FACULTE DES SCIENCES BEN M'SICK
UNIVERSITÉ HASSAN II DE CASABLANCA

Département Mathématiques et Informatique

Filière : LICENCE D'EXCELLENCE

EN INTELLIGENCE ARTIFICIELLE

Rapport du projet de fin de module

Deep Learning

Mettre en place un jeu de données (dataset) contenant des paires Question-Réponse (Q-R) ou des textes explicatifs (des questions de mathématiques (par exemple, niveau collège/lycée)

Réaliser par :

BAHRI ILHAME - Aldiebes Ghanem Israa - Bounadar Douaa

Demander par :

Pr.EL Habib BENLAHMAR

Année universitaire : 2024/2025

Table des matières

Liste des figures

Liste des tableaux

CHAPITRE 1 : Introduction

Dans un contexte éducatif en constante évolution, l'intégration des technologies d'intelligence artificielle dans l'apprentissage devient une nécessité. Le présent projet vise à mettre en place un jeu de données (dataset) structuré contenant des paires Question-Réponse (Q-R) ou des textes explicatifs portant sur des questions de mathématiques niveau collège. Ce dataset constitue une base fondamentale pour le développement d'un système intelligent de soutien pédagogique, capable de répondre aux questions des élèves de manière claire, personnalisée et adaptée à leur niveau.

Notre démarche ne se limite pas à une simple collecte de questions ; nous avons également intégré des explications détaillées et des solutions progressives pour chaque problème. L'objectif est de permettre non seulement la vérification d'une réponse, mais aussi la compréhension du raisonnement qui y mène. Grâce à l'utilisation de modèles de langage de grande taille (LLM), combinés à des techniques de RAG (Retrieval-Augmented Generation), notre système peut exploiter ce dataset pour générer des réponses cohérentes, contextualisées, et alignées avec le programme scolaire.

Ce projet s'inscrit dans une logique d'innovation pédagogique, en mettant la puissance de l'IA au service de l'enseignement des mathématiques et en facilitant l'accès à des ressources claires, structurées et interactives.

Chapitre 2 : Objectifs

Le projet a pour objectif **principal** de constituer un jeu de données structuré et de qualité, contenant :

- Des paires Question-Réponse (Q-R) couvrant les principaux chapitres du programme de mathématiques de niveau collège (arithmétique, géométrie, algèbre, probabilités, etc.).
- Des explications détaillées pour chaque question, permettant aux élèves de comprendre le raisonnement menant à la solution.

Les objectifs **secondaires** incluent :

- Assurer la compatibilité du jeu de données avec des systèmes d'apprentissage automatique, notamment les LLM et les pipelines RAG.
- Permettre une utilisation flexible dans des applications éducatives (plateformes d'apprentissage, chatbots pédagogiques, etc.).
- Garantir la clarté et la précision des questions, réponses, et explications pour un public cible de collégiens.

Chapitre 3 : Intégration et usages fonctionnels

Le jeu de données que nous avons conçu est pensé pour être exploité dans divers contextes pédagogiques et technologiques. Sa structure organisée (paires Question–Réponse avec explications) le rend facilement intégrable dans différents types de systèmes, que ce soit à des fins éducatives, d'analyse ou de recherche en intelligence artificielle.

- ❖ Plateformes éducatives

Le dataset peut être intégré dans des applications d'apprentissage en ligne, permettant la génération automatique d'exercices interactifs accompagnés de réponses détaillées et d'explications pas à pas, favorisant la compréhension autonome des élèves.

- ❖ Chatbots pédagogiques

En l'associant à un modèle de langage, il devient possible de développer des chatbots intelligents capables de répondre en temps réel aux questions des collégiens, avec des réponses adaptées au niveau de l'élève, et enrichies d'exemples ou de rappels de notions.

- ❖ Outils d'analyse pédagogique

Les enseignants et chercheurs en éducation peuvent s'appuyer sur ce dataset pour analyser les types d'erreurs fréquentes, identifier les chapitres les plus problématiques, ou encore évaluer la performance de différents modèles d'IA dans le domaine de la résolution de problèmes mathématiques.

