



MEMOIRE DE FIN D'ÉTUDES

Pour l'obtention du Master :

Data Science et Intelligence Artificielle

Réseaux de Neurones Graphiques pour les Systèmes de Recommandation

Présenté par :

BELAROUI Khaoula

Sous la direction de :

Dr. AYAD Habib

Le jury composé de :

Dr. MOUCHAWRAB Samar : *Examinateur*

Dr. CHADI Mohamed-Amine : *Rapporteur*

Année académique 2023-2024

“The best way to predict the future is to create it.”

- ABRAHAM LINCOLN

À tous ceux qui m'ont soutenue.

Avant-goût

Qu'on le remarque ou non, les systèmes de recommandation sont devenus omniprésents dans notre quotidien numérique. Invisibles, ils œuvrent en arrière-plan, façonnant nos choix, orientant nos découvertes et amplifiant nos intérêts. Pourtant, derrière cette simplicité apparente se cachent des mécanismes complexes, tentant de comprendre nos désirs les plus profonds. Ce mémoire se penche sur l'avenir de ces systèmes, et pose la question : peut-on mieux saisir la richesse des relations humaines, au-delà des simples données ?

Les réseaux neuronaux graphiques (GNN), tels des explorateurs d'un nouveau monde, offrent une réponse fascinante à cette question. Ils vont plus loin que les méthodes classiques, non pas en se contentant de suggérer, mais en capturant les réseaux d'interactions qui nous définissent. Ces modèles, en se nourrissant des liens entre utilisateurs et éléments, révèlent des connexions insoupçonnées, ouvrant des horizons inexplorés pour la recommandation. À travers cette approche, l'intelligence artificielle ne se limite plus à guider nos choix, elle devient un partenaire dans la compréhension de nos propres désirs, dans une mer d'informations où se mêlent le trivial et le profond.

Ce travail, enfin, se veut une exploration de cette alliance entre la technologie et l'humain, où les données et les graphes se rencontrent pour éclairer de nouvelles voies vers des choix qui résonnent avec notre nature la plus intime.

Table de matière

Avant-goût	1
1 Introduction	5
2 Système de recommandation	8
2.1 Qu'est-ce qu'un système de recommandation ?	8
2.2 Un peu d'histoire	10
2.3 L'état de l'art	12
2.4 Familles de recommandation	13
2.4.1 Filtrage basé sur le contenu	14
2.4.2 Filtrage collaboratif	15
2.4.2.1 Le filtrage collaboratif à base de mémoire	17
2.4.2.2 Le filtrage collaboratif à base de modèle	20
2.4.3 Approches Hybrides	24
2.5 Problèmes	24
2.5.1 Problème du démarrage à froid	25
2.5.2 Problème de la rareté des données	25
2.5.3 Taille des tables d'embedding	25
2.5.4 Problème de surspécialisation	26
3 Un peu de théorie de graphe	27
3.1 Introduction	27
3.2 Définitions et Représentation des Graphes	28
3.3 Types de Graphes	29
3.3.1 Graphes Orientés et Non Orientés	29
3.3.1.1 Graphes Pondérés	30
3.3.2 Graphes Étiquetés	31

3.3.3	Concepts Variés : Définitions	32
3.4	E-graphs - graphes sur ordinateurs	34
3.4.1	Matrice d'Incidence	34
3.4.2	Matrice d'Adjacence	35
3.4.3	Matrice des degrés (D)	37
3.4.4	Matrice Laplacienne (L) :	38
4	Réseaux Neuronaux Graphiques	40
4.1	Pourquoi penser aux réseaux neuronaux graphiques ?	40
4.2	Apprentissage Automatique sur les Graphes : Une Vue d'Ensemble .	43
4.2.1	Méthodes Traditionnelles pour l'Apprentissage sur les Graphes	43
4.2.1.1	Extraction de Caractéristiques Manuelles	43
4.2.2	Avènement des Méthodes d'Apprentissage de Représentation sur les Graphes	43
4.2.3	Transition vers les réseaux neuronaux graphiques (GNN) . .	44
4.3	Les concepts principaux	45
4.3.1	Données euclidiennes et Données non euclidiennes	45
4.3.2	Équivariance et invariance par permutation	47
4.3.3	Voisinage dans un graphe	50
4.3.4	Propagation de Messages Neuronaux	51
4.4	Structure générale d'un GNN	54
4.4.1	La structure générale d'un réseau neuronal	54
4.4.2	Pipeline d'un réseau de neurones graphiques	56
4.5	Les types des GNNs	57
4.5.1	Agrégation Simple de Voisinage	59
4.5.2	Réseaux convolutionnels sur graphes	60
4.5.3	Réseaux d'attention sur graphes	60
4.5.4	Échantillonnage et Agrégation sur Graphes	62
4.6	Problèmes sur les graphes	64

