Master 2 : Ingénierie Mathématique Appliquée aux Sciences des Données

MÉMOIRE DE STAGE

présenté et soutenu par

Gérard KRA

le 03 septembre 2025

Mise en place d'un système de recommandation de produits bancaires basé sur l'apprentissage automatique.

Tuteur de stage : **Téahio Boris Arnaud Djadja OURAGA**
Contact universitaire : **Stephane DESCOMBES**

Direction des Systèmes d'Information (DSI)
Service Data & Architecture
Plateau avenue Joseph Anoma, rue des Banques,
01 BP 670 Abidjan 01

Remerciements

Je souhaite exprimer ma sincère gratitude à toutes les personnes qui m'ont accompagné et soutenu durant ce stage de Master 2, tant sur le plan professionnel que personnel.

Je remercie tout particulièrement monsieur Djadja OURAGA, mon tuteur au sein de la BNI, pour sa disponibilité, ses conseils avisés et l'encadrement de qualité dont j'ai bénéficié. Son expertise, sa pédagogie et sa patience ont été déterminantes dans la réussite de ce stage. Je lui suis reconnaissant pour le temps qu'il m'a consacré, notamment pour m'avoir transmis ses connaissances en bases de données et en SQL, ainsi que pour ses explications précieuses sur les scripts d'extraction des données depuis la base ORION.

Je remercie également toute l'équipe de la DSI pour leur accueil chaleureux et leur esprit d'équipe, qui ont grandement facilité mon intégration. Je tiens à exprimer une reconnaissance particulière à monsieur Moussa OUATTARA pour la qualité de nos échanges professionnels et son encouragement tout au long de mon stage.

Je tiens à remercier monsieur Stephane DESCOMBES, mon tuteur universitaire, pour son suivi attentif tout au long de l'année, ses conseils pertinents, ainsi que son accompagnement lors de ma recherche de stage.

Enfin, je remercie l'ensemble des enseignants de l'Université Côte d'Azur et de Polytech Nice-Sophia pour la qualité de leur enseignement et le soutien apporté durant mes deux années de Master.

Résumé

Dans le cadre de ma dernière année de Master 2 « Ingénierie Mathématique pour les Sciences de Données » (IM MSD), j'ai effectué un stage de plusieurs mois au sein de la Direction des Systèmes d'Information (DSI) de la Banque Nationale D'Investissement (BNI) de Côte d'Ivoire. L'objectif principal de ce stage était double : d'une part, participer à l'optimisation des processus de reporting en automatisant certains tableaux de bord existants ; d'autre part, contribuer à la conception d'un système de recommandation basé sur des techniques de Machine Learning.

Mes travaux ont d'abord porté sur l'amélioration des reportings internes, à travers l'utilisation d'outils comme Power BI, SQL ou encore Excel. Cette mission m'a permis de mieux comprendre les besoins métiers et d'adapter les visualisations en fonction des demandes spécifiques.

Dans un second temps, j'ai été impliqué dans un projet plus technique visant à développer un moteur de recommandation personnalisé. Ce projet m'a permis de mobiliser des compétences avancées en Python et en analyse exploratoire de données.

Ce stage m'a permis de développer à la fois mes compétences techniques (programmation, modélisation, visualisation de données) et mes compétences transversales (rigueur, autonomie, travail en équipe). Il représente une étape essentielle dans mon parcours professionnel.

Table des matières

Introduction

À l'ère du numérique, les utilisateurs sont confrontés à une surabondance d'informations qui complique l'identification des contenus réellement pertinents. Pour répondre à ce défi, les systèmes de recommandation se sont imposés comme des outils essentiels permettant de personnaliser l'accès à l'information, d'améliorer l'expérience utilisateur et de renforcer l'engagement. Leur rôle est particulièrement visible dans des secteurs comme le commerce en ligne, les plateformes de streaming ou encore les réseaux sociaux, où ils génèrent une part importante des revenus et de l'activité. Ainsi, plus de 35% des revenus d'Amazon et 80% du temps de visionnage sur Netflix sont issus de ces systèmes (*cf : Ricci,F. et al [1]*).

Dans le secteur bancaire, l'utilisation de ces systèmes ouvre de nouvelles perspectives en matière de personnalisation des services et de fidélisation des clients. La Banque Nationale d'Investissement (BNI), dans un contexte de transformation digitale et de forte concurrence, s'inscrit dans cette dynamique en cherchant à tirer parti des sciences des données pour optimiser son offre.
C'est dans ce cadre que s'inscrit mon stage de fin d'études, réalisé au sein de la Direction des Systèmes d'Information (DSI) de la BNI, dans le cadre du Master 2 "Ingénierie Mathématique appliquée aux Sciences des Données". Ce stage s'est articulé autour de deux axes :

- l'optimisation du reporting décisionnel par l'automatisation de tableaux de bord et d'indicateurs de performance ;
- la mise en œuvre d'un système de recommandation de produits bancaires, basé sur des techniques d'apprentissage automatique.

Ce sujet m'a particulièrement intéressé car il combinait à la fois mes compétences en modélisation mathématique et mon attrait pour l'application concrète de l'intelligence artificielle au service de problématiques réelles. Le présent mémoire propose ainsi, après une présentation de l'entreprise d'accueil, un état de l'art sur les systèmes de recommandation, une description des travaux réalisés, et enfin une discussion critique des résultats obtenus et des perspectives d'amélioration.

Présentation de l'entreprise d'accueil

1.1 Présentation générale

La Banque Nationale d'Investissement (BNI) de Côte d'Ivoire est une institution financière emblématique du paysage bancaire ivoirien, dont l'histoire remonte à plus de six décennies. Créée en 1959 par le décret n° 59-209 du 21 octobre 1959 sous l'appellation de Caisse Autonome d'Amortissement (CAA), cette institution est née de la volonté des autorités ivoiriennes d'asseoir le développement de la Côte d'Ivoire sur une institution forte et crédible.
À ses débuts, la CAA avait pour mission principale la gestion de la dette publique et la recherche de financements pour des projets de développement national. Cette orientation stratégique témoignait de l'ambition du jeune État ivoirien de se doter d'instruments financiers adaptés à ses besoins de développement économique et social. Les décennies 1980 et 1990 ont été marquées par des réformes importantes du secteur bancaire ivoirien, auxquelles la BNI a su s'adapter en modernisant ses structures et en professionnalisant ses équipes. La banque a progressivement étendu son réseau d'agences sur l'ensemble du territoire national, renforçant ainsi sa présence auprès des populations urbaines et rurales.

En 1998, de nouvelles orientations ont été données à la Banque. Elle est ainsi devenue une banque d'investissement après avoir obtenu un agrément d'établissement bancaire auprès de la Commission Bancaire de l'Union Monétaire Ouest-Africaine (UMOA) et de la Banque Centrale des États de l'Afrique de l'Ouest (BCEAO), avec statut de société d'État [1].
Au début des années 2000, la BNI a entamé une nouvelle phase de modernisation, intégrant les technologies de l'information et de la communication dans ses processus opérationnels. Cette digitalisation progressive a permis à la banque de rester compétitive dans un environnement bancaire de plus en plus concurrentiel.
Ainsi, l'ex-CAA a été transformée en Banque Nationale d'Investissement (BNI) depuis 2004 et le capital social a été porté à 20 Milliards 500 millions de FCFA (soit 31,3 millions d'euros).
Pour compléter son offre de services, la BNI a créé deux filiales : BNI Finances et BNI Gestion [2]. BNI Finances est une Société de Gestion et d'Intermédiation (SGI) qui soutient les activités de banque d'investissement et de conseil de la BNI. BNI Gestion, quant à elle, offre des solutions d'investissement personnalisées aux investisseurs institutionnels, entreprises, particuliers et collectivités publiques. En plus de ses deux filiales, la BNI a établi plusieurs partenariats, notamment avec le Fonds de Développement de la Formation Professionnelle (FDFP) [3] pour soutenir les cabinets de formation, et avec la Fédération Ivoirienne de Football (FIF) [4] pour le lancement de cartes bancaires co-brandées. Pour promouvoir le cacao durable et l'emploi des jeunes, elle a signé un partenariat avec l'European Investment Bank

[1] https://www.bni.ci/groupe-bni/historique-bni
[2] https://www.bni.ci/groupe-bni/historique-bni
[3] https://www.bni.ci/actualite
[4] https://www.bni.ci/actualite

(EIB) [5]. La BNI a également des partenariats dans le domaine de l'assurance, notamment avec la LOYALE-VIE, SAHAM Assurance Vie, Wafa Assurance, et SUNU Assurance Vie [6].

Aujourd'hui, avec plus de 65 ans d'existence, la BNI compte parmi les institutions financières les plus expérimentées de Côte d'Ivoire. Elle a su traverser les différentes crises économiques et politiques du pays tout en conservant sa vocation de banque de développement et en s'adaptant aux évolutions du marché bancaire.

La mission de la BNI a évolué au fil du temps pour s'adapter aux défis contemporains du secteur bancaire ivoirien. Depuis 2022, conformément à ses objectifs stratégiques, la BNI a redéfini sa mission en un axe principal : *"Apporter des Solutions Financières, adaptées aux besoins des clients, pour accompagner le développement, les entreprises et la population"*.
Cette mission s'articule autour de plusieurs axes stratégiques :
Le financement du développement économique : Fidèle à son héritage historique, la BNI continue de jouer un rôle clé dans le financement des projets structurants pour l'économie ivoirienne. Elle accompagne les entreprises dans leurs projets d'investissement et de croissance, contribuant ainsi au développement du tissu économique national.
L'inclusion financière : La banque s'engage à démocratiser l'accès aux services financiers pour toutes les couches de la population, en particulier les PME et les particuliers. Cette approche inclusive vise à réduire les inégalités d'accès au crédit et aux services bancaires.
L'innovation et la modernisation : La BNI place l'innovation au cœur de sa stratégie, notamment à travers la digitalisation de ses services et l'adoption de nouvelles technologies. Cette orientation permet à la banque de répondre aux attentes d'une clientèle de plus en plus exigeante et connectée.
Le slogan de la BNI, *"Financer pour développer"*, résume parfaitement sa philosophie d'entreprise. Il reflète la double vocation de la banque : être un acteur économique profitable tout en contribuant au développement socio-économique du pays.
Le marché bancaire ivoirien, composé d'une trentaine d'établissements de crédit, se caractérise par une forte concentration et une concurrence intense. Dans ce contexte, la BNI occupe une position particulière qui allie tradition et modernité.

Selon un rapport récemment publié par le KOACI [7] et le 7info [8], elle occupe la 3ème place dans le classement des banques ivoiriennes, en termes de total bilan, avec 11% de part de marché. En 2024, elle a atteint un total bilan de 2 358 milliards de FCFA (soit 3,6 milliards d'euros), soit une croissance de 34% par rapport à 2023.
La BNI a gravi deux positions dans le classement, passant de la 5ème à la 3ème place grâce à cette performance. Elle est également classée 2ème pour les dépôts et 6ème pour les emplois, avec des parts de marché respectives de 12% et 8%.
La banque a affiché des résultats records en 2024 et ambitionne de consolider sa place dans le Top 3 et de se hisser dans le Top 2 du secteur bancaire ivoirien d'ici 2026.

1.2 Le service d'accueil

La Direction des Systèmes d'Information (DSI) de la BNI joue un rôle stratégique dans la transformation digitale de la banque. En tant que pilier technologique de l'institution, elle assure la conception, le développement, la mise en œuvre et la maintenance de l'ensemble

[5] `https://www.eib.org/fr/press`
[6] `https://www.bni.ci/groupe-bni/partenariats`
[7] `https://www.koaci.com`
[8] `https://www.7info.ci`

des systèmes d'information qui supportent les activités bancaires. De manière générale, elle s'occupe principalement de :
La stratégie technologique : La DSI définit et met en œuvre la stratégie technologique de la banque en alignement avec les objectifs business. Elle identifie les technologies émergentes susceptibles d'améliorer l'efficacité opérationnelle et l'expérience client.
La gestion de l'infrastructure : Elle assure la disponibilité, la sécurité et la performance de l'infrastructure informatique, incluant les serveurs, les réseaux, les bases de données et les applications métiers.
L'intégration et du développement : La DSI développe des solutions sur mesure et intègre des systèmes tiers pour répondre aux besoins spécifiques des différents métiers de la banque.
La sécurité informatique : Elle met en place et maintient les dispositifs de sécurité pour protéger les données sensibles et assurer la conformité réglementaire.
La maintenance : La DSI assure le support technique aux utilisateurs et maintient les systèmes en condition opérationnelle optimale.
L'organisation de la DSI repose sur une structure fonctionnelle articulée autour de cinq services spécialisés : le Service Développement et Intégration, le Service Pilotage, Projets et Transformation, le Service Help Desk et Supervision, le Service Infrastructure, Réseaux et Datacenters, ainsi que le Service Core Banking et Exploitation. Cette répartition permet une gestion méthodique des activités, tout en favorisant une expertise pointue dans chaque domaine stratégique du système d'information.
L'effectif de la DSI compte une trentaine de collaborateurs aux compétences complémentaires, incluant des ingénieurs systèmes, développeurs, architectes solutions, data scientists, experts en sécurité informatique et chefs de projet. Cette diversité de profils garantit une couverture raisonnable des besoins technologiques et assure la synergie nécessaire entre les différents services pour mener à bien les projets stratégiques de la banque.

1.3 Organigramme de la BNI

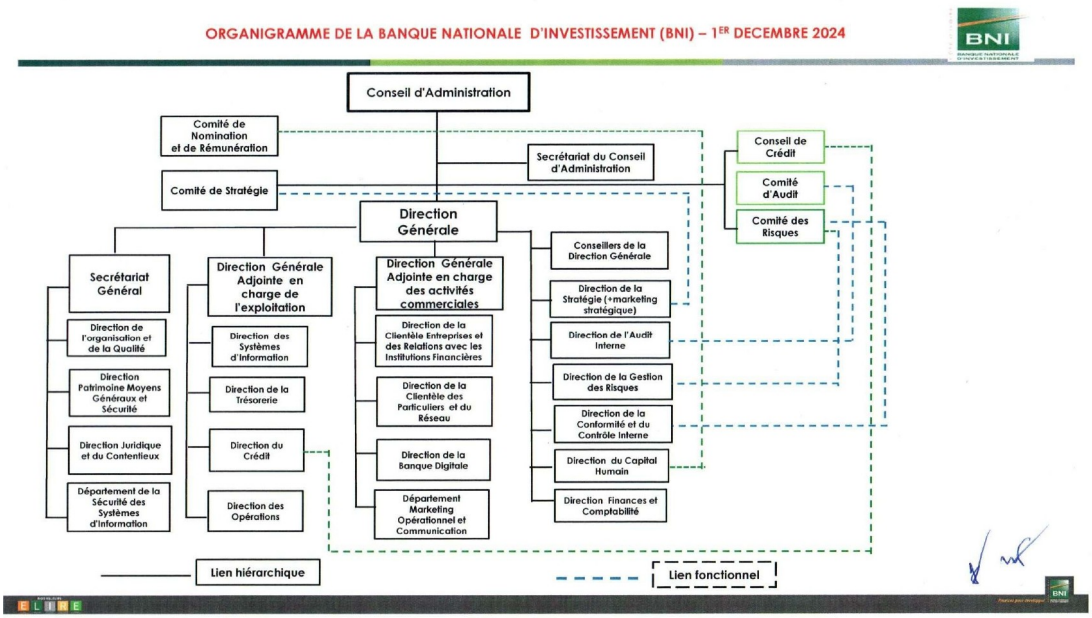

Fig. 1.1 – Organigramme selon le *mémorandum* de la BNI, Décembre 2024.

