes les bonnes réponses).

3.3.2 Évaluation extrinsèque

L'évaluation extrinsèque est essentielle pour mesurer l'efficacité des modèles dans des contextes d'application réelle, comme la correction automatique des copies d'examen. Elle consiste à comparer les performances des modèles avec celles des enseignants humains, ce qui nous permet de déterminer dans quelle mesure les modèles peuvent faciliter la correction d'épreuves tout en maintenant une qualité comparable.

Pour ce faire, la concordance inter-évaluateurs est une méthode clé, qui mesure l'accord entre les corrections générées par l'IA et celles effectuées par les enseignants. Cette approche est particulièrement adaptée pour des tâches comme la correction d'épreuves, où il peut exister plusieurs réponses valides ou des différences d'interprétation, rendant l'évaluation strictement objective difficile. Afin de mener à bien ces évaluations, nous utiliserons **Azure AI Studio** comme outil central pour tester les performances et la fiabilité de notre système. Cet environnement nous permet non seulement d'évaluer les sorties générées par l'IA, mais

également de mesurer les risques et de garantir la sécurité du système. L'utilisation d'Azure AI Studio sera introduit dans la section **3.5** dédiée aux Tests et interprétation des résultats.

3.4 Description succincte de l'implémentation

Comme illustré par la figure **Figure 1.1**, notre système repose sur une architecture en trois tiers. Cette architecture se divise en trois parties principales : la partie cliente, la partie serveur, et la base de données. Nous allons clarifier ces éléments pour mieux expliquer leur fonctionnement et leur rôle dans l'architecture globale. L'implémentation du système repose sur les composantes suivantes :

1°) API REST : Nous avons implémenté une API REST pour traiter les données envoyées par les clients. Ce traitement consiste à gérer toutes les sollicitations venant de la partie cliente, principalement pour collecter les compositions des enseignants ainsi que les réponses des étudiants pour chaque épreuve. Ces données sont ensuite transmises aux modèles LLM pour correction et génération de feedback. Après traitement, les résultats produits par les modèles sont stockés puis renvoyés au client pour affichage.

2°) Base de données : Comme indiqué dans le point précédent, une base de données est essentielle pour notre système. Pour cette première version, nous avons choisi une base de données de type SQLite, adaptée à notre besoin de légèreté et de simplicité pour le prototypage.

3.4.1 API REST

L'API REST constitue le cœur du système. C'est ici que nous avons implémenté les divers modules du système ainsi que les prompts utilisés pour interagir avec les modèles LLM. Nous avons choisi d'utiliser le *framework* Python **FastAPI**, reconnu pour sa robustesse et sa popularité dans l'écosystème de l'ML (Machine Learning) [71]. L'API expose plusieurs *endpoints* que nous avons présenté en détails dans la section **ci-dessus** sur la présentation de l'API. Voici les informations nos deux modèles LLM que nous utilisons pour la correction des épreuves et la génération de feedbacks :

3.4.2 Client Web

La partie cliente, ou frontend, est l'interface visible du système, comme introduit dans la section **3.2.5**. Ces interfaces ont été développée à l'aide du framework JavaScript **ReactJS** pour sa flexibilité et son écosystème dynamique, et de **Tailwind CSS** pour la gestion du style, offrant une mise en page responsive et moderne. Cette partie assure une interaction fluide avec l'utilisateur et se connecte à notre API pour l'affichage des résultats et la gestion des soumissions.

Nous concluons cette section en présentant brièvement les techniques couramment utilisées pour la création de systèmes basés sur l'IA générative, tout en soulignant celles appliquées dans notre projet.

3.4.3 Techniques de réalisation de systèmes basés sur l'IA générative

1. Retrieval-Augmented Generation (RAG) [31]

Le RAG combine la génération de texte par un modèle avec un système de récupération d'informations. Il permet à l'IA de consulter des bases de données ou des documents externes pour fournir des réponses plus précises et actuelles. Dans notre projet, cette technique pourrait améliorer la précision des réponses tout en réduisant les erreurs de contenu.

2. Fine-tuning

Comme introduit dans la section **1.3.2.6**, Utiliser pour personnalise le modèle en fonction des besoins spécifiques du projet, comme la correction d'épreuves et la génération de feedbacks dans notre cas.

3. Exécution sur appareil (On-Device AI) [71]

L'exécution sur appareil permet de déployer des modèles directement sur des dispositifs comme les smartphones ou les ordinateurs, sans dépendre du cloud. Cela réduit les coûts, améliore la réactivité et renforce la sécurité des données, un avantage pour les applications sensibles.

4. Prompt Engineering [31]

L'ingénierie de prompt est essentielle pour orienter les réponses des modèles d'IA. Elle comprend des techniques telles que :

Zero-shot prompting : Le modèle répond à une question sans exemple préalable.

Few-shot prompting : Quelques exemples sont fournis pour guider la réponse.

Chain-of-thought prompting : Le modèle résout un problème en plusieurs étapes.

Self-refinement : Le modèle améliore ses propres réponses.

Pour notre projet, nous avons utilisé principalement *l'instruction tune prompting et le Few-shot prompting* pour optimiser les retours des modèles.

3.5 Tests et interprétation des résultats

Dans cette section, nous présentons les différentes techniques et outils utilisés pour tester notre système. Comme mentionné précédemment, nous avons principalement employé deux méthodes pour évaluer nos deux modèles LLM : l'évaluation manuelle et l'évaluation automatisée. Cette dernière a été réalisée à l'aide des outils avancés d'Azure AI Studio, qui offrent des fonctionnalités spécifiques pour analyser la performance, la sécurité et les risques des applications basées sur l'IA générative.

3.5.1 Datasets utilisés pour les tests

Lorsque nous parlons des tests, il s'agit de ceux effectués pour évaluer non seulement les performances individuelles de nos deux modèles, mais aussi celles de notre système dans son ensemble. Contrairement aux jeux de données généralement utilisés pour l'entraînement des modèles, nous avons opté ici pour des ensembles de données spécifiques, adaptés à notre contexte d'évaluation. Ces *datasets* ne sont pas de grande taille, mais sont soigneusement conçus pour refléter notre cas d'utilisation.

Nous avons structuré nos tests autour de deux petits jeux de données respectivement un par modèle contenant :

- Les données d'entrée utilisateur, incluant des consignes et contextes spécifiques,

- Le contexte supplémentaire fourni aux modèles pour enrichir leurs réponses,
- Les réponses générées par nos modèles (GPT-3.5 Turbo et LLaMA 3),
- La sortie attendue, servant de référence pour comparer et évaluer la performance des modèles.

Ces jeux de données nous ont permis de mesurer l'efficacité des modèles dans un cadre réaliste, tout en tenant compte des exigences de notre système.

Dataset Elect_Eval / GPT 3 Turbo et Llama 3 8B – Instruct

Ce *dataset* a été constitué à partir d'un échantillon de 7 copies d'une interrogation d'électricité générale, comprenant 70 questions au total, soit 10 questions par copie. L'évaluation a été réalisée dans le cadre d'un cours de deuxième année de licence en génie civil à l'ULPGL Goma ; ces *datasets* sont accessibles sur *Git Hub*⁵.

Comme mentionné précédemment, ce dataset est structuré en quatre colonnes : "**answer**", "**question**", "**ground_truth**", et "**context**". Chaque colonne contient 7 entrées correspondant aux copies corrigées. Voici le détail de chaque colonne :

- "**answer**" : Contient les corrections générées par le modèle sous format JSON. Ce format inclut un commentaire global, une note totale, ainsi qu'un commentaire spécifique et une note pour chaque question.
- "**question**" : Contient un copier de la composition fournis par l'enseignant et des réponses données par les étudiants sous forme de texte pour chaque entrée.
- "**context**" : Fournit des informations contextuelles pour chaque entrée, sous forme de texte, permettant au modèle d'avoir un cadre de référence.

⁵ https://github.com/Ashuza11/Elect_Eval-Dataset

- "ground_truth" : Contient les corrections manuelles réalisées par l'enseignant, également au format JSON, structuré de la même manière que les réponses générées par le modèle (avec des commentaires et des notes pour chaque question).

Comme les copies d'examen étaient manuscrites, nous les avons transcrites en format numérique pour pouvoir les traiter. Les copies originales n'incluaient pas de feedback, c'est pourquoi nous avons utilisé **GPT-4o** (GPT-4o, le dernier modèle de langage d'OpenAI, qui se démarque par ses performances de pointe sur de nombreux *benchmarks* standardisés. Sur le *benchmark MMLU* évaluant l'acquisition de connaissances des LLMs en configuration *zéro-shot* et *few-shot*, GPT-4o atteint un score de 88,7% en mode 5-shot, surpassant largement les autres modèles [72] [73].) pour générer les feedbacks. Ces feedbacks sont basés sur les réponses données, la note globale et les notes individuelles par question, et ont été formatés en format JSON (JavaScript Object Notation) de manière similaire à ceux générés par nos deux modèles.