4.6.1	Tâches au niveau des nœuds :	65
4.6.1.1	Application d'un Classificateur Linéaire	65
4.6.2	Tâches au niveau des Liens	66
4.6.2.1	Application d'une Fonction d'Interaction	67
4.6.3	Tâches au Niveau du Graphe	68
4.6.3.1	Opération de Lecture (Readout)	68
5	GNNs pour les systèmes de recommandation	69
5.1	Pourquoi les GNNs sont-ils nécessaires pour les systèmes de recommandation ?	69
5.2	Représentation des données	70
5.2.1	Filtrage collaboratif utilisateur-élément	72
5.2.1.1	Structure générale	73
5.2.1.2	Architectures	74
6	Simulations et Résultats	82
6.1	Objectifs de la Simulation	82
6.2	Configuration des Modèles	84
6.2.1	Modèle GNN	84
6.2.1.1	Construction du Dataset et Représentation	84
6.2.1.2	Architecture du Modèle	87
6.2.1.3	Entraînement et Évaluation du Modèle	90
6.2.2	Factorisation de Matrice	93
6.2.3	Filtrage Collaboratif Basé sur la Mémoire	94
6.3	Résultats Comparés	95
6.4	Analyse et Discussion	96
7	Application Web de Recommandation avec GNN	98
7.1	Introduction	98
7.2	Technologies Utilisées	98

7.3	Structure de l'Application	99
7.4	Fonctionnement de l'Application	100
7.5	Conclusion	103
8	Confusion	104
9	Conclusion	105

Introduction

Dans le monde hyperconnecté d'aujourd'hui, où les choix se multiplient à un rythme vertigineux et où l'information est produite en quantités exponentielles, la capacité de filtrer, de sélectionner, et de personnaliser les contenus proposés aux utilisateurs est devenue un enjeu majeur. Les systèmes de recommandation, autrefois des outils de niche, sont désormais au cœur de notre expérience numérique. Ils influencent nos décisions d'achat, nos choix de divertissement, et même les opinions que nous formons. Pourtant, malgré leur omniprésence, ces systèmes rencontrent des limites importantes lorsqu'il s'agit de traiter des données complexes, hétérogènes, ou hautement interconnectées.

Les approches traditionnelles des systèmes de recommandation, basées sur des techniques comme le filtrage collaboratif ou le filtrage basé sur le contenu, ont démontré leur efficacité dans de nombreux contextes. Cependant, elles sont souvent incapables de capturer la complexité des interactions entre utilisateurs et contenus, en particulier dans des environnements où les relations ne sont pas simplement linéaires, mais tissées au sein de structures de données en réseau. Dans ces cas, les méthodes traditionnelles peuvent échouer à fournir des recommandations pertinentes et personnalisées, laissant entrevoir la nécessité d'approches plus sophistiquées et plus adaptées à la réalité des données modernes.

C'est dans ce contexte que les réseaux neuronaux (GNN) apparaissent comme une avancée prometteuse. Les GNN sont capables de traiter les données structurées en graphes, c'est-à-dire des données où les relations entre les éléments sont aussi importantes que les éléments eux-mêmes. Cette capacité à modéliser les in-

teractions complexes entre utilisateurs et contenus offre une nouvelle dimension aux systèmes de recommandation. Les GNN ne se contentent pas de chercher des similarités superficielles, mais explorent la richesse des connexions sous-jacentes, permettant ainsi des recommandations plus pertinentes, plus personnalisées, et mieux adaptées aux besoins individuels.

Ce mémoire se propose d'explorer en profondeur cette innovation en examinant l'état actuel des systèmes de recommandation et en introduisant les GNN comme une solution aux limitations identifiées. Le travail s'articule autour de plusieurs axes majeurs :

- **chapitre 2** : Systèmes de recommandation - Une revue des méthodes classiques, incluant leurs succès, leurs limites, et les défis auxquels ils sont confrontés dans les environnements modernes.
- **chapitre 3** : Préliminaires mathématiques - Une introduction aux concepts fondamentaux de la théorie des graphes, nécessaire pour comprendre le fonctionnement des GNN.
- **chapitre 4** : Réseaux neuronaux sur les graphes - Une analyse détaillée des GNN, de leur structure, de leur fonctionnement, et de leurs applications potentielles.
- **chapitre 5** : GNN pour les systèmes de recommandation - Une étude spécifique de l'application des GNN dans le domaine des recommandations, illustrée par des exemples et des études de cas.
- **chapitre 6** : Simulation - Une série de simulations et d'expériences visant à évaluer l'efficacité des GNN par rapport aux méthodes traditionnelles, en mettant en évidence leurs forces et leurs faiblesses.
- **chapitre 7** : Mise en œuvre d'une application web de recommandation utilisant la simulation réalisée avec notre modèle de réseaux neuronaux graphiques (GNN).

L’ambition de ce mémoire est double : d’une part, fournir une analyse rigoureuse et critique des systèmes de recommandation actuels et des avancées permises par les GNN ; d’autre part, ouvrir de nouvelles perspectives pour la recherche et le développement dans ce domaine en pleine évolution. En explorant les potentialités des GNN, ce travail aspire à contribuer à l’amélioration des systèmes de recommandation, en les rendant non seulement plus efficaces, mais aussi plus en phase avec la complexité et la richesse des interactions humaines.

Au-delà de l’aspect technique, ce mémoire invite à une réflexion plus large sur la manière dont les outils numériques influencent nos vies, et sur la responsabilité des chercheurs et des ingénieurs à concevoir des systèmes qui respectent et comprennent les individus qu’ils servent. Dans un monde où les choix sont multiples et les parcours individuels diversifiés, l’objectif ultime des systèmes de recommandation ne devrait pas seulement être de prédire nos préférences, mais de nous accompagner dans notre exploration personnelle, en enrichissant notre expérience du monde numérique.