1.4 Organigramme de la DSI

Globalement, la Direction des Systèmes d'Information de la BNI est structurée selon l'organigramme ci-dessous :

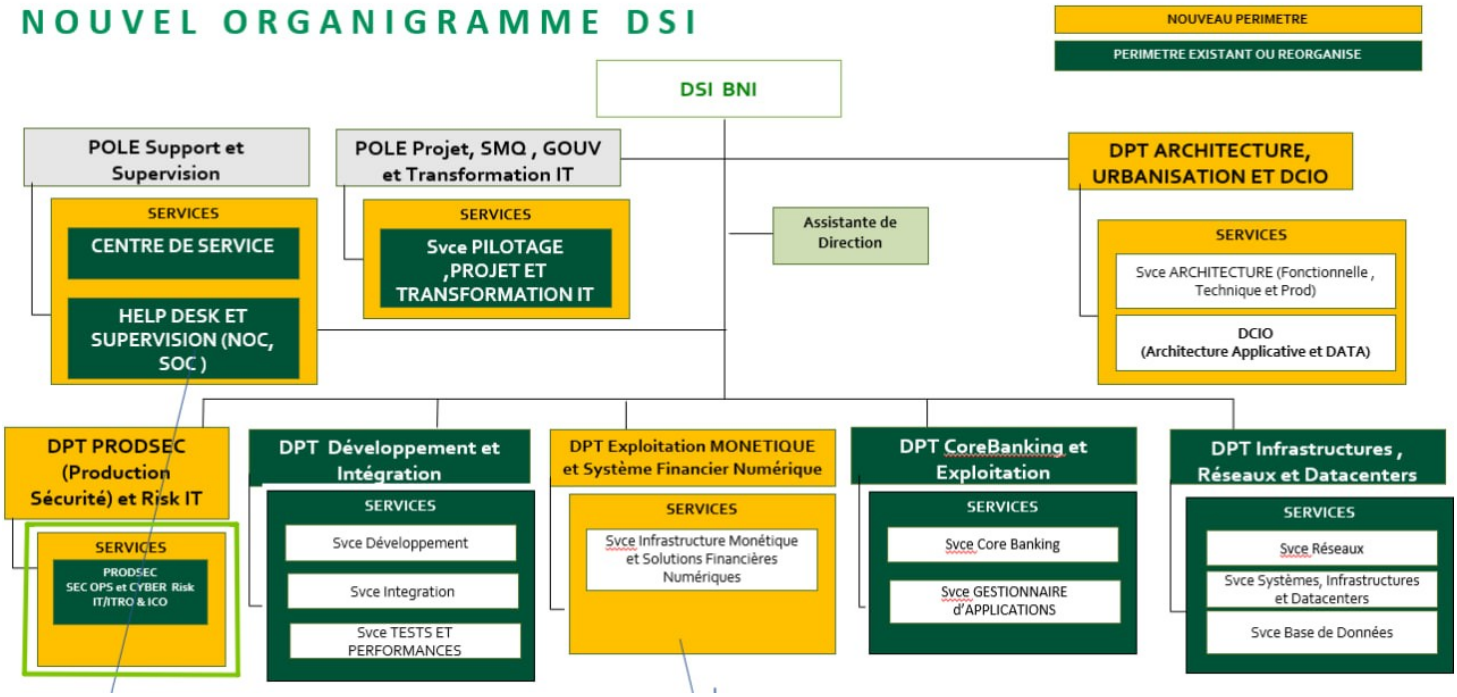

Fig. 1.2 – Organigramme selon le *mémorandum* de la BNI, Décembre 2024.

État de l'art et fondements théoriques des systèmes de recommandation

2.1 État de l'art

Les systèmes de recommandation représentent un domaine d'étude qui a largement gagné en importance depuis le début des années 1990 en raison de l'essor du commerce en ligne [14] et la publication des premiers articles en relation avec le filtrage collaboratif (*Collaborative Filtering*) [17]. Il existe donc aujourd'hui plusieurs stratégies pour le développement de systèmes de recommandation [9].
Selon Ricci et al. (2011) dans [1], « la plupart des systèmes de recommandation modernes reposent sur trois paradigmes fondamentaux : le filtrage collaboratif, le filtrage basé sur le contenu et les approches hybrides ». Dans ce qui suit, nous présentons en détail chacune de ces méthodes, en mettant en lumière leurs principes, avantages et limites respectives.

2.1.1 Formalisation mathématique du problème

Définitions

Définition 2.1 (Système de recommandation)**.** Les systèmes de recommandation sont des outils d'intelligence artificielle conçus pour aider les utilisateurs à découvrir des items (produits, services ou contenus) pertinents, souvent en analysant leurs préférences, historiques et interactions passées [1]. Leur objectif principal est de filtrer intelligemment l'information disponible afin de personnaliser l'expérience utilisateur et de lui proposer des suggestions adaptées à ses goûts, même si ces suggestions ne seraient pas découvertes spontanément [29].
Formellement, un système de recommandation peut être modélisé comme une fonction :

$$f : \mathcal{U} \times \mathcal{I} \times \mathcal{C} \to \mathcal{R} \tag{2.1}$$

où :

- $\mathcal{U}$ est l'ensemble des utilisateurs,
- $\mathcal{I}$ est l'ensemble des items,
- $\mathcal{C}$ est l'ensemble des contextes (par exemple, temps, lieu, appareil),
- $\mathcal{R}$ est l'ensemble des prédictions ou recommandations possibles.

Définition 2.2 (Matrice de notation)**.** La relation entre les utilisateurs et les items est généralement représentée par une matrice de notation $R \in \mathbb{R}^{m \times n}$, où $m = |\mathcal{U}|$ et $n = |\mathcal{I}|$. Chaque élément r_{ui} de R correspond à la note (explicite ou implicite) donnée par l'utilisateur u à l'item i. Comme le souligne [2], cette matrice est typiquement creuse : la plupart des utilisateurs n'ont évalué qu'une petite fraction des items, ce qui mène à un taux de remplissage $\rho = \frac{|\{(u,i) : r_{ui} \neq 0\}|}{mn}$ souvent inférieur à 1%.

Problème de prédiction

Le problème central consiste à prédire les évaluations manquantes dans R. Formellement, nous cherchons à estimer une fonction $\hat{r}_{ui} = f(u, i, \Theta)$ où Θ représente les paramètres du modèle, telle que :

$$\hat{r}_{ui} \approx r_{ui} \quad \forall (u, i) \in \mathcal{T} \tag{2.2}$$

où $\mathcal{T}$ est l'ensemble des couples (utilisateur, item) pour lesquels nous avons des évaluations observées.

2.1.2 Concepts et approches des systèmes de recommandation

Filtrage collaboratif

Le filtrage collaboratif (*Collaborative Filtering, CF*) est une des approches les plus populaires et les plus utilisées dans les systèmes de recommandation modernes. Son principe fondamental repose sur l'idée que les utilisateurs qui ont partagé des préférences similaires dans le passé auront probablement des goûts similaires à l'avenir [16]. En d'autres termes, si deux utilisateurs ont aimé les mêmes articles dans le passé, ils sont susceptibles d'apprécier les mêmes articles dans le futur.
Cette approche reproduit à grande échelle un comportement humain courant : celui du bouche-à-oreille. Lorsqu'une personne cherche un bon film, un restaurant ou un livre, elle a souvent recours à l'avis d'autres personnes ayant des goûts similaires. Le filtrage collaboratif automatise cette démarche via le traitement algorithmique de grandes matrices d'évaluations.

Illustration. La figure ci-dessous illustre de manière simplifiée le fonctionnement d'un système de filtrage collaboratif : les utilisateurs interagissent avec un sous-ensemble d'items (films, produits, articles...), et le système identifie des utilisateurs "voisins" ayant des comportements similaires pour générer des recommandations pertinentes.

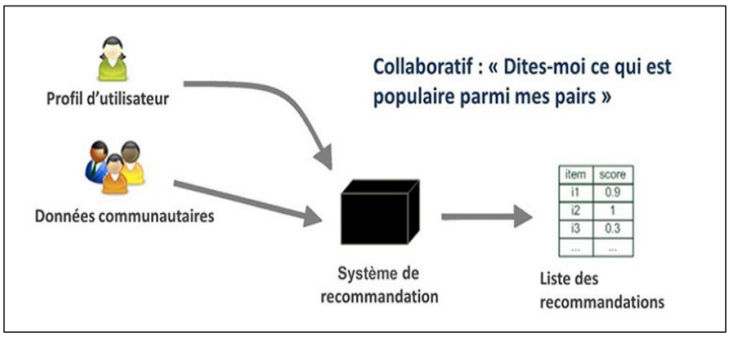

Fig. 2.1 – Système de recommandation collaboratif [29]

Deux grandes familles d'approches. On distingue généralement deux catégories de méthodes dans le filtrage collaboratif :

- **Méthodes basées sur la mémoire** : elles exploitent directement la base de données d'interactions utilisateur-item pour calculer des similarités et produire des recommandations. Elles sont simples à implémenter mais deviennent inefficaces à grande échelle.
- **Méthodes basées sur un modèle** : elles utilisent des techniques d'apprentissage automatique pour construire un modèle prédictif à partir des données, tel que les

matrices de facteurs latents (ex : SVD), les modèles bayésiens ou les réseaux de neurones.

Mesure de la similarité entre utilisateurs. Le cœur des méthodes à base de mémoire repose sur la mesure de la similarité entre les profils utilisateurs. Deux mesures couramment utilisées sont la corrélation de Pearson et la similarité cosinus.

Définition 2.3 (Similarité entre utilisateurs – Corrélation de Pearson)**.** La similarité entre deux utilisateurs u et v peut être mesurée par la corrélation de Pearson :

$$\text{sim}_{\text{Pearson}}(u,v) = \frac{\sum_{i\in\mathcal{I}_{uv}}(r_{ui}-\bar{r}_u)(r_{vi}-\bar{r}_v)}{\sqrt{\sum_{i\in\mathcal{I}_{uv}}(r_{ui}-\bar{r}_u)^2}\sqrt{\sum_{i\in\mathcal{I}_{uv}}(r_{vi}-\bar{r}_v)^2}} \tag{2.3}$$

où $\mathcal{I}_{uv}$ est l'ensemble des items évalués par les deux utilisateurs, r_{ui} la note donnée par u à l'item i, et $\bar{r}_u$ la moyenne des notes de u.

Cette mesure permet de neutraliser l'effet des utilisateurs ayant des biais d'évaluation (par exemple, ceux qui notent systématiquement plus haut ou plus bas que les autres), en centrant les notes autour de la moyenne propre à chaque utilisateur.

Définition 2.4 (Similarité entre utilisateurs – Cosinus)**.** La similarité cosinus entre deux utilisateurs u et v est définie par :

$$\text{sim}_{\cos}(u,v) = \frac{\sum_{i\in\mathcal{I}_{uv}} r_{ui}\cdot r_{vi}}{\sqrt{\sum_{i\in\mathcal{I}_{uv}} r_{ui}^2}\cdot\sqrt{\sum_{i\in\mathcal{I}_{uv}} r_{vi}^2}} \tag{2.4}$$

Elle mesure le cosinus de l'angle entre les vecteurs d'évaluations des deux utilisateurs, sans recentrage.

La mesure cosinus est plus simple à calculer et souvent utilisée lorsque les vecteurs de notes sont normalisés. Toutefois, elle peut être moins robuste dans le cas d'utilisateurs ayant des biais d'échelle.

En résumé, la corrélation de Pearson est souvent privilégiée lorsque les notes attribuées par les utilisateurs présentent une forte hétérogénéité, car elle permet de recentrer les évaluations autour de la moyenne de chaque utilisateur, atténuant ainsi les biais individuels. À l'inverse, la similarité cosinus se montre plus efficace dans les contextes où les données sont peu bruitées ou déjà normalisées (ex : Movielens), en raison de sa simplicité de calcul et de sa capacité à capter la direction des préférences plutôt que leur valeur absolue.

Avantages et limites du CF. L'un des principaux avantages du filtrage collaboratif réside dans le fait qu'il ne nécessite aucune description explicite des items : les recommandations sont générées uniquement à partir des interactions passées entre utilisateurs et items. Cette méthode permet également de mettre en évidence des relations complexes entre les préférences d'utilisateurs, reposant uniquement sur leurs comportements collectifs. Toutefois, cette approche souffre de certaines limites, notamment le problème du démarrage à froid *(cold-start)*, où il devient difficile de faire des recommandations à un nouvel utilisateur ou pour un nouvel item dépourvu d'historique. Par ailleurs, la nature souvent clairsemée *(sparse)* des données utilisateur-item peut nuire à la qualité des prédictions. Enfin, les méthodes basées sur la mémoire présentent des problèmes de scalabilité, car leur

coût computationnel augmente significativement avec le nombre d'utilisateurs et d'items.

En conclusion, le filtrage collaboratif constitue une méthode puissante et intuitive pour générer des recommandations personnalisées. Son efficacité repose sur l'hypothèse de similarité comportementale entre utilisateurs. Bien qu'elle présente certaines limitations, cette méthode constitue la base de nombreuses implémentations industrielles, notamment chez Netflix, Amazon ou YouTube. Dans la pratique, elle est souvent renforcée ou combinée à d'autres approches au sein de systèmes hybrides, qui seront présentés dans les sections ultérieures.

Filtrage basé sur le contenu

Le filtrage basé sur le contenu *(Content-Based Filtering, CBF)* repose sur l'idée que les utilisateurs préfèrent des items similaires à ceux qu'ils ont aimés par le passé. Cette approche exploite les propriétés des items, comme les mots-clés, les catégories ou encore les descripteurs numériques, pour construire un modèle personnalisé de préférences utilisateur [28, 11].
De façon opérationnelle, un modèle (appelé profil utilisateur) est construit pour chaque utilisateur en analysant les items qu'il a évalués ou consommés. Ce profil est ensuite utilisé pour estimer la pertinence d'un nouvel item en fonction de ses caractéristiques. Cela permet par exemple de recommander un film possédant des attributs similaires à ceux des films déjà bien notés par l'utilisateur.
Autrement dit, cette méthode permet l'analyse fine des préférences d'un consommateur vis-à-vis de certaines qualités d'un produit, qu'elles soient explicites (par exemple, un genre de film ou une marque) ou implicites (inférées à partir de données d'usage).

Illustration. Cette approche est couramment utilisée dans les systèmes de recommandation de livres, musiques ou encore d'articles, où les items sont bien décrits par des métadonnées ou du texte. De nombreux systèmes industriels ont d'ailleurs intégré le filtrage basé sur le contenu, comme l'explique [29], notamment dans les premiers moteurs de recommandation d'Amazon ou de Pandora.