- ❖ Recherche en Intelligence Artificielle

Le dataset constitue un excellent support pour l'entraînement et l'évaluation de modèles LLM ou RAG, appliqués à des tâches complexes telles que la compréhension de problèmes, la génération d'explications logiques, ou l'automatisation de la correction.

Grâce à son format structuré et enrichi, ce jeu de données est facilement exploitable au sein de bases de données éducatives, de systèmes de requêtage, ou via des API, pour des traitements en temps réel.

Chapitre 4 : Architecture technique

La mise en place de ce projet repose sur une architecture technique combinant modèles de langage de grande taille (LLM), techniques de RAG (Retrieval-Augmented Generation), et une conception rigoureuse du dataset. Cette approche permet non seulement de générer et structurer les données de manière automatisée et cohérente, mais aussi de les exploiter pour développer des outils intelligents capables de comprendre, résoudre et expliquer des problèmes mathématiques du niveau collège.

Nous avons également exploré l'utilité des API pour rendre le dataset accessible en temps réel à travers des applications web, des interfaces de chatbot, ou des plateformes d'e-learning. L'ensemble du système repose sur un code organisé en modules, facilitant l'intégration, la maintenance et l'évolution du projet.

Cette section présente donc les outils, méthodes et concepts techniques utilisés : génération des données via LLM, enrichissement sémantique avec RAG, structure du code, et potentiel d'interfaçage avec des systèmes externes via API.

1.RAG (Retrieval-Augmented Generation)

1.1. Contexte et motivation

Les **modèles de langage génératifs** comme GPT, BERT ou LLaMA ont transformé la manière dont les machines interagissent avec le langage humain. Cependant, leur principal **point faible** réside dans leur dépendance à leur **connaissance pré-entraînée**. Ils peuvent produire des contenus cohérents mais parfois **hallucinés**, inexacts, ou obsolètes.

Pour remédier à cette limitation, une nouvelle classe d'architectures a émergé : **Retrieval-Augmented Generation (RAG)**. Elle introduit une **connexion explicite entre la génération de texte et une base documentaire dynamique**, permettant aux modèles de s'appuyer sur des **connaissances actualisées, fiables et contextuelles**.

Dans notre projet éducatif, ce paradigme est utilisé pour **générer automatiquement des questions-réponses pédagogiques**, en s'appuyant sur les contenus de cours fournis par l'enseignant (fichiers PDF).

1.2. Définition du RAG

RAG (Retrieval-Augmented Generation) est un paradigme hybride combinant deux sous-domaines du NLP :

- **Information Retrieval (IR)** – la récupération d'information pertinente dans un corpus documentaire à l'aide de méthodes vectorielles avancées ;
- **Text Generation (NLG)** – la génération automatique de texte par des modèles de langage entraînés à prédire la suite logique d'un texte ou à répondre à une question.

L'objectif de RAG est de permettre à un système de génération de texte de **s'appuyer sur des documents externes** pour produire des réponses plus **factuelles, précises et traçables**.

1.3. Architecture conceptuelle de RAG

L'architecture de RAG repose sur deux modules principaux, **interconnectés mais indépendants** :

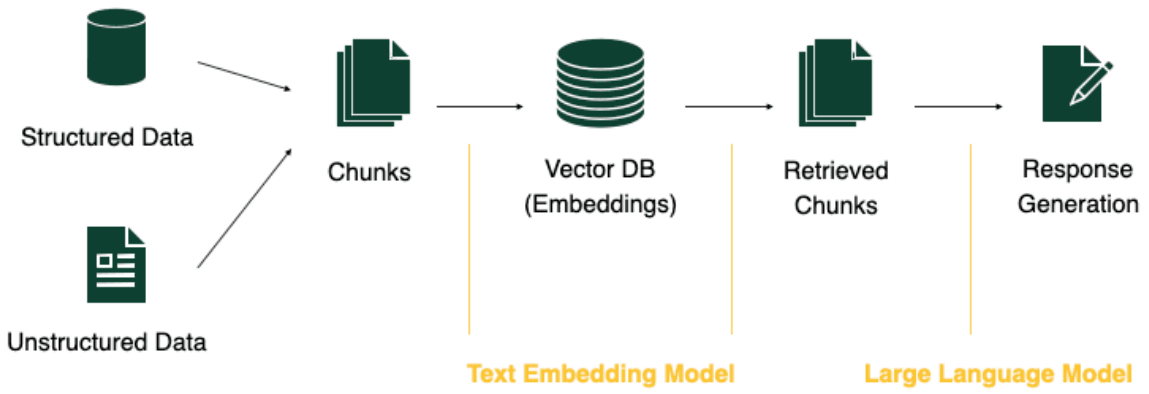

Figure 1 : Architecture conceptuelle de RAG

1.3.1 Module de récupération (Retriever)

Le rôle du Retriever est de rechercher dans une **base de documents vectorisée** les passages les plus pertinents à partir d'une **requête d'entrée**. Cette étape repose sur deux piliers :