Cette introduction élargit la perspective en intégrant non seulement les aspects techniques et méthodologiques de votre travail, mais aussi une réflexion sur l’impact et l’importance des systèmes de recommandation dans la société moderne. Elle pose ainsi un cadre plus complet pour la suite de votre mémoire.

Système de recommandation

“The environment is everything that isn’t me.”

—ALBERT EINSTEIN

L’utilisation des systèmes de recommandation, comme la plupart des méthodes d’apprentissage automatique en production, ne consiste pas seulement à mettre en œuvre le meilleur algorithme, mais à comprendre les utilisateurs et le domaine. Définir une notion commence probablement par une réflexion, accompagnée d’une ou plusieurs questions qu’elles se posent. Cela peut donner lieu à une vue d’ensemble, parfois floue, mais qui constitue certainement le point de départ.

2.1 Qu’est-ce qu’un système de recommandation ?

Un système de recommandation représente une défense intuitive contre la surcharge de choix pour les consommateurs. Face à la croissance exponentielle de l’information disponible sur le web, les utilisateurs sont souvent confrontés à une multitude de produits, films ou restaurants. Ainsi, la personnalisation devient une stratégie essentielle pour améliorer l’expérience utilisateur. Les systèmes de recommandation peuvent être définis de diverses manières. La définition la plus populaire et la plus générale que nous citons ici est celle de Robin Burke (Burke, 2002)[15] :

« un système capable de fournir des recommandations personnalisées ou permettant de guider l’utilisateur vers des ressources intéressantes ou utiles au sein d’un espace de données important. »

Un système de recommandation va au-delà d’un simple algorithme sophistiqué. Il implique également une compréhension approfondie des données et les utilisa-

teurs. Les experts en données ont longuement débattu de l'importance relative d'avoir un algorithme exceptionnellement performant par rapport à disposer de davantage de données. Chacune de ces approches présente des inconvénients : des algorithmes super performants nécessitent des ressources matérielles de pointe et en grande quantité, tandis que l'augmentation du volume de données soulève des défis supplémentaires, tels que celui de garantir un accès suffisamment rapide à ces données.

Une recommandation personnalisée ou non personnalisée ? Les recommandations peuvent être *personnalisées* ou *non personnalisées*. Par exemple, un site web affichant les dix machines à pain les plus vendues fournit des recommandations *non personnalisées*. En revanche, si un site web de vente immobilière ou de billets de concert utilise vos données démographiques ou votre localisation actuelle pour vous proposer des recommandations, celles-ci sont *semi-personnalisées*. Les recommandations entièrement personnalisées sont disponibles sur Amazon, où les clients identifiés voient des suggestions comme "Recommandations pour vous". L'idée de la recommandation *personnalisée* découle du fait que les individus ne sont pas uniquement intéressés par les articles populaires, mais aussi par ceux qui sont moins vendus ou qui se trouvent dans la longue traîne. La personnalisation est une stratégie essentielle pour faciliter une meilleure expérience utilisateur. Dans cette mémoire, on parlera sur la recommandation *personnalisée*.

Globalement, ces systèmes jouent un rôle vital et indispensable dans divers domaines d'accès à l'information pour dynamiser les affaires et faciliter le processus de prise de décision, et ils sont omniprésents sur de nombreux sites web, notamment dans le commerce électronique et les médias en ligne.

2.2 Un peu d'histoire

Les systèmes de recommandation ont évolué depuis leurs débuts pour devenir des outils essentiels dans divers domaines, tels que le commerce électronique, le streaming, et les réseaux sociaux. À l'origine, les approches non apprises (non-learned), qui reposaient sur des règles ou des heuristiques prédéfinies, dominaient le paysage. Parmi ces méthodes figuraient le filtrage collaboratif basé sur les utilisateurs et celui basé sur les items, qui exploitaient les similitudes entre utilisateurs ou items pour formuler des recommandations.

En parallèle, le filtrage basé sur le contenu a émergé comme une autre méthode importante. Cette approche se concentre sur les caractéristiques des items eux-mêmes pour recommander des articles similaires à ceux déjà appréciés par l'utilisateur.

Avec l'essor des techniques d'apprentissage automatique, les approches apprises (learned), qui utilisent des modèles capables d'apprendre et d'optimiser les recommandations en fonction des données, ont émergé. Ces approches incluent des modèles de factorisation matricielle, qui capturent les interactions implicites entre utilisateurs et items, et des réseaux neuronaux, qui introduisent des représentations plus complexes et non linéaires. Aujourd'hui, les réseaux neuronaux sur les graphes (GNN) marquent une nouvelle étape en intégrant directement la structure des relations entre utilisateurs et items dans le processus de recommandation.

