الجمهورية الجزائرية الديمقراطية الشعبية
République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université de 8 Mai 1945 – Guelma
Faculté des Mathématiques, d'Informatique et des Sciences de la matière
Département d'Informatique

Mémoire de Fin d'études Master

Filière : Informatique

Option : Sciences et Technologies de l'Information et de la Communication

Thème :

Large Language Models pour Les Systèmes de Recommandation

Présenté par

Zerguine Nada

Devant le jury

Président : Pr. BENCHERIET Chemesse Ennahar
Encadreur : Pr. KOUAHLA Mohamed Nadjib
Examinateur : Pr. LAFIFI Yacine

Juin 2025

RÉSUMÉ

Avec l'importance croissante de la personnalisation de l'information, il devient essentiel de développer des systèmes de recommandation capables d'exploiter plus en profondeur la richesse du contenu textuel disponible. Cette étude vise à exploiter les LLMs dans ce domaine, en développant un système de recommandation de livres basé sur la modélisation de la similarité sémantique entre les métadonnées et les résumés générés. Pour atteindre cet objectif, cinq modèles préentraînés de type Sentence-Transformer ont été utilisés, et ont tous été ajustés (fine-tuned) pour mieux s'adapter au contexte spécifique de la tâche. Les résultats de l'évaluation ont montré que le modèle all-MiniLM-L6 a obtenu une performance remarquable, avec un score de rappel (Recall) atteignant environ 99,69 % lors de la recommandation de dix livres. Ces résultats mettent en évidence le fort potentiel des LLMs dans la construction de systèmes de recommandation intelligents, interactifs et évolutifs.

Mots-clés : Systèmes de recommandation, LLMs, Fine-tuning, Sentence-Transformers, Recommandation basée sur le contenu, Personnalisation.

ABSTRACT

With the growing importance of information personalization, it has become essential to develop recommendation systems capable of fully leveraging the richness of available textual content. This study proposes the use of Large Language Models (LLMs) in this context, through the development of a book recommendation system based on semantic similarity modeling between metadata and generated summaries. To achieve this goal, five pre-trained models from the Sentence-Transformer family were employed, all of which were fine-tuned to better adapt to the specific task context. Evaluation results showed that the all-MiniLM-L6 model achieved remarkable performance, reaching a recall score of approximately 99.69% when recommending ten books. These findings highlight the strong potential of LLMs in building intelligent, interactive, and scalable recommendation systems.

Keywords : Recommendation Systems, LLMs, Fine-tuning, Sentence-Transformers, Content-Based Recommendation, Personalization.

ملخص

مع تزايد أهمية تخصيص المعلومات، أصبح من الضروري تطوير أنظمة توصية قادرة على الاستفادة الكاملة من ثراء المحتوى النصي المتاح. تقترح هذه الدراسة توظيف النماذج اللغوية الكبيرة (LLMs) في هذا السياق، من خلال تطوير نظام لتوصية الكتب يعتمد على نمذجة التشابه الدلالي بين البيانات الوصفية والملخصات المُولَّدة. لتحقيق هذا الهدف، تم استخدام خمسة نماذج مُسبقة التدريب من عائلة Sentence-Transformer، وقد خضعت جميعها لعملية ضبط دقيق (Fine-tuning) لتتناسب مع سياق المهمة. أظهرت نتائج التقييم أن نموذج all-MiniLM-L6 قد حقق أداءً متميزًا، حيث بلغ مؤشر الاسترجاع (Recall) حوالي 99.69٪ عند التوصية بعشرة كتب. تُبرز هذه النتائج الإمكانات العالية للنماذج اللغوية الكبيرة في بناء أنظمة توصية ذكية، تفاعلية، وقابلة للتوسع.

الكلمات المفتاحية: أنظمة التوصية، النماذج اللغوية الكبيرة (LLMs)، الضبط الدقيق (Fine-tuning)، محولات الجُمل (Sentence-Transformers)، التوصية المعتمدة على المحتوى، التخصيص.

REMERCIEMENTS

Je rends grâce à Allah, le Tout-Puissant, pour m'avoir accordé la force, la patience et la persévérance nécessaires pour mener à bien ce projet.

Je remercie profondément mes parents pour leur amour inconditionnel, leur patience, leurs sacrifices et leur soutien constant, sans lesquels je n'aurais jamais pu mener ce travail à bien. J'exprime également toute ma gratitude à mes deux frères et à l'ensemble de ma famille pour leur présence réconfortante, leurs encouragements et leur soutien moral tout au long de ce parcours.

Je tiens à exprimer ma profonde gratitude à mon encadrant, Pr. Mohamed Nadjib, pour sa disponibilité, son accompagnement attentif et ses conseils précieux tout au long de ce projet de fin d'études. Sa rigueur scientifique et sa bienveillance ont grandement contribué à l'aboutissement de ce travail.

Je remercie également le doctorant Younes Allal, dont l'aide précieuse et les explications claires m'ont soutenue à chaque étape de ce projet.

Enfin, je tiens à remercier l'ensemble des enseignants du département d'informatique pour les connaissances qu'ils m'ont transmises durant mon cursus, ainsi que pour leur engagement et leur accompagnement tout au long de ces années.

TABLE DES MATIÈRES

TABLE DES FIGURES

LISTE DES TABLEAUX

LISTE DES CODES

LISTE DES ACRONYMES

BERT Bidirectional Encoder Representations from Transformers.

DUIP Dynamic User Intent Prediction.

FAISS Facebook AI Similarity Search.

LLMs Large Language Models.

LSTM Long Short-Term Memory.

NLP Natural Language Processing.

INTRODUCTION GÉNÉRALE

Les sociétés numériques connaissent aujourd'hui une croissance énorme de la quantité de données disponibles sur Internet, ce qui a entraîné des transformations profondes dans la manière de rechercher des informations pertinentes. Dans ce contexte, les systèmes de recommandation jouent un rôle central. Ils analysent le contenu et proposent des recommandations personnalisées à chaque utilisateur, en fonction de ses préférences et de son historique d'interactions. Ces systèmes sont devenus essentiels dans de nombreux domaines, comme la recommandation de produits dans le commerce électronique, de vidéos sur les plateformes de streaming, de contenus sur les réseaux sociaux, ou encore de ressources pédagogiques dans le domaine de l'éducation.

En parallèle, les avancées dans le domaine du traitement automatique du langage naturel (NLP) et de l'intelligence artificielle (IA), notamment avec l'apparition des grands modèles de langage (LLMs) comme BERT, GPT ou T5, ont profondément changé la manière dont les machines comprennent et génèrent le langage humain. Grâce à l'architecture Transformer, les modèles comme BERT offrent des capacités d'analyse sémantique avancées, ce qui permet de concevoir des systèmes de recommandation plus pertinents, efficaces et capables de saisir les nuances du sens pour fournir des suggestions plus précises et adaptées aux besoins de chaque utilisateur.

L'intégration des modèles LLM dans les systèmes de recommandation constitue une solution innovante et prometteuse aux limites des approches classiques basées sur le contenu, en exploitant des données textuelles telles que les avis, les descriptions et les métadonnées. Malgré les progrès réalisés dans ce domaine, il existe toujours un écart entre la richesse des données textuelles disponibles et la manière dont elles sont réellement exploitées. En effet, les approches traditionnelles ont souvent du mal à détecter les relations sémantiques implicites entre les éléments, en raison d'une compréhension limitée du contexte, et d'une dépendance excessive aux avis et interactions des utilisateurs, qui peuvent parfois être trompeuses. L'analyse de simples mots-clés ne suffit plus ; une compréhension fine du contexte global et des relations implicites devient indispensable.

Ainsi, la problématique principale de ce travail repose sur la question suivante : comment exploiter

les capacités des modèles LLM pour améliorer la qualité, la précision et la personnalisation des systèmes de recommandation ?

L'objectif principal de ce mémoire est de concevoir un système de recommandation sémantique basé sur les modèles LLM, capable d'exploiter les métadonnées enrichies des livres afin de proposer des recommandations précises et personnalisées à chaque utilisateur. En d'autres termes, il s'agit de construire un modèle capable de faire des suggestions pertinentes non seulement en se basant sur des mots-clés, mais aussi sur le contexte et les significations partagées entre les livres.
Ce travail de recherche s'articule autour de plusieurs questions, notamment :

- ***Q1 :*** *Quels types de modèles LLM sont les plus adaptés pour extraire des représentations sémantiques efficaces à partir de textes littéraires ?*
- ***Q2 :*** *Dans quelle mesure le fine-tuning des LLMs améliore-t-il leurs performances dans un système de recommandation ?*
- ***Q3 :*** *Quels modèles permettent d'obtenir la meilleure similarité entre la requête d'un utilisateur et les éléments recommandés ?*

Cette mémoire est divisée en trois chapitres :

- **Chapitre 1 - État de l'art :** Présentation des bases théoriques des modèles LLM, des principes des systèmes de recommandation, et des travaux récents combinant ces deux domaines.
- **Chapitre 2 - Conception :** Description des différentes étapes de la conception du système de recommandation basé sur les LLM, de la préparation des données à l'évaluation du système.
- **Chapitre 3 - Implémentation :** Présentation de la partie pratique, avec l'environnement technique, les outils utilisés, et les résultats expérimentaux obtenus.

Enfin, les trois chapitres se concluent par un résumé général récapitulant les points clés de ce mémoire.

CHAPITRE I

ÉTAT DE L'ART

Introduction

L'explosion massive du contenu numérique sur Internet a rendu difficile l'accès des utilisateurs au contenu correspondant à leurs intérêts. Les systèmes de recommandation viennent comme une solution efficace, car ils filtrent les informations et suggèrent des produits, des services ou du contenu personnalisé en fonction du comportement et des préférences de l'utilisateur. Ces systèmes sont essentiels dans des domaines tels que le commerce électronique, les plateformes de streaming et la santé numérique. Les Large Language Models (LLMs) tels que BERT contribuent à améliorer la qualité des recommandations grâce à leur capacité à comprendre le langage naturel et à traiter d'énormes quantités de textes.

1 Large Language Models (LLMs)

1.1 Définition des LLMs

Les LLMs sont des modèles du Deep Learning entraînés sur d'énormes quantités de données textuelles pour comprendre et générer du texte naturel de manière cohérente. Ils ont un impact majeur sur le traitement du langage naturel et sont largement utilisés dans divers secteurs tels que la santé, l'éducation, la finance, le droit et la cybersécurité. Parmi les modèles les plus connus figurent GPT et BERT, deux modèles de langage basés sur l'architecture des Transformers, principalement conçus pour des tâches de compréhension et de génération de texte [27].

1.2 Architecture des LLMs

Avant d'examiner le fonctionnement des LLMs, il est important de comprendre le concept d'embeddings (représentations vectorielles). Les embeddings sont des vecteurs de dimension réduite qui représentent des mots ou des entités dans un espace numérique, tout en conservant certaines propriétés sémantiques. En d'autres termes, ils permettent à un modèle de langage de comprendre les similarités et différences entre les mots sur la base de leur usage.
Les premières techniques d'embedding, comme Word2Vec et GloVe, reposaient sur les statistiques de co-occurrence des mots dans de grands corpus. Elles attribuent à chaque mot un vecteur fixe, quelles que soient les phrases dans lesquelles il apparaît. Bien que ces approches capturent certaines relations sémantiques (comme dans l'analogie **roi** $-$ **homme** $+$ **femme** $\approx$ **reine**), elles ne prennent pas en compte le contexte d'apparition du mot.

Une avancée déterminante a été introduite avec l'article "Attention Is All You Need" [28], qui a présenté l'architecture Transformer (Figure I.1). Cette architecture représente aujourd'hui le cœur des LLMs modernes, tels que GPT, BERT, T5. Elle a radicalement transformé le traitement du langage naturel grâce à un mécanisme appelé self-attention.
Le self-attention permet au modèle d'examiner l'ensemble des mots d'une séquence pour déterminer, à chaque étape, lesquels sont les plus pertinents à prendre en compte. Cette approche permet la production de représentations contextuelles, où la signification d'un mot est ajustée en fonction de son environnement linguistique.

Variantes architecturales des modèles Transformer

On peut diviser les modèles Transformer en trois grandes catégories [44] :

- **Modèles à encodeur uniquement (Encoder-Only) :** Ces modèles utilisent seulement la partie encodeur du Transformer. Ils sont très efficaces pour comprendre le sens complet d'une phrase. On les utilise souvent pour des tâches comme la classification de phrases. Un exemple bien connu est BERT [4].

- **Modèles à décodeur uniquement (Decoder-Only) :** Ces modèles utilisent uniquement la partie décodeur du Transformer. Ils sont spécialisés dans la génération de texte. La plupart des grands modèles de langage actuels, comme GPT, utilisent cette architecture.

- **Modèles encodeur-décodeur (Encoder-Decoder/Seq2Seq) :** Aussi appelés sequence-to-sequence models, ils utilisent à la fois l'encodeur et le décodeur. Ces modèles sont utiles pour des tâches qui nécessitent à la fois de comprendre un texte et de produire une réponse, comme la traduction ou la génération de résumés. Un exemple connu est T5.

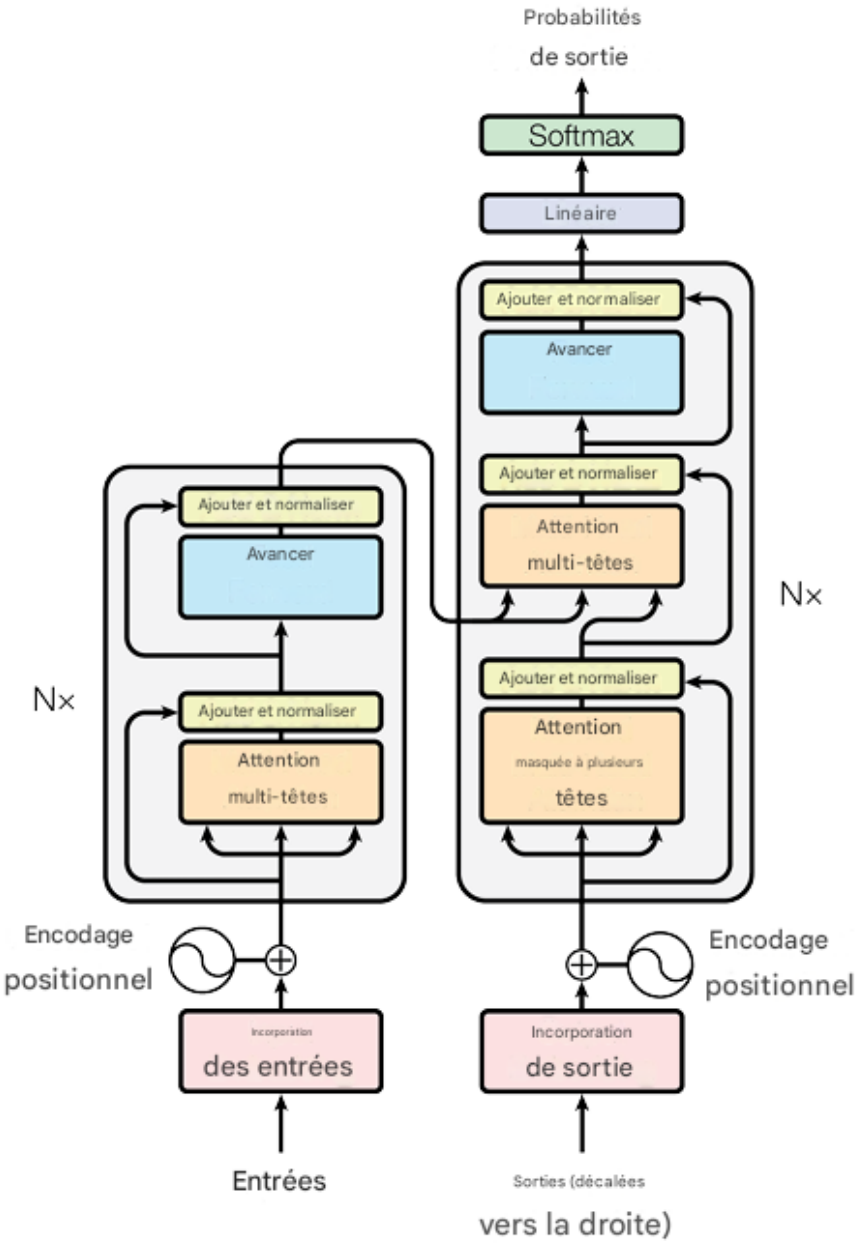


FIG. I.1 : Architecture du Transformer : mécanisme d'attention multi-têtes. [28]

2 Les systèmes de recommandation

2.1 Définition

Les systèmes de recommandation sont des algorithmes qui donnent des suggestions à l'utilisateur afin qu'elles correspondent à ses propres intérêts et comportements, et qu'elles soient déduites de manière intelligente. Ce type de système a été largement adopté par de nombreuses entreprises dans divers secteurs tels que la musique, la littérature, le commerce en ligne et les plateformes de streaming [25].