Voici les prompts utilisés pour générer ces feedbacks :

Text

```
Tu es un enseignant bienveillant qui évalue les réponses d'un étudiant.  
Ton rôle est d'aider l'étudiant à comprendre ses erreurs et à s'améliorer.  
Voici la consigne:  
Commencerez par donner un avis général sur sa performance. Puis passer en
```

JSON

```
Voici le format attendu
{
  "advice": "Exemple de commentaire global ici.",
  "grading": {
    "1": {
      "note": 2,
      "commentaires": "Bonne réponse, mais peut être améliorée en ajoutant plus de détails."
    },
    "2": {
      "note": 1.5,
      "commentaires": "Réponse partiellement correcte, mais manque de précision."
    },
    "3": {
      "note": 1,
      "commentaires": "Bonne compréhension du sujet, mais quelques erreurs mineures."
    }
  }
}
```

Figure 3.29 Prompt utilisateur utilisée pour la génération de feedback pour l'évaluation.

3.5.2 Tests

Nous avons mené plusieurs tests, à la fois sur les deux modèles et sur notre système, avec pour objectif de vérifier le bon fonctionnement global et de démontrer les performances des modèles pour notre tâche d'assistance à la correction d'épreuves théoriques et à la génération de feedbacks constructifs. Nous avons effectué deux types de tests :

- **Tests fonctionnels** : Vérification du bon fonctionnement de chaque composant du système (API REST, base de données, interface utilisateur), à la fois de manière isolée et en intégration, pour assurer une cohésion globale.
- **Tests de performance des LLMs** : Évaluation comparative des performances des deux modèles (GPT-3.5 Turbo et LLaMA 3 8B) dans la correction d'exams et la génération de feedbacks, en utilisant nos deux *datasets* mentionnés précédemment ainsi que l'outil Azure AI Studio.

Les tests ont été réalisés en se basant sur des métriques de performance telles que **le Score F1, la cohérence, la fluidité, la similarité, l'enracinement, et la pertinence**, fournies par l'outil Azure AI Studio.

Tests sur la performance des LLMs

Comme mentionné précédemment, nous avons effectué deux types de tests pour évaluer les performances de nos modèles : des tests manuels (où nous avons jugé nous-mêmes les sorties des modèles) et des tests automatisés (avec l'assistance de l'IA pour évaluer les résultats). Pour ces évaluations, nous avons utilisé **Azure AI Studio** pour déployer et tester nos deux modèles, ainsi que le modèle **GPT-4o** qui nous a assisté dans l'évaluation de leurs performances.

Azure AI Studio est une plateforme clé pour évaluer les applications d'IA générative. Elle propose des méthodes d'évaluation manuelle et automatisée, permettant de tester la performance des modèles en mesurant des métriques comme la précision, le rappel ou le F1-

Score. De plus, elle aide à identifier les risques associés à l'utilisation de l'IA dans divers contextes, offrant un cadre structuré pour analyser la fiabilité et l'efficacité des modèles déployés.

D'une part, l'évaluation manuelle permet aux experts de passer en revue les sorties générées par les modèles sur des petits ensembles de données. Cela est particulièrement utile pour identifier et corriger les erreurs avant d'étendre les tests à une plus grande échelle.

D'autre part, Azure AI Studio propose des outils d'évaluation automatisée qui mesurent la qualité des modèles avec des métriques comme **la précision** et **le F1-Score**. Ces métriques sont utiles dans des contextes où il existe des réponses prédéfinies, comme c'est le cas pour notre tâche. Dans des scénarios plus ouverts, où il n'existe pas de réponse unique correcte, les évaluations assistées par IA permettent d'annoter les réponses générées en fonction de critères comme **la pertinence**. Azure AI Studio permet également de suivre en continu les performances des modèles, ce qui est crucial pour détecter d'éventuelles régressions et assurer que les corrections et feedbacks restent de haute qualité au fil du temps [75].

3.5.3 Résultats des tests

Les résultats des tests effectués sont détaillés ci-dessous, avec une analyse des différentes métriques :

- **Résultats des performances des LLMs** : Les résultats présentés ci-dessous ont été obtenus en utilisant le dataset **Elect_eval** mentionné à la section **3.5.1**.

Test manuel : Nous avons inscrit 7 étudiants et 1 enseignant, créé un cours, composé une évaluation, puis partagé le code du cours avec les étudiants pour qu'ils soumettent leurs réponses. Les données issues de ce test ont été utilisées pour évaluer la performance sur **Azure AI Studio**. Voici les résultats obtenus :

Tableau 12 Résultat du test manuel ici de système

Index	Note Enseignant /20	Note GPT 3.5 Turbo /20	Note Llama 3 8b – Instruct /20
1	4,5	5	5
2	15,25	12,5	7,5
3	11	11	11,5
4	13,5	9	11
5	12	12	15
6	12,5	13,5	4
7	9,5	5,5	4,5

Il est à noter que les notes dans la colonne 'Not Enseignant' sont celles attribuées par l'enseignant pour les 7 copies de cette épreuve.

Pour obtenir ce résultat, nous avons défini les paramètres suivants et un même prompt (présenté dans les annexes) sur nos deux modèles :

Tableau 13 Configuration des paramètres de model GPT3.5 Turbo et Llama 3 8B -instruct

Temperature	Max tokens	Top_p	Frequency_penalty	Presence_penalty
0,45	3000	0,8	0	0

Les paramètres présentés dans le **Tableau 13** concernent les réglages de l'API pour nos deux modèles. Une **température** de **0,45** a été choisie pour garantir des réponses plus cohérentes et moins aléatoires, essentielle pour des tâches nécessitant une grande précision, comme ce le cas pour notre tâche. Le paramètre **max_tokens** est fixé à **3000** pour permettre des réponses suffisamment longues et détaillées, tout en évitant les coupures prématurées. Le paramètre **top_p** est défini à **0,8**, un choix modéré qui équilibre diversité et qualité des réponses, favorisant des réponses créatives tout en évitant les informations non pertinentes. Le **frequency_penalty** est réglé à **0** pour ne pas limiter la répétition des termes, ce qui est pertinent pour garantir une fluidité maximale du texte. Enfin, **l'absence de pénalité** de présence

(presence_penalty à 0) permet la réitération de concepts si nécessaire, assurant ainsi la clarté et la couverture complète des sujets.

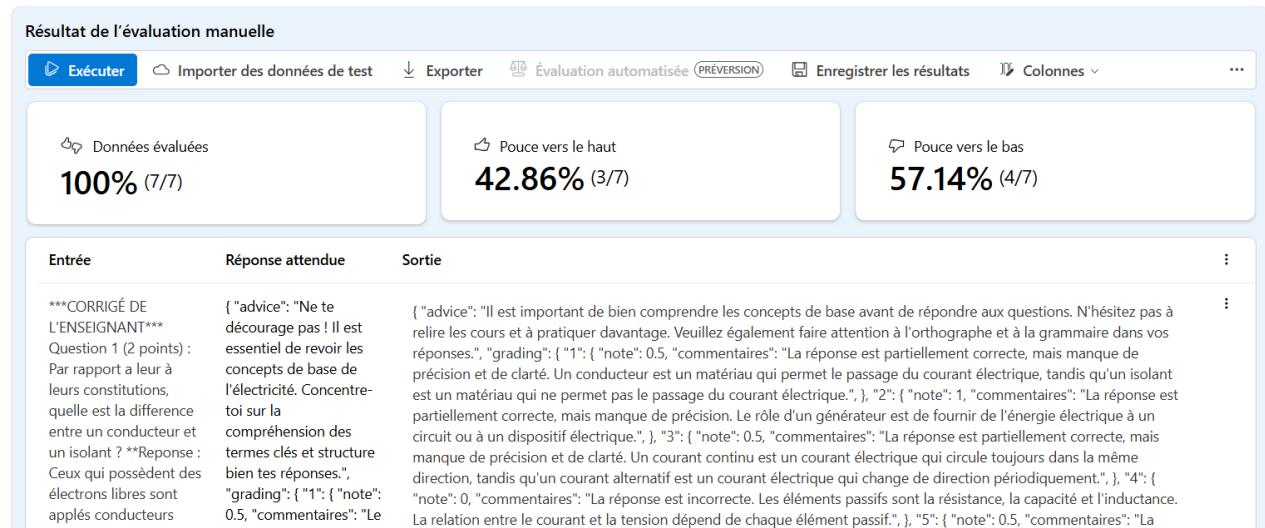


Figure 3.30 Résultat du test manuel sur la plateforme Azure AI studio

1. Test automatisé :

Résultats des tests sur le modèle GPT-3.5 Turbo : métriques de performance et de qualité.