a) Encodage sémantique des documents et de la requête

- Chaque document (ou passage de document) est **vectorisé** grâce à un **encodeur** (souvent un modèle type BERT, Sentence-BERT ou MiniLM), qui transforme chaque fragment de texte en un vecteur dense de dimension fixe (par exemple, 768 dimensions).
- La requête est encodée de la même manière. Ainsi, on dispose de vecteurs comparables dans un même espace sémantique.

b) Recherche par similarité vectorielle

- Une fois les vecteurs construits, on utilise une **métrique de distance** (par ex. **cosine similarity** ou **dot product**) pour évaluer la proximité entre la requête et tous les documents.
- On sélectionne alors les **top-k documents** les plus proches, qui sont supposés être **les plus informatifs et contextuels**.

c) Indexation et performance

- Pour améliorer les performances, on utilise des structures comme **FAISS (Facebook AI Similarity Search)** ou **Milvus**, qui permettent une **recherche vectorielle rapide** dans des bases contenant potentiellement des millions de documents.

1.3.2. Module de génération (Generator)

Une fois les documents pertinents récupérés, ils sont transmis en entrée à un **modèle de langage génératif**, comme **GPT, T5, FLAN-T5** ou **LLaMA**.

a) Mécanisme de conditionnement

Le modèle de génération ne reçoit pas uniquement la question (requête), mais aussi **les documents récupérés concaténés** à la requête. Cela constitue un **contexte enrichi**, sur lequel le modèle se base pour produire la réponse.

Exemple schématique du prompt :

> [Document 1]
> [Document 2]
> ...
> Question : "Quels sont les types de fonctions en mathématiques ?"
> Réponse :

b) Génération contrôlée

Grâce aux documents fournis, le modèle n'a plus besoin d'« halluciner » une réponse. Il **génère un texte qui reflète le contenu des documents** tout en gardant la fluidité et la richesse linguistique qu'on attend d'un modèle LLM.

c) Approches hybrides

Certains systèmes RAG utilisent plusieurs stratégies pour la génération :

- **Fusion de contextes** : concaténer les passages sélectionnés.
- **RAG-Sequence vs. RAG-Token** : le modèle peut choisir dynamiquement quel document utiliser pour chaque token généré (plus complexe).