2.2 Techniques de Base

Les systèmes de recommandation reposent sur plusieurs techniques fondamentales :

- **Filtrage collaboratif :** Le filtrage collaboratif est basé sur le comportement des utilisateurs précédents, comme les produits qu'ils ont achetés ou les avis qu'ils ont laissés. Ainsi, les préférences d'un utilisateur peuvent être prédites à partir des décisions d'autres utilisateurs, en exploitant les similitudes dans leur comportement pour améliorer les recommandations personnalisées dans diverses applications [12].

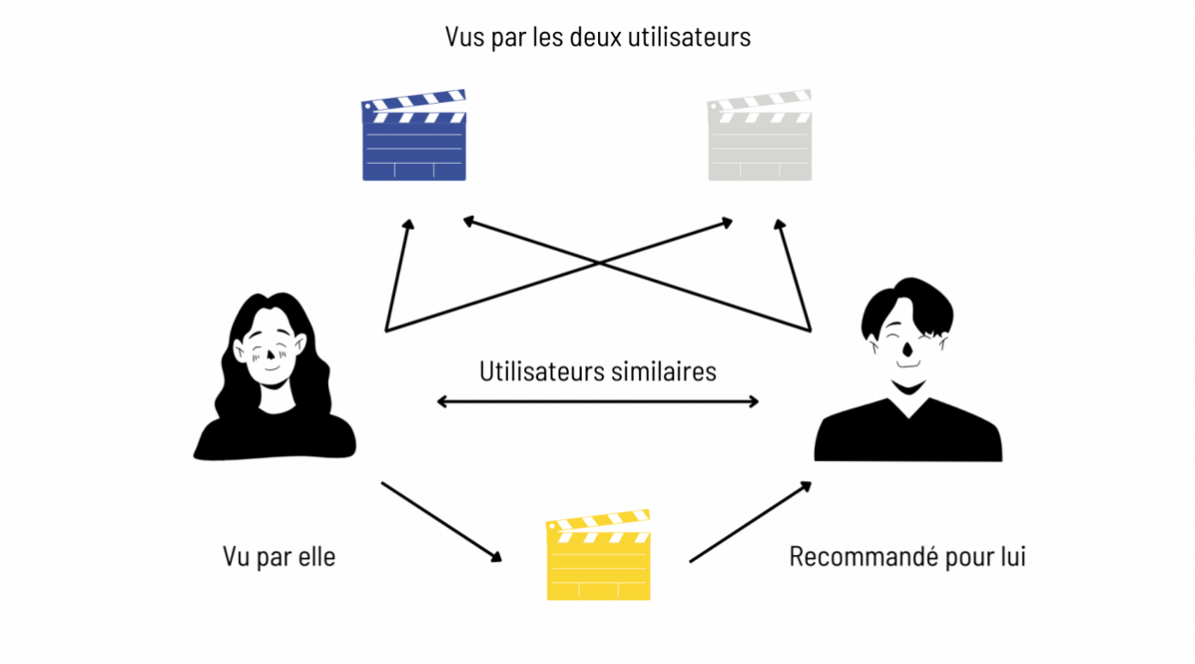


FIG. I.2 : Illustration du filtrage collaboratif. [38]

- **Filtrage basé sur le contenu :** Le filtrage basé sur le contenu est une technique fondamentale qui repose sur les caractéristiques des éléments et les préférences des utilisateurs. Des mots-clés sont utilisés pour décrire chaque élément et établir un profil utilisateur, afin d'identifier les types d'éléments susceptibles de lui plaire [24].

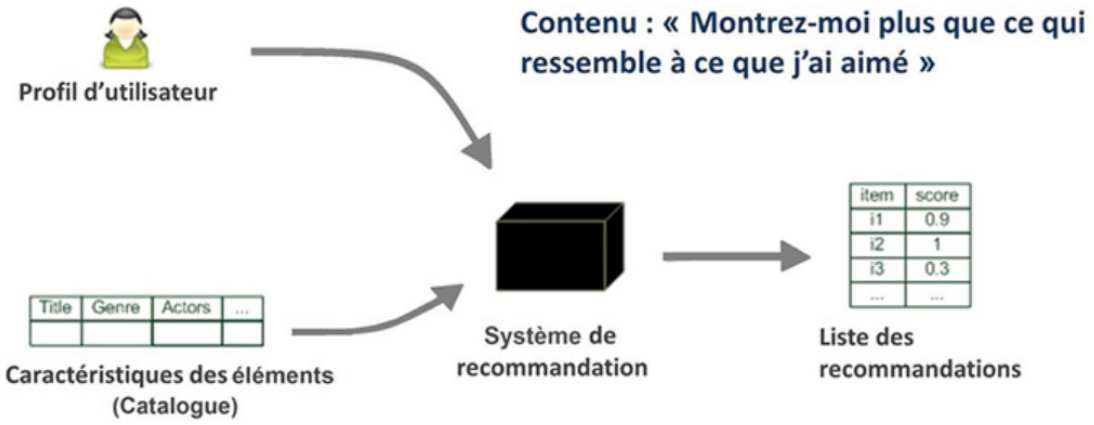


FIG. I.3 : Illustration du filtrage basé sur le contenu. [41]

- **Filtrage hybride :** Le filtrage hybride combine des techniques de filtrage collaboratif et de filtrage basé sur le contenu afin d'optimiser les systèmes de recommandation. Cette approche permet de surmonter des problèmes tels que la rareté des données et le démarrage à froid, en exploitant à la fois les interactions des utilisateurs et les caractéristiques des éléments. Elle s'adapte aux besoins variés des utilisateurs et aux différents contextes d'utilisation, ce qui permet de générer des recommandations plus précises, pertinentes et satisfaisantes [7].

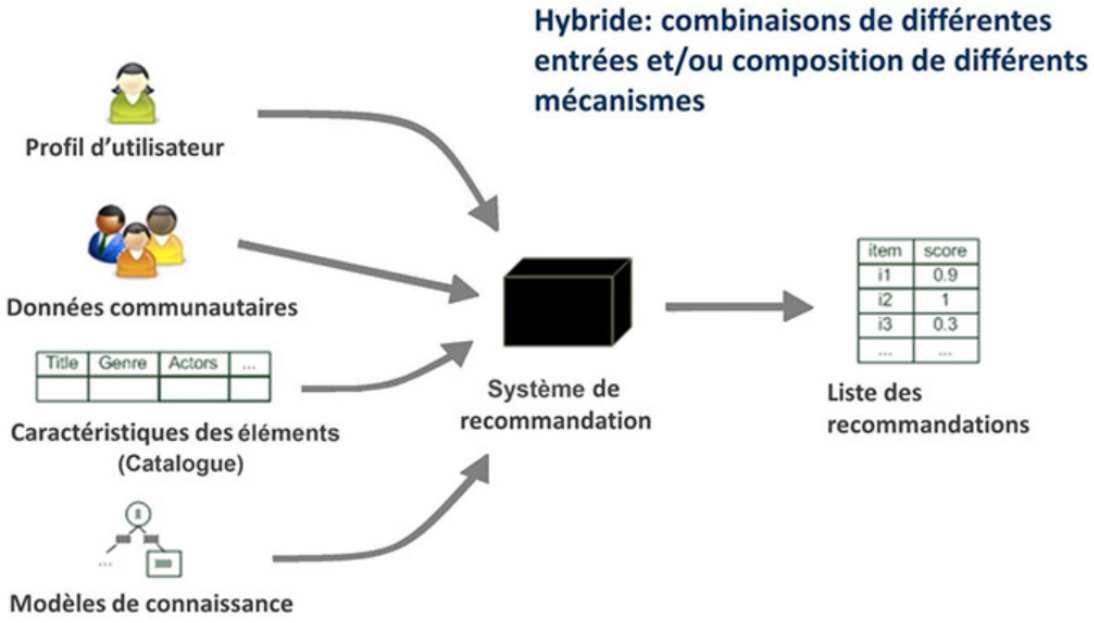


FIG. I.4 : Illustration du système de recommandation hybride. [41]

2.3 Limites des systèmes de recommandation traditionnels

Bien que les systèmes de recommandation traditionnels soient largement utilisés et efficaces dans de nombreux contextes, ils présentent plusieurs limites. Celles-ci peuvent réduire la qualité des recommandations fournies et limiter leur capacité à répondre aux besoins des utilisateurs. Parmi ces limites, on retrouve celles identifiées dans l'étude de Gao [8] :

- **Démarrage à froid :** L'utilisation des données des utilisateurs présente certains inconvénients, notamment le problème du démarrage à froid. Ce problème survient lorsqu'un nouvel utilisateur rejoint le système ou lorsqu'un utilisateur existant ajoute de nouveaux éléments. En l'absence d'historique suffisant, il devient difficile pour l'algorithme de prédire les préférences, ce qui entraîne des recommandations moins précises.
- **Sparsité des données :** Le problème de la rareté des données se produit lorsque les transactions et les commentaires ne sont pas suffisants pour déduire la similarité entre les utilisateurs ou entre les éléments, ce qui affecte la précision et la performance du système de recommandation.
- **La scalabilité :** La scalabilité des données signifie que, lorsque la quantité d'informations augmente fortement, les systèmes peuvent avoir du mal à rester précis et efficaces, tout en maintenant la qualité et la cohérence des données.

3 Intégration des LLMs dans les systèmes de recommandation

L'intégration des LLMs dans les systèmes de recommandation facilite les interactions conversationnelles en permettant une communication en langage naturel. Ces modèles exploitent des dialogues riches en texte, intégrant à la fois les préférences des utilisateurs et la description des éléments, afin de mieux traiter les requêtes complexes et d'améliorer la qualité des recommandations [11].

3.1 Formes d'intégration des LLMs

Les LLMs peuvent être intégrés dans un système de recommandation de plusieurs manières, selon le rôle qu'ils jouent dans la chaîne de traitement :

- **Comme encodeurs d'embeddings :** Les données textuelles, telles que les avis des utilisateurs ou les descriptions de produits, sont encodées (par les LLMs) en vecteurs sémantiques exploitables par des modèles de recommandation classiques.
- **Comme agents conversationnels :** Le LLM agit comme une interface intelligente capable de dialoguer en langage naturel afin de cerner les préférences de l'utilisateur. Il en extrait des intentions ou des préférences, sous forme sémantique ou structurée, qu'il transmet ensuite à un moteur de recommandation chargé d'identifier les items les plus pertinents (ex. [6]).

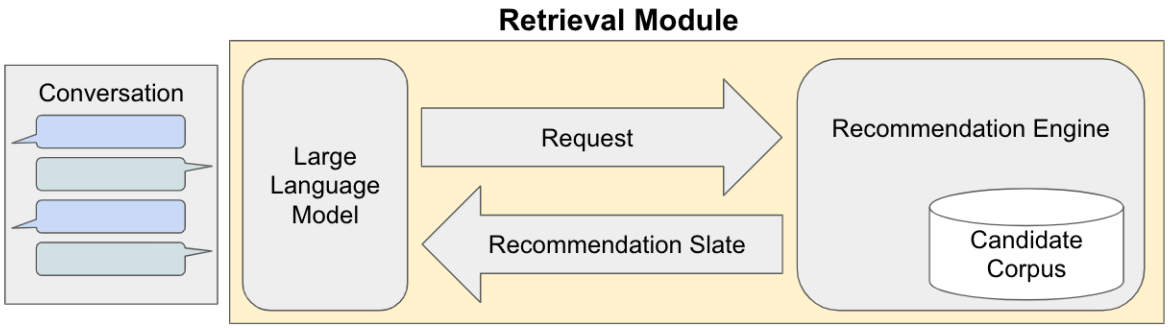


FIG. I.5 : Les LLMs comme des agents conversationnels [6]

- **Comme moteur de recommandation autonome :** Dans certains cas, le LLM peut effectuer le raisonnement et produire directement les recommandations à partir d'un prompt contextualisé avec les historiques ou les métadonnées.
- **Comme outil d'explication :** Les LLMs peuvent également générer des explications compréhensibles des recommandations proposées, renforçant la transparence du système [17].

3.2 Les modèles LLMs utilisés

D'après le survey de WU et al. [30], les LLMs utilisés pour la recommandation peuvent être classés en deux grandes familles :

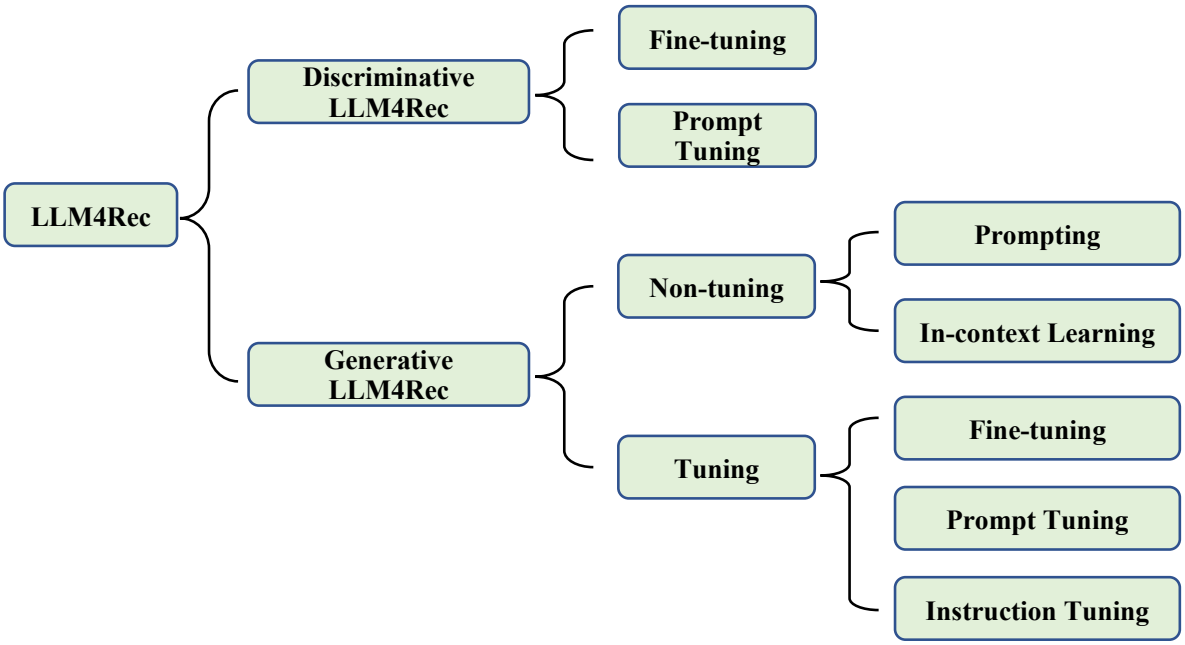


FIG. I.6 : Les modèles LLMs utilisés dans les systèmes de recommandation [30]

Modèles discriminatifs

Ces modèles, comme BERT, sont optimisés pour comprendre et encoder des textes. Ils sont particulièrement efficaces pour des tâches comme la recherche sémantique, la classification ou la prédiction de similarité utilisateur-produit.