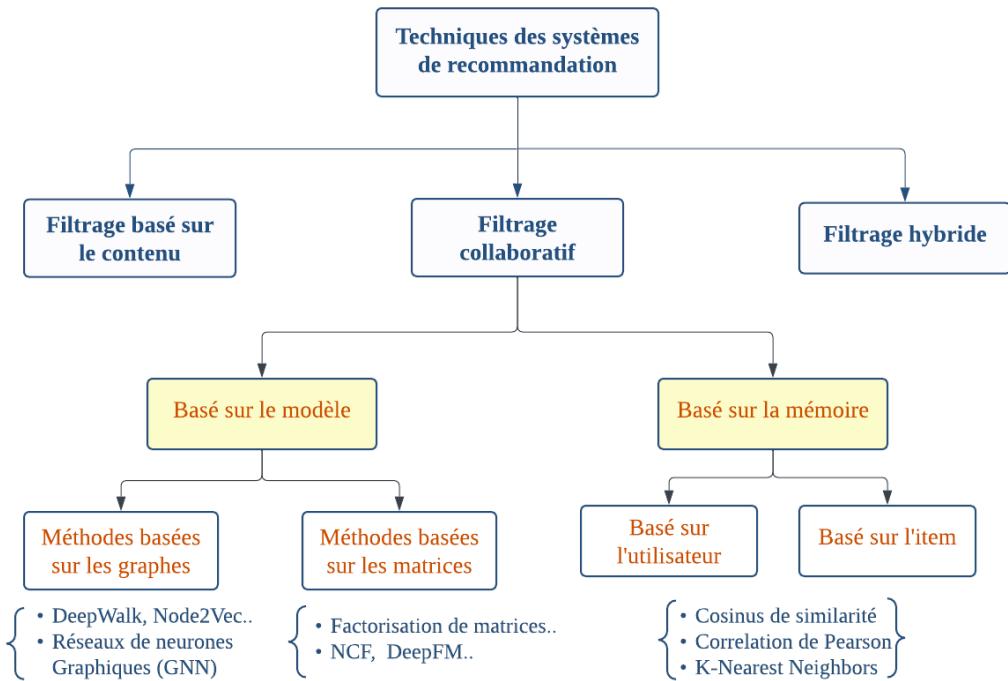


FIGURE 2.1 – Entre l'histoire et l'état de l'art : Évolution des techniques pour les Systèmes de Recommandation

Au début, les modèles de recommandation ont capturé l'effet de filtrage collaboratif en calculant directement la similarité entre les interactions. Du coup, les recommandations sont générées en fonction des actions et des préférences des utilisateurs. En d'autres termes, les utilisateurs "collaborent" indirectement en partageant leurs opinions et leurs interactions passées avec les articles ou les produits. Ensuite, des méthodes de filtrage collaboratif basées sur les modèles ont été proposées, telles que la factorisation de matrices ou les machines de factorisation, pour aborder la recommandation comme un problème d'apprentissage de représentation. Cependant, ces méthodes sont confrontées à des défis critiques tels que les comportements d'utilisateurs complexes ou les données d'entrée.

Pour surmonter ces défis, des modèles basés sur les réseaux neuronaux ont été proposés. Par exemple, le filtrage collaboratif neuronal a été développé pour

étendre le produit interne de la factorisation de matrices avec des perceptrons multicouches afin d'améliorer sa capacité. De même, la machine de factorisation profonde a combiné le modèle superficiel de la machine de factorisation avec des perceptrons multicouches.

Cependant, malgré ces avancées, ces méthodes restent fortement limitées car leurs paradigmes de prédiction et d'entraînement ignorent les informations structurelles d'ordre élevé dans les données observées. Par exemple, l'objectif d'optimisation du filtrage collaboratif neuronal est de prédire l'interaction entre l'utilisateur et l'article, et les échantillons d'entraînement comprennent des interactions positives observées entre l'utilisateur et l'article, ainsi que des interactions négatives non observées. Cela signifie que lors de la mise à jour des paramètres pour un utilisateur spécifique, seuls les articles avec lesquels il a interagi sont pris en compte, c'est-à-dire on considère uniquement les voisins de premier ordre (les noeuds directement connectés à un noeud donné).

En somme, au fur et à mesure de l'évolution des systèmes de recommandation, nous avons vu une progression des modèles superficiels aux modèles neuronaux, puis aux modèles basés sur les réseaux de neurones. Cependant, malgré les améliorations apportées par ces modèles plus récents, ils restent confrontés à des défis importants liés à la complexité des comportements des utilisateurs et à la gestion des données d'entrée. C'était la motivation d'introduire les réseaux de neurones graphiques (GNN).

2.3 L'état de l'art

Récemment, les progrès dans les réseaux neuronaux graphiques offrent une opportunité solide et fondamentale pour répondre aux problèmes mentionnés dans les systèmes de recommandation. Plus précisément, les réseaux neuronaux sur les graphes adoptent une propagation d'incorporation pour agréger itérativement les

incorporations des voisins. En empilant les couches de propagation, chaque nœud peut accéder à l'information des voisins de haut niveau, plutôt que seulement aux voisins de premier niveau comme le font les méthodes traditionnelles (Le [chapitre 4](#)).

En d'autres termes, les réseaux neuronaux graphiques sont capables de capturer des informations structurales de haut niveau dans les données, en explorant les relations entre les différents éléments de manière plus profonde que les méthodes traditionnelles. Ainsi qu'ils offrent plusieurs autres avantages distincts qui les rendent particulièrement adaptés à des tâches impliquant des données structurées sous forme de graphes, de manière générale.