1.4. Application du RAG dans notre projet

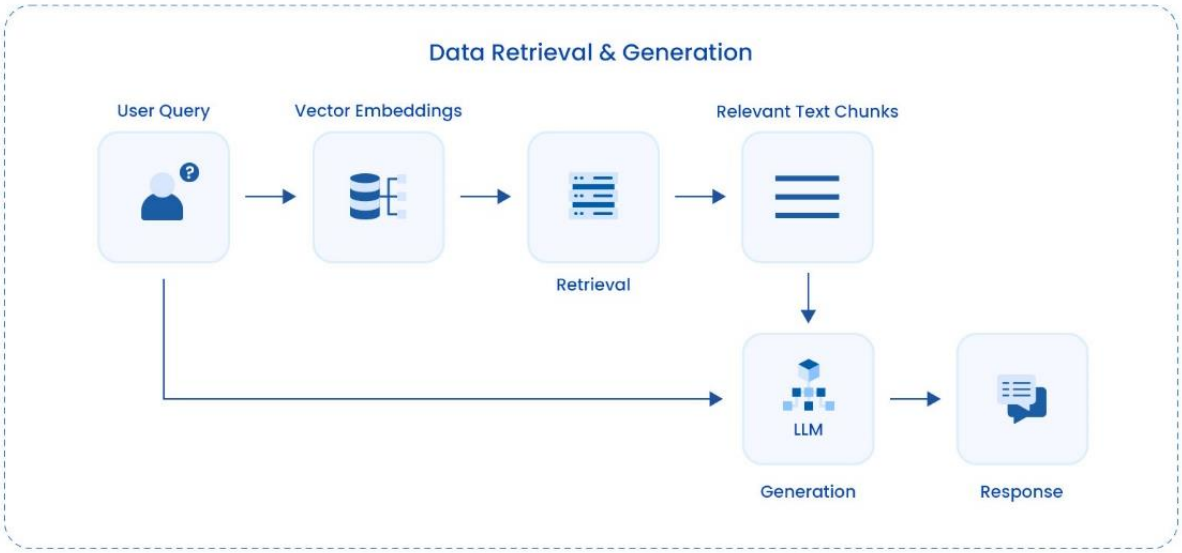

Figure 2 : Application du RAG dans notre projet

Dans notre projet, l'objectif était de générer des **questions pédagogiques ciblées** (QCM, Vrai/Faux, questions ouvertes) à partir de **cours PDF**. Voici comment chaque étape du RAG a été adaptée :

Étape 1 : Requête de l'utilisateur (Query)

L'utilisateur soumet une requête textuelle. Dans notre projet, cette requête est générée automatiquement sous la forme :

"À partir du contenu de ce cours, génère 20 questions pratiques (avec réponses et explications)."

C'est cette requête qui lance tout le processus RAG.

Étape 2 : Encodage de la requête

La requête est **encodée sous forme de vecteur** à l'aide d'un **modèle d'embedding** (dans notre cas : GoogleGenerativeAIEmbeddings de Gemini).
Cela permet de **comparer sémantiquement** cette requête à une base de documents encodés.

Étape 3 : Recherche dans la base vectorielle (Retrieval)

Avant toute génération, RAG interroge une **base de connaissances vectorisée** :

- Les **cours PDF sont d'abord découpés** en petits morceaux cohérents (appelés *chunks*).
- Ces morceaux sont vectorisés avec le même modèle d'embedding et **stockés dans une base vectorielle (ici FAISS)**.
- Lorsqu'une requête arrive, le système cherche dans cette base les **k fragments les plus proches** sémantiquement.

Étape 4 : Augmentation du contexte

Les fragments récupérés sont **ajoutés au prompt d'entrée du modèle de langage**.
C'est cela l'**augmentation** : le LLM ne reçoit pas juste la requête, il reçoit aussi le **contexte textuel associé**, ce qui lui permet de :

- Générer du contenu spécifique à un document,
- Limiter les erreurs d'hallucination,
- S'aligner sur des connaissances à jour.

Exemple de prompt généré :

"Voici le contenu d'un cours sur le Théorème de Pythagore [...] Génère 20 questions avec réponses et explications."

Étape 5 : Génération par le LLM

Le modèle de langage (**LLM**, ici : ChatGoogleGenerativeAI) prend en entrée :

- La requête utilisateur,
- Les documents top-k récupérés.

Il produit alors une **réponse complète**, souvent longue et structurée : dans notre cas, un **bloc de 20 questions/réponses avec explication**, dans un format JSON.

Étape 6 : Sortie structurée et réutilisable

La réponse est formatée dans une structure standardisée (JSON), contenant :

- "cours" : nom du chapitre traité,
- "question" : la consigne posée,
- "réponse" : la réponse correcte,
- "explication" : justification ou raisonnement associé.

1.5. Avantages concrets de l'approche RAG dans un contexte éducatif

Avantage	Détail
Précision pédagogique	Chaque question est enracinée dans un contenu réel du cours, évitant les erreurs conceptuelles.
Personnalisation	Le système peut générer des questions adaptées à un chapitre précis ou au niveau de difficulté souhaité.
Évolutivité	Il suffit d'ajouter de nouveaux documents pour étendre le champ des connaissances sans réentraîner le modèle.
Réduction des hallucinations	L'ancrage dans le cours empêche le modèle de produire des réponses fausses ou décontextualisées.
Traçabilité	On peut associer chaque question générée à sa source documentaire exacte (utile pour la validation par un enseignant).