- **Fine-tuning :** Le fine-tuning adapte un LLMs pré-entraîné à une tâche précise, comme la recommandation, en l'ajustant avec des données spécifiques (avis, clics, achats, profils, etc.). L'objectif est de conserver sa connaissance générale tout en le spécialisant pour générer des recommandations personnalisées et pertinentes.
- **Prompt tuning :** le comportement du modèle est ajusté en concevant des formulations d'entrée spécifiques (prompts), sans modifier ses paramètres internes.

Modèles génératifs

Comme GPT, T5 ou LLaMA, ces modèles peuvent produire du texte, générer des recommandations complètes ou simuler des dialogues. Ils sont de plus en plus utilisés pour les systèmes de recommandation conversationnels et explicatifs.

- **Sans tuning (non-tuning) :** L'utilisation du modèle se fait de manière directe, en formulant des prompts adaptés (prompting) ou en fournissant des exemples dans l'entrée (in-context learning), sans nécessiter de réentraînement.
- **Avec tuning :** Cette approche inclut différentes méthodes pour adapter le modèle à la tâche de recommandation : le fine-tuning, le prompt tuning et l'instruction tuning, qui consiste à entraîner le modèle à suivre des instructions explicites, ce qui améliore sa capacité à générer des recommandations cohérentes à partir de consignes générales.

4 Applications des LLMs dans les systèmes de recommandation

E-commerce

Les LLMs contribuent au développement des systèmes de recommandation pour le commerce électronique en améliorant la personnalisation et la pertinence des suggestions de produits. Ces modèles s'appuient sur une compréhension approfondie des noms et descriptions des produits afin de mieux cerner les préférences des utilisateurs, ce qui permet de générer des recommandations directement, sans se limiter au classement basé uniquement sur les évaluations. Cette approche permet de traiter des tâches telles que la classification, la recherche d'information, et la modélisation de motifs complexes dans les interactions utilisateur-produit. Elle renforce ainsi la capacité des modèles linguistiques à interpréter des comportements riches et à transformer l'expérience du commerce électronique grâce à des recommandations plus ciblées et personnalisées (ex. [31, 32]).

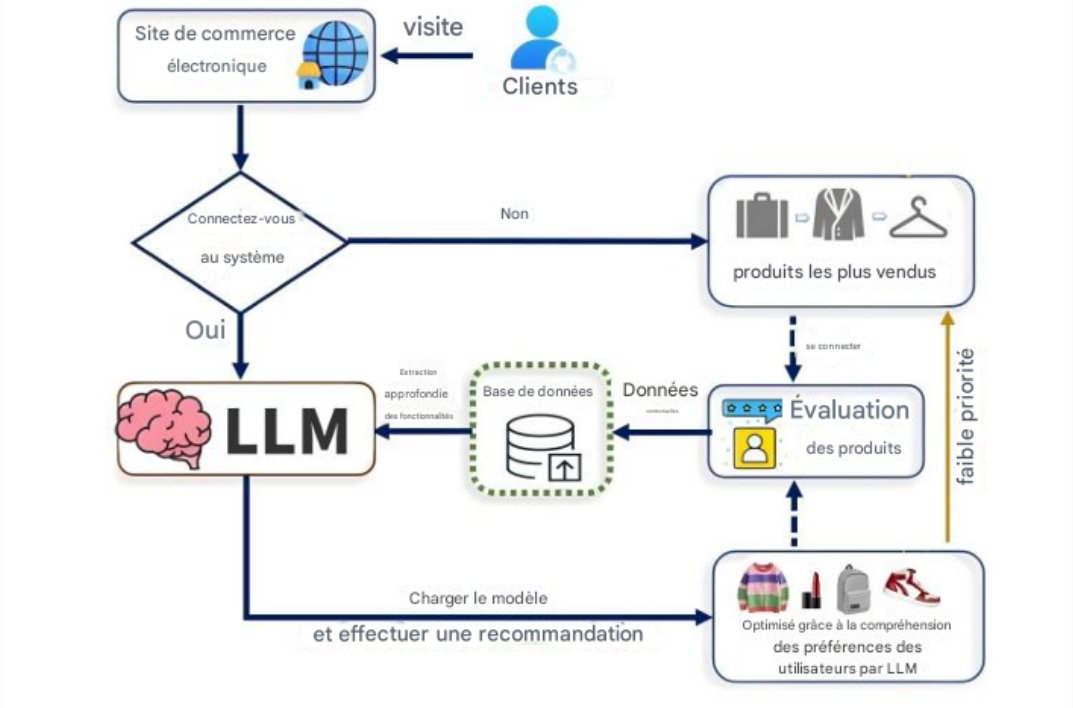


FIG. I.7 : Application du LLM au systéme de recommandation des plateformes de commerce électronique. [32]

Streaming

Dans le domaine du streaming (musique, vidéo), les LLMs sont utilisés pour comprendre le contexte de consommation (mood, préférences implicites, ..) et générer des recommandations plus expressives. Par exemple, certains modèles peuvent suggérer des playlists ou des contenus similaires à partir de requêtes comme *"je veux regarder un film inspirant ce soir"*, en combinant compréhension linguistique et historique utilisateur (ex. [26]).

Réseaux sociaux

Les LLMs jouent un rôle central dans les recommandations sur les réseaux sociaux en interprétant le langage naturel des publications, commentaires ou requêtes utilisateurs. Ils permettent des suggestions personnalisées en temps réel, en s'appuyant sur l'analyse contextuelle, les émotions exprimées et les interactions passées (ex. [36]).

Education

Les LLMs sont de plus en plus utilisés dans le domaine de l'éducation pour personnaliser l'apprentissage et accompagner les enseignants. Un exemple représentatif est l'étude de MA et al. [18], qui explore

l'efficacité des LLMs pour la recommandation de cours adaptés au profil des apprenants.

5 Les obstacles de l'intégration des LLMs dans les systémes de recommandation

- **Inadéquation entre les LLMs et les tâches de recommandation :** Malgré les capacités avancées des LLMs en raisonnement en langage naturel, ils présentent des limites pour modéliser les interactions complexes entre utilisateurs et items, qui sont au cœur des tâches de recommandation. Ce qui résulte une baisse de performance lorsque les LLMs sont appliqués à des tâches classiques de recommandation telles que la recherche, le classement ou la prédiction de notes [1].
- **Problèmes d'évolutivité (Scalability Issues) :** L'intégration des LLMs dans les systèmes de recommandation soulève des problèmes importants en raison de leur forte demande en ressources de calcul. Lorsque le volume de données comportementales utilisateur augmente, le coût et le temps d'inférence deviennent élevés [23].
- **Hallucinations et limitations du contexte d'entrée :** Les LLMs peut donner des recommandations érronées en générant des éléments inexistants, un phénomène connu sous le nom d'hallucination [15], ce qui contient la fiabilité des résultats. De plus, la limitation du nombre de jetons dans les invites (par exemple 2048 pour LLaMA) restreint l'intégration de signaux collaboratifs globaux, réduisant ainsi leur efficacité pour modéliser fidèlement les préférences globales des utilisateurs [23].

6 Études et travaux connexes

Une étude approfondie menée par JIANG et al. [10] améliore les systèmes de recommandation en combinant les MLV et le filtrage collaboratif. L'étude développe trois variantes de RecLM (GPT_KD, Naive, Mask) pour évaluer les techniques de réglage des instructions. En utilisant l'apprentissage par renforcement pour mieux capturer la diversité des préférences et gérer les scénarios zero-shot, elle affine les profils des utilisateurs et des items à partir de leur historique.
Des expériences sur les ensembles de données MIND et Netflix valident la performance du modèle par des comparaisons avec des modèles de référence. RecLM améliore la précision et la diversité des recommandations sur différents modèles. L'étude d'ablation confirme l'efficacité de l'apprentissage par renforcement, rendant RecLM adapté aux données massives et éparses pour des applications réelles. Le modèle RecLM a surpassé le modèle de référence sur trois ensembles de données :

- **MIND :** Rappel de 0.75 et NDCG (Normalized Discounted Cumulative Gain) de 0,80. Le NDCG indique si les éléments les plus utiles sont bien placés en haut de la liste, avec un score compris entre 0 et 1.
- **Netflix :** Rappel de 0.78 et NDCG de 0,82.
- **Industriel :** Rappel de 0.74 et NDCG de 0.79.

Dans l'étude [14], les auteurs proposent une approche hybride combinant le filtrage collaboratif et les LLMs pour améliorer la recommandation. Elle exploite le filtrage collaboratif pour modéliser les préférences des utilisateurs et les LLMs pour mieux comprendre les informations textuelles. Cette approche permet de surmonter les défis du démarrage à froid et de la rareté des données, tout en améliorant la précision et la diversité des recommandations. L'étude développe un système de recommandation hybride combinant le filtrage collaboratif et les LLMs. En utilisant MovieLens (films pertinents) et Amazon Product Review (produits adaptés), elle applique la factorisation matricielle et l'extraction sémantique pour fusionner les scores de recommandation. Les performances sont évaluées selon la précision, le rappel et la satisfaction des utilisateurs. et les résultats montrent que le modèle hybride surpasse le filtrage collaboratif et les LLMs seuls. Par exemple, sur le dataset MovieLens, il atteint une précision de 75,6% , soit une amélioration de +3,3% par rapport au filtrage collaboratif seul (72,3%) et +5,5% par rapport au modèle basé uniquement sur les LLMs (70,1%). La satisfaction des utilisateurs s'est également améliorée, confirmant l'efficacité de cette approche pour des recommandations plus précises et diversifiées.

Dans cette recherche, ZHENG et al. [37] proposent RSLLM, un cadre innovant intégrant les systèmes de recommandation séquentielle aux grands modèles de langage (LLMs). En traitant les interactions utilisateur comme un langage distinct, RSLLM aligne les recommandations traditionnelles sur les capacités des LLMs pour prédire les interactions futures à partir des comportements passés. RSLLM est une mé-

thode d'invitation innovante qui combine des modèles de recommandation classiques avec des caractéristiques textuelles, modélisant ainsi le comportement utilisateur comme un langage. Elle ajuste LLMs en deux étapes : d'abord avec des invitations textuelles, puis en les unifiant avec les connaissances comportementales issues des modèles de recommandation traditionnels. La recherche démontre que le modèle RSLLM surpasse les modèles de référence dans trois bases de données : MovieLens, Steam et LastFM. Sur MovieLens 0.53, 0.52 sur Steam, et 0.5 sur LastFM.

Dans leur étude, Chen et al. [2] s'attachent à renforcer les performances des systèmes de recommandation basés sur les LLMs, en se concentrant spécifiquement sur l'amélioration de leurs capacités de classement. Pour cela, ils proposent la méthode LLM4IDRec pour une modélisation précise des préférences des utilisateurs. L'étude se base sur des ensembles de données provenant de plateformes telles que Yelp et Amazon pour la préparation des données. Le cadre LLM4IDRec encode les préférences des utilisateurs à travers la tokenisation des textes et la formulation de requêtes textuelles. L'efficacité du système est évaluée à l'aide des métriques RecallK et NDCGK, et comparée aux performances de modèles de référence tels que BPR et SimGCL pour démontrer son efficacité à saisir les préférences des utilisateurs. LLM4IDRec améliore de la performance des recommandations, avec une augmentation de 13% sur l'indicateur NDCG et de 7,8% sur l'indicateur de rappel, prouvant ses capacités à améliorer le classement des éléments pertinents et à correspondre aux préférences des utilisateurs.

Dans une autre étude, Xu et al. [33] proposent Dynamic User Intent Prediction (DUIP), un cadre innovant visant à optimiser les systèmes de recommandation en saisissant de manière dynamique l'intention des utilisateurs et en produisant des suggestions d'articles sur mesure. Cette approche est réalisée en remédiant aux insuffisances des méthodes conventionnelles, fréquemment incapables de s'ajuster à l'évolution des goûts des utilisateurs. DUIP est un cadre qui associe des Long Short-Term Memory (LSTM) et des modèles linguistiques à LLMs pour modéliser de manière dynamique l'intention de l'utilisateur. Le LSTM saisit le comportement utilisateur symbiotique et dépendant du temps, tandis que le LLMs exploite les invitations générées pour anticiper l'intérêt de l'utilisateur. Le cadre a été évalué en utilisant trois ensembles de données réels : MovieLens-1M, Amazon Games et Amazon Bundle. Les performances du modèle DUIP attestent de sa capacité à optimiser les systèmes de suggestion. Parmi les critères d'évaluation utilisés, le Hit Rate (HR@k) mesure la proportion de fois où l'élément pertinent apparaît parmi les k premières recommandations. Ces métriques ont été appliquées au jeu de données MovieLens-1M :

- HR@1 : 0,1883
- HR@5 : 0,5348
- NDCG@1 : 0,1883
- NDCG@5 : 0,3674

Ces résultats montrent que DUIP surpasse plusieurs modèles de référence, offrant ainsi des recommandations plus précises et personnalisées basées sur l'intention et le comportement des utilisateurs.

Dans leur étude, YANG et al. [34] visent à optimiser et améliorer les algorithmes de filtrage collaboratif grâce à des LLMs en abordant des défis tels que les coûts de calcul élevés, la rareté des données, les problèmes de démarrage à froid et l'évolutivité. Il utilise des ensembles de données disponibles publiquement comme MovieLens 1M et Netflix Prize ainsi que des techniques d'optimisation telles que la factorisation matricielle et le calcul parallèle et distribué. L'article améliore le filtrage collaboratif dans les LLMs afin d'optimiser la précision et l'évolutivité des recommandations. Sur MovieLens 1M, le modèle hybride diminue le RMSE de 13,2% (0,812 au lieu de 0,935 pour le modèle initial) et l'ANN améliore la durée d'entraînement (26,7 s comparé à 35,6 s). Dans le cadre du Netflix Prize, les modèles parviennent à diminuer le RMSE de 16,2% à 19% en situation de forte sparsité (80%), attestant ainsi de leur performance face aux problématiques liées aux données dispersées et au phénomène du démarrage à froid (cold start).

Dans leur étude, XU, XIAO et CHEN [32] examinent l'utilisation des LLMss pour perfectionner les systèmes de suggestion en e-commerce, en dépassant les contraintes des techniques conventionnelles et en enrichissant l'expérience client ainsi que l'expansion des ventes grâce à des recommandations plus exactes et variées. Cette étude utilise une méthodologie pour collecter le comportement des utilisateurs, les caractéristiques des produits et les informations contextuelles, en utilisant une structure de modèle hybride qui combine LLMs avec des algorithmes existants et évalue ses performances sur la base de données de commerce électronique réelles. L'étude révèle une amélioration notable des performances du modèle de recommandation basé sur les LLMs par rapport aux approches traditionnelles.
Les résultats obtenus sont les suivants :

- **Précision :** augmentation de 0.75% à 0.82%
- **Rappel :** progression de 0.68% à 0.77%
- **F1 Score :** amélioration de 0.71% à 0.79%
- **Taux de clic moyen (CTR) :** hausse de 0.56% à 0.63%
- **Diversité des recommandations :** passant de 0.34% à 0.48%

Dans une autre étude, YANG et al. [35] explorent comment Les systèmes de recommandation basés sur les LLMs faire un lien entre l'espace des éléments (comme les produits) et celui du langage naturel. Les approches précédentes présentent des limites de pertinence et de cohérence des identifiants. Et c'est pour ça, cette étude propose le modèle TransRec, testé avec deux modèles, BART-large et LLaMA-7B, sur trois domaines différents Beauty(produits de beauté – Amazon),Toys(jouets – Amazon), Yelp (restaurants

– Yelp Dataset).
La méthodologie de cette étude tourne autour du développement et de la mise en œuvre du cadre TransRec, un modèle fondé sur l'indexation et la génération à partir des interactions utilisateur.
Les résultats ont été obtenus en utilisant les métriques couramment employées Recall@K et NDCG@K pour évaluer les performances des modèles, avec K fixé à 5 ou 10 :

- **Beauty :** Recall@5 : 0.0504 | Recall@10 : 0.0735 | NDCG@5 : 0.0365 | NDCG@10 : 0.0450
- **Toys :** Recall@5 : 0.0518 | Recall@10 : 0.0764 |NDCG@5 : 0.0360 | NDCG@10 : 0.0420
- **Yelp :** Recall@5 :0.0354 | Recall@10 :0.07457 | NDCG@5 :0.0262 | NDCG@10 :0.0306

L'étude prouve que les facettes (identifiant) et (titre) sont essentielles pour la qualité des recommandations.