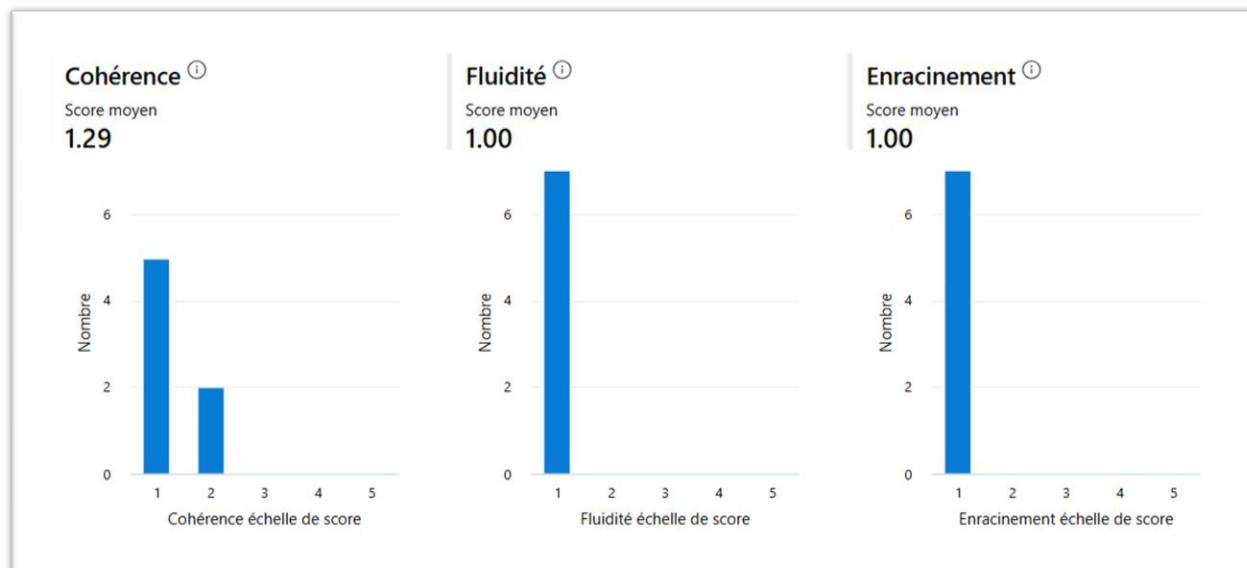


Figure 3.31 Résultat de test du modèle GPT 3.5 Turbo (cohérence, fluidité, enracinement)



Figure 3.32 Résultat de test du modèle GPT 3.5 Turbo (pertinence, similarité)

Tableau 14 Résultat détaillé des métriques GPT3.5 Turbo

Index	Score F1	Cohérence	Fluidité	Similarité	Enracinement	Pertinence
1	0,49	1	1	1	1	1
2	0,47	1	1	3	1	3
3	0,51	1	1	2	1	3
4	0,48	2	1	2	1	4
5	0,47	1	1	1	1	4
6	0,47	1	1	3	1	3
7	0,49	2	1	1	1	4

Le tableau **Tableau 14** présente les résultats des tests réalisés avec l'assistance du modèle **GPT-4o**, portant sur le modèle GPT-3.5 Turbo. Ce tableau détaille les données illustrées dans les **Figure 3.32** et **Figure 3.33**, qui exposent respectivement les métriques de performance et de qualité du modèle testé.

Résultats des tests sur le modèle Llama 3 8B-Instruct : métriques de performance et qualité.

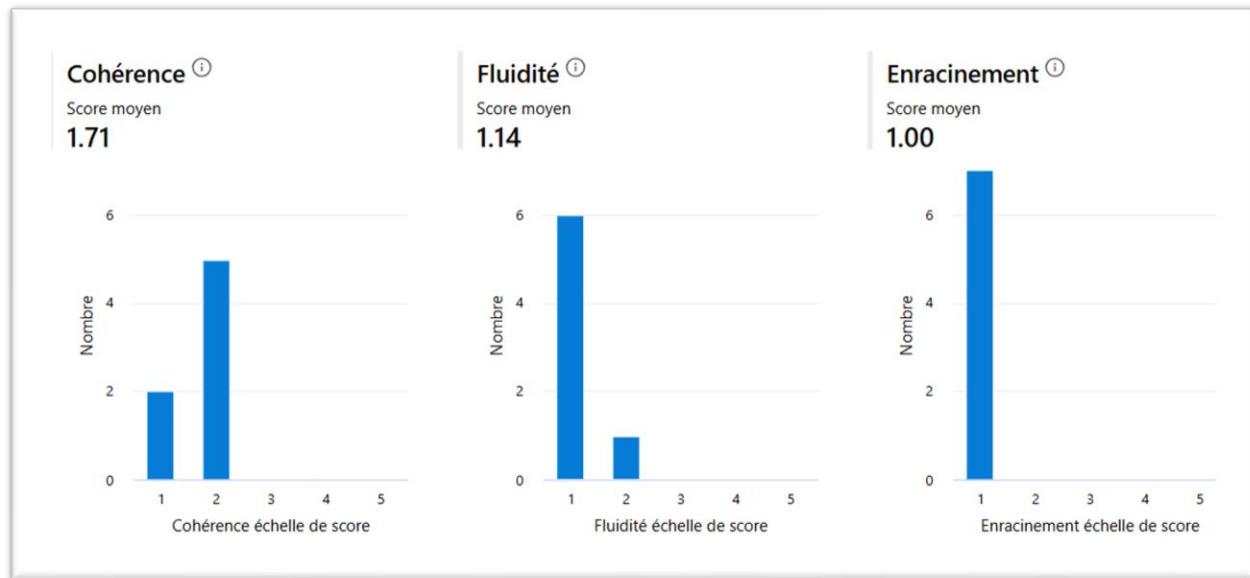


Figure 3.33 Résultat de test du modèle Llama 3 8B-Instruct (cohérence, fluidité, enracinement)

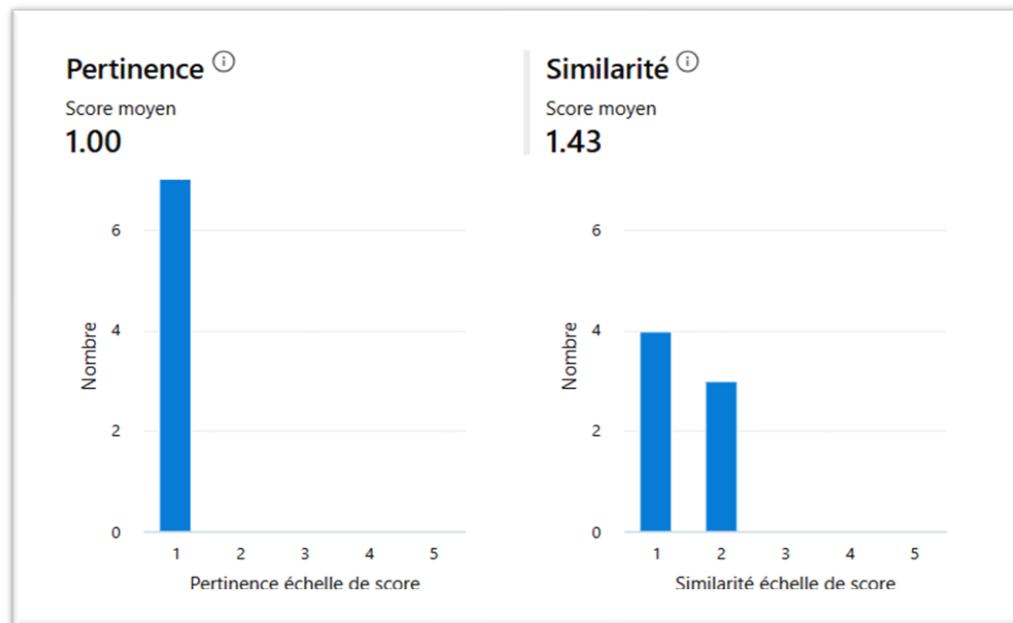


Figure 3.34 Résultat de test du modèle Llama 3 8B-Instruct (pertinence, similarité)

Tableau 15 Résultat détaillé des métriques Llama 3 8B – Instruct

Index	Score F1	Cohérence	Fluidité	Similarité	Enracinement	Pertinence
1	0,49	1	1	1	1	1
2	0,44	2	2	1	1	1
3	0,54	1	1	1	1	2
4	0,49	2	1	1	1	2
5	0,42	2	1	1	1	2
6	0,44	2	1	1	1	1
7	0,49	1	1	1	1	1

Le tableau **Tableau 15** présente les résultats des tests réalisés avec l'assistance du modèle **GPT-4o**, portant sur le modèle **Llama 3 8B -Instruct**. Ce tableau détaille les données illustrées dans les figures 23 et 32, qui exposent respectivement les métriques de performance et de qualité du modèle testé.