Tableau 1 : Avantages concrets de l'approche RAG dans un contexte éducatif

1.6. Limites et perspectives

Bien que puissant, RAG présente quelques limites :

- **Dépendance à la qualité des documents** : si les cours sont mal structurés ou flous, la génération peut être imprécise.
- **Sélection de contexte imparfaite** : si le Retriever ne choisit pas les bons documents, la génération sera peu pertinente.
- **Limite de mémoire** : les modèles LLM ont une fenêtre de contexte limitée (ex. 4k à 32k tokens), ce qui peut poser problème pour de longs documents.

Perspectives :

- Affiner le ranking des passages par des modèles plus sophistiqués (reranking).
- Introduire une boucle de **feedback humain** pour corriger/améliorer les générations.
- Utiliser des techniques comme **Long Context RAG** ou **Multi-hop RAG** pour répondre à des questions plus complexes.

1.7. Conclusion

L'architecture RAG permet de **combiner la rigueur des systèmes de recherche d'information** avec la **richesse expressive des modèles génératifs**, créant ainsi un système intelligent capable de produire des contenus fiables, contextualisés et pédagogiques.

Dans notre cas, cette approche a été décisive pour générer automatiquement des **exercices contextualisés**, tout en assurant une **fidélité aux notions enseignées**. Elle représente un levier prometteur pour construire des outils d'enseignement assisté par l'IA, adaptables à tous les niveaux scolaires.

2- LLM (Large Language Models)

2.1. Définition générale des LLM

Un **LLM** (Large Language Model), ou **Modèle de Langage à Grande Échelle**, est un modèle d'intelligence artificielle spécialisé dans la compréhension et la génération de texte humain. Il est entraîné sur des milliards de mots et phrases issus de livres, d'articles, de pages web, de forums, etc., ce qui lui permet de reconnaître les structures, les patterns grammaticaux et le contexte des mots.

Les LLM appartiennent à la famille des modèles **fondés sur l'architecture Transformer**, introduite par l'article fondamental *"Attention is All You Need"* (Vaswani et al., 2017). Cette architecture est particulièrement efficace pour le traitement séquentiel des données textuelles, permettant de capturer à la fois le sens global d'un texte et les relations entre les mots, même lorsqu'ils sont éloignés dans la phrase.

Exemples populaires de LLM :

- GPT-3.5 et GPT-4 par OpenAI
- Gemini par Google DeepMind
- Claude par Anthropic
- LLaMA par Meta AI
- Mistral, Falcon, Bloom (open-source)

2.2. Comment les LLM sont-ils entraînés ?

Le cœur de l'apprentissage d'un LLM repose sur la **prédiction du mot suivant** dans une séquence de texte. Le processus suit plusieurs étapes :

a) Pré-entraînement

Durant cette phase, le modèle lit des milliards de phrases sans supervision humaine. Il apprend à:

- Prédire le mot suivant (apprentissage auto-supervisé),
- Comprendre la grammaire, la syntaxe et les nuances du langage,
- Développer des représentations sémantiques profondes du texte.

b) Fine-tuning (ajustement)

Parfois, les LLM subissent un entraînement supplémentaire sur des jeux de données spécialisés (éducation, médecine, finance…). Cela permet d'adapter leur comportement à des tâches spécifiques.

c) Reinforcement Learning from Human Feedback (RLHF)

Pour certains modèles (comme GPT-4), une dernière phase d'entraînement consiste à affiner les réponses à l'aide de retours humains. Cela permet :

- Rendre les réponses plus utiles,
- D'éviter les propos inappropriés ou erronés.

2.3. Fonctionnement général : les concepts clés

Un LLM transforme le texte en **vecteurs numériques** (embeddings), traite ces vecteurs dans des **couches de neurones profondes**, et génère un texte de sortie **mot par mot**, en se basant sur les probabilités apprises.

Éléments importants :

- **Tokenisation** : le texte est découpé en unités appelées *tokens* (souvent des morceaux de mots).
- **Auto-attention** : permet au modèle de se "concentrer" sur les parties du texte les plus pertinentes.
- **Positionnal encoding** : indique au modèle l'ordre des mots dans une phrase.