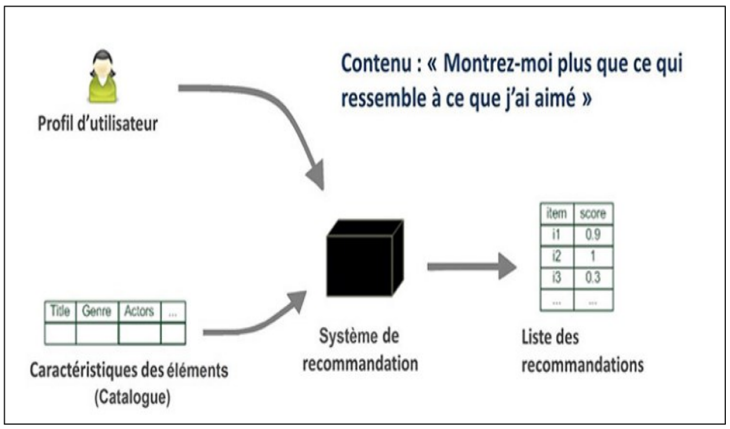

Fig. 2.2 – Système de recommandation basé sur le contenu [29]

Avantages. Le filtrage basé sur le contenu présente plusieurs atouts majeurs. Contrairement au filtrage collaboratif, il ne nécessite pas de données provenant d'autres utilisateurs. Il peut ainsi générer des recommandations même pour des profils très spécifiques ou rares, ce qui permet de contourner en partie le problème du démarrage à froid du côté utilisateur. De plus, son implémentation reste relativement simple, en particulier dans des environnements caractérisés par un faible volume de trafic [28].

Limites. Malgré ces avantages, cette approche comporte également certaines limites bien connues. Elle tend en effet à produire une sur-spécialisation en recommandant principalement des items très proches de ceux déjà consommés, ce qui réduit la diversité des suggestions proposées [11]. Son efficacité dépend par ailleurs fortement de la qualité et de la richesse des descripteurs utilisés pour caractériser les items ; lorsque ces derniers sont insuffisants ou bruités, les performances du modèle peuvent s'en trouver fortement dégradées. Enfin, cette méthode ne prend généralement pas en considération les effets de popularité ou les tendances sociales, alors même que ceux-ci jouent souvent un rôle important dans les choix des utilisateurs.

Conclusion. En résumé, le filtrage basé sur le contenu constitue une méthode robuste et autonome, particulièrement adaptée aux contextes où les interactions sociales sont limitées ou indisponibles. Toutefois, ses limites, en particulier le problème de sur-spécialisation, incitent à l'intégrer dans une approche hybride qui combine plusieurs techniques, comme cela sera présenté dans la section suivante.

Approche hybride

Les systèmes de recommandation hybrides combinent plusieurs techniques de recommandation principalement le filtrage collaboratif et le filtrage basé sur le contenu afin de bénéficier des avantages de chacune, tout en atténuant leurs inconvénients. Cette approche s'est imposée comme une solution efficace pour répondre aux limites structurelles des systèmes classiques, notamment les problèmes de démarrage à froid, de sparsité des données, ou encore de sur-spécialisation.

Motivations. Le filtrage collaboratif repose sur l'hypothèse que des utilisateurs ayant eu des comportements similaires dans le passé continueront à le faire dans le futur [10, 8]. Bien que cette méthode soit largement utilisée, elle dépend fortement de la densité des interactions, ce qui limite ses performances en présence de données clairsemées. De son côté, le filtrage basé sur le contenu exploite les caractéristiques des items pour faire des suggestions [11, 28]. Il est efficace pour recommander de nouveaux items, mais souffre d'un manque de diversité et d'une tendance à proposer toujours des éléments similaires (effet de sur-spécialisation).
Dans ce contexte, l'approche hybride permet de tirer parti de la complémentarité entre ces méthodes [1, 12]. Comme le souligne [12], l'hybridation améliore la robustesse globale du système, augmente la couverture et favorise une personnalisation plus fine des recommandations.

Méthodes d'hybridation. Plusieurs stratégies d'hybridation sont proposées dans la littérature :

- **Pondérée :** cette technique consiste à combiner les scores générés par plusieurs algorithmes à l'aide de poids numériques. Par exemple, on peut attribuer un poids de 0,6 au score d'un système de filtrage collaboratif et 0,4 à celui d'un système basé sur le contenu. Le score final est alors une moyenne pondérée. Cette approche est simple à implémenter et permet un ajustement flexible selon la performance de chaque composant [12].
- **Cascade :** dans cette stratégie, les algorithmes sont organisés en pipeline. Un premier système génère une liste d'items candidats, et un second système raffine cette liste, souvent par réordonnancement ou filtrage [12]. Par exemple, un système basé sur le contenu peut fournir une première sélection d'articles pertinents, que l'on

affine ensuite avec un filtre collaboratif tenant compte des préférences d'utilisateurs similaires.

— **Switching :** cette méthode consiste à alterner dynamiquement entre différents systèmes selon les cas d'usage. Par exemple, si un utilisateur est nouveau (peu de données d'interaction), on privilégie le filtrage basé sur le contenu ; sinon, on applique le filtrage collaboratif [1].

— **Feature Combination :** cette technique combine les informations issues de différents systèmes sous forme d'un vecteur unifié, souvent traité ensuite par un modèle de machine learning (régression, réseau de neurones) [18, 15].

— **Modèles unifiés :** certaines approches hybrides vont plus loin en intégrant l'hybridation directement au sein d'un modèle unique. C'est le cas par exemple des autoencodeurs dans AutoRec [9], des réseaux de neurones dans NCF [3], ou encore des modèles de graphes comme LightGCN [5] qui exploitent conjointement les interactions et les attributs.

Résultats empiriques. Des études ont démontré que les approches hybrides surpassent généralement les méthodes pures sur des métriques classiques telles que la précision, le rappel ou le NDCG [1, 12]. En particulier, elles permettent :

— de mieux traiter les utilisateurs ou items nouveaux (problème du démarrage à froid),

— d'augmenter la couverture des systèmes (proposer des recommandations même en cas de peu d'historique),

— de réduire la redondance des recommandations (diversité accrue),

— de tirer profit des techniques avancées de représentation comme les embeddings ou les réseaux neuronaux profonds [13, 4, 23].

Les systèmes hybrides représentent aujourd'hui l'une des directions les plus prometteuses pour la recommandation personnalisée. Leur capacité à combiner les points forts de différentes méthodes en fait une solution adaptée aux environnements complexes et à grande échelle, comme ceux rencontrés dans l'e-commerce, la vidéo à la demande ou les réseaux sociaux [20, 17]. Leur succès repose toutefois sur une conception judicieuse de l'architecture d'intégration, ainsi qu'une évaluation rigoureuse de la complémentarité entre les modèles. La section 2.3 est entièrement consacré à une analyse détaillée de l'approche hybride.

Tableau comparatif des approches de recommandation

On résume, dans le tableau récapitulatif ci-dessous, les points forts et les points faibles des différentes approches présentées.

Méthode	Avantages	Inconvénients
Filtrage collaboratif	— Pas besoin de métadonnées sur les items — Le système n'a, dans une certaine mesure, besoin que de la matrice de feedback pour entraîner un modèle de factorisation matricielle. — Le système n'a pas besoin de caractéristiques contextuelles — S'adapte aux goûts évolutifs	— Problème du démarrage à froid (utilisateur/item) — Matrices utilisateur-item souvent creuses — La complexité des algorithmes est proportionnelle à la taille de la matrice donc d'ordre $\mathcal{O}(l \times c)$ (l : nombre de lignes ; c : nombre de colonnes)
Filtrage basé sur le contenu	— Ne dépend pas des autres utilisateurs — Plus facile de l'adapter à un grand nombre d'utilisateurs — Recommande des items nouveaux (cold-start item) — Recommandations personnalisées	— Besoin de descripteurs détaillés pour les items — Risque de surspécialisation (peu de diversité) — Ne capte pas les tendances globales
Approche hybride	— Combine les points forts des deux approches — Meilleure précision globale — Réduction du problème de cold start	— Implémentation plus complexe — Coût computationnel plus élevé — Requiert des données multiples (contenu + interactions)

Tab. 2.1 – Comparaison des approches de recommandation : contenu, collaboratif et hybride.

2.1.3 Métriques d'évaluation

Les systèmes de recommandation basés sur le filtrage collaboratif font des prédictions sur la matrice R. La méthode consiste à faire des prédictions automatiques sur les intérêts d'un utilisateur en recueillant les diverses évaluations des autres utilisateurs. Pour mesurer la précision de celles-ci, les interactions estimées sont comparées avec les interactions réelles, c'est-à-dire celles qui ont été créées par l'utilisateur.
La précision d'un système de recommandations est généralement évaluée par deux mesures principales : l'erreur quadratique moyenne (RMSE) et l'erreur absolue moyenne (MAE). Ces deux métriques permettent une interprétation facile de par leur même échelle que les notes d'origine. Cependant, il peut être préférable d'en utiliser l'une ou l'autre selon le contexte.

Erreur quadratique moyenne

Définition 2.5 (RMSE)**.** La Root Mean Square Error est définie par :

$$\text{RMSE} = \sqrt{\frac{1}{|\mathcal{T}|} \sum_{(u,i)\in\mathcal{T}} (\hat{r}_{ui} - r_{ui})^2} \tag{2.5}$$

L'erreur quadratique moyenne a pour caractéristique principale qu'elle a tendance à pénaliser davantage les erreurs importantes puisqu'elles sont mises au carré. Cela signifie que la métrique RMSE est plus susceptible d'être affectée par des valeurs aberrantes ou de mauvaises prédictions. Par définition, la métrique RMSE ne sera jamais aussi petite que la métrique MAE. De plus, la métrique RSME n'utilise pas de valeurs absolues, ce qui est beaucoup plus pratique mathématiquement lors du calcul de la distance, du gradient ou d'autres mesures. C'est pourquoi la plupart des fonctions de coût de l'apprentissage automatique évitent d'utiliser la métrique MAE.

Erreur absolue moyenne

Définition 2.6 (MAE)**.** La Mean Absolute Error est définie par :

$$\text{MAE} = \frac{1}{|\mathcal{T}|} \sum_{(u,i)\in\mathcal{T}} |\hat{r}_{ui} - r_{ui}| \tag{2.6}$$

L'erreur moyenne absolue a pour caractéristique qu'elle ne donne aucun biais aux erreurs extrêmes. S'il y a des valeurs aberrantes ou des termes d'erreur importants, leur erreur pèsera de la même manière que les autres erreurs de prédiction. Par conséquent, la métrique MAE doit être privilégiée lorsque l'on recherche davantage l'exactitude des notes plutôt que de donner de l'importance aux valeurs aberrantes. Pour obtenir une vue ou une représentation holistique du système de recommandation, on utilisera donc la métrique MAE.

Ainsi, si les métriques RMSE et MAE restent les plus couramment utilisées dans l'évaluation de la qualité prédictive des systèmes de recommandation en particulier pour les approches basées sur la prédiction de notes, elles ne permettent pas à elles seules de rendre compte de la pertinence réelle des recommandations proposées à l'utilisateur. Dans les scénarios top-N, où l'objectif est de suggérer une liste restreinte d'items susceptibles d'intéresser l'utilisateur, il devient essentiel de recourir à des métriques issues de la classification, telles que la précision, le rappel, l'accuracy ou encore le F1-score.
Ces dernières permettent de mieux évaluer l'adéquation des éléments recommandés avec les attentes ou comportements réels de l'utilisateur, en tenant compte non seulement des prédictions correctes mais aussi des erreurs d'omission ou de suggestion. En définitive, le

choix des métriques dépend fortement du contexte applicatif : certaines prioriseront la réduction des erreurs numériques, tandis que d'autres chercheront à maximiser la pertinence perçue ou l'engagement utilisateur. Une évaluation rigoureuse et complète d'un système de recommandation doit donc s'appuyer sur un ensemble complémentaire de métriques, permettant de capturer à la fois la justesse statistique et l'efficacité pratique du système.

2.2 Fondements théoriques

Le choix des algorithmes de machine learning à implémenter dépend du type de système de recommandation envisagé, des objectifs poursuivis ainsi que des caractéristiques des données disponibles.

2.2.1 Filtrage collaboratif et factorisation matricielle

Approches basées sur la mémoire

L'approche basée sur la mémoire utilise les interactions passées des utilisateurs pour calculer les similitudes entre ceux-ci ou entre les items. Pour trouver la note r_{ui} qu'un utilisateur donnerait à un item i, l'approche recherche les utilisateurs similaires à u qui ont noté l'item i et calcule la note r_{ui} en fonction des notes des utilisateurs trouvés à l'étape précédente. Afin de trouver les U utilisateurs les plus similaires à l'utilisateur u, on calcule la similarité sur base des items communément notés avec l'utilisateur comparé en calculant leur distance ou leur similarité.

Filtrage collaboratif utilisateur-utilisateur

L'approche utilisateur-utilisateur prédit la note r_{ui} d'un utilisateur pour un item en utilisant les notes des utilisateurs similaires.

Théorème 2.1 (Prédiction basée sur les k plus proches voisins)**.** La prédiction basée sur les k plus proches voisins (k-NN) dans un système de recommandation est une méthode qui utilise les préférences d'utilisateurs similaires à l'utilisateur actif pour prédire ses goûts potentiels pour un item non évalué. Cette approche repose sur la similarité des préférences entre utilisateurs et utilise les évaluations de ces utilisateurs proches pour estimer le score que l'utilisateur actif pourrait donner à un item.
Soit $\mathcal{N}_k(u)$ l'ensemble des k utilisateurs les plus similaires à u. La prédiction pour l'utilisateur u et l'item i est donnée par :

$$\hat{r}_{ui} = \bar{r}_u + \frac{\sum_{v \in \mathcal{N}_k(u)} \text{sim}(u, v) \cdot (r_{vi} - \bar{r}_v)}{\sum_{v \in \mathcal{N}_k(u)} |\text{sim}(u, v)|} \tag{2.7}$$

Démonstration. Cette formulation découle directement de l'hypothèse de moyenne pondérée, où les poids sont proportionnels à la similarité. La normalisation par la somme des similarités absolues garantit que la prédiction reste dans la plage des notes possibles. □

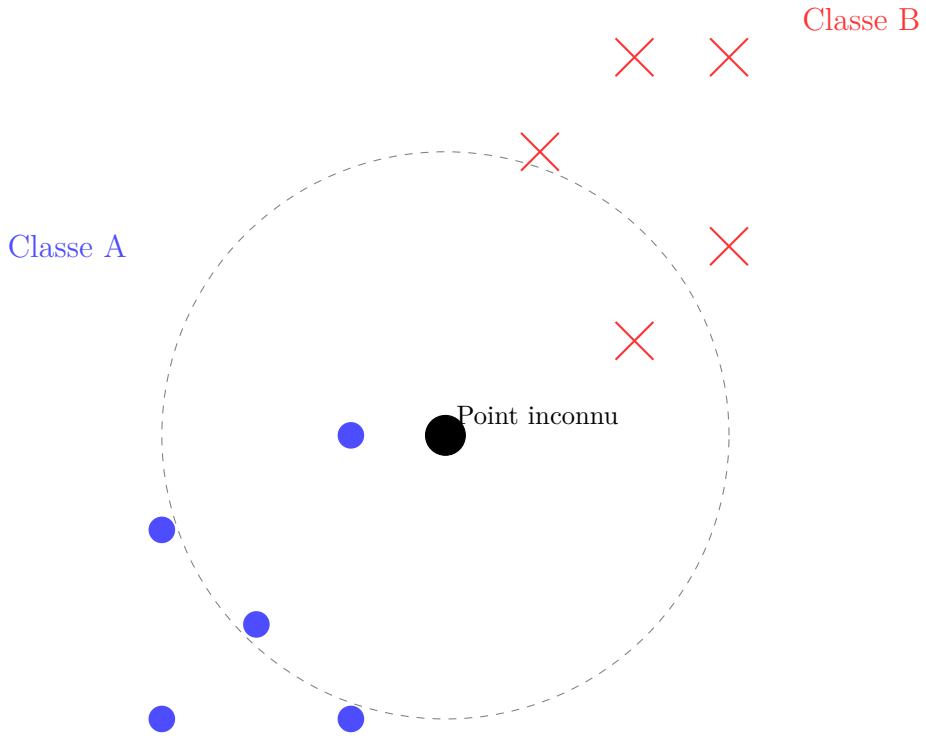

FIG. 2.3 – Exemple de classification par la méthode des k-plus proches voisins ($k = 3$). Le point noir représente un point inconnu à classer. Les trois plus proches voisins dans le cercle de rayon donné appartiennent majoritairement à la Classe A (points bleus), ce qui conduit à sa classification dans cette classe.