Résultats des tests sur le modèle GPT-3.5 Turbo : métriques de performance et qualité après ajustement des paramètres et amélioration du prompt.

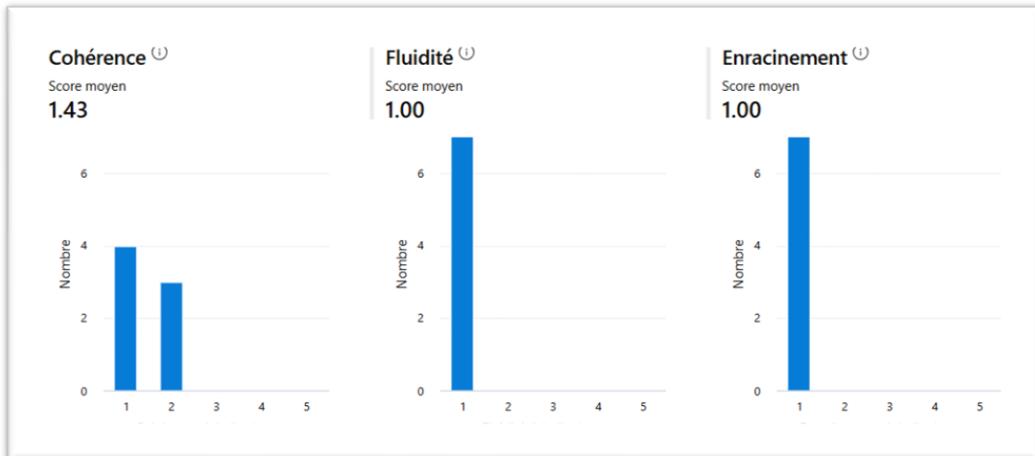


Figure 3.35 Résultat de test du modèle GPT 3.5 Turbo après ajustement et amélioration
(cohérence, fluidité, enracinement)

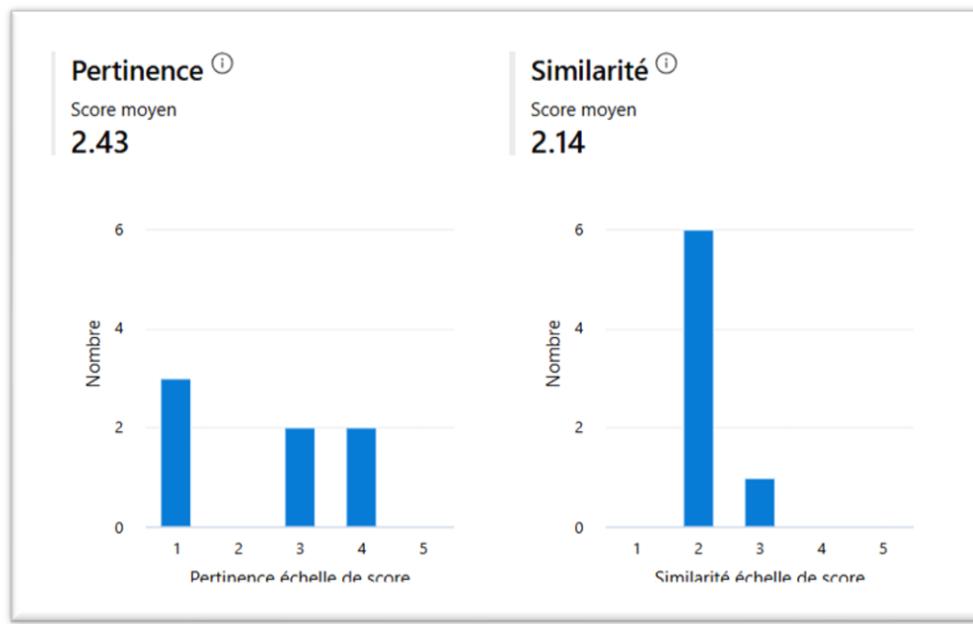


Figure 3.36 Résultat de test du modèle GPT 3.5 Turbo après ajustement et amélioration
(pertinence, similarité)

Tableau 16 Résultats détaillés des métriques de GPT-3.5 Turbo après ajustements et améliorations

Index	Score F1	Cohérence	Fluidité	Similarité	Enracinement	Pertinence
1	0,53	1	1	2	1	2
2	0,49	1	1	2	1	2
3	0,51	1	1	2	1	2
4	0,41	2	1	3	1	4
5	0,47	2	1	2	1	2
6	0,47	2	1	2	1	2
7	0,49	1	1	2	1	2

Le tableau **Tableau 14** présente les résultats des tests réalisés avec l'assistance du modèle **GPT-4o** sur le modèle GPT-3.5 Turbo après ajustement et amélioration. Ce tableau détaille les

données illustrées dans les **Figure 3.35** et **Figure 3.36**, qui exposent respectivement les métriques de performance et de qualité du modèle testé. Pour obtenir ce résultat, nous avons ajusté les paramètres suivants ainsi que le prompt (présenté ci-dessous dans les annexes) sur le modèle GPT-3.5 Turbo.

Tableau 17 Configuration des paramètres de model GPT3.5 Turbo

Temperature	Max tokens	Top_p	Frequency_penalty	Presence_penalty
0,30	3000	0,7	0	0

Voir l'interprétation des différentes métriques ci-dessus au **Tableau 13**.

3.6 Interprétation des résultats

1. Test manuel

Comme on peut le voir dans les tableaux **Tableau 12**, IGPT-3.5 Turbo semble être plus aligné avec les évaluations humaines, tandis que Llama 3 8b - Instruct montre des divergences notables, ce qui suggère des différences dans la façon dont ces modèles interprètent et attribuent les notes.

La **Figure 3.30** présente les résultats du test manuel réalisé sur Azure AI Studio. Ce test révèle que **GPT-3.5 Turbo** fonctionne relativement bien pour certaines questions, mais qu'il nécessite des ajustements pour offrir des réponses plus précises et complètes. Cela est particulièrement important dans des contextes éducatifs où des explications détaillées sont requises. Pour améliorer la qualité des réponses, il serait pertinent d'ajuster le prompt, de fournir des exemples plus précis, ou de réviser les critères d'évaluation.

2. Test automatisé :

Model GPT 3.5 Turbo

Les résultats, illustrés par les **Résultats des tests sur le modèle GPT-3.5 Turbo : métriques de performance et de qualité**.

Figure 3.31 Résultat de test du modèle GPT 3.5 Turbo (cohérence, fluidité, enracinement)

et **Figure 3.32**, ainsi que détaillés dans le **Tableau 14**, sont très similaires à ceux obtenus lors de l'évaluation manuelle. De plus, l'introduction de plusieurs métriques permet d'analyser la performance du modèle sous différents angles pour chaque instance de notre petit jeu de données.

Interprétation des métriques :

1. Score F1 :

- Le score F1 est une mesure qui combine la précision et le rappel d'un modèle. Il s'agit de la moyenne harmonique de ces deux valeurs. Il est particulièrement utile lorsqu'il y a un déséquilibre entre les classes positives et négatives [76].
- Un score F1 proche de 1 indique une bonne performance du modèle, tandis qu'un score plus proche de 0 montre des lacunes. Dans le **Tableau 14**, les scores varient de 0,47 à 0,51, ce qui montre une performance modérée du modèle.

2. Cohérence :

- La cohérence évalue si les réponses générées par le modèle sont logiques et se tiennent bien dans leur structure.
- Un score de 1 signifie que la réponse manque de cohérence, tandis qu'un score de 5 indiquerait une très bonne cohérence.

3. Fluidité :

- La fluidité mesure la manière dont les réponses sont formulées. Une réponse fluide est facile à lire et suit les règles grammaticales.

- Dans le **Tableau 14**, toutes les réponses ont obtenu un score de 1, indiquant que la fluidité des réponses doit être améliorée.

5. Similarité :

- La similarité évalue dans quelle mesure la réponse générée par le modèle correspond à la réponse attendue.
- Un score de 1 ici signifie que les réponses du modèle sont très différentes des réponses attendues.

6. Enracinement :

- L'enracinement mesure la connexion entre la réponse et le sujet ou le contexte général. Une bonne réponse est bien ancrée dans le sujet discuté.
- Les scores varient entre 1 et 4, ce qui montre une variation dans la profondeur de la connexion au sujet.

7. Pertinence :

- La pertinence mesure dans quelle mesure la réponse est appropriée et répond directement à la question posée.
- Les scores varient entre 1 et 3, ce qui suggère que la pertinence des réponses est parfois en dessous des attentes.