Avec ses avantages dans le traitement des données structurelles et l'exploration des informations structurales, les méthodes basées sur les réseaux neuronaux graphiques sont devenues les nouvelles approches de pointe dans les systèmes de recommandation. En résumé, les réseaux neuronaux graphiques permettent une modélisation plus sophistiquée des relations entre les utilisateurs, les articles et d'autres éléments, ce qui améliore la précision et la pertinence des recommandations.

2.4 Familles de recommandation

La littérature sur les systèmes de recommandation[6] est vaste et se distingue par diverses approches académiques et industrielles, ce qui peut conduire à des catégorisations et taxonomies complexes. Dans cette section, nous fournirons une vue d'ensemble des principales familles de systèmes de recommandation. Il est également à noter que différents auteurs peuvent interpréter ces familles de manière légèrement différente ; dans les catégorisations qui suivent, nous essayerons de présenter une seule interprétation en mettant l'accent sur les principales distinctions.

2.4.1 Filtrage basé sur le contenu

Les méthodes de filtrage basées sur le contenu utilisent les similarités entre les “contenus” des éléments ou des utilisateurs, exprimées à travers leurs caractéristiques (comme des informations supplémentaires ou d’autres attributs non numériques), pour effectuer des recommandations. Elles partent du principe qu’un utilisateur consommera plus volontiers des éléments similaires à ceux qu’il a déjà consommés. Ces méthodes cherchent généralement à modéliser la fonction :

$$f(\text{caractéristiques_utilisateur}, \text{caractéristiques_élément})$$

où les caractéristiques de l’utilisateur peuvent inclure l’âge, le sexe, le revenu, et celles de l’élément, le prix et la couleur.

Les systèmes de recommandation modernes basés sur le contenu intègrent les éléments dans un espace d’embedding de faible dimension. Dans cet espace, des fonctions comme le produit scalaire ou la distance euclidienne permettent de trouver les éléments les plus proches. Ces approches ont particulièrement bien réussi dans le domaine des recommandations multimédia, et nous évoquerons quelques exemples considérés comme des recommandations basées sur le contenu.

L’article *Deep content based music recommendation* [31] présente un système automatique de recommandation musicale qui utilise un réseau de neurones convolutifs (CNN) pour produire des représentations latentes compactes des mél spectrogrammes des chansons appréciées par l’utilisateur, et propose à ce dernier des morceaux similaires à ceux qu’il a déjà écoutés.

Les modèles généraux de récupération d’images, tels que le modèle de « Deep Metric Learning via Lifted Structured Feature Embedding » [32], le reconnaissleur facial DeepID [50], ainsi que le reconnaissleur de produits e-commerce Groknet de Facebook [33], peuvent également être considérés comme des modèles de recom-

mandation basés sur le contenu, car ils apprennent à intégrer des images dans un espace latent, sur lequel plusieurs tâches peuvent être résolues, y compris la recommandation d'articles.

2.4.2 Filtrage collaboratif

Les méthodes de filtrage collaboratif sont une famille de méthodes qui reposent sur les similarités entre utilisateurs. Le concept de filtrage collaboratif a été décrit pour la première fois dans [5] comme une alternative au filtrage basé sur le contenu, où les auteurs ont proposé une approche efficace pour filtrer des documents sur une plateforme grâce à la collaboration des utilisateurs. Le système enregistrait les réactions des utilisateurs aux documents qu'ils lisaient et utilisait ces informations pour aider d'autres utilisateurs à filtrer les documents particulièrement intéressants ou non.

Ces méthodes décrivent généralement les utilisateurs et les articles à travers leurs identifiants, en partant du principe que des utilisateurs avec des comportements similaires auront aussi des préférences similaires pour les articles. Par exemple, dans le cas d'une recommandation de films, pour prédire comment un utilisateur u noterait un film m (note inconnue lors de la requête), un système de recommandation collaboratif chercherait d'abord des utilisateurs ayant des goûts similaires à ceux de u (c'est-à-dire ayant noté les films de manière similaire), puis essaierait d'estimer la note en se basant sur celles de ces utilisateurs similaires. Plus généralement, pour prédire la consommation d'un article s par un utilisateur u , la probabilité de consommation est estimée en fonction de la probabilité de consommation pour des utilisateurs similaires à u .

Ces méthodes se divisent généralement en deux catégories principales :

- **Filtrage collaboratif à base de mémoire :** Dans cette approche, les systèmes de recommandation utilisent directement les interactions historiques

entre les utilisateurs et les items pour calculer des similarités. Par exemple, dans une recommandation de films, pour prédire comment un utilisateur u noterait un film m (note inconnue au moment de la requête), un système collaboratif chercherait d'abord des utilisateurs aux goûts similaires à ceux de u (ayant noté les films de manière similaire) et estimerait ensuite la note en se basant sur celles de ces utilisateurs. Plus généralement, pour prédire la consommation d'un article s par un utilisateur u , la probabilité est estimée en fonction de celle des utilisateurs similaires à u .