Factorisation matricielle

La factorisation matricielle est l'une des techniques les plus efficaces pour le filtrage collaboratif, particulièrement adaptée à la nature parcellaire des matrices utilisateur–item. Elle consiste à approximer la matrice des notations $R \in \mathbb{R}^{m \times n}$ par le produit de deux matrices de rang réduit. Cette approche permet de capturer les interactions latentes entre utilisateurs et items tout en réduisant la dimensionnalité du problème, ce qui améliore l'efficacité computationnelle et la qualité des recommandations [2]. Pour y arriver, on peut utiliser la méthode décomposition en valeurs singulières *(Singular Value Decomposition (SVD))* de la matrice R.

La SVD permet d'approximer la matrice R par :

$$R \approx U \Sigma V^{\top} \tag{2.8}$$

où :

- $U \in \mathbb{R}^{m \times r}$ contient les vecteurs propres des utilisateurs,
- $\Sigma \in \mathbb{R}^{r \times r}$ est une matrice diagonale des valeurs singulières,
- $V \in \mathbb{R}^{n \times r}$ contient les vecteurs propres des items,
- $r \ll \min(m, n)$ est le rang réduit (nombre de facteurs latents).

Cette approximation présente plusieurs avantages :

- réduction de la dimension des données,
- complétion des notes manquantes dans R,
- extraction de facteurs latents interprétables.

La méthode SVD a été popularisée dans le contexte des systèmes de recommandation par la participation de Simon Funk au Netflix Prize, où elle a démontré une capacité remarquable à prédire des notes manquantes [7].

$$\underbrace{\begin{bmatrix} 5 & ? & 3 \\ 4 & 1 & ? \\ ? & 2 & 4 \end{bmatrix}}_{\text{Matrice des notes } R} \approx \underbrace{\begin{bmatrix} u_{11} & u_{12} \\ u_{21} & u_{22} \\ u_{31} & u_{32} \end{bmatrix}}_{U_{m\times r}} \cdot \underbrace{\begin{bmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{bmatrix}}_{\Sigma_{r\times r}} \cdot \underbrace{\begin{bmatrix} v_{11} & v_{12} & v_{13} \\ v_{21} & v_{22} & v_{23} \end{bmatrix}^\top}_{V^\top_{r\times n}}$$

Cependant, compte tenu de la sparsité des matrices de notations, la SVD classique peut être inadaptée car elle nécessite une matrice complète. Pour pallier ce problème, Salakhutdinov et Mnih ont proposé une approche probabiliste de la factorisation matricielle [22] qui approxime R par :

$$R \approx PQ^\top \tag{2.9}$$

où $P \in \mathbb{R}^{m\times k}$ et $Q \in \mathbb{R}^{n\times k}$ sont les matrices latentes des utilisateurs et des items respectivement, avec $k \ll \min(m, n)$.

Théorème 2.2 (Optimisation par descente de gradient)**.** Les matrices P et Q sont apprises en minimisant la fonction de coût :

$$J(P, Q) = \frac{1}{2} \sum_{(u,i)\in\mathcal{K}} (r_{ui} - p_u^\top q_i)^2 + \frac{\lambda}{2} \left(\sum_{u=1}^{m} \|p_u\|^2 + \sum_{i=1}^{n} \|q_i\|^2 \right) \tag{2.10}$$

où $\mathcal{K}$ est l'ensemble des couples (utilisateur, item) observés et λ est un coefficient de régularisation pour éviter le surapprentissage.

Démonstration. La minimisation est effectuée par calcul des gradients :

$$\frac{\partial J}{\partial p_{uk}} = - \sum_{i:(u,i)\in\mathcal{K}} (r_{ui} - p_u^\top q_i) q_{ik} + \lambda p_{uk} \tag{2.11}$$

$$\frac{\partial J}{\partial q_{ik}} = - \sum_{u:(u,i)\in\mathcal{K}} (r_{ui} - p_u^\top q_i) p_{uk} + \lambda q_{ik} \tag{2.12}$$

□

Méthodes d'optimisation

Deux méthodes d'optimisation principales sont couramment utilisées dans ce contexte :

Descente de gradient stochastique *(Stochastic Gradient Descent (SGD)).* La méthode SGD met à jour les vecteurs p_u et q_i à chaque interaction observée (u, i). Elle est efficace pour les grands ensembles de données mais sensible au choix du taux d'apprentissage et peut converger lentement [2].

Moindres carrés alternés pondérés *(Weighted Alternated Least Squares (WALS)).* WALS repose sur une optimisation alternée des matrices P et Q, en résolvant à chaque étape un problème quadratique fermé. Cette méthode converge plus rapidement que SGD dans de nombreux cas et est bien adaptée aux architectures distribuées. Elle a été intégrée à des systèmes industriels comme celui de Google [8].

Factorisation matricielle avec biais

Théorème 2.3 (Modèle avec biais)**.** Une extension du modèle incorpore les biais globaux, utilisateur et item :

$$\hat{r}_{ui} = \mu + b_u + b_i + p_u^\top q_i \tag{2.13}$$

où μ est la note moyenne globale, b_u est le biais de l'utilisateur u, et b_i est le biais de l'item i.

La fonction de coût devient :

$$J = \frac{1}{2} \sum_{(u,i)\in\mathcal{K}} (r_{ui} - \mu - b_u - b_i - p_u^\top q_i)^2 + \frac{\lambda}{2} \left(\|P\|_F^2 + \|Q\|_F^2 + \|b_u\|^2 + \|b_i\|^2 \right) \tag{2.14}$$

Factorisation matricielle non négative

Définition 2.7 (NMF)**.** La factorisation matricielle non négative impose les contraintes $P \geq 0$ et $Q \geq 0$, permettant une interprétation additive des facteurs latents.

Théorème 2.4 (Algorithme multiplicatif pour NMF)**.** Les règles de mise à jour multiplicatives pour NMF sont :

$$P_{uk} \leftarrow P_{uk} \frac{(RQ)_{uk}}{(PQ^\top Q)_{uk}} \tag{2.15}$$

$$Q_{ik} \leftarrow Q_{ik} \frac{(R^\top P)_{ik}}{(QP^\top P)_{ik}} \tag{2.16}$$

2.2.2 Filtrage basé sur le contenu : principes et ingénierie des caractéristiques

Représentation des caractéristiques

Dans une approche de filtrage basée sur le contenu, chaque item i est modélisé par un vecteur de caractéristiques $x_i \in \mathbb{R}^d$, où d désigne le nombre total d'attributs décrivant les items. Ces caractéristiques peuvent inclure, selon le domaine, des informations telles que le genre, le thème, la durée, l'auteur, ou toute autre métadonnée pertinente.

Dans le cas particulier des documents textuels, une méthode de représentation très utilisée est le TF-IDF (*Term Frequency-Inverse Document Frequency*). Cette méthode attribue à chaque mot un poids reflétant son importance dans un document donné, pondérée par sa fréquence d'apparition dans l'ensemble du corpus. Elle permet ainsi de capturer la pertinence discriminante des termes pour caractériser un item.

Définition 2.8 (Poids TF-IDF - Implementation scikit-learn)**.** Le poids TF-IDF d'un terme t dans un document d selon l'implémentation de `TfidfVectorizer` de scikit-learn [1] est défini comme :

$$\text{TF-IDF}(t, d) = \text{TF}(t, d) \times \text{IDF}(t) \tag{2.17}$$

où :

$$\text{TF}(t, d) = \frac{\text{freq}(t, d)}{\sum_{t' \in d} \text{freq}(t', d)} \tag{2.18}$$

$$\text{IDF}(t) = \log \left(\frac{1 + N}{1 + |\{d : t \in d\}|} \right) + 1 \tag{2.19}$$

[1] `https://scikit-learn.org/stable/index.html`

Le vecteur TF-IDF final est normalisé selon la norme euclidienne (L2) :

$$\text{TF-IDF}_{\text{norm}}(t,d) = \frac{\text{TF-IDF}(t,d)}{\sqrt{\sum_{t' \in d} \text{TF-IDF}(t',d)^2}} \tag{2.20}$$

avec :

- $\text{freq}(t,d)$: fréquence du terme t dans le document d,
- N : nombre total de documents dans le corpus,
- $|\{d : t \in d\}|$: nombre de documents contenant le terme t,
- Le facteur de lissage $+1$ dans l'IDF évite la division par zéro et stabilise les calculs,
- Le terme $+1$ ajouté à l'IDF est spécifique à l'implémentation scikit-learn.

Afin d'assurer la pertinence de la représentation des documents, un prétraitement des données textuelles est indispensable. Il est crucial de nettoyer les textes en supprimant les mots vides (ou *stop words*) qui n'apportent pas d'information sémantique utile. Il s'agit généralement des articles, des prépositions, des pronoms ou des conjonctions. Pour les textes en français, une liste personnalisée de mots vides peut être utilisée, tandis que pour l'anglais, des listes standards comme celles proposées par `NLTK` ou `scikit-learn` sont disponibles. Ce nettoyage améliore significativement la qualité des représentations TF-IDF.

En complément, l'efficacité du filtrage basé sur le contenu repose sur la qualité du profil utilisateur. Celui-ci doit refléter avec précision les préférences spécifiques de l'utilisateur dans l'espace des caractéristiques. Une mauvaise représentation du profil utilisateur ou des caractéristiques non nettoyées peuvent fortement nuire à la pertinence des recommandations générées.

Plongements de mots (Word Embeddings)

Le *word embedding*, ou *plongement lexical*, est une technique de représentation vectorielle des mots largement utilisée en traitement automatique du langage naturel (TALN). L'idée centrale est de projeter chaque mot d'un vocabulaire V dans un espace vectoriel continu de dimension d (souvent $\mathbb{R}^d$), où les relations sémantiques et syntaxiques entre les mots sont préservées. Contrairement à une représentation *one-hot*, qui attribue à chaque mot un vecteur binaire et creux de taille $|V|$, les plongements produisent des vecteurs denses et continus qui permettent de mesurer la similarité entre les mots.
Ainsi, des mots apparaissant dans des contextes similaires auront des vecteurs proches dans cet espace, ce qui permet au système de « comprendre » qu'ils partagent un sens ou une fonction similaire. Cette proximité est généralement mesurée via la *similarité cosinus* :

$$\text{sim}_{\cos}(w_i, w_j) = \frac{v_{w_i}^\top \cdot v_{w_j}}{\|v_{w_i}\| \, \|v_{w_j}\|}, \tag{2.21}$$

où v_{w_i} et v_{w_j} sont les vecteurs des mots w_i et w_j.
Dans un système de filtrage basé sur le contenu, cette approche est particulièrement utile : si la description d'un élément (produit, article, film, etc.) contient un mot w, il est possible de trouver d'autres éléments dont la description contient des mots proches de w dans l'espace d'embedding, même si les termes exacts diffèrent. Par exemple, un utilisateur ayant apprécié un article mentionnant le mot « *cinéma* » pourra se voir recommander des articles contenant « *film* » ou « *projection* », si ces termes sont proches dans l'espace vectoriel.
Comme le résument Goldberg et Levy (2014) :

> “Les plongements de mots capturent des relations syntaxiques et sémantiques de manière que les représentations discrètes ne peuvent pas, permettant une généralisation au-delà des cooccurrences observées.”

Word2Vec et modélisation Skip-gram

Parmi les méthodes d'apprentissage de plongements, le modèle *Word2Vec* (Mikolov et al., 2013) est l'un des plus influents. Il existe principalement sous deux architectures : *Continuous Bag of Words* (CBOW) et *Skip-gram*. La version *Skip-gram*, que nous détaillons ici, apprend à prédire les mots du contexte à partir d'un mot central.

Soit une séquence de mots $(w_1, w_2, \dots, w_T)$ et une taille de fenêtre de contexte c. L'objectif est de maximiser la probabilité de prédire chaque mot du contexte w_{t+j} à partir du mot central w_t, avec $-c \leq j \leq c$ et $j \neq 0$:

$$J(\theta) = \frac{1}{T} \sum_{t=1}^{T} \sum_{\substack{-c \leq j \leq c \\ j \neq 0}} \log p(w_{t+j} \mid w_t; \theta). \tag{2.22}$$

La probabilité conditionnelle $p(w_O \mid w_I)$ est définie par une fonction softmax :

$$p(w_O \mid w_I) = \frac{\exp(v_{w_O}^\top v_{w_I})}{\sum_{w=1}^{W} \exp(v_w^\top v_{w_I})}, \tag{2.23}$$

où :

— v_{w_I} est le vecteur associé au mot d'entrée (mot central),

— v_{w_O} est le vecteur associé au mot de sortie (mot du contexte),

— W est la taille du vocabulaire.

Dans un système de recommandation basé sur le contenu, l'utilisation de *Word2Vec* permet de représenter les mots des descriptions d'éléments (produits, documents, etc.) dans un espace vectoriel où les relations de sens sont capturées. Les éléments peuvent alors être comparés non pas en fonction de mots strictement identiques, mais en fonction de la proximité de leurs représentations vectorielles. Cette approche augmente la capacité du système à proposer des recommandations pertinentes, même lorsque les descriptions diffèrent légèrement dans la formulation.

Ainsi, *Word2Vec* permet de dépasser les limites des simples modèles de cooccurrence, en offrant une représentation continue et généralisable du vocabulaire, ouvrant la voie à des recommandations plus riches et plus précises.

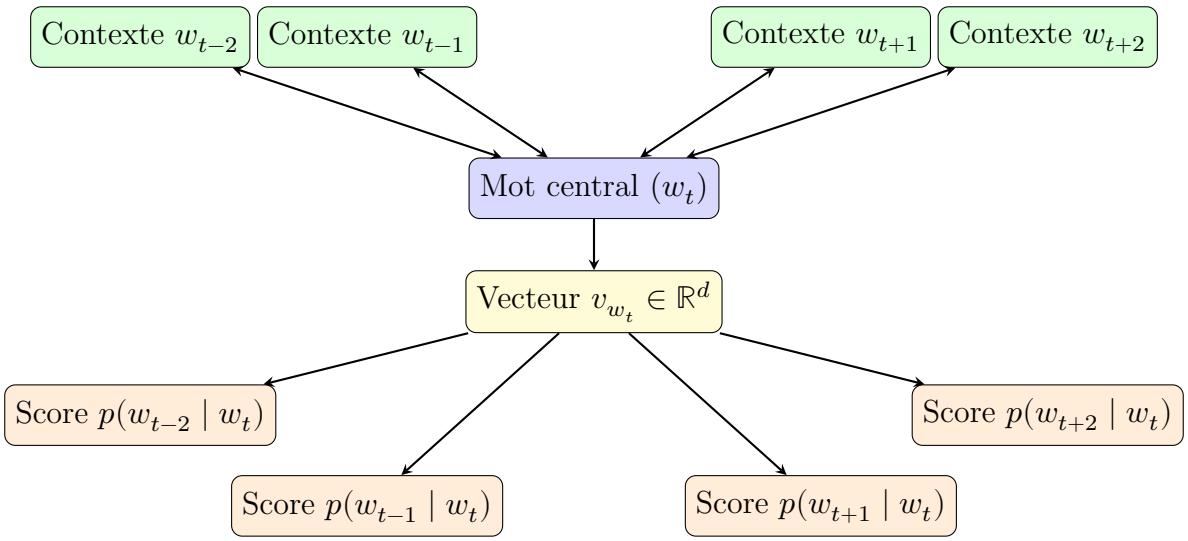

FIG. 2.4 – Schéma simplifié du modèle *Word2Vec (Skip-gram)* appliqué à des descriptions d'items dans un filtrage basé sur le contenu.