Chaque métrique est évaluée sur une échelle de 1 à 5, avec **5 étant une très bonne valeur** et **1 une mauvaise valeur**, à l'exception du **score F1**, qui a une signification légèrement différente en tant que mesure de la performance globale du modèle. Les résultats dans ce tableau indiquent que les performances du modèle sont globalement modérées, avec des points d'amélioration clairs en fluidité, similarité, et parfois en pertinence. Pour les résultats obtenus après ajustement des paramètres et du prompt, on remarque une amélioration globale en pertinence, similarité, cohérence, et en Score F1, ce qui démontre que ce modèle est plus adapté à cette tâche

comparativement au modèle Llama 3 8B - Instruct. Cependant, plusieurs ajustements et améliorations du prompt et des paramètres sont encore nécessaires, ou bien l'application d'autres techniques mentionnées ci-dessus dans **la section 3.4.3**, afin d'obtenir de très bons résultats.

Model Llama 3 8B -Instruct

Comme indiqué dans le **Tableau 14** pour le test manuel, ce modèle montre une légère baisse des performances globales, en particulier au niveau du Score F1, qui oscille entre 0,42 et 0,54, des valeurs inférieures à celles du modèle **GPT-3 Turbo**. Bien que certaines métriques, comme la fluidité dans certains cas, aient légèrement progressé, le modèle continue de présenter des faiblesses notables en termes de pertinence et de similarité, ce qui affecte la qualité globale des réponses.

3.7 Justification des choix d'implémentation

Voici donc quelques justifications très sommaires des choix d'implémentation que nous avons faits. Les points saillants sont principalement ici présentés :

- Comme mentionné précédemment dans la section 34 sur l'implémentation de l'API, nous avons choisi d'utiliser **FastAPI** en raison de la grande liberté qu'il offre aux développeurs. Il ne suit pas une structure rigide tout en restant robuste, simple et très populaire dans l'écosystème du développement d'applications basées sur l'IA.
- Pour la partie client, l'utilisation de **ReactJS** s'explique non seulement par sa popularité, mais aussi par notre familiarité avec sa syntaxe et son architecture. De même, nous avons opté pour le framework **Tailwind** pour la stylisation de nos interfaces, en raison de son efficacité et de sa flexibilité.

- Pour le modèle LLM, nous avons choisi deux modèles : GPT-3.5 Turbo, un modèle propriétaire accessible via OpenAI, et LLaMA 3 8B - Instruct, un modèle open source. Nous avons sélectionné ces deux modèles en raison de leur coût, accessibilité, et performance, permettant ainsi de comparer les avantages des approches propriétaires et open source pour notre cas d'utilisation.
- Concernant la méthode d'évaluation de nos modèles, nous avons opté pour la plateforme Azure AI Studio. Celle-ci offre une gamme complète d'outils, notamment pour le déploiement, les tests, et la surveillance à grande échelle, ce qui nous permettrait d'évaluer les performances et d'assurer la fiabilité des modèles en production.

3.8 Conclusion partielle

Dans ce chapitre, nous avons finalisé la conception détaillée de notre système en couvrant tous ses aspects essentiels : la base de données, l'API et les interfaces utilisateur. Cette structure repose sur une architecture à trois niveaux, garantissant une séparation claire entre la logique métier, la gestion des données et l'interface utilisateur, assurant ainsi une meilleure évolutivité et maintenabilité du système.

Les tests effectués ont permis d'évaluer les performances de notre système, notamment à travers des données réelles. En utilisant Azure AI Studio, nous avons réalisé des évaluations manuelles et automatisées, en analysant les résultats avec des métriques telles que la précision et le F1-Score. Les résultats révèlent que GPT-3.5 Turbo s'aligne mieux avec les évaluations humaines, bien qu'il nécessite des ajustements pour améliorer la précision et la complétude de ses réponses. À l'inverse, LLaMA3 8B-Instruct affiche des performances inférieures, particulièrement en termes de pertinence et de similarité, ce qui indique que des ajustements sont nécessaires pour les deux modèles afin d'optimiser leur performance pour notre tâche.

Conclusion générale

Ce mémoire a exploré la conception et la mise en œuvre d'un système automatisé de correction d'épreuves théoriques, **Bic Rouge**, reposant sur des modèles de langage de pointe comme GPT-3.5 Turbo et LLAMA-3. L'objectif principal était de tester l'hypothèse selon laquelle l'intégration de modèles de langage (LLM) dans un système de correction automatisée pourrait améliorer l'efficacité, la précision et la cohérence du processus d'évaluation.

Au début de ce travail, nous avons émis plusieurs hypothèses :

1. **Amélioration de l'efficacité, précision et cohérence** : Nous avons postulé qu'un système d'aide à la correction basé sur les LLM améliorerait ces aspects. Nos résultats ont en grande partie confirmé cette hypothèse, en particulier pour GPT-3.5 Turbo, qui a démontré un meilleur alignement avec les évaluations humaines que le modèle LLAMA-3. Cependant, bien que GPT-3.5 Turbo soit plus performant, certaines limitations en matière de cohérence et de nuance ont été observées, nécessitant des ajustements supplémentaires.
2. **Amélioration de l'expérience des enseignants** : Nous avions également hypothéqué que l'intégration des LLMs dans un outil convivial permettrait aux enseignants d'améliorer l'expérience de correction. Cette hypothèse a été confirmée dans une certaine mesure, avec la capacité du système à générer des retours positifs pour les étudiants et la possibilité pour les enseignants de modifier et valider les corrections effectuées par les LLMs. Toutefois, des ajustements du prompt et des paramètres sont encore nécessaires pour garantir une précision et une pertinence optimales.
3. **Utilisation des métriques d'évaluation** : Nous avons supposé que l'utilisation de métriques d'évaluation appropriées permettrait de mieux mesurer l'impact du système sur la précision et la cohérence des évaluations. Cette hypothèse s'est avérée fondée, les métriques comme le F1-Score et la précision nous ayant permis d'identifier des écarts de performance entre les modèles et d'évaluer objectivement leur fiabilité.

Les contributions majeures de ce travail incluent l'identification des défis liés à l'évaluation automatique, la comparaison des performances des LLMs pour la correction d'épreuves

théoriques, ainsi que le développement d'une architecture modulaire capable de s'adapter à différentes tâches d'évaluation.

Néanmoins, certaines hypothèses n'ont pas été pleinement confirmées. Par exemple, bien que le système ait montré des améliorations notables, il reste des lacunes dans la gestion de réponses complexes et dans l'interprétation des subtilités des réponses des étudiants, particulièrement avec LLAMA-3. Cela indique qu'une adaptation plus poussée, comme le **fine-tuning** ou l'usage de techniques telles que le **RAG**, est nécessaire pour obtenir des résultats plus robustes.

Pour les perspectives, des axes de recherche supplémentaires pourraient inclure :

- L'affinement des prompts afin d'améliorer la précision des évaluations générées.
- L'utilisation de techniques telles que du **RAG** pour renforcer les capacités du système dans la correction des épreuves.
- Le **fine-tuning** des LLMs afin de les adapter spécifiquement à la tâche de correction et à la génération de feedback personnalisé.
- L'intégration de nouveaux LLMs plus performants pour mieux capter la subjectivité des réponses.
- L'intégration de modèles de vision pour faciliter la correction dans des contextes éducatifs où les enseignants n'ont pas accès à des ordinateurs.
- Le développement d'une application mobile pour améliorer l'accessibilité du système dans les écoles, en particulier dans les régions moins équipées en infrastructure numérique.

Ces extensions visent à rendre le système plus flexible, accessible et performant dans divers environnements éducatifs.

En somme, ce travail ouvre des perspectives prometteuses pour l'automatisation de l'évaluation académique tout en mettant en lumière les défis techniques et éthiques qui doivent être surmontés pour atteindre une solution robuste et généralisable.

Bibliographie

- [1] V. S. T. a. M. Murgia, "Generative AI exists because of the transformer," FINANCIAL TIMES , 12 SEPTEMBER 2023. [Online]. Available: <https://ig.ft.com/generative-ai/>. [Accessed 28 Avril 2024].
- [2] N. S. N. P. J. U. e. a. Ashish Vaswani, "Attention Is All You Need," p. 15, 2017.
- [3] F. H. W. Miao, Guidance for generative AI in education and research, Paris : UNESCO, 2024.
- [4] J. W. R. C. D. L. a. a. Alec Radford, "Language Models are Unsupervised Multitask Learners," OpenAI, 2019.
- [5] S. K. S. Dadi Ramesh, "An automated essay scoring systems: a systematic literature," 23 September 2021. [Online]. Available: <https://doi.org/10.1007/s10462-021-10068-2>. [Accessed 28 Avril 2024].
- [6] Y. K. G. G. L. Heejin Do, "Autoregressive Score Generation for Multi-trait Essay Scoring," no. arXiv:2403.08332v1, p. 8, 2024.
- [7] M. C. K. M. e. a. Ig Ibért Bittencourt, "Artificial Intelligence in Education," in *21st International Conference, AIED 2020*, Morocco, 2020.
- [8] M. Uto, "Automated Assessment of Student Writing with Transformers," *Behaviormetrika*, p. 48, 2021.
- [9] A. Cader, "The Potential for the Use of Deep Neural Networks in e-Learning Student Evaluation with New Data Augmentation Method," in *Artificial Intelligence in Education*, Morocco, Springer, 2020, p. 458.
- [10] A. Condor, "Exploring Automatic Short Answer Grading as a Tool to Assist in Human Rating," in *Artificial Intelligence in Education*, Morocco, Springer , 2020, p. 458.
- [11] A. Ghaicha, "Theoretical Framework for Educational Assessment: A Synoptic," *Journal of Education and Practice*, vol. 7, p. 20, 2016.