Dans l'article de Colangelo et al. [3], L'objectif principal de cette étude est de développer un système automatisé de recommandation de revues scientifiques, en évaluant les performances de différents modèles Sentence Transformer, afin d'identifier les revues les plus appropriées pour un manuscrit donné à partir de son titre et de son résumé.
Les auteurs comparent cinq modèles préentraînés (all-mpnet-base-v2, all-MiniLM-L6-v2, all-MiniLM-L12-v2, multi-qa-distilbert-cos-v1, et all-distilroberta-v1) afin d'évaluer leur capacité d'alignement thématique, leur efficacité computationnelle et la pertinence des recommandations, notamment dans un contexte biomédical.
La méthodologie interroge PubMed pour récupérer jusqu'à 5 000 articles similaires, Ensuite, les titres et résumés de ces articles ainsi que de l'article test sont encodés à l'aide de cinq modèles Sentence Transformer. L'évaluation se base sur la similarité maximale et moyenne, l'entropie de Shannon, le coefficient de Gini, ainsi que sur la présence de la vraie revue de publication dans les recommandations proposées.
Parmi les cinq modèles Sentence Transformers évalués, le modèle all-mpnet-base-v2 s'est révélé être le plus performant. Il atteint une similarité maximale de 0.83 et une similarité moyenne de 0.71, tout en identifiant la revue cible dans le top 10 des recommandations dans 14.5 % des cas.

Dans leur étude, les auteurs Liao et al. [13] proposent LLaRA, un assistant de recommandation basé sur les LLMs, qui intègre à la fois les connaissances comportementales et sémantiques afin d'améliorer la recommandation séquentielle. L'approche repose sur trois éléments clés : des prompts hybrides, un alignement via un adaptateur (adapter), et un entraînement progressif selon une stratégie de curriculum learning.
Les résultats expérimentaux montrent une amélioration notable : en particulier, LLaRA atteint un HitRatio@1 de 0.4421 sur MovieLens, 0.4949 sur Steam, et 0.4508 sur LastFM.

Étude	Titre	Approche	Datasets	Résultats clés
[10]	*RecLM : Recommendation Instruction Tuning*	Réglage des instructions (LLM + CF + RL)	MIND, Netflix, Industrie	Rappel : 0,75–0,78 ; NDCG : 0,79–0,82
[14]	*Enhanced Recommendation Combining Collaborative Filtering and Large Language Models*	Hybridation CF + LLM	MovieLens, Amazon Review	Amélioration : +3,3% CF, +5,5% LLM
[37]	*Towards a Unified Paradigm : Integrating Recommendation Systems as a New Language in Large Models*	RSLLM (Recommandation comme langage)	MovieLens, Steam, LastFM	HitRatio@1 : MovieLens 0.53, Steam 0.52, LastFM 0.5
[2]	*Enhancing ID-based Recommendation with Large Language Models*	LLM4IDRec : Tokenisation + prompts LLM	Yelp, Amazon	Gain NDCG : +13%, Recall : +7,8%
[33]	*Enhancing User Intent for Recommendation Systems via Large Language Models*	DUIP : Intention utilisateur via LSTM + LLM	MovieLens-1M, Amazon Games, Amazon Bundle	HR@1 : 0,1883 ; HR@5 : 0,5348 ; NDCG@5 : 0,3674
[34]	*Optimization and Scalability of Collaborative Filtering Algorithms in Large Language Models*	CF + LLM, optimisation de la factorisation matricielle	MovieLens 1M, Netflix Prize	RMSE réduit de 13,2% (0,812 vs. 0,935)
[32]	*Leveraging Large Language Models to Enhance Personalized Recommendations in E-commerce*	LLM + algorithmes de recommandation + données comportementales	Données e-commerce réelles	Précision : 0,82 ; Rappel : 0,77 ; F1 : 0,79 ; Diversité : 0.48
[35]	*Bridging Items and Language : A Transition Paradigm for Large Language Model-Based Recommendation*	BART-large, LLaMA-7B (basés sur Transformer)	Beauty, Toys, Yelp	Meilleure performance sur Toys (Recall@10 = 0.076, NDCG@10 = 0.042).
[3]	*A Comparative Analysis of SentenceTransformer Models for Automated Journal Recommendation Using PubMed Metadata*	Sentence-BERT (Mpnet, MiniLM, DistilRoBERTa, DistilBERT, multi-qa)	PubMed (articles biomédicaux)	Mpnet : similarité max 0.83, inclusion dans le top 10 des revues : 14.5%. Temps d'encodage plus long (98s).

Étude	Titre	Approche	Datasets	Résultats clés
[13]	*LLaRA : Large Language–Recommendation Assistant*	LLARA	MovieLens, Steam, LastFM	HitRatio@1 de 0.4421 sur MovieLens, 0.4949 sur Steam, et 0.4508 sur LastFM.

TAB. I.1 : Études et travaux connexes

Conclusion

Nous avons exploré dans ce chapitre les bases des systèmes de recommandation, leurs techniques fondamentales, ainsi que leurs limites. Nous avons également abordé les LLMs et leur rôle dans l'amélioration des recommandations en tirant parti des interactions des utilisateurs et du contexte textuel. Cette évolution vers des modèles plus avancés permet de mieux comprendre les préférences des utilisateurs et d'améliorer les recommandations. Dans le chapitre suivant, nous aborderons notre méthodologie de travail, en particulier comment intégrer les LLMs dans notre système de recommandation, ainsi que les différentes étapes de l'entraînement du modèle et l'amélioration de ses performances.

CHAPITRE II

CONCEPTION

Introduction

Ce chapitre présente la méthodologie adoptée pour développer notre système de recommandation basé sur les LLMs. Nous décrivons d'abord les choix liés aux données (sélection, préparation, prétraitement), puis détaillons l'intégration du LLM dans le système, en mettant l'accent sur le fine-tuning.
Cette méthodologie fait le lien entre le cadre théorique du chapitre précédent et les expérimentations à venir.

1 Architecture générale

Dans notre méthodologie (Figure II.1), nous avons commencé par la définition et la sélection de l'ensemble de données, constitué de descriptions de livres accompagnées de leurs métadonnées. Cette première étape a été suivie d'un processus de nettoyage et de prétraitement des données, afin de garantir leur qualité et leur cohérence. Nous avons ensuite utilisé un modèle pré-entraîné performant pour générer automatiquement des résumés à partir des métadonnées, ce qui nous a permis de constituer l'ensemble de données final.
Dans l'étape suivante, nous avons procédé au fine-tuning de plusieurs modèles sur cet ensemble, dans le but d'apprendre des représentations vectorielles (embeddings) spécialisées, mieux adaptées au domaine ciblé (les livres). Ces modèles ont ensuite été évalués à l'aide de métriques spécifiques, afin de mesurer leur capacité à capturer la similarité sémantique entre les métadonnées et les résumés.
Enfin, les modèles fine-tunés ont été utilisés pour encoder l'ensemble des résumés, dans le cadre de la mise en place d'un système de recommandation basé sur la similarité. Ce système vise à suggérer des livres correspondant aux besoins ou préférences exprimés par un utilisateur, en comparant les représentations vectorielles.

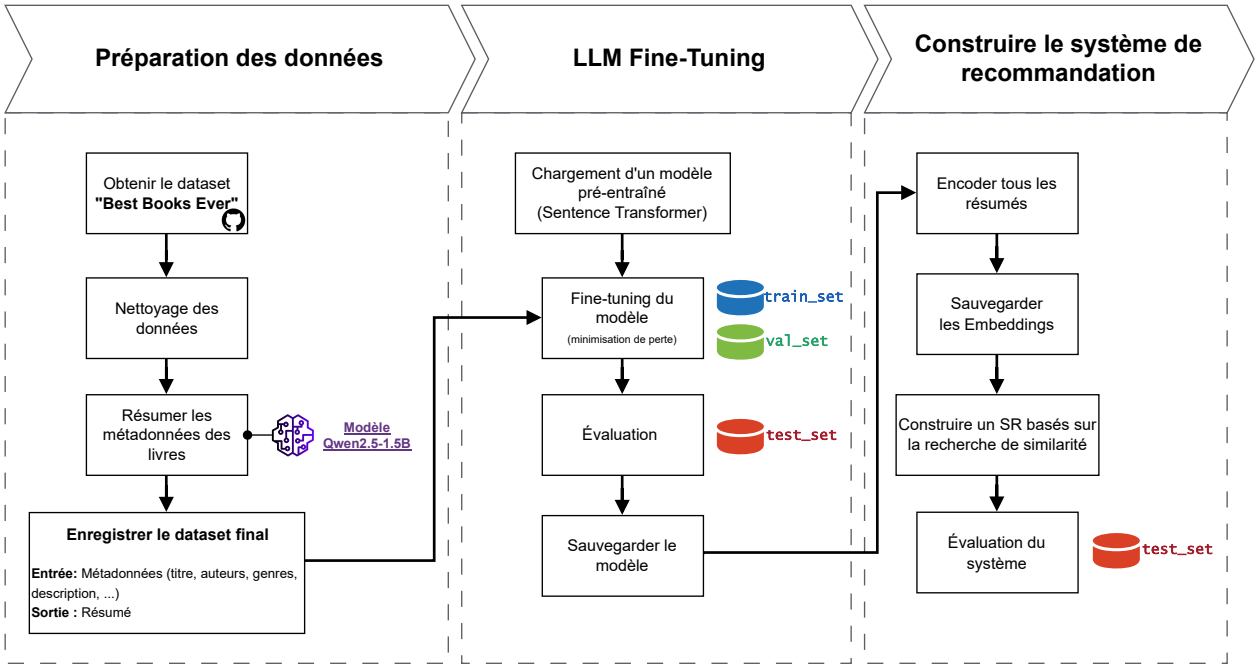


FIG. II.1 : Architecture générale

2 Méthodologie

2.1 Préparation des données

Dans notre travail, cette étape de préparation des données s'assure que les données sont propres, cohérentes et adaptées. Elle joue un rôle crucial avant l'entraînement du modèle, car la qualité des recommandations dépend directement de la qualité des données traitées en amont.

Source des données

L'ensemble de données utilisé dans ce travail concerne des livres et est issu du site Goodreads, une plateforme populaire dédiée aux critiques et aux évaluations littéraires par les utilisateurs. Cet ensemble de données est disponible publiquement sur Zenodo [16] ainsi que sur GitHub [40]. Il contient 52 478 lignes réparties sur 25 colonnes, offrant une richesse d'informations sur chaque livre.
Parmi les colonnes les plus pertinentes pour notre analyse, on peut citer :

- `title` : Titre du livre.
- `author` : Auteur(s) du livre.
- `description` : description du livre.
- `language` : Langue d'écriture du livre.

- `isbn` : Numéro ISBN du livre.
- `genres` : Liste des genres associés au livre.
- `publisher` : Maison d'édition.
- `publishDate` : Date de publication.

Nettoyage des données

Dans notre cas, le nettoyage des données suit plusieurs étapes essentielles. Premièrement, nous supprimons toutes les lignes dupliquées de toutes les colonnes pour éviter les redondances, ce qui pourrait biaiser les résultats du modèle ou attribuer une importance excessive à certains livres.
Après avoir analysé la distribution du nombre de tokens dans les descriptions, nous avons décidé de nettoyer la colonne correspondante en supprimant celles qui sont manquantes, trop courtes ou trop longues. Ce filtrage vise à garantir que chaque description contienne une quantité d'information suffisante sans être trop verbeuse, afin d'en faciliter le traitement par les LLMs.
Enfin, l'anglais a été choisi à la suite d'une analyse des différentes langues disponibles, qui a montré qu'il était prédominant. Un filtrage a ensuite été appliqué pour ne conserver que les livres en anglais, afin d'unifier la langue utilisée et d'assurer la cohérence des recommandations récupérées.

Résumer les métadonnées des livres

La phase de résumé des métadonnées des livres (exemple illustrée dans la Figure II.2) joue un rôle essentiel, car elle permet de standardiser les descriptions souvent longues, hétérogènes ou difficilement exploitables en l'état. Ces résumés structurés facilitent l'analyse sémantique et permettent de représenter chaque livre de manière optimisée et informative pour le fine-tuning d'un LLMs.
Dans cette étape, nous avons utilisé le modèle Qwen [43] (version 2.5-1.5B), développé par Alibaba Cloud, pour générer automatiquement des résumés à partir des métadonnées des livres (titre, auteur, genres, description, éditeur, date de publication).
La méthode consiste d'abord à construire un prompt clair et structuré, présenté sous forme de conversation adaptée aux modèles de type chat. Ce prompt est ensuite tokenisé et envoyé au modèle, qui génère alors un résumé. La réponse du modèle est ensuite extraite et décodée pour ne conserver que le texte généré, ce qui permet d'obtenir des résumés précis, normalisés et adaptés à un traitement automatique à grande échelle.
Nous avons choisi d'utiliser le modèle Qwen-2.5-1.5B, car il est à la fois open source, performant [1], et librement accessible, contrairement à de nombreux modèles propriétaires comme ChatGPT-4.
Cette tâche n'a pas été réalisée manuellement, car elle serait trop chronophage, difficile à standardiser et

[1] Des comparaisons de performance sont disponibles sur le rapport technique [21].

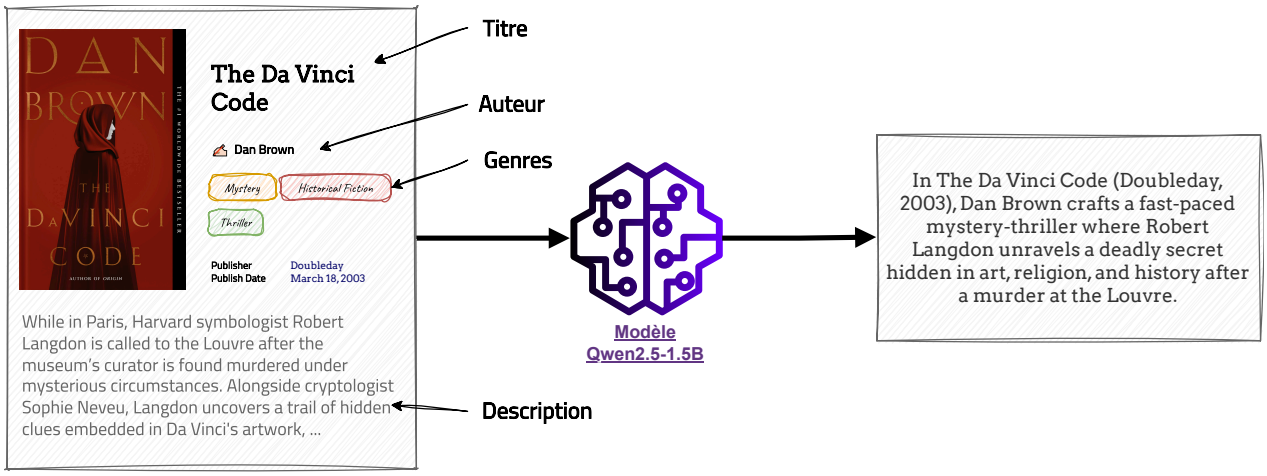


FIG. II.2 : Exemple illustratif du mécanisme de génération de résumé à partir des métadonnées d'un livre.

sujette à la subjectivité. Le modèle Qwen automatise ce processus rapidement, de manière cohérente et reproductible.