2.4. Cas d'usage pédagogiques des LLM

Dans un contexte éducatif, les LLM peuvent :

- Générer automatiquement des **questions d'examen**, des **quizz**, ou des **QCM** ;
- Créer des **explications pédagogiques personnalisées** adaptées à différents niveaux (ex : collège, lycée) ;
- Traduire ou reformuler des consignes ;
- Corriger des réponses ou proposer des pistes de réflexion.

Dans notre cas, le LLM est utilisé pour **générer des paires question-réponse** ainsi que des **explications détaillées**, dans un format structuré, en se basant sur un *prompt* bien défini contenant le nom du chapitre du cours.

2.5. Avantages des LLM dans un projet éducatif

Avantage	Détail
Scalabilité	Génération rapide de centaines de questions sans effort manuel.
Compréhension du contexte	Le LLM comprend le niveau de difficulté et adapte ses formulations.
Gain de temps	Plus besoin de créer manuellement chaque Q-R.
Uniformité	Les questions ont une structure cohérente et une terminologie homogène.
Richesse pédagogique	Possibilité d'ajouter des explications complètes, des alternatives, ou des conseils méthodologiques.

Tableau 2 : Avantages des LLM dans un projet éducatif

2.6. Application du LLM dans notre projet

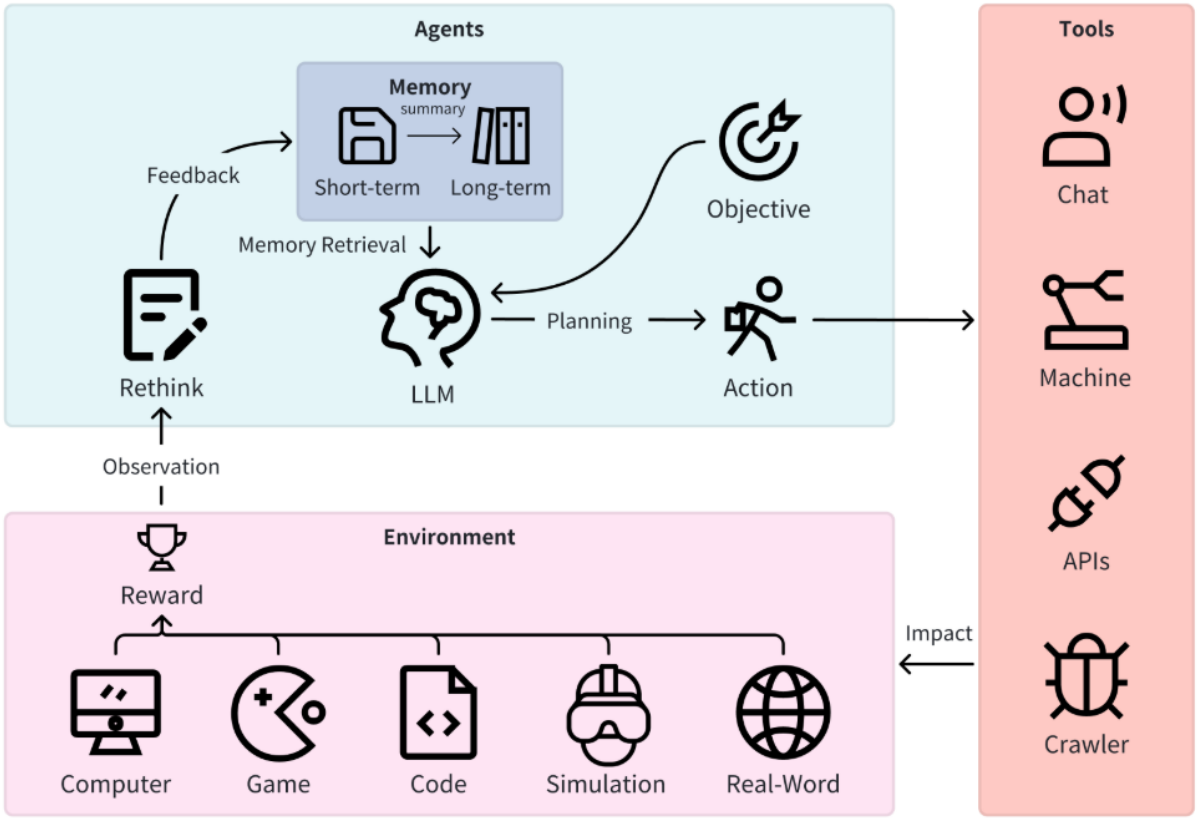

Figure 3 : Application du LLM dans notre projet

A. Choix du modèle
Utilisation du **modèle gemini-2.0-flash** via l'**API Gemini** (Google Generative AI), adapté à la génération rapide de contenus courts et pédagogiques.