- [12] N. D. e. D. O. Ravachol, "L'évaluation et l'enseignement des sciences et des technologies," *Open Edition Journals*, p. 1, 2023.
- [13] V. G. Thomas Kellaghan, "L'évaluation pour améliorer la qualité de l'enseignement Paris 2002," *UNESCO : Institut international de planification de l'éducation*, vol. 7, p. 119, 2002.
- [14] M. B. C. e. a. M. Kourakoro Bagayoko, Évaluation des apprentissages scolaires, Bamako: UNESCO Développement des capacités pour l'éducation pour tous /CapEFA, 2015.
- [15] B. M. N. R. M. S. e. a. Tom B. Brown, "Language Models are Few-Shot Learners," *arxiv*, vol. 4, p. 75, 2020.
- [16] J. D. M.-W. C. K. L. K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," in *Proceedings of NAACL-HLT 2019*, 2019.
- [17] J. H. M. Daniel Jurafsky, "Speech and Language Processing (3rd ed. draft)," Stanford University, 3 Feb 2024 . [Online]. Available: <https://web.stanford.edu/~jurafsky/slp3/>. [Accessed 31 5 2024].
- [18] J. Alammar, "The Illustrated Transformer," 15 11 2020. [Online]. Available: <https://jalammar.github.io/illustrated-transformer/>. [Accessed 02 06 2024].
- [19] F. CHOLLET, Deep Learning with Python, Shelter Island: Manning Publications Co, 2021.
- [20] T. L. G. I. e. a. Hugo Touvron, "LLaMA: Open and Efficient Foundation Language Models," *arxiv*, vol. 1, p. 27, 2023.
- [21] D. A. H. E. A. e. a. Rishi Bommasani, "On the Opportunities and Risks of Foundation Models," p. 214, 2022.
- [22] N. S. A. R. e. a. Colin Raffel, "Exploring the Limits of Transfer Learning with a Unified," *Journal of Machine Learning Research*, p. 67, 2020.

- [23] S. R. Jeremy Howard, "Universal Language Model Fine-tuning for Text Classification," *Association for Computational Linguistics*, vol. Volume 1, p. 12, 2018.
- [24] P. R. H. S. Christopher D. Manning, "Introduction to Information Retrieval," Cambridge University Press, 04 07 2009. [Online]. Available: <https://nlp.stanford.edu/IR-book/html/htmledition/irbook.html>. [Accessed 06 04 2024].
- [25] P. Koehn, Statistical Machine Translation, New York: Cambridge University Press, 2012.
- [26] T. D. Albert Gu, "Mamba: Linear-Time Sequence Modeling with Selective State Spaces," 2024.
- [27] .. Z. Y. Y. Y. a. a. Zihang Dai, "Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context," 2019.
- [28] OpenAI, "GPT-4 Technical Report," 2024.
- [29] K. K. David, "Conception et realisation d'un systeme base sur l'intelligence artificielle pour resumer automatiquement les textes," p. 142, 2022.
- [30] "Introducing Claude," Anthropic, 14 03 2023. [Online]. Available: <https://www.anthropic.com/news/introducing-claude>. [Accessed 04 06 2024].
- [31] N. N. L. S. a. a. John Aziz, "Generative AI for Beginners (Version 2) - A Course," Microsoft , 2024. [Online]. Available: <https://github.com/microsoft/generative-ai-for-beginners/tree/main?tab=readme-ov-file>. [Accessed 05 06 2024].
- [32] H. Z. A. M. M. A. Alexei Baevski, "wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations," 2020.
- [33] L. L. A. P. a. a. Raphael Tang, "What the DAAM: Interpreting Stable Diffusion Using Cross Attention," p. 13, 2022.
- [34] Stability.AI, "Stable Diffusion 3: Research Paper," Stability.AI, 5 03 2024. [Online]. Available: <https://stability.ai/news/stable-diffusion-3-research-paper>. [Accessed 14 06 2024].

- [35] A. F. C. A. a. a. Teven Le Scao, "BLOOM: A 176B-Parameter Open-Access Multilingual," 2023.
- [36] N. S. A. R. a. a. Colin Raffel, "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer," *Journal of Machine Learning Research*, p. 67, 2023.
- [37] L. B. A. Y. Z. e. a. Raymond Li, "StarCoder: may the source be with you!," *Transactions on Machine Learning Research*, p. 55, 2023.
- [38] J. G. F. G. e. a. Baptiste Rozière, "Code Llama: Open Foundation Models for Code," Meta AI, 2024.
- [39] J. W. K. C. H. a. a. Alec Radford, "Learning Transferable Visual Models From Natural Language Supervision," p. 48, 2021.
- [40] Google Research, "Tacotron 2: Generating Human-like Speech from Text," Google, 19 12 2017. [Online]. Available: <https://research.google/blog/tacotron-2-generating-human-like-speech-from-text/>. [Accessed 14 06 2024].
- [41] ElevenLabs, "Online AI Text to Speech for Free," ElevenLabs, [Online]. Available: <https://elevenlabs.io/>. [Accessed 14 06 2024].
- [42] D. D. F. J. H. a. a. Romal Thoppilan, "LaMDA: Language Models for Dialog Applications," 2022.
- [43] I. G. a. a. Rohan Taori, "Alpaca: A Strong, Replicable Instruction-Following Model," Stanford Center for Research on Foundation Models, 2021. [Online]. Available: <https://crfm.stanford.edu/2023/03/13/alpaca.html>. [Accessed 05 06 2024].
- [44] Microsoft, "Phi-3 Technical Report A Highly Capable Language Model Locally on Your Phone," 2024.
- [45] J. S. Tim Pearce, "Reconciling Kaplan and Chinchilla Scaling Laws," *arXiv*, vol. 1, p. 8, 2024.
- [46] T. X. H. L. a. a. Shen Wang, "Large Language Models for Education: A Survey and Outlook," *arXiv*, vol. 2, p. 14, 2024.

- [47] OpenAI, "Introducing ChatGPT Edu," OpenAI, 30 05 2024. [Online]. Available: <https://openai.com/index/introducing-chatgpt-edu/>. [Accessed 05 06 2024].
- [48] M. Goyal, "LLM Use-Cases: Top 10 Industries Using Large Language Models," Bot Penguin, [Online]. Available: <https://botpenguin.com/blogs/llm-use-cases>. [Accessed 05 06 2024].
- [49] B. R. a. a. Zhenqin Wu, "MoleculeNet: A Benchmark for Molecular Machine Learning," p. 65, 2018.
- [50] Google-DeepMind, "AlphaFold is accelerating research in nearly every field of biology.,," Google-DeepMind, [Online]. Available: <https://deepmind.google/technologies/alphafold/>. [Accessed 05 06 2024].
- [51] A. A. S. M. Ummara Mumtaz, "Artificial Intelligence in Health," *AccScience*, vol. 1, no. 2, p. 13, 2024.
- [52] i. X. S. L. a. a. Xiaolan Chen, "Evaluating large language models in medical applications: a survey," p. 42.
- [53] S. N. D. K. a. Tareq Rasul, "The role of ChatGPT in higher education: Benefits, challenges, and future research directions," *Journal of Applied Learning & Teaching*, p. 16, 2023.
- [54] A. G. F. S. Han Liu, "Categorization and Construction of Rule Based Systems," in *15th International Conference on Engineering Applications of Neural Networks*, Sofia, Bulgaria, 2014.
- [55] S. M. W. e. a. Hwee Tou Ng, "The CoNLL-2014 Shared Task on Grammatical Error Correction," in *Eighteenth Conference on Computational Natural Language Learning: Shared Task*, Baltimore, Maryland, 2014.
- [56] C. Jacquet-Pfau, "Correcteurs orthographiques et grammaticaux," *Revue française de linguistique appliquée*, vol. VI, no. 2, pp. 81-94, 2001.
- [57] J. H. M. Dan Jurafsky, "Speech and Language Processing (3rd ed. draft)," Stanford University, 3 Feb 2024. [Online]. Available: <https://web.stanford.edu/~jurafsky/slp3/>. [Accessed 15 June 2024].