Distance de Jaccard

La distance de Jaccard est une mesure de dissimilarité utilisée principalement pour comparer des ensembles ou des vecteurs binaires. Elle repose sur l'idée de quantifier la proportion de désaccord entre deux ensembles en tenant compte à la fois de leurs éléments communs et de leurs éléments distincts.

Pour deux vecteurs binaires x_i et x_j, l'indice de Jaccard est défini comme le rapport entre la taille de l'intersection et celle de l'union des ensembles :

$$\text{Jaccard}(i,j) = \frac{|x_i \cap x_j|}{|x_i \cup x_j|} \tag{2.24}$$

La distance de Jaccard est alors obtenue en soustrayant cet indice de 1, ce qui donne :

$$\text{D}_{\text{Jaccard}}(i,j) = 1 - \frac{|x_i \cap x_j|}{|x_i \cup x_j|} \tag{2.25}$$

Cette distance est particulièrement adaptée à la comparaison de profils utilisateur ou d'objets représentés par des attributs binaires (présence/absence d'une caractéristique, d'un clic, etc.).

2.2.3 Apprentissage du profil utilisateur

Régression linéaire

Théorème 2.5 (Profil utilisateur par régression)**.** Le profil optimal d'un utilisateur u est obtenu en résolvant :

$$p_u^* = \arg\min_p \sum_{i \in \mathcal{I}_u} (r_{ui} - p^\top x_i)^2 + \lambda \|p\|^2 \tag{2.26}$$

La solution analytique est :

$$p_u^* = (X_u^\top X_u + \lambda I)^{-1} X_u^\top r_u \tag{2.27}$$

où X_u est la matrice des caractéristiques des items évalués par u et r_u est le vecteur des notes correspondantes.

Régression logistique pour la recommandation binaire

Dans le cas d'une recommandation binaire (aimer / ne pas aimer), on peut modéliser la probabilité d'appréciation d'un item i par un utilisateur u à l'aide de la régression logistique.

Définition 2.9 (Modèle de régression logistique)**.** La probabilité que l'utilisateur u aime l'item i, représenté par un vecteur de caractéristiques x_i, est donnée par :

$$P(y_{ui} = 1 \mid x_i, p_u) = \frac{1}{1 + \exp(-p_u^\top x_i)} = \sigma(p_u^\top x_i) \tag{2.28}$$

où $\sigma(z)$ est la fonction sigmoïde.
La log-vraisemblance associée à un ensemble d'observations $\{(x_i, y_{ui})\}_{i=1}^n$, avec $y_{ui} \in \{0, 1\}$, est :

$$\log \mathcal{L}(p_u) = \sum_{i=1}^{n} \left[y_{ui} \log \sigma(p_u^\top x_i) + (1 - y_{ui}) \log \left(1 - \sigma(p_u^\top x_i)\right) \right] \tag{2.29}$$

Cette fonction est ensuite maximisée pour estimer les préférences p_u de chaque utilisateur. La régression logistique est simple, efficace, et adaptée aux systèmes de recommandation à retour implicite.

2.2.4 Réduction de dimensionnalité

Analyse en composantes principales

L'Analyse en Composantes Principales (ACP), également appelée PCA *(Principal Component Analysis)*, est une technique statistique qui réduit la dimensionnalité des données en identifiant les axes qui maximisent la variance des données, permettant ainsi de capturer l'essentiel de l'information dans un nombre réduit de dimensions.

Théorème 2.6 (PCA pour la réduction de dimensionnalité)**.** L'ACP trouve la projection linéaire qui maximise la variance préservée :

$$\max_W \operatorname{tr}(W^\top \Sigma W) \quad \text{s.t.} \quad W^\top W = I \tag{2.30}$$

où Σ est la matrice de covariance des caractéristiques.

La solution est donnée par les k premiers vecteurs propres de Σ.

Analyse discriminante linéaire

Définition 2.10. L'Analyse Discriminante Linéaire *(Linear Discriminant Analysis (LDA))*, est une technique utilisée en apprentissage supervisé pour résoudre les problèmes de classification multi-classes.
Elle trouve la projection qui maximise la séparabilité entre classes :

$$\max_w \frac{w^\top S_B w}{w^\top S_W w} \tag{2.31}$$

où S_B est la matrice de variance inter-classes et S_W la matrice de variance intra-classes.

2.3 Approches hybrides et réseaux de neurones profonds

2.3.1 Stratégies d'hybridation

Approche pondérée

Définition 2.11 (Hybridation pondérée)**.** Les prédictions de différents algorithmes sont combinées linéairement :

$$\hat{r}_{ui} = \alpha \hat{r}_{ui}^{(CF)} + (1-\alpha)\hat{r}_{ui}^{(CB)}, \quad \alpha \in [0,1] \tag{2.32}$$

où $\hat{r}_{ui}^{(CF)}$ et $\hat{r}_{ui}^{(CB)}$ sont les prédictions issues respectivement du filtrage collaboratif et du filtrage basé sur le contenu.

Approche en cascade

Théorème 2.7 (Hybridation en cascade)**.** La méthode en cascade applique un algorithme primaire lorsque la confiance associée à sa prédiction dépasse un seuil τ, sinon un second algorithme est utilisé :

$$\hat{r}_{ui} = \begin{cases} \hat{r}_{ui}^{(1)} & \text{si confidence}(\hat{r}_{ui}^{(1)}) > \tau \\ \hat{r}_{ui}^{(2)} & \text{sinon} \end{cases} \tag{2.33}$$

avec $\tau \in [0,1]$.

2.3.2 Réseaux de neurones pour la recommandation

Perceptron multicouche

Le perceptron multicouche *(ou multilayer perceptron (MLP))* est un type de réseau neuronal artificiel organisé en plusieurs couches. Il possède au moins trois couches : une couche d'entrée, au moins une couche cachée, et une couche de sortie. L'information circule de la couche d'entrée vers la couche de sortie uniquement, ce qui en fait un réseau à propagation directe *(feedforward)*. Les neurones de la dernière couche sont les sorties du système global.

Il permet de surmonter les limites du perceptron à couche unique en offrant une puissance de calcul nettement supérieure. Il s'agit en effet d'un type de réseaux de neurones artificiels inspirés du fonctionnement des neurones biologiques. Il est organisé en plusieurs couches au sein desquelles circulent des données entre la première couche également appelée « couche d'entrée » et la couche de sortie *(output layer)* uniquement [2].

Définition 2.12 (MLP pour recommandation)**.** Un perceptron multicouche pour la recommandation peut être formalisé comme :

$$h^{(1)} = \sigma(W^{(1)}[p_u; q_i] + b^{(1)}) \tag{2.34}$$

$$h^{(2)} = \sigma(W^{(2)}h^{(1)} + b^{(2)}) \tag{2.35}$$

$$\hat{r}_{ui} = W^{(3)}h^{(2)} + b^{(3)} \tag{2.36}$$

où $[p_u; q_i]$ est la concaténation des vecteurs utilisateur et item, σ est la fonction d'activation, et $W^{(l)}, b^{(l)}$ sont les paramètres de la couche l.

[2] `https://www.jedha.co/formation-ia/algorithme-perceptron`

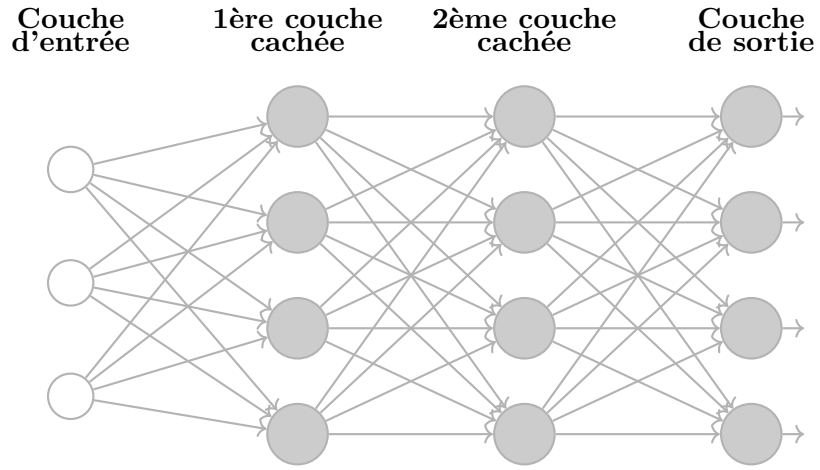

Illustration graphique d'un perceptron multicouche

Autoencodeurs pour la recommandation

Un auto-encodeur est un type de réseau neuronal utilisé dans le domaine de l'apprentissage non supervisé, notamment pour la modélisation des données et la réduction de leur dimensionnalité. Il est conçu pour reconstruire ses propres données d'entrée, ce qui lui permet de capturer les caractéristiques essentielles des données sans avoir besoin de labels. Dans le contexte des systèmes de recommandation, les auto-encodeurs sont utilisés pour améliorer la précision des recommandations en capturant les préférences des utilisateurs et les caractéristiques des items.

Réseaux de neurones convolutifs

Définition 2.13 (Utilisation des CNN pour la recommandation textuelle)**.** Les réseaux de neurones convolutifs (CNN) sont utilisés pour extraire automatiquement des représentations sémantiques à partir des descriptions textuelles des items. Une couche convolutive applique un filtre w sur une fenêtre de h mots consécutifs du texte :

$$c_i = \sigma(w \cdot x_{i:i+h-1} + b) \tag{2.37}$$

où :

- $x_{i:i+h-1}$ est la concaténation des vecteurs d'embedding de h mots consécutifs,
- w est un filtre convolutif appris durant l'entraînement,
- b est un biais,
- σ est une fonction d'activation non linéaire (souvent ReLU).

Ce mécanisme permet de capturer des motifs textuels importants pour prédire la pertinence d'un item vis-à-vis d'un utilisateur.

2.3.3 Factorisation matricielle neuronale

La factorisation matricielle neuronale vise à exploiter la puissance des réseaux de neurones pour capturer des interactions complexes entre utilisateurs et items, en allant au-delà des modèles linéaires classiques tels que ceux décrits dans [2].

Filtrage Collaboratif Neuronal *(Neural Collaborative Filtering (NCF))*

Proposé par He et al. [3], le NCF unifie la factorisation matricielle (FM) avec des réseaux neuronaux profonds afin de modéliser de manière non linéaire les interactions utilisateur-item. Le modèle NCF est formulé comme suit :

Définition 2.14 (NCF)**.** Le NCF combine deux branches : la factorisation matricielle généralisée (GMF) et un perceptron multicouche (MLP). La prédiction finale est obtenue par :

$$\hat{r}_{ui} = \sigma(h^\top [p_u^{GMF} \odot q_i^{GMF}; \text{MLP}(p_u^{MLP}, q_i^{MLP})]) \tag{2.38}$$

Où :

- p_u^{GMF}, q_i^{GMF} : vecteurs latents issus de la GMF,
- p_u^{MLP}, q_i^{MLP} : vecteurs latents en entrée du MLP,
- $\odot$: produit à composante (Hadamard),
- σ : fonction sigmoïde,
- h : vecteur de sortie linéaire.

Fonction de perte

Dans le cas d'un apprentissage implicite, la fonction de perte utilisée est une log-vraisemblance binaire, adaptée pour distinguer les interactions observées des interactions négatives [3] :

Théorème 2.8 (Optimisation NCF)**.**

$$L = -\sum_{(u,i)\in\mathcal{Y}\cup\mathcal{Y}^-} y_{ui} \log \hat{y}_{ui} + (1 - y_{ui}) \log(1 - \hat{y}_{ui}) \tag{2.39}$$

avec :

- $\mathcal{Y}$: ensemble des interactions positives observées,
- $\mathcal{Y}^-$: interactions négatives *négative sampling*,
- $y_{ui} \in \{0, 1\}$: label d'interaction,
- $\hat{y}_{ui}$: probabilité prédite que l'utilisateur u apprécie l'item i.

2.3.4 Réseaux de neurones graphiques (GNN)

Les approches neuronales récentes ont intégré la structure du graphe des interactions utilisateur-item via des réseaux de neurones graphiques [4].

Réseaux Convolutionnels Graphiques *(Graph Convolutional Networks (GCN))*

Les GCN modélisent la propagation d'information sur un graphe biparti utilisateur-item, en combinant les embeddings des voisins :

Définition 2.15 (GCN pour recommandation)**.**

$$h_u^{(l+1)} = \sigma\left(\sum_{i\in\mathcal{N}(u)} \frac{1}{\sqrt{|\mathcal{N}(u)||\mathcal{N}(i)|}} W^{(l)} h_i^{(l)} + b^{(l)}\right) \tag{2.40}$$

Où :

- $\mathcal{N}(u)$: items avec lesquels u a interagi,
- $h_i^{(l)}$: embedding de l'item i à la couche l,
- $W^{(l)}, b^{(l)}$: poids et biais à la couche l,
- σ : fonction d'activation (ex : ReLU).

LightGCN : Simplification des réseaux de neurones convolutifs sur graphes

Les modèles de type GCN ont récemment montré de très bonnes performances pour les systèmes de recommandation, en exploitant la structure de graphe biparti utilisateur-item. Cependant, ces modèles incluent généralement des fonctions d'activation non linéaires (comme ReLU) et des matrices de poids d'apprentissage à chaque couche, ce qui augmente leur complexité computationnelle sans toujours améliorer les performances.

Pour pallier cela, He et al. [5] ont proposé un modèle plus simple et plus efficace : LightGCN *(Light Graph Convolutional Network)*. Ce modèle élimine les fonctions d'activation et les matrices de poids intermédiaires, ne conservant que la phase essentielle : la propagation de voisinage.

Définition 2.16 (Propagation dans LightGCN)**.** À chaque couche $l+1$, les représentations latentes des utilisateurs et des items sont mises à jour uniquement par agrégation des représentations des nœuds voisins selon la règle :

$$e_u^{(l+1)} = \sum_{i \in \mathcal{N}(u)} \frac{1}{\sqrt{|\mathcal{N}(u)||\mathcal{N}(i)|}} e_i^{(l)} \tag{2.41}$$

$$e_i^{(l+1)} = \sum_{u \in \mathcal{N}(i)} \frac{1}{\sqrt{|\mathcal{N}(i)||\mathcal{N}(u)|}} e_u^{(l)} \tag{2.42}$$

où :

- $e_u^{(l)}$ et $e_i^{(l)}$ désignent les représentations (ou embeddings) de l'utilisateur u et de l'item i à la couche l.
- $\mathcal{N}(u)$ est l'ensemble des items interagis par l'utilisateur u, et $\mathcal{N}(i)$ l'ensemble des utilisateurs ayant interagi avec l'item i.
- Le facteur de normalisation $\frac{1}{\sqrt{|\mathcal{N}(u)||\mathcal{N}(i)|}}$ permet une mise à l'échelle équilibrée de l'information propagée selon la connectivité.