À l'issue de cette étape, nous avons constitué un ensemble de données enrichi, composé pour chaque exemple d'une entrée comprenant le titre, l'auteur, les genres, la description, l'éditeur et la date de publication, ainsi que d'une sortie correspondant au résumé généré automatiquement. Le dataset a ensuite été divisée en trois ensembles : 80 % des données ont été réservées à l'entraînement (train set), 10 % à la validation (validation set), et 10 % aux tests (test set).

2.2 LLM Fine-Tuning

Sentence Transformer

Sentence Transformer est un modèle NLP conçu pour produire des représentations numériques de taille fixe (embeddings), à partir de phrases entières, Par contre les approches traditionnelles centrées sur les mots, il capture le sens global d'une phrase, ce qui facilite la compréhension sémantique pour des tâches telles que la comparaison de phrases, la recherche d'informations ou la détection de similarités [39].

Les modèles de type LLMs classiques, comme BERT, nécessitent que deux phrases soient fournies ensemble en entrée pour évaluer leur similarité. Cette méthode devient rapidement très coûteuse en calcul, notamment lorsqu'on travaille avec de grands volumes de données. Par exemple, identifier la paire la plus similaire parmi 10 000 phrases peut prendre jusqu'à 65 heures, en raison du nombre massif de comparaisons à effectuer (près de 50 millions) [22].

Contrairement, Sentence-BERT (SBERT) [22], basé sur l'architecture de BERT, est spécialement conçu pour améliorer ce processus. Grâce à une architecture siamoise ou triplet, SBERT encode chaque phrase indépendamment en un vecteur (embedding) qui capture son sens global. Cela permet ensuite de comparer les phrases très rapidement via des distances vectorielles (cosine similarity), sans avoir à les traiter par

paires à chaque fois.
Ce mécanisme permet d'effectuer des recherches sémantiques rapides, précises et extensibles, même sur de très grands ensembles de données.
Dans notre approche, nous avons utilisé les modèles suivants :

- `all-MiniLM-L6-v2` : variante compacte basée sur MiniLM (6 couches), optimisée pour la similarité sémantique.
- `all-MiniLM-L12-v2` : variante plus grande avec 12 couches, basée sur l'architecture de Microsoft `MiniLM-L12-H384-uncased` .
- `multi-qa-MiniLM-L6-cos-v1` : modèle entraîné pour des tâches de question-réponse multilingues, utilisant une métrique de similarité cosinus.
- `stsb-roberta-base-v2` : basé sur l'architecture RoBERTa, spécialement conçu pour les tâches de similarité textuelle sémantique.
- `DistilBERT` : est une version allégée du modèle BERT, obtenue par une technique de distillation des connaissances. Il génère des vecteurs pour chaque token, mais il ne fournit pas directement une représentation unique de la phrase. Pour cela, une couche de pooling est ajoutée au modèle pour qu'il fonctionne comme un Sentence Transformer.

Fine-tuning des modèles

Dans cette étape de notre travail, nous avons procédé au fine-tuning des modèles en utilisant comme entrée les métadonnées d'un livre (titre, auteurs, genres, description) et comme sortie cible le résumé correspondant. Ce fine-tuning a été réalisé sur un jeu de données composé de paires (métadonnées, résumé), divisées en ensembles d'entraînement et de validation.
L'objectif principal est de minimiser la perte entre l'entrée (métadonnées) et la sortie (résumé), en rapprochant l'embedding des métadonnées de celui du résumé correspondant (du même livre), tout en s'assurant que cet embedding reste bien distinct de ceux des autres résumés.

Évaluation des modèles

Chaque modèle a ensuite été évalué sur les données de test afin de vérifier leur capacité à encoder les métadonnées et les résumés de manière à produire des embeddings proches lorsqu'ils correspondent au même contenu.

Les modèles retenus seront conservés pour l'étape suivante, qui consiste à construire le système de recommandation.

2.3 Construire le système de recommandation

Encodage des résumés

Dans un premier temps, nous encodons l'ensemble des résumés présents dans notre base initiale, incluant les ensembles d'entraînement, de validation et de test, à l'aide du modèle déjà affiné (fine-tuned) lors de l'étape précédente. L'objectif est de simuler un scénario réaliste dans lequel une base de livres complète est déjà disponible. Lorsqu'un utilisateur soumet une requête (par exemple à partir de métadonnées ou d'une description) le système doit être capable d'identifier, parmi tous les livres existants, ceux dont le contenu est le plus similaire, afin de les lui recommander.
Les embeddings des résumés sont calculés une seule fois, puis stockés dans une base vectorielle. Ces représentations vectorielles sont ensuite utilisées pour comparer la requête de l'utilisateur aux résumés disponibles, dans une opération appelée "**similarity search**" (recherche par similarité).

Évaluation du système

Pour évaluer les performances du système, nous utilisons à nouveau le jeu de test, mais selon une approche différente de celle basée uniquement sur la perte (loss). Au lieu de mesurer l'erreur entre l'entrée et la sortie, nous examinons si, pour chaque entrée du test, le système est capable de recommander (parmi les Top-k résultats) le résumé correct ou des résumés sémantiquement similaires.
L'évaluation repose sur plusieurs métriques standard de la recherche d'information :

- **Recall@k**
  - ❝ *Est-ce que le bon livre figure parmi les k recommandations proposées ?*
  - ➜ Cette métrique mesure la proportion de cas où le bon résumé (celui qui est réellement lié aux métadonnées saisies) figure parmi les k premiers résultats suggérés par le système. Un bon score de Recall@k signifie que le système a été capable de recommander le bon livre dans la plupart des cas.
- **MRR@k (Mean Reciprocal Rank)**
  - ❝ *À quelle position moyenne se trouve la première recommandation pertinente ?*
  - ➜ Cette mesure évalue à quel rang le premier résumé pertinent apparaît dans les résultats. Plus cette position est élevée (donc proche du haut de la liste), plus la note est bonne. Un bon MRR@k indique que le bon livre est souvent proposé parmi les premiers résultats, ce qui améliore l'expérience utilisateur.

$$\text{MRR@k} = \frac{1}{|\mathcal{Q}|} \sum_{i=1}^{|\mathcal{Q}|} \frac{1}{\text{rank}_i}$$

- **Cosine Similarity**

  ❝ *Dans quelle mesure la requête et les résumés recommandés sont-ils sémantiquement proches ?*

  ➜ Calcule la proximité sémantique entre l'embedding de la requête et ceux des résumés récupérés.

$$\text{cosine\_similarity}(A, B) = \frac{A \cdot B}{\|A\|\|B\|}$$

Conclusion

Dans ce chapitre, nous avons expliqué toutes les étapes de la conception nécessaires au développement d'un système de recommandation basé sur les LLMss, allant de la préparation des données jusqu'à l'évaluation du système. Dans le chapitre suivant, nous aborderons l'application pratique de cette méthodologie et discuterons des résultats obtenus.

CHAPITRE III

IMPLÉMENTATION

Introduction

Après avoir présenté dans le chapitre précédent sur la méthodologie de notre travail en va explique l'aspect pratique de chaque étape de la conception théorique. Nous y décrivons l'environnement de travail utilisé, les langages de programmation, ainsi que les bibliothèques . Chaque étape de la conception sera illustrée par du code, des visualisations, afin de montrer comment la méthodologie théorique a été traduite en une solution opérationnelle, capable de générer des recommandations pertinentes et personnalisées.

1 Environnement de travail

1.1 googel colab

Google Colaboratory (également connu sous le nom de Colab) est un service cloud basé sur les notebooks Jupyter, utilisé dans l'enseignement et la recherche en intelligence artificielle. Il offre un environnement d'exécution entièrement configuré pour le machine learning et le deep learning, avec un accès à une carte graphique (GPU) haute performance, ce qui le rend particulièrement adapté aux projets dans ces domaines. Intégré à Google Drive, Colab permet de sauvegarder automatiquement ses travaux et de collaborer en temps réel, à la manière de Google Docs.

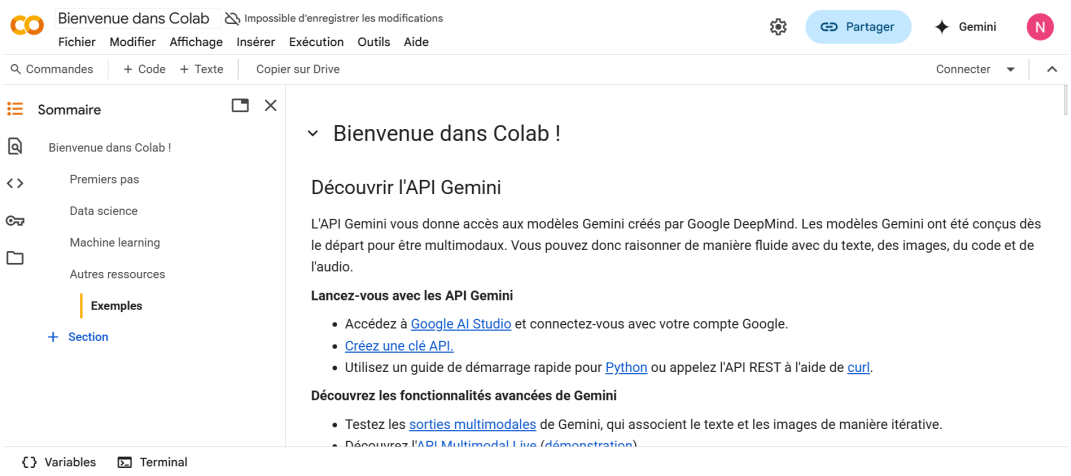


FIG. III.1 : Page principale de Google Colab.

1.2 kaggle

Kaggle est une plateforme en ligne dédiée aux spécialistes de la science des données, de l'apprentissage automatique et de l'apprentissage profond. Elle permet aux utilisateurs de développer des modèles pour résoudre des problèmes spécifiques ou analyser des ensembles de données variés. Kaggle offre un environnement flexible pour l'entraînement des modèles grâce à des ressources gratuites, telles que des GPU (jusqu'à 30 heures par semaine) et des TPU (jusqu'à 20 heures par semaine). En plus de cela, la plateforme propose une communauté active permettant de collaborer sur des projets, de partager du code et des jeux de données, et d'apprendre des travaux réalisés par d'autres membres.

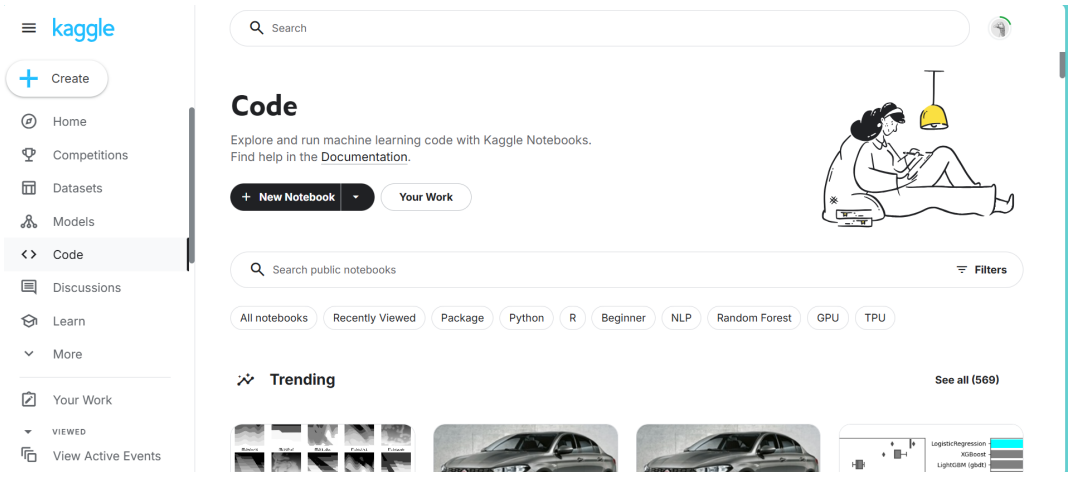


FIG. III.2 : Page d'accueil de la section "Code" de Kaggle.

2 Langage de programmation (Python)

Python est un langage de programmation facile à apprendre et à utiliser. Il intègre des structures de données de haut niveau efficaces et adopte une approche simple, tout en étant interprété et orienté objet. Grâce à sa syntaxe élégante et à son typage dynamique, associés à sa nature interprétée, il constitue un langage idéal pour le scripting et le développement rapide d'applications, dans de nombreux domaines et sur la plupart des plateformes.

3 bibliothèques

3.1 Pandas

Pandas est définie comme une bibliothèque open source de Python qui peut manipuler et analyser des données tabulaires. Elle offre des structures comme DataFrame et Series, qui rendent le nettoyage, la transformation et l'exploration des données faciles. Fonctionnant avec d'autres outils comme NumPy, scikit-learn et matplotlib, Pandas donne un traitement performant des données qui résulte en un code clair et concis. [9]

3.2 Pytorch

PyTorch est définie comme une bibliothèque Python qui a la capacité de l'exécution directe des calculs dynamiques sur des tenseurs, tout en intégrant la différentiation automatique. Elle fait l'accélération matérielle via les GPU. PyTorch donne les mêmes performances que celles des bibliothèques les plus rapides qui sont maintenant utilisées pour l'apprentissage profond.[20]

3.3 Transformers

Transformers est une bibliothèque qui implémente les architectures Transformer. Elle offre des implémentations robustes de modèles populaires, conçues pour être faciles à lire, à étendre et à déployer. La bibliothèque utilise de nombreux modèles préentraînés grâce à un hub centralisé .[29]

3.4 datasets

Datasets est définie comme une bibliothèque qui facilite la découverte et la réutilisation des jeux de données ayant déjà été utilisés pour l'entraînement de ces modèles.[19]

3.5 FAISS

C'est une bibliothéque qui a le principe de la recherche par similarité ou elle doit trouver un équilibre entre plusieurs contraintes et sa repose sur deux techniques principales : la compression des vecteurs et la

recherche non exhaustive. FAISS est utilisée dans l'indexation à très grande échelle, la recherche textuelle, l'extraction de connaissances et la modération de contenu.[5]

4 Implementation

4.1 Préparation des données

Nettoyage des données

La distribution du nombre de tokens dans les descriptions était comme le montre la Figure III.3, la majorité des descriptions contiennent entre 0 et 300 tokens, avec un pic autour de 100 tokens. On observe une diminution progressive du nombre de livres au fur et à mesure que la longueur des descriptions augmente, certaines atteignant et même dépassant 1000 tokens.