- [58] D. A. E. & R. P. Pérez-Marín, "Automatic Assessment of Open Ended Questions with a Bleu-Inspired Algorithm and Shallow NLP," in *4th International Conference*, Alicante, 2004.
- [59] pearson, "ExamWeezaard," pearson, [Online]. Available: <https://qualifications.pearson.com/en/support/Services/examwizard>. [Accessed 29 June 2024].
- [60] Turnitin, "Gradescope," Turnitin, [Online]. Available: <https://www.turnitin.com/products/gradescope/>. [Accessed 29 06 2024].
- [61] Grammarly, " Acceil," Grammarly, [Online]. Available: <https://www.grammarly.com/>. [Accessed 29 June 2024].
- [62] S. MOR, "Top 8 leaderboards to choose the right AI model for your task," AI Tidbits, 29 Jun 2024. [Online]. Available: <https://www.aitidbits.ai/p/leaderboards-for-choosing-best-model>. [Accessed 17 Feb 2024].
- [63] X. C. N. X. e. a. Junjie Ye, "AComprehensive Capability Analysis of GPT-3 and GPT-3.5 Series Models," *arxiv* , vol. 2, p. 47, 2023.
- [64] X. M. H. Q. e. a. Wei Huang, "How Good Are Low-bit Quantized LLaMA3 Models? An Empirical Study," *arxiv* l, vol. 1, p. 7, 2024.
- [65] X. M. H. Q. e. a. Wei Huang, "How Good Are Low-bit Quantized LLaMA3 Models? An Empirical Study," *arxiv* , vol. 1, p. 7, 2024.
- [66] AI, Vellum, "GPT-3.5 Turbo vs Llama 3.1 8b," Vellum AI, 2024. [Online]. Available: <https://www.vellum.ai/comparison/gpt-3-5-turbo-vs-llama-3-1-8b>. [Accessed 01 09 2024].
- [67] Artificial analysis , "Llama 3 Instruct 8B: Quality, Performance & Price Analysis," Artificial analysis , 2024. [Online]. Available: https://artificialanalysis.ai/models/llama-3-instruct-8b?models_selected=gpt-4o-2024-08-06%2Cgpt-4o-mini%2Cllama-3-1-instruct-405b%2Cllama-3-1-instruct-70b%2Cllama-3-1-instruct-8b%2Cgemini-1-5-pro%2Cgemini-1-5-flash%2Cclaude-35-sonnet%2Cclaude-3-haiku%2Cm. [Accessed 01 09 2024].

- [68] Open AI, "Pricing," Open AI, 2024. [Online]. Available: <https://openai.com/api/pricing/>. [Accessed 24 08 2024].
- [69] M. Carteau, Introduction aux bases de données, Paris: AgroParisTech - UFR d'informatique, 2014.
- [70] G. GOY, Conception des bases de données avec UML, Québec : Presses de l'université du Québec , 2009.
- [71] S. Mor, "Top 8 leaderboards to choose the right AI model for your task," AI Tidbits, 17 02 2024. [Online]. Available: <https://www.aitidbits.ai/p/leaderboards-for-choosing-best-model>. [Accessed 27 08 2024].
- [72] DataScientest, "FastAPI: Everything you need to know about the most widely used Python web framework for Machine Learning," DataScientest, 29 08 2023. [Online]. Available: <https://datascientest.com/en/fastapi-everything-you-need-to-know-about-the-most-widely-used-python-web-framework-for-machine-learning>. [Accessed 09 09 2024].
- [73] H. S. L. J. C. e. a. Ji Joong Moon, "A New Frontier of AI: On-Device AI Training and Personalization," *arXiv*, vol. 3, p. 11, 2024.
- [74] J. A. S. A. e. a. OpenAI, "GPT-4 Technical Report," *arXiv*, vol. 6, p. 100, 2024.
- [75] Open AI, "GPT-4o Evaluations A Comprehensive Analysis," Open AI, 2024. [Online]. Available: <https://gpt4mni.com/evaluations/>. [Accessed 13 09 2024].
- [76] R. M. E. U. Lauryn Gayhardt, "Évaluation des applications d'IA générative," Microsoft , 03 09 2024. [Online]. Available: <https://learn.microsoft.com/fr-fr/azure/ai-studio/concepts/evaluation-approach-gen-ai>. [Accessed 12 09 2024].
- [77] F. Chollet, Deep Learning with Python, Shelter Island: Manning Publications Co, 2021.
- [78] JUnit.org. [Online]. Available: <http://www.junit.org>. [Accessed 2 Janvier 2019].
- [79] W. M. S. X. X. K. Z. Y. W. Q. F. Changrong Xiao, "Improving Automated Essay Scoring Using Large Language Models," *cs.CL*, p. 14, 2024.

- [80] O. ‘. Buruk, "Academic Writing with GPT-3.5: Reflections on Practices, Efficacy, and Transparency," p. 93.
- [81] N. P. M. Garcia, "Comparative Analysis of Large Language Models for Text Correction Tasks".
- [82]
- [83] A. Ghaicha, "Theoretical Framework for Educational Assessment: A Synoptic," *Journal of Education and Practice*, vol. 7, p. 20, 2016.
- [84] P. T. a. a. Yanis Labrak, "Correction automatique d'examens écrits par approche neuronale profonde et attention croisée bidirectionnelle," *Open science* , vol. 7, p. 10, 2022.
- [85] Microsoft Azure , "Azure AI Studio: gpt-35-turbo-instruct," Microsoft , [Online]. Available: <https://ai.azure.com/explore/models/gpt-35-turbo-instruct/version/3/registry/azure-openai?tid=84c31ca0-ac3b-4eae-ad11-519d80233e6f>. [Accessed 29 06 2024].
- [86] Microsft Azure, "Azure AI Studio," Microsft , 2024. [Online]. Available: <https://ai.azure.com/explore/models/Meta-Llama-3-8B-Instruct/version/5/registry/azureml-meta?wsid=/subscriptions/4becce75-83d2-4dcc-9afb-1a41ed1af77a/resourceGroups/AzureAIStudioTest/providers/Microsoft.MachineLearningServices/workspaces/ulpgl-mlsa-event&>. [Accessed 5 June 2024].
- [87] J. S. Tim Pearce, "Reconciling Kaplan and Chinchilla Scaling Laws," *arxiv* , vol. 1, p. 8, 2024.
- [88] Hugging Face, "meta-llama/Meta-Llama-3.1-405B-Instruct," Hugging Face, 2024. [Online]. Available: <https://huggingface.co/meta-llama/Meta-Llama-3.1-405B-Instruct>. [Accessed 12 09 2024].

Annexe A

A.1 Exemples de données de test

A.1.1 Copies et compositions

À gauche, une des copies utilisées pour le test et à droite, le corrigé de l'enseignant également utilisé à cette fin.

INTERROGATION DE RATTRAP D'ELECTRICITE GENERALE (90 min)
Année Académique 2023/2024 - Deuxième Licence GC

Consignes : Répondez sur la même feuille ; Soyez précis et concis ; Téléphone interdit et voisin fermé

Question 1 (2 points) : Par rapport à leurs constitutions, quelle est la différence entre un conducteur et un isolant ? 20

Un conducteur est en métal tandis qu'un isolant peut être en porcelaine.

Question 2 (2 points) : Quel est le rôle d'un générateur ?

Le rôle d'un générateur est de mettre en mouvement simultanément les charges mobiles situées dans les matériaux conducteurs.

Question 3 (2 points) : Quelle est la différence entre un courant continu et un courant alternatif ?

Un courant continu est un courant qui ne change pas son sens au cours du temps alors qu'un courant alternatif est celui qui périodiquement change son sens.

Question 4 (3 points) : Citez trois éléments passifs et dites comment est le courant par rapport à la tension dans chaque élément en régime alternatif

- Un condensateur : le courant y est stocké.
- Un récepteur : l'énergie électrique se transforme en diverses énergies.
- Une inductance :

Question 5 (2 points) : Qu'est-ce qu'une Source idéale de tension ?

Une source idéale de tension est une source qui fournit une intensité constante quelque soit la charge qui la traverse, si corde que cette charge ne soit pas nulle.

Question 6 (1 point) : Quel est l'élément passif qui stock l'énergie dans un champ magnétique ?

Un condensateur.

INTERROGATION D'ELECTRICITE GENERALE (90 min)

Consignes : Répondez sur la même feuille ; Soyez précis et concis ; Telephone interdit et voisin fermé :

Question 1 (2 points) : Par rapport à leur à leurs constitutions, quelle est la différence entre un conducteur et un isolant ?

Reponse : Ceux qui possèdent des électrons libres sont appelés conducteurs (essentiellement les métaux), et ceux qui n'en possèdent pas, sont appelés isolants.

Question 2 (2 points) : Quel est le roles d'un générateur ?

Reponse : Le role du générateur consiste non pas à fabriquer des charges, mais à mettre en mouvement simultané les charges mobiles situé dans les matériaux conducteurs du circuit électrique.