B. Préparation des thématiques
Élaboration d'une liste de chapitres (fractions, équations, géométrie, proportionnalité, etc.) pour guider la génération.

C. Conception des prompts dynamiques
Création de prompts spécifiques pour chaque thème, permettant d'orienter le modèle vers une génération cohérente et variée.

D. Structure des données générées
Chaque élément suit un format JSON :

```
{
 "cours" : "...",
 "question" : "...",
 "reponse" : "...",
 "explication" : "..."
}
```

E. Contrôle qualité

- Nettoyage automatique via scripts Python.
- Validation manuelle de la structure et du fond.
- Rejet automatique des réponses incomplètes ou mal formées.

F. Résultats obtenus

- Plus de 1000 questions générées.
- Variété dans les types (problèmes, QCM, vrai/faux…).
- Explications pédagogiques intégrées et compréhensibles.

2.7. Limites et défis des LLM

Malgré leur puissance, les LLM présentent des risques à ne pas négliger :

- **Hallucinations** : le modèle peut "inventer" des faits ou des réponses incorrectes (surtout en mathématiques).
- **Dépendance au prompt** : une mauvaise consigne peut entraîner des résultats incohérents ou hors-sujet.

- **Manque de raisonnement profond** : les modèles peuvent résoudre des problèmes simples, mais ne maîtrisent pas toujours la logique mathématique complexe.
- **Biais dans les données** : les modèles peuvent refléter des stéréotypes ou des erreurs présents dans leurs données d'entraînement.

Solution appliquée dans notre projet : nous avons mis en place une validation manuelle des résultats et utilisé des *prompts* précis et bien encadrés pour limiter ces effets.

2.8. Conclusion :

L'intégration d'un LLM tel que *gemini-2.0-flash* dans notre projet a permis d'automatiser efficacement la génération de contenus éducatifs pertinents et bien structurés. Grâce à une approche méthodique, combinant prompts ciblés, format de sortie standardisé et validation rigoureuse, nous avons pu produire un dataset riche, adapté à des usages pédagogiques variés. Cette étape constitue une base solide pour des applications ultérieures dans les domaines de l'intelligence artificielle éducative, de la recherche, et du développement de solutions interactives pour l'apprentissage des mathématiques au collège.

Chapitre 5 : Structure et volume du dataset généré

À l'issue du processus de génération, nous avons obtenu un fichier au format JSON contenant 5000 objets, chacun représentant une entrée complète avec les champs "cours", "question", "reponse" et "explication".

Ce dataset riche et structuré couvre l'ensemble des thématiques ciblées du programme de mathématiques niveau collège, avec une grande diversité de types de questions. Le format JSON choisi facilite l'intégration dans des bases de données, des API ou des outils d'analyse. Ci-dessous, un aperçu de la structure du fichier généré est présenté sous forme de capture d'écran :


```
[
{
        "cours": "Équations du premier degré",
        "question": "Résoudre l'équation : x + 5 = 10",
        "reponse": "x = 5",
        "explication": "Pour isoler x, soustrayez 5 des deux côtés de l'équation."
},
    {
        "cours": "Équations du premier degré",
        "question": "Résoudre l'équation : 2x = 6",
        "reponse": "x = 3",
        "explication": "Pour isoler x, divisez les deux côtés de l'équation par 2."
    },
    {
        "cours": "Équations du premier degré",
        "question": "Résoudre l'équation : -3x + 7 = 1",
        "reponse": "x = 2",
        "explication": "Soustraire 7 des deux côtés : -3x = -6. Diviser par -3 : x = 2."
    },
    {
        "cours": "Équations-produit",
        "question": "Résoudre l'équation : (x - 2)(x + 3) = 0",
        "reponse": "x = 2 ou x = -3",
        "explication": "Si un produit est nul, alors au moins un des facteurs est nul. Donc x-2=0 ou x+3=0."
    },
```


Figure 4 : Structure du dataset final