- **Filtrage collaboratif à base de modèle :** Contrairement au filtrage collaboratif à base de mémoire, cette approche modélise les interactions utilisateur-item à l'aide de modèles statistiques ou d'apprentissage automatique. Ces modèles apprennent des représentations latentes pour les utilisateurs et les items, permettant de mieux capturer les relations complexes et les préférences implicites. Les modèles de filtrage collaboratif basés sur l'apprentissage modélisent la fonction :

$$f(\text{utilisateur_id}, \text{élément_id})$$

et se composent de deux parties principales :

- **Embedding**¹ : les utilisateurs et les articles sont d'abord intégrés via une transformation d'embedding apprenable, dans des représentations latentes pour faciliter la comparaison.
- **Modélisation des interactions** : les interactions historiques sont reconstruites à partir des représentations latentes des utilisateurs et des articles, via une fonction d'interaction, qui peut être apprenable (par exemple, un perceptron multicouche) ou non (comme un produit sca-

1. Un embedding est une représentation continue de données discrètes dans un espace vectoriel de faible dimension, préservant les propriétés structurelles et sémantiques de ces données. Cette technique permet de capturer des relations complexes tout en facilitant les calculs dans des tâches d'apprentissage automatique

laire).

2.4.2.1 Le filtrage collaboratif à base de mémoire

Le filtrage collaboratif à base de mémoire rassemble deux types : *le filtrage collaboratif basé sur les utilisateurs* et *Le filtrage collaboratif à base de mémoire* (*voir la figure 2.2*).

Le filtrage collaboratif basé sur les utilisateurs a été introduit pour la première fois dans le système GroupLens (Resnick and Varian, 1997), son principe de fonctionnement est très simple : déterminer les utilisateurs qui sont similaires à l'utilisateur courant, puis calculer une valeur de prédition pour chaque item candidat à la recommandation en analysant les notes que les voisins de l'utilisateur courant ont exprimées sur cet item. Du coup, ce type de filtrage passe par deux phases importantes : le calcul de la similarité et le calcul de la prédition de la note.

1. **La similarité** : le calcul de la similarité se fait soit sur l'ensemble des utilisateurs soit sur l'ensemble des items en utilisant des mesures de similarité et de distance, les plus utilisées dans la littérature sont la similarité de cosinus, ou la corrélation de Pearson.
2. **La prédition** : il s'agit de prédire la note des items non notés par les utilisateurs en se basant sur les notes données par ses voisins.

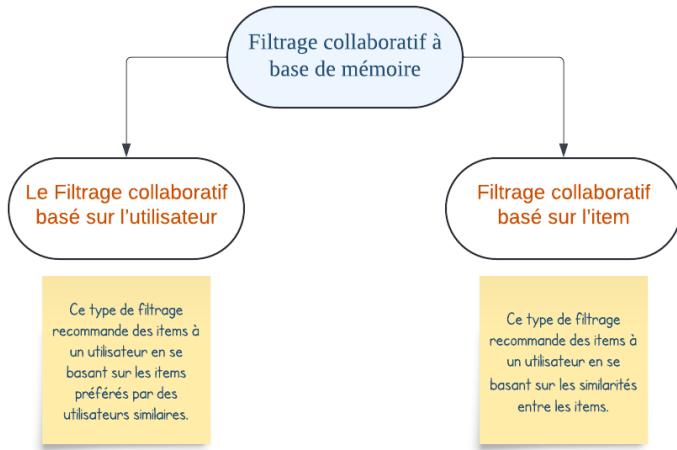


FIGURE 2.2 – Les types de filtrage collaboratif à base de mémoire

Ici, nous donnons dans ce qui suit les détails de calcul de ces deux mesures en se basant sur les utilisateurs dans ce travail.

Le calcul de la similarité :

1. ***La similarité du Cosinus*** [16] : La similarité cosinus entre deux utilisateurs u et w se base sur les vecteurs de leurs évaluations. La similarité entre ces deux utilisateurs est donnée par :

$$\text{similarité cosinus } (u, w) = \cos(v_u, v_w) = \frac{\sum_{i \in I_{uw}} r_{u,i} \times r_{w,i}}{\sqrt{\sum_{i \in I_{uw}} r_{u,i}^2} \cdot \sqrt{\sum_{i \in I_{uw}} r_{w,i}^2}} \quad (2.1)$$

Où :

- r_{ui} est la note donnée par l'utilisateur u à l'item i ,
- r_{wi} est la note donnée par l'utilisateur w à l'item i ,
- I_{uw} est l'ensemble des items évalués à la fois par les utilisateurs u et w .

Le cosinus varie entre 0 et 1. Une valeur égale à 1 indique que les deux

utilisateurs ont des préférences identiques, une valeur égale à 0 indique qu'ils n'ont rien en commun. La similarité cosinus se base sur l'angle entre deux vecteurs dans un espace multidimensionnel. Un inconvénient majeur de l'utilisation du cosinus dans le filtrage collaboratif est qu'il ne tient pas compte de la variation dans le jugement des utilisateurs.

2. ***Le calcul de la corrélation de Pearson*** [16] : Le coefficient de corrélation de Pearson mesure la liaison linéaire entre deux variables numériques en calculant le rapport entre leur covariance et le produit non nul de leur écart type. Il permet ainsi de mesurer la similarité en supprimant l'inconvénient de la variation des évaluations. L'équation (1.2) donne la formule de calcul de la mesure de corrélation entre deux utilisateurs u et w. En raison des données manquantes, seuls les items notés à la fois par u et w sont pris en compte. La similarité entre ces deux utilisateurs est donnée par :

$$simPear(u, w) = r_{uw} = \frac{\sum_{i \in I_{uw}} (r_{ui} - \bar{r}_u)(r_{wi} - \bar{r}_w)}{\sqrt{\sum_{i \in I_{uw}} (r_{ui} - \bar{r}_u)^2} \sqrt{\sum_{i \in I_{uw}} (r_{wi} - \bar{r}_w)^2}} \quad (2.2)$$

Où :

- r_{ui} est la note donnée par l'utilisateur u à l'item i .
- r_{wi} est la note donnée par l'utilisateur w à l'item i .
- \bar{r}_u est la note moyenne de l'utilisateur u .
- \bar{r}_w est la note moyenne de l'utilisateur w .
- I_{uw} est l'ensemble des items évalués à la fois par les utilisateurs u et w .