Question 3 (2 points) : Quelle est la différence entre un courant continu et un courant alternatif ?

Reponse : Un courant continu (c.c) est donc un courant indépendant du temps. Les courant alternatif est un courant électrique qui périodiquement change de sens.

Question 4 (3 points) : Citez trois éléments passifs et dites comment est le courant

A.1.2 Sorties des LLMs

À gauche, la sortie du modèle LLAMA-3 8B-Instruct, à droite celle du modèle GPT-3 Turbo après transformation et nettoyage préalable au processus pour l'enregistrement dans la base de données.

```

{
    "advice": "Bien joué pour votre effort ! Vous avez des points forts, mais il est important de travailler sur la précision et la clarté de vos réponses.",
    "grading": {
        "1": {
            "note": 0,
            "commentaires": "La réponse est incomplète et contient des erreurs. Un conducteur est un matériau qui permet le passage du courant, tandis qu'un isolant est un matériau qui empêche le passage du courant. Il est important de préciser les caractéristiques des conducteurs et des isolants."
        },
        "2": {
            "note": 1,
            "commentaires": "La réponse est partiellement correcte. Un générateur est un dispositif qui produit de l'énergie électrique, mais il n'est pas juste pour la production d'énergie électrique. Il est important de préciser les différents types de générateurs et leurs utilisations."
        },
        "3": {
            "note": 1,
            "commentaires": "La réponse contient des erreurs. Le courant alternatif est un courant qui change de direction et de magnitude périodiquement, tandis que le courant continu est un courant qui circule dans une direction constante. Il est important de comprendre les différences entre les courants alternatif et continu."
        }
    }
}

{
    "advice": "Tu as montré une compréhension générale des concepts, mais il y a des erreurs à corriger. Assure-toi de fournir des explications claires et précises pour chaque réponse.",
    "grading": {
        "1": {
            "note": 0.5,
            "commentaires": "La réponse montre une confusion entre conducteur et isolant. Travaille sur la distinction entre les deux et ajoute des détails sur les électrons libres et leur rôle."
        },
        "2": {
            "note": 0.5,
            "commentaires": "La réponse est incomplète. Explique plus en détail le rôle d'un générateur dans un circuit électrique."
        },
        "3": {
            "note": 0.5,
            "commentaires": "Il y a des confusions entre courant continu et courant alternatif. Précise la définition de chaque type de courant."
        },
        "4": {
            "note": 0.5,
            "commentaires": "Les éléments passifs cités ne sont pas corrects. Revois la définition des éléments passifs et leur relation avec la tension en régime alternatif."
        },
        "5": {
            "note": 0.5,
            "commentaires": "La réponse manque de précision sur la source idéale de tension. Développe davantage sur les caractéristiques d'une telle source."
        }
    }
}

```

A.2 Exemple de prompts utilisés et configuration du modèle

Voici un exemple de prompt utilisé pour la correction et la génération de feedback.

```

role_prompt = f"""
>>>> ROLE_PROMPT >>>>
Tu es un enseignant précis, qui se concentre sur l'évaluation minutieuse des copies et la communication claire des erreurs.
"""

grading_elements = f"""
>>>> SUPPORTS D'ÉVALUATION >>>>>

Voici les éléments dont tu disposeas:
    >>>> CORRIGÉ DE L'ENSEIGNANT >>>>
    {data.teacher_corrected_assessment}

    >>>> CRITÈRES DE CORRECTION >>>>
    {data.grading_criteria}

    >>>> RÉPONSES DE L'ÉTUDIANT >>>>
    {data.student_responses}
"""

instruction_prompt = f"""
>>>> INSTRUCTION_PROMPT >>>>
Ton rôle est de corriger chaque réponse en fonction des critères suivants :
1. Analyse chaque réponse en fonction de la compréhension conceptuelle, exactitude des faits, et clarté de l'explication
2. Attribue une note précise pour chaque réponse.
3. Fournis des commentaires explicatifs détaillés sur chaque erreur, en insistant sur ce qui doit être corrigé et pourquoi.
4. Inclue un retour global au début qui indique clairement les domaines où l'étudiant doit s'améliorer.
5. Utilise un ton pédagogique, mais reste rigoureux dans la notation.
6. Retourne tout en français.
7. **Le résultat doit être un JSON strict, sans texte explicatif supplémentaire.**
8. **Retourne uniquement un JSON conforme au format ci-dessous, sinon la réponse sera rejetée.**

"""

output_structre = f"""
>>>> OUTPUT_STRUCTURE >>>>
Le JSON doit être structuré exactement comme ceci : sans texte explicatif supplémentaire.
Ne commencez pas par un texte comme "Here is the corrected evaluation in JSON format".

```

Voici sa configuration pour le modèle LLM GPT-3.5 Turbo.

```
import openai
import json
import re
from core.config import settings

# Access the OpenAI API key
openai.api_key = settings.openai_api_key

async def get_grading_from_llm(
    role_prompt: str,
    grading_elements: str,
    instruction_prompt: str,
    output_structre: str,
):
    try:
        # Call OpenAI API
        response = openai.ChatCompletion.create([
            model="gpt-3.5-turbo",
            messages=[
                {"role": "system", "content": role_prompt},
                {"role": "user", "content": grading_elements},
                {"role": "user", "content": output_structre},
                {"role": "user", "content": instruction_prompt},
            ],
            temperature=0.2,
            max_tokens=3500,
            top_p=0.5,
            frequency_penalty=0,
            presence_penalty=0,
        ])

        # Get the response text
        result = response["choices"][0]["message"]["content"]
        print("Response from OpenAI:", result)
    except Exception as e:
        print(f"Error occurred while calling OpenAI API: {e}")

```

A.3 Documentation sur les endpoints disponibles de l'API

Cette annexe a pour objectif de fournir des précisions sur quelques endpoints de l'implémentation finale de notre API. Pour l'endpoint POST `/api/teachers/register`, voici un exemple de requête réussie :

The screenshot shows a POST request to `/api/teachers/register` with the following details:

- Parameters:** No parameters.
- Request body (required):** application/json
- Body Content:**

```
{
  "name": "Mbuyi",
  "postname": "Kabila",
  "last_name": "Nsunda",
  "email": "mbuyi.nsunda@unigom.cd",
  "password": "string",
  "new_institution": {
    "name": "Université de Goma (UNIGOM)",
    "description": "Université publique à Goma, en province du Nord-Kivu."
  },
  "is_active": true
}
```
- Servers:** (empty)
- Server response:**
 - Code:** 200
 - Response body:**

```
{
  "name": "Mbuyi",
  "postname": "Kabila",
  "last_name": "Nsunda",
  "email": "mbuyi.nsunda@unigom.cd",
  "id": 1,
  "institutions": [
    {
      "name": "Université de Goma (UNIGOM)",
      "description": "Université publique à Goma, en province du Nord-Kivu.",
      "id": 1
    }
  ]
}
```
 - Response headers:**

```
access-control-allow-credentials: true
content-length: 234
content-type: application/json
date: Thu, 19 Sep 2024 03:15:41 GMT
server: uvicorn
```
- Responses:**
 - Code:** 200
Description: Successful Response
Media type: application/json
- Links:** No links

En cas d'erreur, voici le retour de l'indpint :

The screenshot shows a 422 Validation Error response with the following details:

- Code:** 422
- Validation Error**
- Media type:** application/json
- Example Value | Schema:**

```
{
  "detail": [
    {
      "loc": [
        "string",
        0,
        "msg",
        "type"
      ],
      "msg": "string",
      "type": "string"
    }
  ]
}
```
- Links:** No links

Pour l'endpoint

GET/api/student_responses/assignment/{assignment_id}/student/{student_id}/question/{question_id}, {assignment_id} représente l'ID de la tâche, {student_id} l'ID de l'étudiant, et {question_id} l'ID de la question sont les paramètres de l'endpoint. Cet endpoint a pour objectif d'obtenir les réponses des étudiants en fonction de la tâche.

GET /api/student_responses/assignment/{assignment_id}/student/{student_id}/question/{question_id} Get Student Responses By Assignment Student And Question Endpoint

Parameters

Name	Description
assignment_id * required	assignment_id
student_id * required	student_id
question_id * required	question_id

Try it out

En voici un exemple de requête réussie :

Code Description Links

200 Successful Response No links

Media type

application/json

Controls Accept header.

Example Value | Schema

```
[  
  {  
    "question_id": 0,  
    "response_text": "string",  
    "file": "string",  
    "id": 0,  
    "assignment_id": 0,  
    "student_id": 0,  
    "grade": 0,  
    "state": true  
  }]
```

En cas d'erreur, voici le retour est :

422 Validation Error No links

Media type

application/json

Example Value | Schema

```
{  
  "detail": [  
    {  
      "loc": [  
        "string",  
        0  
      ],  
      "msg": "string",  
      "type": "string"  
    }  
  ]}
```