Définition 2.17 (Représentation finale)**.** La représentation finale d'un utilisateur u (resp. item i) est obtenue par une agrégation (souvent une moyenne) des représentations à chaque couche de propagation :

$$e_u = \sum_{l=0}^{L} \alpha_l e_u^{(l)}, \quad e_i = \sum_{l=0}^{L} \alpha_l e_i^{(l)} \tag{2.43}$$

où α_l est un poids d'attention associé à chaque couche l. Dans la pratique, on utilise souvent des poids uniformes $\alpha_l = \frac{1}{L+1}$.

LightGCN présente plusieurs avantages significatifs, tels que soulignés par He et al.[5]. Tout d'abord, sa simplicité est l'un de ses points forts : en supprimant les fonctions d'activation non linéaires et les matrices de transformation, LightGCN se concentre uniquement sur la propagation des informations de voisinage. Cette simplification réduit la complexité computationnelle du modèle, le rendant plus léger et plus rapide à entraîner, tout en conservant l'essentiel des signaux structurels du graphe (*cf.* Section 3.2 de [5]).
Ensuite, LightGCN se distingue par son efficacité en matière de performance. Il surpasse plusieurs variantes de GCN traditionnels sur de nombreux benchmarks de recommandation, notamment sur des ensembles de données implicites comme Amazon, Yelp ou Gowalla. Cela montre que même sans couches non linéaires, le modèle capture efficacement les relations utilisateur-item (*cf.* résultats expérimentaux dans la Section 5.2 de [5]).

Enfin, LightGCN démontre une meilleure capacité de généralisation. En effet, l'absence de transformations complexes permet de limiter les risques de surapprentissage, ce qui est confirmé empiriquement par les gains observés dans des contextes où les données sont fortement bruitées ou incomplètes (*cf.* discussion dans la Section 4 de [5]).

Fonction de perte. La fonction de perte utilisée est généralement une version du Classement Bayésien Personnalisé *(Bayesian Personalized Ranking (BPR))*, exprimée comme suit :

$$\mathcal{L}_{BPR} = -\sum_{(u,i,j)\in\mathcal{D}} \log \sigma\left(\hat{y}_{ui} - \hat{y}_{uj}\right) + \lambda\|\Theta\|^2 \tag{2.44}$$

où :

- $\hat{y}_{ui} = e_u^\top e_i$ est le score de préférence pour l'utilisateur u sur l'item i.
- (u, i, j) représente un triplet où i est un item cliqué (positif) et j un item non cliqué (négatif).
- σ est la fonction sigmoïde, et Θ les paramètres du modèle (ici les embeddings).
- λ est un coefficient de régularisation.

Ainsi, LightGCN représente une approche simple mais puissante, permettant d'exploiter efficacement la structure de graphe dans les systèmes de recommandation, tout en conservant une complexité algorithmique raisonnable.

Travaux réalisés

3.1 Reporting et automatisation

Au cours de mon stage, une part importante de mon travail a été consacrée à la gestion des rapports extraits de la base de données bancaire ORION. Ces rapports, principalement destinés au service Core Banking, permettent le suivi quotidien des opérations internes et externes de la banque. Le travail accompli s'est structuré en plusieurs phases successives, allant de l'extraction manuelle des données à la mise en place d'un processus d'automatisation partielle, avec des perspectives d'optimisation future.

1. Extraction et structuration des données

La première étape a consisté à rédiger et exécuter plusieurs requêtes SQL afin d'extraire les informations pertinentes des tables de la base ORION. Ces requêtes ont été élaborées sur mesure, en réponse aux besoins spécifiques exprimés par les utilisateurs métiers. Les données extraites concernaient, entre autres, les opérations bancaires journalières, l'état des comptes clients, ou encore certaines transactions internes.
Une fois collectées, les données ont été exportées et structurées sous forme de fichiers `.xlsx`, un format largement utilisé par les équipes pour leurs analyses et traitements quotidiens. Cette étape a constitué la base du processus de formalisation des rapports.

2. Création de tables dédiées et automatisation via procédures SQL

Pour pérenniser ce processus, j'ai ensuite créé de nouvelles tables dédiées dans la base ORION. Ces tables ont été conçues pour centraliser les résultats consolidés des différentes requêtes, facilitant ainsi leur exploitation à long terme.
Afin de les alimenter automatiquement, j'ai développé une procédure stockée incluant des curseurs SQL. Ceux-ci permettent de parcourir dynamiquement les résultats et de gérer les opérations d'insertion, de mise à jour ou de suppression selon les règles définies. Cette procédure a ensuite été encapsulée dans un job SQL programmé pour s'exécuter automatiquement chaque jour à 6h00 du matin, garantissant ainsi la production régulière d'un rapport bancaire mis à jour sans intervention manuelle.

3. Intégration et visualisation des rapports via Power BI

Une fois les rapports générés, ils sont intégrés dans Power BI, un outil de visualisation interactif permettant la création de tableaux de bord dynamiques et personnalisés. La mise en forme des données dans Power BI repose sur une requête préalablement élaborée par mon maître de stage et ses collaborateurs, ce qui permet d'automatiser l'organisation des visuels et d'assurer une cohérence dans la présentation.
Les rapports ainsi produits sont ensuite exportés manuellement au format PDF, puis envoyés par email aux différents responsables concernés. Bien que cette diffusion ne soit pas encore automatisée, elle constitue une étape clé permettant une prise de décision rapide, fondée sur des données fiables et actualisées quotidiennement.

4. Automatisation complète : objectif partiellement atteint

Dans une volonté de perfectionnement, j'avais prévu d'automatiser intégralement le processus de diffusion des rapports en reliant Power BI à Power Automate. L'idée était de déclencher automatiquement l'envoi quotidien des fichiers PDF une fois le job SQL terminé. Cette automatisation aurait permis de supprimer totalement l'intervention manuelle dans la chaîne de production.
Cependant, en raison de contraintes de temps et de la complexité de certaines configurations liées à Power Automate, cette dernière phase n'a pas pu être mise en œuvre durant mon stage. Elle demeure néanmoins une piste d'amélioration à intégrer ultérieurement pour renforcer encore davantage l'autonomie et la robustesse du système.

3.2 Mise en place d'un système de recommandation

3.2.1 Système de recommandation hybride à espace d'embedding unifié

Parmi les approches étudiées, nous avons retenu un système hybride reposant sur un espace d'embedding unique, intégrant à la fois les caractéristiques des clients et les produits qu'ils détiennent. Ce choix répond à deux contraintes majeures des données que nous avions à disposition à la BNI : l'absence de descriptions détaillées des produits et le manque d'historique d'appréciations, rendant inefficaces les approches purement basées sur le contenu ou collaboratives.
Dans ce système, le profil de chaque client est constitué d'attributs sociodémographiques (âge, sexe, profession, segment, ancienneté, etc.) et de la liste des produits détenus. Après prétraitement textuel (mise en minuscules, suppression de la ponctuation et normalisation des espaces), ces informations sont converties en une séquence de tokens, chaque attribut et produit correspondant à un token distinct.
Ces tokens sont ensuite représentés par des vecteurs denses grâce à un modèle *Skip-gram* entraîné sur l'ensemble des profils et produits. La représentation finale du client est obtenue par agrégation des embeddings de ses tokens, avec une pondération favorisant les informations rares mais discriminantes. La similarité entre clients est ensuite mesurée afin d'identifier les voisins les plus proches d'un client cible.
Les produits recommandés sont sélectionnés parmi ceux absents du portefeuille du client mais présents chez ses voisins, puis rerankés via la méthode de la Pertinence Marginale Maximale *(Maximal Marginal Relevance, MMR)* [33], pour équilibrer pertinence et diversité. Cette approche permet de privilégier les produits les plus représentatifs des profils proches tout en évitant la redondance dans la liste de recommandation.
Enfin, l'évaluation des recommandations s'appuie sur plusieurs métriques opérationnelles : la similarité moyenne entre produits recommandés et produits possédés, la diversité intra-liste et la couverture globale. Si les résultats ne sont pas satisfaisants, le processus peut être réajusté, ce qui assure une amélioration itérative des recommandations.
Cette approche unifiée offre une solution robuste et flexible, adaptée aux profils peu renseignés et performante même en situation de *cold start*, là où les approches classiques seraient limitées.

3.2.2 Description et prétraitement des données

L'ensemble de données utilisé dans ce projet provient de la BNI et regroupe l'historique des interactions entre les clients et les produits bancaires proposés par l'établissement. Afin de garantir la confidentialité et de respecter les réglementations en matière de protection

des données personnelles, toutes les informations permettant d'identifier directement un client ont été anonymisées.

Avant toute manipulation, le jeu de données consolidé présentait une taille de (337 712, 21), correspondant respectivement au nombre de lignes (observations) et au nombre de colonnes (variables). Ces variables incluaient aussi bien des informations sociodémographiques (âge, sexe, ancienneté) que des métadonnées produits et des traces d'interactions. Une première exploration, réalisée en `Python`, a permis d'identifier les attributs pertinents, de détecter certaines incohérences de formatage et de mieux comprendre la distribution des données. Les résultats de cette analyse exploratoire ont guidé la sélection des colonnes utiles à la construction du système de recommandation.

L'analyse descriptive met en évidence que les comptes d'épargne particuliers constituent le produit le plus souscrit, avec environ 80 000 ouvertures, loin devant les cartes Visa et d'autres produits bancaires classiques (Figure 3.1). Par ailleurs, la répartition par tranche d'âge (Figure 3.2) montre une prédominance des clients âgés de 35 à 44 ans (plus de 110 000 souscripteurs), suivis par la tranche 45-54 ans ($\approx$ 78 000), confirmant une clientèle globalement mature et active professionnellement.

En termes de distribution fine par âge (Figure 3.3), la concentration maximale se situe entre 35 et 45 ans (environ 39 000 clients), tandis que les moins de 25 ans et les plus de 70 ans restent minoritaires. L'ancienneté dans la relation bancaire (Figure 3.4) révèle quant à elle une majorité de clients récents, avec un pic notable entre 0 et 3 ans, puis une décroissance progressive au-delà de 20 ans.

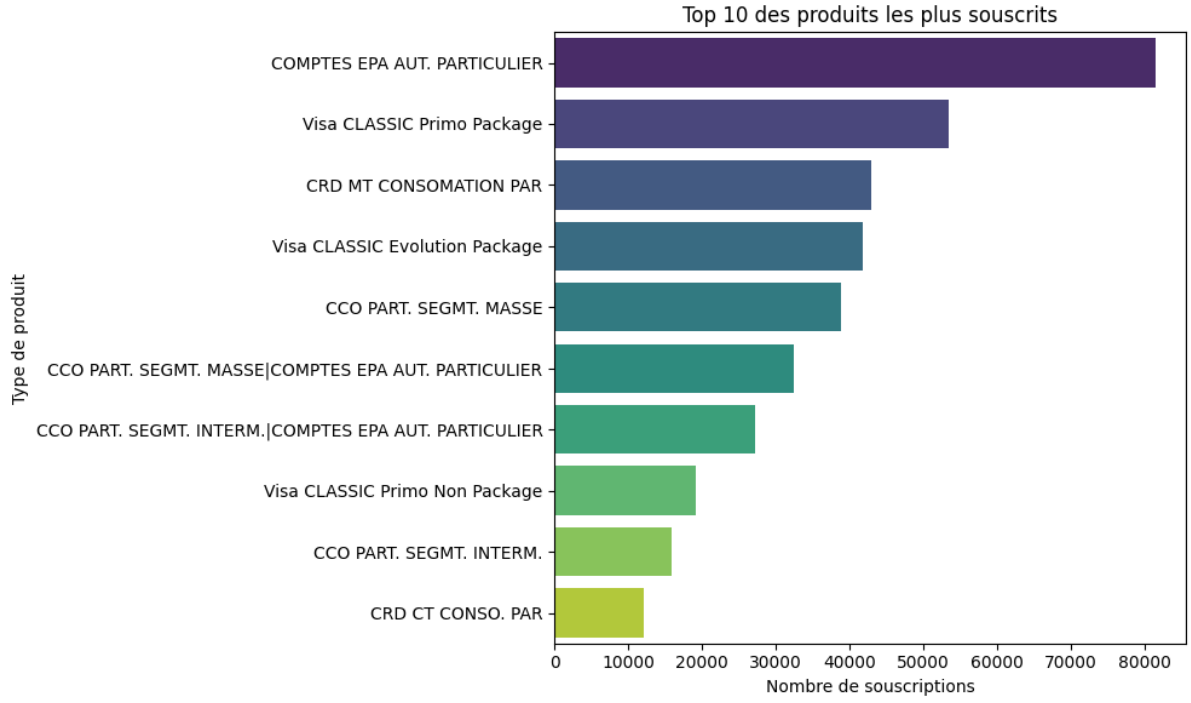

Fig. 3.1 – Top 10 des produits les plus souscrits

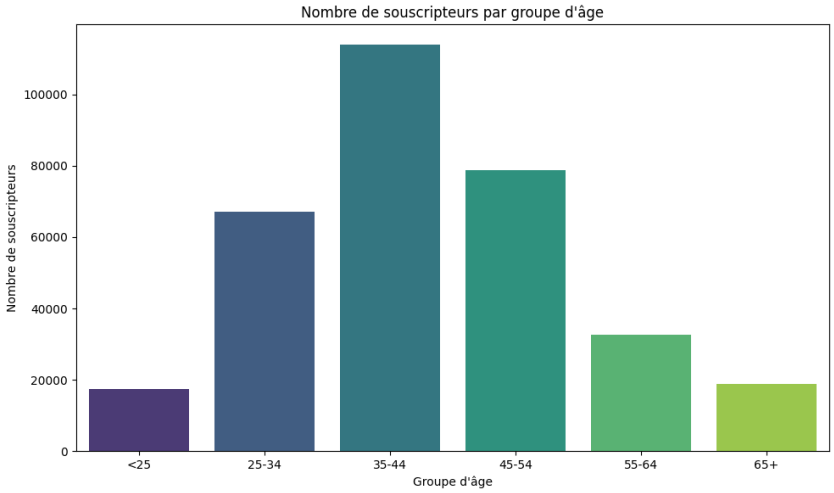

Fig. 3.2 – Répartition des clients par tranche d'âge

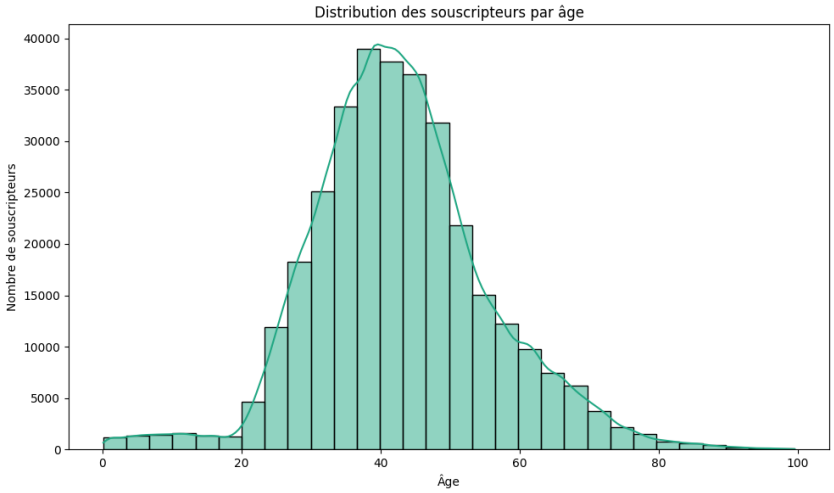

Fig. 3.3 – Distribution du nombre de souscripteurs par âge

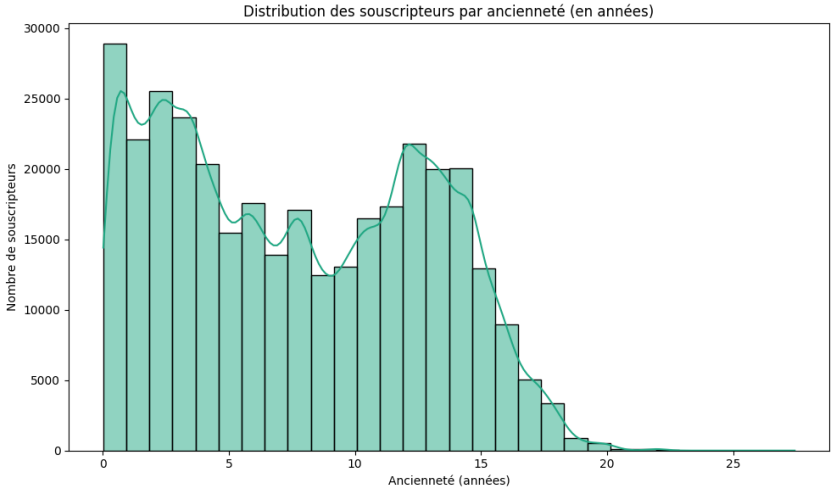

Fig. 3.4 – Distribution du nombre de souscripteurs par ancienneté

Une fois la phase descriptive des données réalisée, un prétraitement rigoureux a été appliqué afin de préparer la mise en œuvre du système de recommandation hybride. Les informations