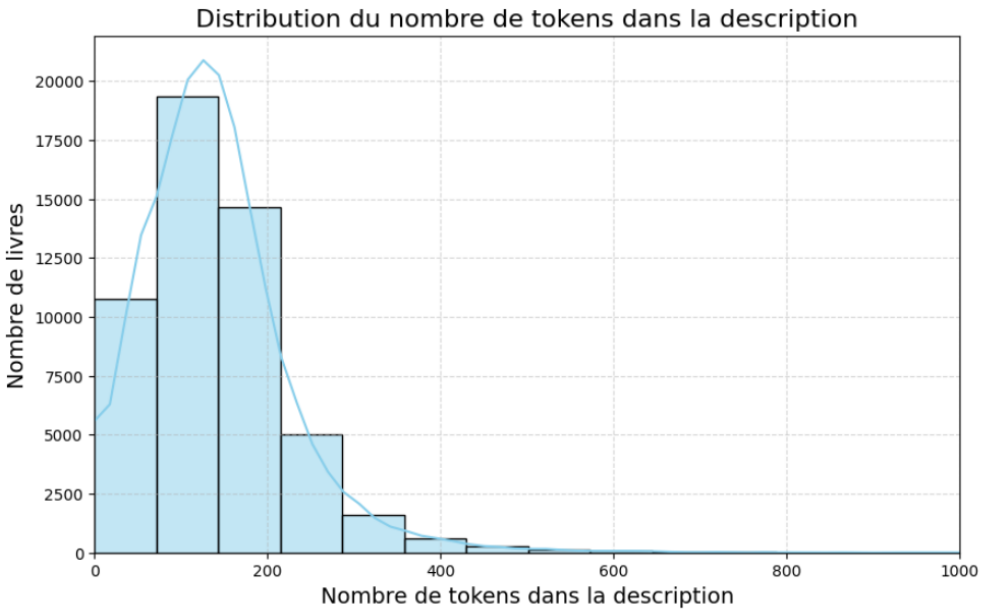


FIG. III.3 : Distribution du nombre de tokens dans la description (0..1000).

Le pourcentage des langues les plus représentées dans le jeu de données est illustré par la Figure III.4. On observe une nette domination de la langue anglaise, qui représente 93.8% des descriptions. Les autres langues par l'arabe (2.3%), l'espagnol (1.5%), le français (1.3%) et l'allemand (1.2%), sont beaucoup moins présentes. Cette forte prédominance de l'anglais montre que le dataset est principalement anglophone.

Les livres ont été filtrés et nettoyés, une partie du processus étant illustrée dans le Code III.1. Cette implémentation utilise la bibliothèque Pandas, incluant la suppression des doublons, le filtrage des descriptions trop courtes ou longues, ainsi que la sélection des livres rédigés en anglais. Le filtrage repose principalement sur les analyses préalables du dataset.

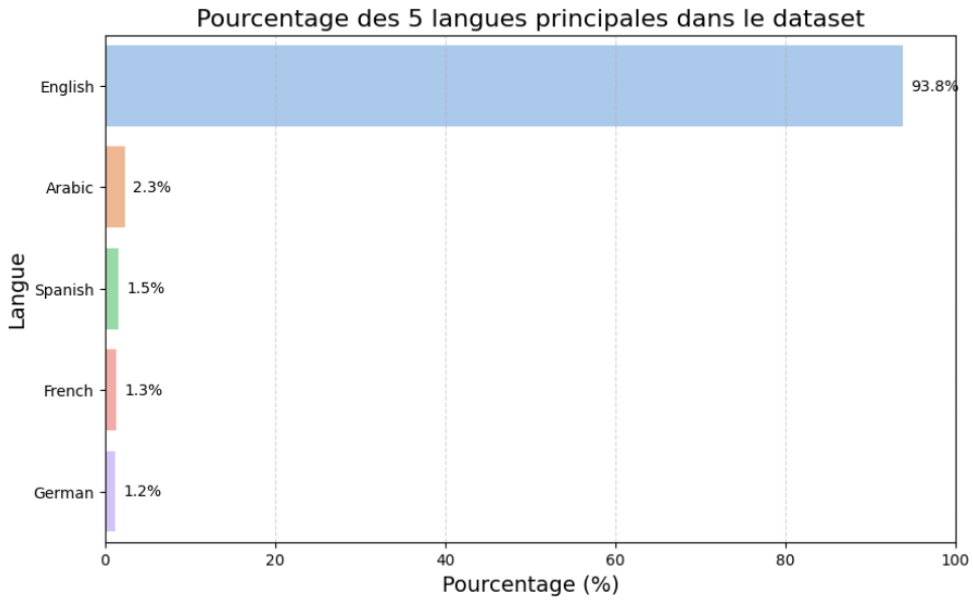


FIG. III.4 : Pourcentage des langues les plus représentées dans le jeu de données.

```python
import pandas as pd

df = pd.read_csv('books_1.Best_Books_Ever.csv')

# Supprimer doublons exacts
df = df.drop_duplicates()

# Supprimer livres avec description manquante ou trop courte/longue
df = df[df['description'].notna()]
df['desc_token_count'] = df['description'].apply(lambda x: len(x.split()))
df = df[(df['desc_token_count'] >= 20) & (df['desc_token_count'] <= 300)]
# Garder uniquement les livres en anglais
df = df[df['language'] == 'English']
```

Code III.1 : Partie de nettoyage des données

Résumer les métadonnées des livres

La phase de résumé des métadonnées des livres, décrite en détail dans le deuxième chapitre, se compose de plusieurs étapes. L'implémentation débute par la sélection des métadonnées pertinentes (telles que le titre, l'auteur, les genres, la description, l'éditeur et la date de publication) suivie de la construction d'un prompt structuré (Code III.2). Ce prompt est élaboré selon des consignes strictes, afin de produire un résumé concis et fidèle, sans ajout d'informations non présentes dans les données d'origine. Ensuite, le

modèle de langage Qwen (chargé comme illustré dans Code III.3) est utilisé pour effectuer la tokenisation du prompt et générer le résumé.

```
def make_prompt(book):
    return f""" You are a precise book summarization assistant.
Based strictly on the metadata provided, write a concise summary of the book in one
↪  or one and a half sentences (no more than 3 short sentences, and under 100
↪  tokens). The summary should explicitly include the listed genres and authors. Do
↪  not invent, infer, or add any details not found in the metadata. Reflect only
↪  what is present, with a focus on the core idea, tone, and themes. Avoid generic
↪  phrases like "This book is about..." and do not copy the description verbatim.
↪  End with a complete, meaningful statement. The summary must not contain any
↪  markdown formatting, titles like "Summary:", emojis, or hashtags. Only generate
↪  plain English text without any decorations.

---
Title: {book['title']}

Authors: {book['author']}

Genres: {book['genres']}

Description: {book['description']}

Publisher: {book['publisher']}

Publish Date: {book['publishDate']}
---

Summary:
"""
```

Code III.2 : Création de prompt pour résumer un livre à partir de ses métadonnées

```
model_name = "Qwen/Qwen2.5-1.5B-Instruct"

model = AutoModelForCausalLM.from_pretrained(
    model_name,
    torch_dtype="auto",
    device_map="auto"
)
tokenizer = AutoTokenizer.from_pretrained(model_name)
tokenizer.padding_side = "left"
```

Code III.3 : Chargement du modèle de langage Qwen2.5-1.5B-Instruct

4.2 Fine-tuning du modèle

Les modèles que nous avons utilisés pour le fine-tuning dans cette phase sur les trois ensembles (train, val et test) issus de l'étape précédente sont des modèles de type SentenceTransformer, dans le but d'apprendre à produire des représentations vectorielles similaires pour des paires de textes de type (`métadonnées` → `résumé`). Par exemple, nous avons utilisé le modèle `all-MiniLM-L6-v2` comme illustré dans le Code III.4.
Le choix de la fonction de perte `MultipleNegativesRankingLoss` s'explique par la nature particulière de nos données : nous ne disposons que de paires positives, c'est-à-dire des couples de textes sémantiquement proches. Ce type de perte est particulièrement adapté dans ce cas, car elle permet d'optimiser l'espace d'embedding en rapprochant les textes similaires (par exemple, métadonnées et résumés correspondants), sans nécessiter de paires négatives explicites [42].

Évaluation du modèle

La classe MeanCosineSimilarityEvaluator (Code III.5) permet d'évaluer les performances de notre modèle en calculant la similarité cosinus moyenne entre deux ensembles de phrases. Dans notre cas, elle mesure la similarité entre les entrées (métadonnées) et les sorties correspondantes (résumés), ce qui reflète la capacité du modèle à générer des embeddings cohérentes pour des paires de textes sémantiquement liées.

```python
# 1. Charger le modèle de transformation de phrases
model_name = "all-MiniLM-L6-v2"
model = SentenceTransformer(f"sentence-transformers/{model_name}")

# 2. Définir la fonction de perte (loss)
loss = MultipleNegativesRankingLoss(model)

# 3. Définir les paramètres d'entraînement
args = SentenceTransformerTrainingArguments(
    num_train_epochs=5,  # Nombre d'époques d'entraînement
    per_device_train_batch_size=32,  # Taille du batch pour l'entraînement
    per_device_eval_batch_size=32,  # Taille du batch pour l'évaluation
    learning_rate=5e-5,  # Taux d'apprentissage
    load_best_model_at_end=True,  # Recharger le meilleur modèle à la fin
    metric_for_best_model="eval_loss",  # Métrique pour sélectionner le meilleur
    ↪  modèle
)

# 4. Définir l'évaluateur basé sur la similarité cosinus moyenne
evaluator = MeanCosineSimilarityEvaluator(
    sentences1=val_dataset["anchor"], # métadonnées
    sentences2=val_dataset["positive"], # résumés
)

# 5. Entraîner le modèle
trainer = SentenceTransformerTrainer(
    model=model,
    args=args,
    train_dataset=train_dataset,
    eval_dataset=val_dataset,
    loss=loss,
    evaluator=evaluator,
)

trainer.train()
```

Code III.4 : Fine-tuning (`all-MiniLM-L6-v2`)

```python
class MeanCosineSimilarityEvaluator(SentenceEvaluator):
    def __init__(self, sentences1, sentences2, name=None):
        # ...

    def __call__(self, model, output_path=None, epoch=-1, steps=-1):
        # Encoder les deux listes de phrases en vecteurs
        embeddings1 = model.encode(self.sentences1, ...)
        embeddings2 = model.encode(self.sentences2, ...)

        # Calcul des similarités cosinus pour chaque paire de phrases
        cosine_similarities = np.sum(embeddings1 * embeddings2, axis=1)
        mean_cosine = np.mean(cosine_similarities)  # Moyenne des similarités

        return float(mean_cosine)
```

Code III.5 : Évaluateur de Similarité Cosinus Moyenne

4.3 Construire le système de recommandation

4.3.1 Encodage des résumés

la construction d'un index de recherche sémantique à l'aide de la bibliothèque FAISS s'effectue en exploitant les représentations vectorielles (embeddings) générées par un de les modèles de SentenceTransformer.

```python
def build_faiss_index(model, outputs):
    output_embeddings = model.encode(outputs, convert_to_numpy=True)

    dim = output_embeddings.shape[1]

    index = faiss.IndexFlatIP(dim)
    index.add(output_embeddings)
    return index, output_embeddings
```

Code III.6 : Construction d'un index FAISS pour la recherche de similarité cosinus

4.3.2 Évaluation du système

Nous évaluons les performances des modèle en utilisant trois métriques essentielles : Recall@k, MRR@k, et similarité cosinus moyenne.

```python
def evaluate_with_faiss(model, test_inputs, test_outputs, all_outputs, faiss_index,
↪  all_output_embeddings, k=10):
    hits = 0
    mrr_total = 0
    sim_total = 0

    for input_text, true_output in tqdm(zip(test_inputs, test_outputs),
    ↪  total=len(test_inputs), disable=False):
        # Encode input query with no progress bar
        query_embedding = model.encode([input_text], convert_to_numpy=True)

        D, I = faiss_index.search(query_embedding, k)
        top_k = [all_outputs[i] for i in I[0]]

        # Recall@k
        if true_output in top_k:
            hits += 1

        # MRR
        try:
            mrr = 1 / (top_k.index(true_output) + 1)
        except ValueError:
            mrr = 0
        mrr_total += mrr

        # Encode true output with no progress bar
        true_emb = model.encode([true_output], convert_to_numpy=True)
        retrieved_embs = all_output_embeddings[I[0]]  # assumed normalized already
        sims = cosine_similarity(true_emb, retrieved_embs)[0]
        sim_total += np.mean(sims)

    recall = hits / len(test_inputs)
    mrr = mrr_total / len(test_inputs)
    sim = sim_total / len(test_inputs)

    return recall, mrr, sim
```

Code III.7 : Évaluation de système

5 Résultats obtenus

5.1 Le dataset

Le dataset final comprend 35163 livres après le nettoyage (Un échantillon de livres est illustré dans la Figure III.5). Chaque livre est défini par plusieurs attributs (le détail est disponible en Annexe A), mais nous nous intéressons uniquement à ceux qui sont pertinents pour notre étude, tels que le titre, les auteurs, les genres, la description, la date de publication, l'éditeur et le résumé.
Un exemple de résumé issu des métadonnées est le suivant :

- **Métadonnées**

  ➜ **Titre :** The Da Vinci Code

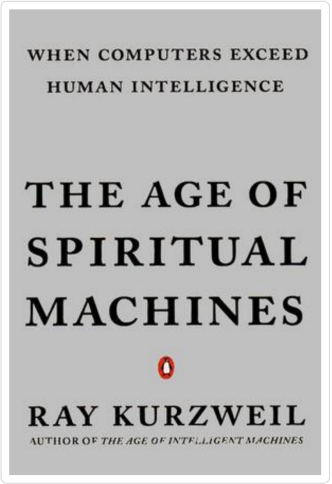
  ➜ **Auteur :** Dan Brown

  ➜ **Genres :** ['Fiction', 'Mystery', 'Thriller', 'Suspense', 'Mystery Thriller', 'Historical Fiction', 'Adventure', 'Novels', 'Crime', 'Adult']

  ➜ **Description :** While in Paris, Harvard symbologist Robert Langdon is awakened by a phone call in the dead of the night. The elderly curator of the Louvre has been murdered inside the museum, his body covered in baffling symbols. As Langdon and gifted French cryptologist Sophie Neveu sort through the bizarre riddles, they are stunned to discover a trail of clues hidden in the works of Leonardo da Vinci—clues visible for all to see and yet ingeniously disguised by the painter.Even more startling, the late curator was involved in the Priory of Sion—a secret society whose members included Sir Isaac Newton, Victor Hugo, and Da Vinci—and he guarded a breathtaking historical secret. Unless Langdon and Neveu can decipher the labyrinthine puzzle—while avoiding the faceless adversary who shadows their every move—the explosive, ancient truth will be lost forever.

- **Résumé**

  ➜ In *The Da Vinci Code*, renowned Harvard symbologist Robert Langdon investigates the murder at the Louvre, uncovering hidden symbols linked to Leonardo da Vinci and a mysterious secret society, while also facing threats from an unknown adversary.



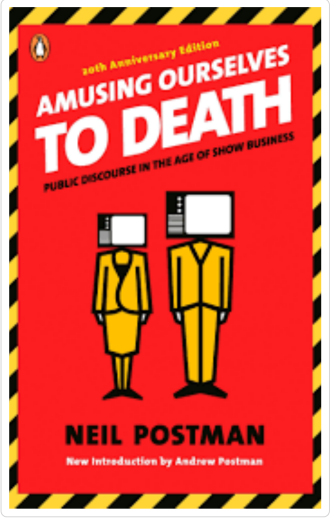
The Age of Spiritual Machines: When Computers Exceed Human Intelligence

Author: Ray Kurzweil

Rating: ⭐ 3.9 | **Pages:** 388 | **Language:** English

Genres: Science | Nonfiction | Technology | Philosophy | Artificial Intelligence | Futurism | Computers | Psychology | Sociology | Computer Science

Summary:
In this groundbreaking work, Ray Kurzweil explores how advancements in artificial intelligence will lead to machines surpassing human cognitive abilities around 2020, transforming society profoundly.