Le calcul de la prédiction

la prédiction de la note de l'utilisateur [16] courant u pour un item candidat à la recommandation i revient à calculer une moyenne pondérée de ses notes

sur l'ensemble des items similaires à i . Le calcul de la prédiction, p est donné par la formule (1.2) :

$$\text{pred}(u, i) = \bar{r}_u + \frac{\sum_{w \in N} (\text{sim}(u, w) * (r_{w,i} - \bar{r}_w))}{\sum_{w \in N} \text{sim}(u, w)} \quad (2.3)$$

Où :

- N est l'ensemble des utilisateurs similaires à u qui ont évalué l'item i .
- r_{ui} est la note donnée par l'utilisateur u à l'item i .
- r_{wi} est la note donnée par l'utilisateur w à l'item i .
- $\text{sim}(u, w)$: Similarité entre les utilisateurs u et w .
- \bar{r}_u est la note moyenne de l'utilisateur u .
- \bar{r}_w est la note moyenne de l'utilisateur w .

2.4.2.2 Le filtrage collaboratif à base de modèle

[Mais pourquoi penser à un autre algorithme de filtrage collaboratif ?](#)

Le filtrage collaboratif à base de mémoire a connu beaucoup de limites (Voir la figure 2.3) et pour y faire face, les algorithmes utilisés étaient améliorés pour atteindre aux algorithmes du filtrage collaboratif à base de modèle.

Contrairement aux algorithmes basés sur la mémoire, seules les évaluations faites par l'utilisateur sont prises en compte dans les algorithmes basés sur le modèle afin de construire un modèle de prédiction des évaluations. La différence entre FBA et FBM est que le premier se base directement sur les interactions passées, tandis que le second utilise des modèles pour capturer les relations sous-jacentes entre les données d'interaction. Les algorithmes basés sur un modèle construisent en offline une image réduite de la matrice des évaluations dans un objectif de réduire la complexité des calculs.

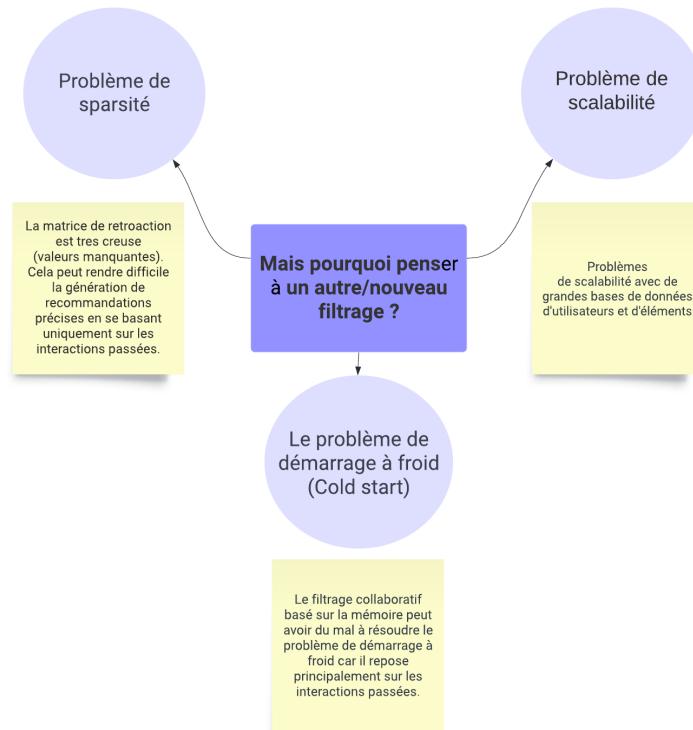


FIGURE 2.3 – Quelques limites du filtrage collaboratif à base de mémoire

Ainsi, il faut d'abord créer des modèles qui ressemblent aux comportements des utilisateurs et ensuite, utiliser ces modèles afin d'émettre les recommandations. En pratique, il a été démontré que l'approche par mémoire offre de meilleures performances en termes de précision alors que l'approche par modèle est plus efficace à grande échelle avec de gros ensembles de données. Ces modèles capturent les relations sous-jacentes entre utilisateurs et éléments, souvent en utilisant des techniques telles que la factorisation de matrices, les réseaux de neurones, les arbres de décision, etc.

Nous allons commencer notre discussion sur les modèles de filtrage collaboratif avec des approches classiques, telles que la Factorisation en Matrice (MF) et Le filtrage collaboratif neuronal NFC.

Factorisation de matrices

La factorisation de matrices (MF) [7] est une méthode classique pour effectuer du filtrage collaboratif, qui exploite le concept mathématique de la factorisation matricielle pour découvrir des caractéristiques latentes à la fois pour les utilisateurs et les éléments, et utiliser ces caractéristiques pour reconstruire la matrice d'interaction.