textuelles des profils clients et des produits ont d'abord été nettoyées : conversion en minuscules, suppression des caractères spéciaux, normalisation des espaces et élimination des termes rares. Les mots vides *(stop words)*, spécifiques au français et au contexte bancaire, ont été retirés à l'aide d'un dictionnaire personnalisé, réduisant ainsi le bruit dans les représentations vectorielles.
Ces tokens nettoyés ont ensuite été convertis en vecteurs numériques grâce à un modèle *Skip-gram*, produisant un espace d'embedding unifié pour clients et produits. Cette représentation vectorielle constitue la base des calculs de similarité entre clients, ainsi que de la sélection et du reranking des produits recommandés à l'aide du KNN et de la méthode MMR.

3.2.3 Implémentation de la recommandation basée sur le contenu (formalisme mathématique)

Dans le moteur de recommandation proposé, on note c, un client donné. Ainsi, un modèle *Word2Vec (Skip-gram)* entraîné sous Keras associe à chaque token t un vecteur dense $e_t \in \mathbb{R}^d$ dans un espace d'embedding commun.

Construction du vecteur client. Soit T_c l'ensemble des tokens observés pour le client c. Le vecteur représentant c est obtenu par la moyenne simple des embeddings des tokens :

$$v_c \;=\; \frac{1}{|T_c|} \sum_{t \in T_c} e_t.$$

(On notera que si $T_c = \emptyset$ "cas *cold-start*", alors v_c est considéré comme vecteur nul et un mécanisme look-alike est utilisé, voir ci-dessous.)

Voisinage produit (cas client avec historique). Pour un client c disposant d'un vecteur v_c non nul, on calcule les similarités cosinus entre v_c et les vecteurs clients de la base, puis on agrège les produits observés chez les voisins pour former un score de candidature. Soit C l'ensemble de tous les clients et v_j le vecteur du client $j \in C$. On note :

$$s_{c,j} \;=\; \mathrm{sim}(v_c, v_j),$$

où $\mathrm{sim}(v_c, v_j)$ est la similarité cosinus entre les vecteurs v_c et v_j.
Pour tout index j tel que $s_{c,j} \geq \tau_{\min}$ (seuil de similarité), on ajoute aux scores des produits p possédés par j une contribution proportionnelle à $s_{c,j}$. En pratique :

$$\mathrm{score\_prod}(p) \;=\; \sum_{j : p \in P_j} \alpha \cdot s_{c,j},$$

où P_j est l'ensemble des produits de j et α est un facteur de normalisation (souvent $\alpha = 0.5$ dans l'implémentation pour atténuer la contribution par voisin).
Les scores sont ensuite normalisés en mode « relative » (diviser par le maximum) si nécessaire :

$$\widetilde{\mathrm{score\_prod}}(p) \;=\; \frac{\mathrm{score\_prod}(p)}{\max_q \mathrm{score\_prod}(q)}.$$

Look-alike par profil (*cold-start*). Pour un client c sans produits, on encode son profil socio-démographique via un encodage des variables catégorielles (one-hot) et une normalisation des variables numériques. Soit X l'espace profil et x_c la représentation profil de c. On construit un kNN en distance cosinus (ou euclidienne sur vecteurs normalisés) et on récupère les k plus proches voisins $\mathcal{N}_k(c) = \{j_1, \dots, j_k\}$ avec distances $d_{c,j}$.
On transforme les distances en poids de voisinage :

$$w_{c,j} \;=\; \max(1 - d_{c,j},\, 0).$$

Pour limiter les voisins effectifs, on ne conserve que le plus petit préfixe des voisins dont la somme des poids couvre une fraction τ (ex. $\tau = 0.8$) du poids total. Formellement, soit l'ordre des voisins trié par poids décroissant $(j_{(1)}, j_{(2)}, \dots)$ et $W_m = \sum_{r=1}^{m} w_{c,j_{(r)}}$. On choisit $m_{\text{eff}} = \min\{m : W_m \geq \tau \cdot W_k\}$ et la liste de voisins effectifs est $\mathcal{N}_{\text{eff}}(c) = \{j_{(1)}, \dots, j_{(m_{\text{eff}})}\}$.
La contribution de ces voisins aux scores produits s'écrit alors :

$$\text{base\_score}(p) \;=\; \sum_{j \in \mathcal{N}_{\text{eff}}(c)} w_{c,j} \cdot \mathbb{1}_{\{p \in P_j\}}.$$

On normalise ensuite par la somme des poids effectifs $W_{m_{\text{eff}}}$ pour obtenir une proportion pondérée :

$$\widehat{\text{score}}(p) \;=\; \frac{\text{base\_score}(p)}{W_{m_{\text{eff}}}}.$$

Pondération anti-popularité (IDF douce). Pour réduire la propension à recommander les produits sur-représentés, on applique un facteur $\text{idf}(p) \in [0, 1]$ (normalisé) :

$$\text{score\_idf}(p) \;=\; (1 - \alpha) \cdot \widehat{\text{score}}(p) \;+\; \alpha \cdot \widehat{\text{score}}(p) \cdot \text{idf}(p),$$

où $\alpha \in [0, 1]$ règle la force de la pénalisation des produits populaires (si $\alpha = 0$ pas d'effet IDF, si $\alpha = 1$, on a un fort effet anti-popularité).

Reranking par MMR (diversification). Pour produire la liste finale de recommandations et contrôler la redondance, on applique un algorithme de reranking *Maximal Marginal Relevance* (MMR) [33]. Notons C l'ensemble des candidats (produits) triés par score décroissant issu des étapes précédentes, et S l'ensemble courant des items déjà sélectionnés. À chaque itération on choisit l'item $p^\star$ maximisant :

$$p^\star \;=\; \arg\max_{p \in C \setminus S} \; \lambda \cdot \text{rel}(p) \;-\; (1 - \lambda) \cdot \max_{q \in S} \text{sim}(e_p, e_q),$$

où $\text{rel}(p)$ est le score de pertinence normalisé (par exemple $\text{score\_idf}(p)$) et $\lambda \in [0, 1]$ contrôle le compromis pertinence/diversité. Initialement $S = \emptyset$ et l'on répète jusqu'à obtenir le top-N. Ce procédé garantit que chaque nouvel item apporte une utilité marginale tout en restant pertinent.

Évaluation (indicateurs qualitatifs). Étant donné les contraintes sur le jeu de données, l'évaluation se fait via trois indicateurs calculés sur l'ensemble de test C_{test} :

— **Similarité moyenne aux produits possédés.** Pour un client c avec produits possédés P_c et recommandations R_c :

$$S(c) \;=\; \frac{1}{|R_c|} \sum_{p \in R_c} \left(\frac{1}{|P_c|} \sum_{q \in P_c} \text{sim}(e_p, e_q) \right).$$

On reporte ensuite la moyenne $\overline{S} = \frac{1}{|C_{\text{test}}|} \sum_{c \in C_{\text{test}}} S(c)$.

— **Diversité intra-liste.** Pour un client c :

$$D(c) \;=\; \frac{1}{|R_c|(|R_c|-1)} \sum_{\substack{p,q \in R_c \\ p \neq q}} \mathrm{sim}(e_p, e_q).$$

Une valeur faible de $D(c)$ indique une forte diversité. On reporte $\overline{D} = \frac{1}{|C_{\text{test}}|} \sum_{c \in C_{\text{test}}} D(c)$.

— **Couverture globale.** Soit $\mathcal{P}$ l'ensemble complet des produits. La couverture est :

$$\mathrm{Cov} \;=\; \frac{\left|\bigcup_{c \in C_{\text{test}}} R_c\right|}{|\mathcal{P}|},$$

c'est-à-dire la proportion de produits du catalogue apparaissant au moins une fois dans les recommandations.

Ces métriques permettent d'évaluer l'équilibre entre *pertinence* (similarité), *diversité* (MMR) et *couverture* (exploration du catalogue), critères essentiels dans un moteur de recommandation bancaire.

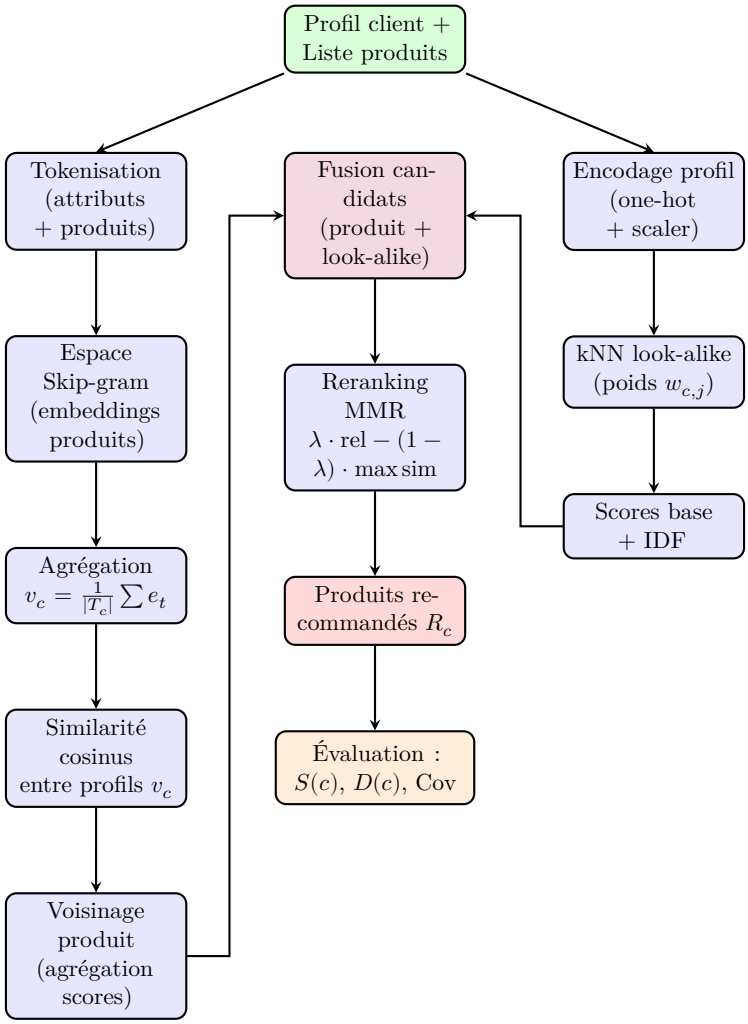

Fig. 3.5 – Pipeline de recommandation hybride : traitement parallèle des embeddings produits *(Skip-gram)* et des profils look-alike (kNN), fusion des candidats, diversification MMR et évaluation des performances.

Résultats et bilan du stage

Ce chapitre présente de manière synthétique les résultats obtenus au cours des expérimentations menées durant mon stage. L'évaluation porte sur le système de recommandation hybride combinant embeddings unifiés, recherche par plus proches voisins (*KNN*) et réordonnancement par *MMR* (*Maximal Marginal Relevance*). Les analyses mettent en avant la pertinence, la diversité et la couverture des recommandations, tout en identifiant les points forts et les limites de la solution proposée.

4.1 Synthèse des résultats

Afin d'illustrer concrètement le fonctionnement du moteur de recommandation, le tableau ci-dessous présente les résultats obtenus pour les cinq premiers clients de notre dictionnaire de données. Pour chacun d'eux, les produits déjà possédés sont indiqués, suivis des cinq recommandations principales classées par ordre décroissant de score de similarité.

Client	Produits déjà possédés	Recommandations avec score
1	cco part segmt masse	1. comptes epa aut particulier (1.0000) 2. cco part segmt interm (0.5816) 3. visa classic primo non package (0.2203) 4. plans epargne logement (0.0993) 5. crd mt consomation par (0.0525)
2	comptes epa aut particulier	1. cco part segmt masse (1.0000) 2. cco part segmt interm (0.8453) 3. visa classic primo non package (0.4503) 4. eburnie bronze emv (0.3715) 5. crd mt consomation par (0.0739)
3	cco part segmt interm plus, comptes epa aut particulier, visa classic primo non package	1. cco part segmt masse (1.0000) 2. cco part segmt interm (0.8362) 3. crd mt consomation par (0.2050) 4. plans epargne logement (0.2034) 5. plan epa autre credit (0.0954)
4	cco part segmt interm, crd ct conso par, plans epargne logement, visa classic primo non package, visa electron	1. comptes epa aut particulier (1.0000) 2. cco part segmt masse (0.2930) 3. crd mt consomation par (0.0806) 4. plan epa autre credit (0.0302) 5. visa classic primo package (0.0066)
5	(aucun)	1. comptes epa aut particulier (1.0000) 2. visa classic evolution package (0.5174) 3. cco part segmt interm (0.5130) 4. plan epa autre credit (0.2578) 5. crd mt consomation par (0.2565)

Commentaires : Les résultats montrent que le moteur de recommandation parvient à identifier des produits cohérents avec le profil des clients. Par exemple, un client ne possédant encore aucun produit (Client 5) se voit proposer en priorité l'ouverture d'un compte épargne particulier, ce qui correspond à une étape d'entrée classique dans une relation bancaire. De même, les clients disposant déjà d'une base de produits (Clients 2

et 3) se voient suggérer des cartes bancaires ou des crédits, ce qui illustre la logique de montée en gamme.
Cependant, certaines recommandations restent redondantes ou peu pertinentes (ex : produits très proches déjà détenus par le client), ce qui met en évidence la nécessité d'améliorer la diversité et la personnalisation fine, notamment via un meilleur équilibrage entre similarité et couverture.

4.2 Compétences développées

Au-delà des aspects purement techniques, ce stage m'a permis de développer un ensemble de compétences variées, à la fois professionnelles et personnelles.