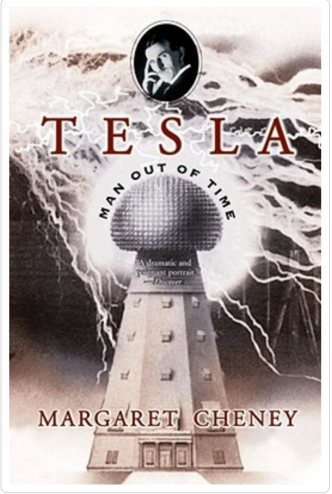
Amusing Ourselves to Death: Public Discourse in the Age of Show Business

Author: Neil Postman, Andrew Postman (Introduction)

Rating: ⭐ 4.14 | **Pages:** 184 | **Language:** English

Genres: Nonfiction | Philosophy | Sociology | Cultural | Politics | Psychology | Technology | Education | History | Society

Summary:
Neil Postman's **"Amusing Ourselves to Death"** explores how television has transformed society into a culture that tolerates superficial entertainment at the expense of critical thinking and meaningful discourse.



Tesla: Man Out of Time

Author: Margaret Cheney

Rating: ⭐ 3.73 | **Pages:** 400 | **Language:** English

Genres: Biography | Nonfiction | Science | History | Biography Memoir | Unfinished | Physics | Technology | Historical | History Of Science

Summary:
Nikola Tesla, a brilliant yet controversial scientist and inventor, revolutionized modern technology through groundbreaking discoveries and visionary experiments, despite being labeled as both a madman and a genius throughout his life.

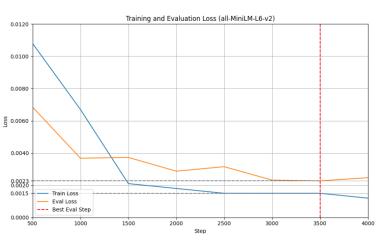
FIG. III.5 : Exemple de dataset (genre : Technlogy)

5.2 Fine-tuning

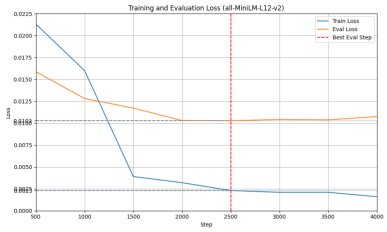
Les résultats montrent que tous les modèles présentent des pertes faibles et des similarités cosinus élevées (Table III.1), ce qui indique une bonne performance sur la tâche de similarité sémantique. Le modèle `all-MiniLM-L6-v2` offre un bon équilibre entre les pertes et la similarité, tandis que `all-MiniLM-L12-v2` affiche des pertes légèrement plus élevées, mais des similarités comparables. Le modèle `multi-qa-MiniLM-L6-cos-v` se distingue par une bonne stabilité et une similarité cosinus élevée, proche de celle du modèle `L12-v2`. Par contre, `stsb-roberta-base-v2` montre les performances les plus faibles, avec des pertes nettement plus élevées et des similarités en retrait, suggérant une adaptation moins efficace. Enfin, `DistilBERT` surpasse tous les autres modèles avec les meilleurs scores de similarité (0.7577 en validation et 0.7594 en test) et des pertes très faibles, confirmant ainsi sa grande capacité de généralisation.

Modèle	**train_loss**	**val_loss**	**val_sim**	**test_loss**	**test_sim**
`all-MiniLM-L6-v2`	0.0015	0.0023	0.7285	0.0028	0.7391
`all-MiniLM-L12-v2`	0.0023	0.0095	0.7442	0.0087	0.7431
`multi-qa-MiniLM-L6-cos-v1`	0.0016	0.0028	0.7452	0.0048	0.7463
`stsb-roberta-base-v2`	0.0072	0.051	0.6908	0.045	0.686
`DistilBERT`	0.0012	0.0034	0.7577	0.0033	0.7594

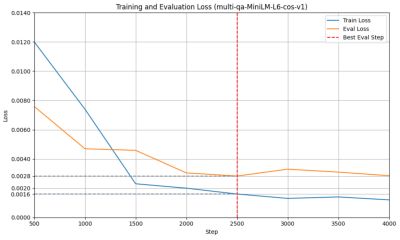
TAB. III.1 : Résultats du fine-tuning des modèles



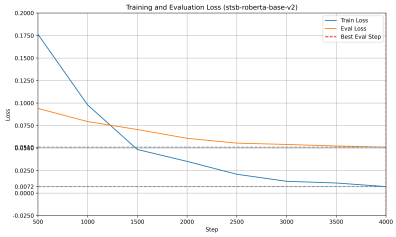
(a) `all-MiniLM-L6-v2`



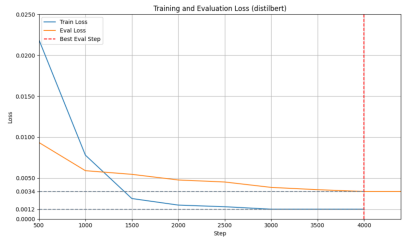
(b) `all-MiniLM-L12-v2`



(c) `multi-qa-MiniLM-L6-cos`



(d) `stsb-roberta-base-v2`



(e) `DistilBERT`

FIG. III.6 : Les pertes d'entraînement et de validation lors du fine-tuning

5.3 Le système de recommandation

Pour ($k = 1$)

Le fine-tuning améliore significativement les performances de l'ensemble des modèles évalués, comme le montre la Table III.2.

- Le modèle `all-MiniLM-L6-v2` bénéficie d'un gain notable : le Recall@1 passe de 0.8703 à **0.9349**, le MRR suit la même progression (de 0.8703 à **0.9349**), tandis que la similarité cosinus grimpe de 0.94 à **0.97**.
- Le modèle `all-MiniLM-L12-v2` enregistre lui aussi des améliorations sensibles avec un Recall@1 qui passe de 0.7574 à **0.8976** et une similarité cosinus de 0.8736 à **0.9503**.
- Le modèle `multi-qa-MiniLM-L6-cos-v1`, déjà performant sans fine-tuning (Recall@1 = 0.8268), voit ses scores atteindre **0.9323** en Recall@1, et **0.9681** en similarité cosinus.
- Pour `stsb-roberta-base-v2`, l'amélioration est particulièrement marquée : le Recall@1 passe de 0.3979 à **0.7352**, et la similarité cosinus progresse de 0.6812 à **0.8627**.
- Enfin, le modèle `DistilBERT`, qui affichait de faibles performances sans fine-tuning (Recall@1 = 0.1251), atteint **0.9261** après adaptation au domaine, avec une similarité cosinus passant de 0.9050 à **0.9649**. Cela s'explique par le fait que sa structure n'est pas conçue pour produire directement des représentations de phrases optimisées pour la comparaison sémantique.

Ces résultats confirment l'importance du fine-tuning pour l'adaptation des modèles aux tâches spécifiques, en particulier dans un contexte de recommandation personnalisée comme ici.

Modèle	**REC**	**MRR**	**COS_SIM**
Sans fine-tuning			
`all-MiniLM-L6-v2`	0.8703	0.8703	0.94
`all-MiniLM-L12-v2`	0.7574	0.7574	0.8736
`multi-qa-MiniLM-L6-cos-v1`	0.8268	0.8268	0.9104
`stsb-roberta-base-v2`	0.3979	0.3979	0.6812
`DistilBERT`	0.1251	0.1251	0.9050
Avec fine-tuning			
`all-MiniLM-L6-v2`	0.9349	0.9349	0.97
`all-MiniLM-L12-v2`	0.8976	0.8976	0.9503
`multi-qa-MiniLM-L6-cos-v1`	0.9323	0.9323	0.9681
`stsb-roberta-base-v2`	0.7352	0.7352	0.8627
`DistilBERT`	0.9261	0.9261	0.9649

TAB. III.2 : Performances du système dans la recommandation d'un livre

Pour ($k = 5$)

Le passage à $k = 5$ améliore nettement les performances de tous les modèles, en particulier après fine-tuning (Table III.3). Les modèles deviennent plus robustes et cohérents dans leurs recommandations, même ceux initialement moins performants. Les différences entre les modèles sont moins marquées, et les réponses sont mieux classées. La légère baisse de la similarité cosinus suggère une priorité donnée à la pertinence plutôt qu'à la distance vectorielle.

Modèle	**REC**	**MRR**	**COS_SIM**
Sans fine-tuning			
`all-MiniLM-L6-v2`	0.9571	0.9062	0.613
`all-MiniLM-L12-v2`	0.8811	0.807	0.5935
`multi-qa-MiniLM-L6-cos-v1`	0.9263	0.8676	0.5809
`stsb-roberta-base-v2`	0.5193	0.4447	0.5395
`DistilBERT`	0.2261	0.1628	0.8966
Avec fine-tuning			
`all-MiniLM-L6-v2`	0.9935	0.9604	0.5409
`all-MiniLM-L12-v2`	0.977	0.9315	0.5453
`multi-qa-MiniLM-L6-cos-v1`	0.9883	0.9567	0.5434
`stsb-roberta-base-v2`	0.8558	0.7832	0.5461
`DistilBERT`	0.9866	0.9523	0.5717

TAB. III.3 : Performances du système dans la recommandation de 5 livres

Pour ($k = 10$)

En augmentant le k à 10, le système atteint des performances proches de la perfection (Table III.4), offrant une liste de livres où les bonnes réponses sont presque toujours présentes.

Modèle	**REC**	**MRR**	**COS_SIM**
Sans fine-tuning			
`all-MiniLM-L6-v2`	0.9716	0.9083	0.5476
`all-MiniLM-L12-v2`	0.909	0.8107	0.5338
`multi-qa-MiniLM-L6-cos-v1`	0.9477	0.8704	0.5144
`stsb-roberta-base-v2`	0.564	0.4506	0.5067
`DistilBERT`	0.2787	0.1698	0.894
Avec fine-tuning			
`all-MiniLM-L6-v2`	0.9969	0.9609	0.4512
`all-MiniLM-L12-v2`	0.9861	0.9328	0.4581
`multi-qa-MiniLM-L6-cos-v1`	0.9954	0.9578	0.4546
`stsb-roberta-base-v2`	0.8919	0.7881	0.476
`DistilBERT`	0.9915	0.9530	0.4873

TAB. III.4 : Performances du système dans la recommandation de 10 livres

Test personnalisé du système (`all-MiniLM-L6-v2`)

Pour évaluer les capacités du système à générer des recommandations pertinentes, un test personnalisé a été effectué à partir d'un prompt spécifique (Figure III.8) : ***"I'm interested in exploring the mechanisms behind how the brain operates"***. Ce test montre que le système est capable de comprendre des requêtes complexes et d'identifier la recherche des livres les plus similaires dans un index d'embeddings.

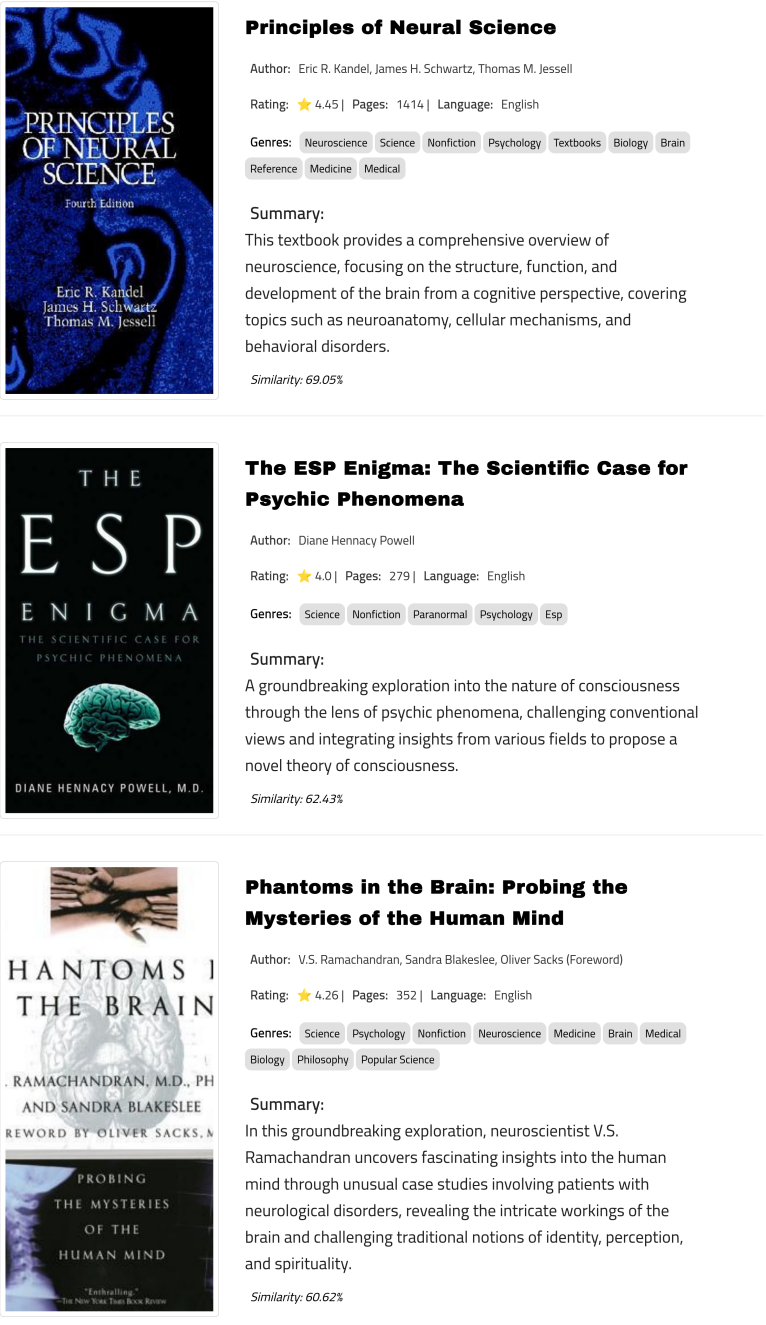


FIG. III.7 : Test personnalisé (`all-MiniLM-L6-v2`)

5.4 Interface de notre système

Une figure décrivant l'interface utilisateur de notre système de recommandation de livres, basé sur une description textuelle, avec sélection du nombre de résultats et soumission de la requête.

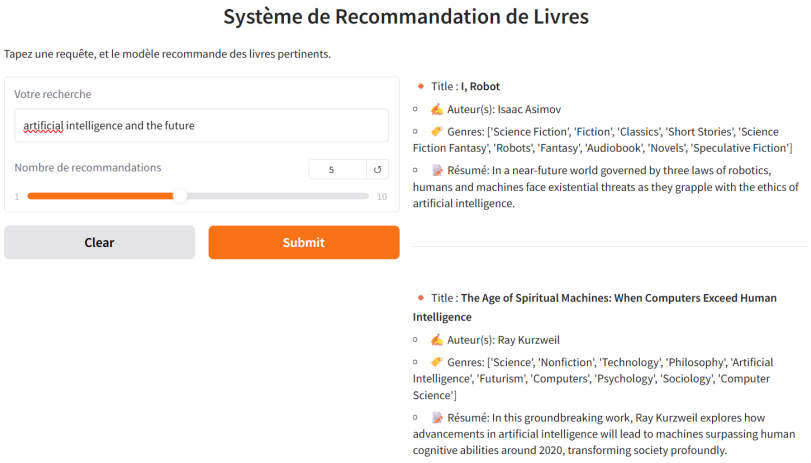


FIG. III.8 : Interface utilisateur du système de recommandation de livres basé sur la description de livre

Conclusion

Le chapitre III présente l'implémentation pratique de notre système de recommandation basé sur les LLMs. Chaque étape de la méthodologie a été illustrée par des extraits de code. Ces résultats ouvrent également des perspectives intéressantes pour l'application de notre système à d'autres domaines.

CONCLUSION GÉNÉRALE

La croissance rapide du volume des données numériques, en particulier textuelles, soulève d'importants défis en matière de filtrage et de personnalisation de l'information. Bien que les systèmes de recommandation traditionnels aient démontré une certaine efficacité, ils révèlent leurs limites face à la diversité des contenus et à la nécessité d'une compréhension fine du langage naturel. Dans ce contexte, l'émergence des LLM constitue une avancée majeure, en ouvrant la voie à la conception de systèmes plus intelligents, capables de générer des textes précis et expressifs, tout en contextualisant les préférences des utilisateurs.

L'objectif principal de ce travail est d'explorer l'intégration des LLMs dans un système de recommandation, en s'appuyant sur leur capacité à produire des représentations sémantiques riches. Pour ce faire, plusieurs étapes méthodologiques ont été suivies. Tout d'abord, un nettoyage du dataset de livres a été effectué à partir des métadonnées disponibles. Ensuite, des résumés pertinents ont été générés à l'aide d'un LLM afin d'enrichir les descriptions textuelles. Plusieurs modèles de type Sentence-Transformer ont ensuite été fine-tunés, notamment `all-MiniLM-L6-v2`, `all-MiniLM-L6-v2`, `multi-qa-MiniLM-L6-cos-v1`, `stsb-roberta-base-v2` et `DistilBERT`, afin d'apprendre à modéliser la similarité sémantique entre les métadonnées et les résumés générés. Enfin, les performances du système ont été évaluées en termes de qualité de recommandation, en mesurant la pertinence des suggestions proposées.

Notre système a démontré une performance remarquable (surtout après le fine-tuning) selon les critères d'évaluation mentionnés précédemment. En fixant le nombre de suggestions à dix (k=10), les résultats ont été quasiment parfaits : le modèle `all-MiniLM-L6-v2`, qui a obtenu les meilleurs résultats (de manière marginale), a atteint un score de rappel (Recall) de 99.69% pour k=10, 99.35% pour k=5, et 93.49% pour k=1.

Les principaux défis identifiés lors du développement :

- La génération de résumés parfois trop brefs ou redondants, en fonction de la richesse des métadonnées d'entrée.
- Le coût computationnel élevé lié au fine-tuning et à la génération, même avec des GPU, limitant les possibilités d'expérimentation à grande échelle.

- L’absence de données réelles d’utilisateurs (clics, avis, profils) restreint les recommandations au contenu textuel seul, sans prise en compte du comportement utilisateur.
- L’évaluation centrée uniquement sur la similarité sémantique, sans retour d’utilisateurs finaux, ce qui limite la portée pratique du système dans un contexte réel.

Perspectives d’amélioration :

- Étendre le système vers une recommandation conversationnelle, où l’utilisateur interagit en langage naturel.
- Intégrer des données comportementales (clics, historique) pour combiner contenu et usage.
- Étudier l’apport de la génération explicative à l’aide des LLM pour renforcer la transparence et la confiance dans le système.
- Recourir à des techniques comme le prompt tuning ou l’in-context learning pour éviter un fine-tuning complet.

Ce mémoire a démontré l’efficacité des modèles de langage de grande taille dans la structuration et la comparaison de contenus textuels au sein d’un système de recommandation. Il propose un prototype complet, fondé sur le langage naturel, qui prépare le terrain pour des recherches futures en vue de recommandations plus intelligentes et personnalisées.

BIBLIOGRAPHIE

[1] Yuwei Cao et al. « Aligning Large Language Models with Recommendation Knowledge ». In : *arXiv preprint arXiv :2404.00245* (2024). URL : https://arxiv.org/abs/2404.00245.

[2] Lei Chen et al. « Enhancing ID-based Recommendation with Large Language Models ». In : *arXiv preprint arXiv :2411.02041* (2024). URL : https://arxiv.org/abs/2411.02041.

[3] Maria Teresa Colangelo et al. *A Comparative Analysis of Sentence Transformer Models for Automated Journal Recommendation Using PubMed Metadata*. 2025. DOI : 10.20944/preprints202501.1334.v1. URL : https://www.preprints.org/manuscript/202501.1334/v1.

[4] Jacob Devlin et al. « BERT : Pre-training of Deep Bidirectional Transformers for Language Understanding ». In : *arXiv preprint arXiv :1810.04805* (2018). URL : https://arxiv.org/abs/1810.04805.

[5] Matthijs Douze et al. « The Faiss library ». In : *arXiv preprint arXiv :2401.08281* (fév. 2025). Version 3. DOI : 10.48550/arXiv.2401.08281. URL : https://arxiv.org/abs/2401.08281.

[6] Luke Friedman et al. *Leveraging Large Language Models in Conversational Recommender Systems*. 2023. arXiv : 2305.07961 [cs.IR]. URL : https://arxiv.org/abs/2305.07961.

[7] Swathi Mirthika G. L. et B. Sivakumar. « Enhancing Hybrid Filtering for Evolving Recommender Systems : Dealing with Data Sparsity and Cold Start Problems ». In : *2024 Second International Conference on Inventive Computing and Informatics (ICICI)*. Bangalore, India : IEEE, juin 2024, p. 141-146. ISBN : 9798350373295. DOI : 10.1109/ICICI62254.2024.00033. URL : https://ieeexplore.ieee.org/document/10675674/.

[8] Ruiyan Gao. « Exploring the Landscape of Recommendation Systems : A Qualitative Analysis of Types, Challenges, and Potential Solutions ». In : *Applied and Computational Engineering* 64 (2024), p. 216-221. DOI : 10.54254/2755-2721/64/20241444. URL : https://www.ewadirect.com/proceedings/ace/article/view/13134.

[9] Prateek GUPTA et Arijit BAGCHI. « Introduction to Pandas ». In : *Essentials of Python for Artificial Intelligence and Machine Learning*. Springer, Cham, 2024, p. 161-196. DOI : 10.1007/978-3-031-43725-0_5. URL : https://doi.org/10.1007/978-3-031-43725-0_5.

[10] Yangqin JIANG et al. « RecLM : Recommendation Instruction Tuning ». In : *arXiv preprint arXiv :2412.19302* (2024). DOI : 10.48550/arXiv.2412.19302. URL : https://arxiv.org/abs/2412.19302.

[11] Anton KORIKOV et al. *Large Language Model Driven Recommendation*. 2024. DOI : 10.48550/ARXIV.2408.10946. URL : https://arxiv.org/abs/2408.10946.

[12] Praveen KUMAR et al. « A Comparative Analysis of Collaborative Filtering Similarity Measurements for Recommendation Systems ». In : *International Journal on Recent and Innovation Trends in Computing and Communication* 11.3s (mars 2023), p. 184-192. ISSN : 2321-8169. DOI : 10.17762/ijritcc.v11i3s.6180. URL : https://ijritcc.org/index.php/ijritcc/article/view/6180.

[13] Jiayi LIAO et al. « LLaRA : Large Language–Recommendation Assistant ». In : *arXiv preprint arXiv :2312.02445* (2023). URL : https://arxiv.org/abs/2312.02445.

[14] Xueting LIN et al. « Enhanced Recommendation Combining Collaborative Filtering and Large Language Models ». In : *arXiv preprint arXiv :2412.18713* (2024). DOI : 10.48550/arXiv.2412.18713. URL : https://arxiv.org/abs/2412.18713.

[15] Junling LIU et al. « Is ChatGPT a Good Recommender ? A Preliminary Study ». In : *arXiv preprint arXiv :2304.10149* (2023). URL : https://arxiv.org/abs/2304.10149.

[16] Lorena Casanova LOZANO et Sergio Costa PLANELLS. *Best Books Ever Dataset*. Version 1.0.0. Zenodo, nov. 2020. DOI : 10.5281/zenodo.4265096. URL : https://doi.org/10.5281/zenodo.4265096.

[17] Sebastian LUBOS et al. « LLM-generated Explanations for Recommender Systems ». In : *Adjunct Proceedings of the 32nd ACM Conference on User Modeling, Adaptation and Personalization*. Cagliari Italy : Association for Computing Machinery, 2024, p. 276-285. ISBN : 9798400704666. DOI : 10.1145/3631700.3665185. URL : https://dl.acm.org/doi/10.1145/3631700.3665185.

[18] Boxuan MA et al. *How Good Are Large Language Models for Course Recommendation in MOOCs ?* 2025. DOI : 10.48550/arXiv.2504.08208. URL : http://arxiv.org/abs/2504.08208.

[19] Angelina MCMILLAN-MAJOR et al. « Reusable Templates and Guides For Documenting Datasets and Models for Natural Language Processing and Generation : A Case Study of the HuggingFace and GEM Data and Model Cards ». In : *Proceedings of the 1st Workshop on Natural Language Generation, Evaluation, and Metrics (GEM 2021)*. Sous la dir. d'Antoine BOSSELUT et al. Online : Association for Computational Linguistics, août 2021, p. 121-135. DOI : 10.18653/v1/2021.gem-1.11. URL : https://aclanthology.org/2021.gem-1.11/.

[20] Adam Paszke et al. « PyTorch : An Imperative Style, High-Performance Deep Learning Library ». In : *Advances in Neural Information Processing Systems*. Sous la dir. de H. Wallach et al. T. 32. Curran Associates, Inc., 2019. URL : https://arxiv.org/abs/1912.01703.

[21] Qwen et al. *Qwen2.5 Technical Report*. 2025. arXiv : 2412.15115 [cs.CL]. URL : https://arxiv.org/abs/2412.15115.

[22] Nils Reimers et Iryna Gurevych. « Sentence-BERT : Sentence Embeddings using Siamese BERT-Networks ». In : *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, nov. 2019. URL : https://arxiv.org/abs/1908.10084.

[23] Xiaoxue Ren et al. « Representation Learning with Large Language Models for Recommendation ». In : *Proceedings of the ACM Web Conference 2024 (WWW '24)*. Association for Computing Machinery, 2024, p. 3464-3475. DOI : 10.1145/3589334.3645458. URL : https://dl.acm.org/doi/10.1145/3589334.3645458.

[24] Dziky Ridhwanullah, Yovita Kinanti Kumarahadi et Bayu Dwi Raharja. « Content-Based Filtering pada Sistem Rekomendasi Buku Informatika ». In : *Jurnal Ilmiah SINUS* 22.2 (juill. 2024), p. 57-66. ISSN : 2548-4028. DOI : 10.30646/sinus.v22i2.840. URL : https://p3m.sinus.ac.id/jurnal/index.php/e-jurnal_SINUS/article/view/840.

[25] Garvit Sharma et al. « Research Paper on Exploring the Landscape of Recommendation Systems : A Comparative Analysis of Techniques and Approaches ». In : *International Journal of Engineering and Computer Science* 13 (juin 2024), p. 26196-26218. DOI : 10.18535/ijecs/v13i06.4827.

[26] Ruixuan Sun et al. *Large Language Models as Conversational Movie Recommenders : A User Study*. 2024. URL : https://arxiv.org/abs/2404.19093v1.

[27] Mitat Uysal, Aynur Uysal et Yasemin Karagül. « A Comprehensive Overview of Large Language Models (LLMs) ». In : *International Journal for Multidisciplinary Research (IJFMR)* 7.1 (2025). URL : https://www.ijfmr.com/papers/2025/1/34609.pdf.

[28] Ashish Vaswani et al. « Attention is all you need ». In : *Advances in neural information processing systems* 30 (2017).

[29] Thomas Wolf et al. « Transformers : State-of-the-Art Natural Language Processing ». In : *arXiv preprint arXiv :1910.03771* (2020). URL : https://arxiv.org/abs/1910.03771.

[30] Likang Wu et al. *A Survey on Large Language Models for Recommendation*. 2024. DOI : 10.48550/arXiv.2305.19860. URL : http://arxiv.org/abs/2305.19860.

[31] Nuofan Xu et Chenhui Hu. *Enhancing E-Commerce Recommendation using Pre-Trained Language Model and Fine-Tuning*. 2023. DOI : 10.48550/arXiv.2302.04443. URL : http://arxiv.org/abs/2302.04443.

[32] Wei XU, Jue XIAO et Jianlong CHEN. *Leveraging Large Language Models to Enhance Personalized Recommendations in E-commerce*. 2024. DOI : 10.48550/arXiv.2410.12829. URL : http://arxiv.org/abs/2410.12829.

[33] Xiaochuan XU et al. « Enhancing User Intent for Recommendation Systems via Large Language Models ». In : *Preprints* 202501.0627 (2025). DOI : 10.20944/preprints202501.0627.v1. URL : https://www.preprints.org/manuscript/202501.0627/v1.

[34] Haowei YANG et al. « Optimization and Scalability of Collaborative Filtering Algorithms in Large Language Models ». In : *arXiv preprint arXiv :2412.18715* (2024). DOI : 10.48550/arXiv.2412.18715. URL : https://arxiv.org/abs/2412.18715.

[35] Zhengyi YANG et al. « Bridging Items and Language : A Transition Paradigm for Large Language Model-based Recommender Systems ». In : *Proceedings of the 17th ACM International Conference on Web Search and Data Mining (WSDM '24)*. Association for Computing Machinery, 2024, p. 1816-1826. DOI : 10.1145/3637528.3671884. URL : https://doi.org/10.1145/3637528.3671884.

[36] Bowen ZHENG et al. *A Large Language Model Enhanced Sequential Recommender for Joint Video and Comment Recommendation*. 2024. DOI : 10.48550/arXiv.2403.13574. URL : http://arxiv.org/abs/2403.13574.

[37] Kai ZHENG et al. « Towards a Unified Paradigm : Integrating Recommendation Systems as a New Language in Large Models ». In : *arXiv preprint arXiv :2412.16933* (2024). URL : https://arxiv.org/abs/2412.16933.

WEBOGRAPHIE

[38] *Concevoir un système de recommandation — uluumy.com*. https://www.uluumy.com/courses/Concevoir-un-systeme-de-recommandation. (Visité le 05/03/2025).

[39] GeeksforGeeks. *Sentence Transformer*. 2025. URL : https://www.geeksforgeeks.org/sentence-transformer/ (visité le 05/06/2025).

[40] *GitHub - scostap/goodreads_bbe_dataset : GoodReads Best Books Ever dataset repository — github.com*. https://github.com/scostap/goodreads_bbe_dataset. (Visité le 03/06/2025).

[41] *Les systèmes de recommandation : une catégorisation - Interstices — interstices.info*. https://interstices.info/les-systemes-de-recommandation-categorisation/. (Visité le 14/04/2025).

[42] *Losses — Sentence Transformers documentation*. URL : https://sbert.net/docs/package_reference/sentence_transformer/losses.html#multiplenegativesrankingloss (visité le 19/06/2025).

[43] Qwen Team. *Qwen2.5: A Party of Foundation Models*. Sept. 2024. URL : https://qwenlm.github.io/blog/qwen2.5/ (visité le 09/06/2025).

[44] *Transformer Architectures - Hugging Face LLM Course — huggingface.co*. https://huggingface.co/learn/llm-course/chapter1/6. (Visité le 11/06/2025).

ANNEXE A : DESCRIPTION DES ATTRIBUTS DU DATASET

- `bookId` : Identifiant unique du livre (souvent un mélange ID-titre).
- `title` : Titre du livre.
- `series` : Nom de la série (si le livre fait partie d'une série).
- `author` : Auteur(e) du livre.
- `rating` : Note moyenne attribuée au livre.
- `description` : description du livre.
- `language` : Langue d'écriture du livre.
- `isbn` : Numéro ISBN du livre.
- `genres` : Liste des genres associés au livre.
- `characters` : Personnages principaux du livre.
- `bookFormat` : Format du livre (ex. Hardcover, Paperback, eBook, etc.).
- `edition` : Édition du livre.
- `page` : Nombre de pages.
- `publisher` : Maison d'édition.
- `publishDate` : Date de publication.

- `firstPublishDate` : Première date de publication.
- `Awards` : Prix littéraires remportés ou nominations.
- `numRatings` : Nombre total de votes.
- `ratingsByStars` : Répartition des notes par étoiles (format string de liste).
- `likedPercent` : Pourcentage de lecteurs ayant aimé le livre.
- `setting` : Lieux où se déroule l'action.
- `coverImg` : URL de l'image de couverture du livre.
- `bbeScore` : Score sur BookBub Editors
- `bbeVotes` : Nombre de votes associés au score BookBub.
- `price` : Prix du livre.