Compétences techniques. J'ai renforcé ma maîtrise des outils de traitement et d'analyse de données, notamment à travers l'usage de Python (pandas, scikit-learn, tensorflow ou keras), SQL et Power BI. La mise en œuvre d'algorithmes de recommandation m'a permis d'approfondir mes connaissances en apprentissage automatique et en modélisation mathématique. J'ai également acquis de l'expérience dans le prétraitement de données réelles, une étape souvent complexe mais essentielle pour garantir la qualité des résultats. Enfin, j'ai appris à intégrer différents outils au sein d'un même flux de travail, de l'extraction de données jusqu'à la visualisation des résultats.

Compétences transversales. Ce stage m'a offert l'opportunité de progresser sur des aspects non techniques mais tout aussi essentiels. J'ai appris à gérer mon temps de manière plus rigoureuse et à m'adapter aux contraintes propres au milieu professionnel. Le travail en collaboration avec les équipes de la DSI m'a permis de développer mes capacités de communication et d'adaptation, en vulgarisant des notions techniques auprès de publics non spécialistes. J'ai aussi pris conscience de l'importance de la clarté et de la précision dans la présentation des résultats.

Bilan personnel. Sur le plan personnel, cette expérience a été très enrichissante. Elle m'a permis de confirmer mon intérêt pour la data science appliquée au secteur bancaire, et de prendre confiance en ma capacité à contribuer à des projets à fort impact. J'ai apprécié la diversité des missions qui m'ont été confiées, alliant analyse de données, développement et restitution auprès des décideurs. Ce stage représente une étape déterminante de mon parcours, car il m'a donné une vision concrète de l'application des sciences des données dans un contexte professionnel exigeant et en pleine transformation.

4.3 Difficultés rencontrées

Comme tout projet technique, la mise en œuvre de ce travail a été confrontée à plusieurs difficultés qu'il a fallu surmonter au fil du stage.

Disponibilité et qualité des données. L'une des principales difficultés rencontrées concernait l'accès et la qualité des données. Certaines informations étaient incomplètes, hétérogènes ou dispersées dans différents systèmes. Leur extraction et leur préparation ont demandé un effort considérable de nettoyage et de structuration avant de pouvoir être utilisées pour l'analyse et la modélisation. Cette étape m'a appris à être patient et méthodique, et à utiliser des techniques de prétraitement adaptées.

Contraintes techniques et matérielles. J'ai également dû faire face à certaines limitations techniques liées aux environnements de travail et aux ressources disponibles. Par exemple, la capacité de calcul parfois limitée ralentissait l'entraînement des modèles de machine learning. Cela m'a conduit à optimiser le code, à privilégier des approches plus légères et à exploiter au mieux les ressources disponibles.

Apprentissage et montée en compétences. La diversité des outils mobilisés (SQL, Power BI, Python, etc.) représentait un autre défi. Si j'avais déjà des bases dans certains, d'autres m'étaient nouveaux ou peu familiers. J'ai dû apprendre rapidement, souvent en autonomie, en m'appuyant sur la documentation (internet) et sur les conseils de mon encadrant. Ce fut le cas par exemple du développement de l'application web (*streamlit*) pour la présentation de notre modèle. Ainsi, ces difficultés de base se sont transformées en une opportunité de progression significative.

Adaptation au contexte professionnel. Enfin, il m'a fallu m'adapter au rythme et aux exigences d'un environnement bancaire structuré, avec des délais, des contraintes de sécurité et des processus stricts. Cette expérience m'a appris à mieux communiquer mes besoins, à prioriser les tâches et à travailler en coordination avec différentes équipes.

Ces difficultés, loin de constituer des obstacles insurmontables, ont au contraire renforcé mes compétences techniques et ma capacité d'adaptation. Elles ont donné à ce stage toute sa valeur formatrice, en m'apportant une vision réaliste des défis rencontrés dans la mise en place de projets de data science en entreprise.

Conclusion et perspectives

Ce mémoire a présenté les travaux réalisés au cours de mon stage de fin d'études au sein de la Banque Nationale d'Investissement (BNI). Deux contributions principales ont été apportées : d'une part, l'automatisation des processus de reporting afin de faciliter la prise de décision, et d'autre part, la conception d'un prototype de système de recommandation de produits bancaires. Ces travaux ont mobilisé des compétences variées allant du traitement de données à la mise en œuvre de techniques d'apprentissage automatique.

Au-delà des résultats techniques obtenus, ce stage m'a permis de développer des compétences transversales précieuses : rigueur dans la conduite de projet, autonomie, et capacité à collaborer efficacement au sein d'une équipe pluridisciplinaire. J'ai également pu constater concrètement l'impact que peuvent avoir les sciences des données dans un environnement bancaire en pleine mutation, notamment en matière de personnalisation de l'offre et d'amélioration de l'expérience client.

Ce travail a confirmé mon intérêt pour la data science et son application au secteur financier. Pour les perspectives futures, plusieurs pistes d'amélioration sont envisageables : l'enrichissement des données utilisées (en intégrant par exemple des informations comportementales ou contextuelles ainsi que la description détaillée des produits), l'expérimentation de modèles avancés tels que les réseaux de neurones profonds, ainsi que la mise en production d'un moteur de recommandation opérationnel.

En conclusion, ce stage a représenté une étape déterminante de mon parcours, en me permettant de mettre en pratique mes connaissances théoriques et d'acquérir une expérience concrète au sein d'une institution bancaire de premier plan. Il constitue une base solide pour mes futurs projets professionnels dans le domaine de la science des données appliquée à la finance.

Annexes

Visualisation des embeddings produits

La figure 1 illustre la projection bidimensionnelle des embeddings des produits bancaires, obtenue à l'aide de l'algorithme *t-SNE* (*t-distributed Stochastic Neighbor Embedding*) [34]. Cette méthode de réduction de dimension permet de représenter les vecteurs de grande dimension associés aux produits tout en conservant leurs similarités locales. Ainsi, les produits similaires forment un cluster.

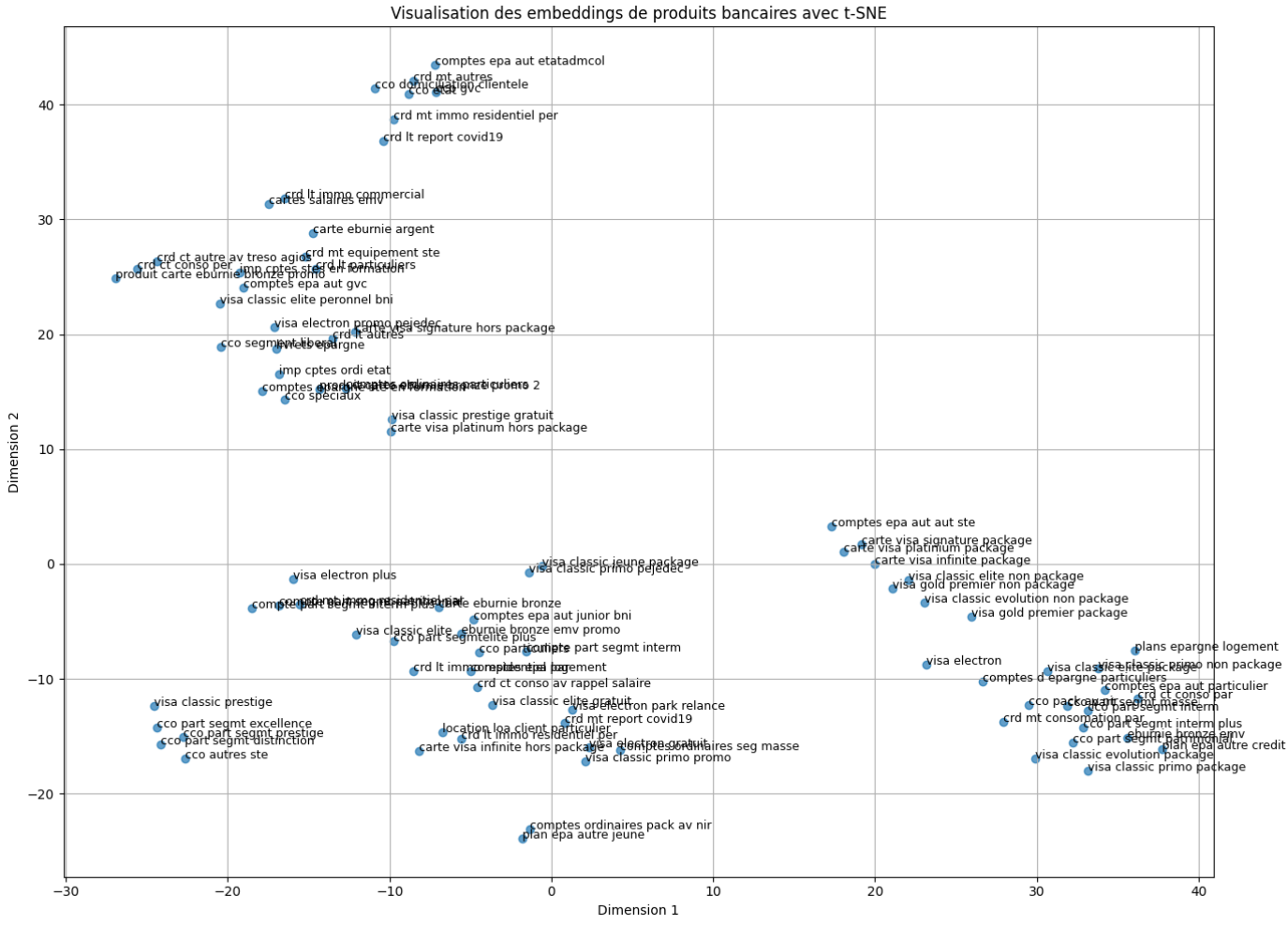

Fig. 1 – Projection 2D des embeddings des produits bancaires via t-SNE

Code source

Voici le lien vers mon profil GitHub où j'ai mis les codes source des algorithmes présentés : github.com/gerardkra/Code-source-Stage

Bibliographie

[1] Ricci, F., Rokach, L., & Shapira, B. (2015). *Recommender Systems Handbook*. Springer.

[2] Koren, Y., Bell, R., & Volinsky, C. (2009). Matrix factorization techniques for recommender systems. *Computer*, 42(8), 30–37.

[3] He, X., Liao, L., Zhang, H., Nie, L., Hu, X., & Chua, T.-S. (2017). Neural collaborative filtering. In *Proceedings of the 26th International Conference on World Wide Web* (pp. 173–182).

[4] Wang, X., He, X., Wang, M., Feng, F., & Chua, T.-S. (2019). Neural graph collaborative filtering. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 165–174).

[5] He, X., Deng, K., Wang, X., Li, Y., Zhang, Y., & Wang, M. (2020). LightGCN : Simplifying and powering graph convolution network for recommendation. In *Proceedings of the 43rd International ACM SIGIR Conference* (pp. 639–648).

[6] Salakhutdinov, R., Mnih, A., & Hinton, G. (2007). Restricted Boltzmann machines for collaborative filtering. In *Proceedings of the 24th International Conference on Machine Learning* (pp. 791–798).

[7] Funk, S. (2006). Netflix update : Try this at home. *Blog post*, `https://sifter.org/~simon/journal/20061211.html`.

[8] Hu, Y., Koren, Y., & Volinsky, C. (2008). Collaborative filtering for implicit feedback datasets. In *Proceedings of the 2008 IEEE International Conference on Data Mining (ICDM)* (pp. 263–272). IEEE.

[9] Sedhain, S., Menon, A. K., Sanner, S., & Xie, L. (2015). AutoRec : Autoencoders meet collaborative filtering. In *Proceedings of the 24th International Conference on World Wide Web* (pp. 111–112).

[10] Sarwar, B., Karypis, G., Konstan, J., & Riedl, J. (2001). Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th International Conference on World Wide Web* (pp. 285–295).

[11] Lops, P., De Gemmis, M., & Semeraro, G. (2011). Content-based recommender systems : State of the art and trends. In F. Ricci et al. (Eds.), *Recommender Systems Handbook* (pp. 73–105). Springer.

[12] Burke, R. (2002). Hybrid recommender systems : Survey and experiments. *User Modeling and User-Adapted Interaction*, 12(4), 331–370.

[13] Covington, P., Adams, J., & Sargin, E. (2016). Deep neural networks for YouTube recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems* (pp. 191–198).

[14] Zhou, G., Zhu, X., Song, C., Fan, Y., Zhu, H., Ma, X., ... & Gai, K. (2018). Deep interest network for click-through rate prediction. In *Proceedings of the 24th ACM SIGKDD Conference* (pp. 1059–1068).

[15] Cheng, H.-T., Koc, L., Harmsen, J., Shaked, T., Chandra, T., Aradhye, H., ... & Shah, H. (2016). Wide & deep learning for recommender systems. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems* (pp. 7–10).

[16] Aciar, S., Zhang, D., Simoff, S., & Debenham, J. (2007). Informed recommender : Basing recommendations on consumer product reviews. *IEEE Intelligent Systems*, 22(3), 39–47.

[17] Park, D. H., Kim, H. K., Choi, I. Y., & Kim, J. K. (2012). A literature review and classification of recommender systems research. *Expert Systems with Applications*, 39(11), 10059–10072.

[18] Rendle, S. (2010). Factorization machines. In *Proceedings of the 2010 IEEE International Conference on Data Mining* (pp. 995–1000).

[19] Provost, F., & Fawcett, T. (2013). *Data Science for Business : What You Need to Know About Data Mining and Data-Analytic Thinking*. O'Reilly Media.

[20] Davenport, T. H., & Ronanki, R. (2018). Artificial intelligence for the real world. *Harvard Business Review*, 96(1), 108–116.

[21] Hidasi, B., Karatzoglou, A., Baltrunas, L., & Tikk, D. (2015). Session-based recommendations with recurrent neural networks. *arXiv preprint arXiv :1511.06939*.

[22] Mnih, A., & Salakhutdinov, R. R. (2007). Probabilistic matrix factorization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 20, 1257–1264.

[23] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. In *Advances in Neural Information Processing Systems* (pp. 5998–6008).

[24] Kang, W.-C., & McAuley, J. (2018). Self-attentive sequential recommendation. In *2018 IEEE International Conference on Data Mining (ICDM)* (pp. 197–206).

[25] Chen, Q., Zhao, H., Li, W., Huang, P., & Ou, W. (2019). Behavior sequence transformer for e-commerce recommendation in Alibaba. In *Proceedings of the 1st International Workshop on Deep Learning Practice for High-Dimensional Sparse Data* (pp. 1–4).

[26] Sun, F., Liu, J., Wu, J., Pei, C., Lin, X., Ou, W., & Jiang, P. (2019). BERT4Rec : Sequential recommendation with bidirectional encoder representations from transformer. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management* (pp. 1441–1450).

[27] Salton, G., & Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. *Information Processing & Management*, 24(5), 513–523.

[28] Pazzani, M. J., & Billsus, D. (2007). Content-based recommendation systems. In *The Adaptive Web*, Springer, pp. 325–341.

[29] Jannach, D., Zanker, M., Felfernig, A., & Friedrich, G. (2010). *Recommender Systems : An Introduction*. Cambridge University Press.

[30] Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv :1301.3781.*

[31] Bengio, Y., Ducharme, R., Vincent, P., & Jauvin, C. (2003). A neural probabilistic language model. *Journal of Machine Learning Research*, 3, 1137–1155.

[32] LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, 521(7553), 436–444.

[33] Carbonell, Jaime and Goldstein, Jade. (1998). Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval. *The use of MMR, diversity-based reranking for reordering documents and producing summaries*, 335–336.

[34] Van der Maaten, L., & Hinton, G. (2008). Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9, 2579–2605.