Étant donné U , un ensemble d'utilisateurs, et I , un ensemble d'éléments, avec leur matrice d'interaction respective $R \in \mathbb{R}^{|U| \times |I|}$, où certaines entrées peuvent être inconnues (par exemple, un utilisateur n'a pas interagi avec un élément, donc la note n'est pas connue), et un hyperparamètre k , indiquant la dimensionnalité des caractéristiques latentes, l'objectif de la factorisation matricielle est de reconstruire :

$$\hat{R} = P \times Q^\top$$

où $P \in \mathbb{R}^{|U| \times k}$ and $Q \in \mathbb{R}^{|I| \times k}$ sont respectivement les représentations latentes des utilisateurs et des éléments. La formule peut également être écrite de manière non vectorisée pour mettre en évidence les contributions de P et Q :

$$\hat{R}_{ui} = Q_i^\top P_u$$

L'objectif d'entraînement devient une minimisation de l'erreur quadratique moyenne de reconstruction entre \hat{R} and R :

$$\min \sum_{(u,i)|\hat{R}_{ui} \neq 0} (R_{ui} - \hat{R}_{ui})^2$$

où différentes variantes du modèle peuvent inclure des termes de régularisation appliqués aux matrices factorisées.

L'approche de factorisation de matrice peut également être vue comme

un problème d'apprentissage profond dans des cadres modernes, en intégrant simplement les identifiants des utilisateurs et des articles dans un espace latent via des couches d'embedding, et en appliquant un produit scalaire pour obtenir la matrice d'interaction reconstruite.

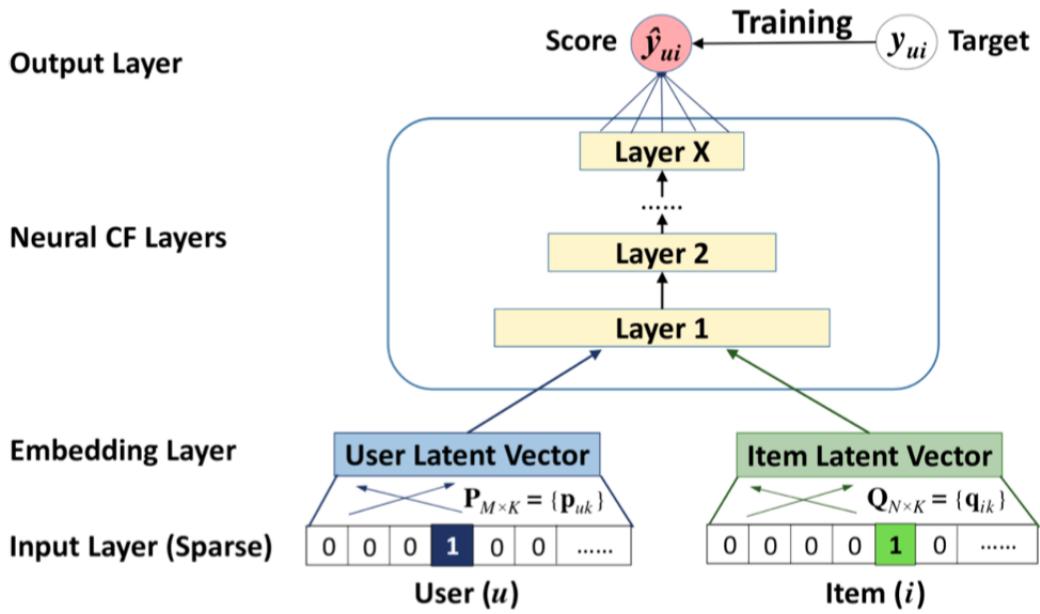


FIGURE 2.4 – Architecture du filtrage collaboratif neuronal

Le filtrage collaboratif neuronal

Le filtrage collaboratif neuronal (NCF ou NeuroCF) [34] peut être considéré comme une généralisation profonde de l'approche de Factorisation de Matrice décrite précédemment. Le concept principal que le NCF étend est la fonction d'interaction, que les auteurs remplacent par un MLP (Multi-Layer Perceptron) arbitrairement profond. En utilisant la même notation que dans la Factorisation de Matrice, nous pouvons définir l'équation du NCF comme suit :

$$\hat{R}_{ui} = \text{MLP}(Q_i, P_u)$$

où Q_i et P_u représentent respectivement les embeddings des articles et des utilisateurs, obtenus à l'aide d'une table de recherche paramétrable (c'est-à-dire des couches d'embedding dans les frameworks modernes d'apprentissage profond).

Cette extension est entraînée en utilisant l'erreur de reconstruction et permet au réseau profond d'apprendre des interactions plus puissantes. Une visualisation de l'architecture peut être vue dans la Figure 2.4.

2.4.3 Approches Hybrides

Les approches hybrides ont pour objectif de pallier les limitations des modèles de filtrage basés sur le contenu et des modèles de filtrage collaboratif (comme le problème de démarrage à froid, qui sera abordé plus loin dans ce chapitre) en combinant ces deux types d'approches. En d'autres termes, tout modèle qui combine à la fois les identifiants et les caractéristiques de contenu peut être considéré comme hybride. Comme ces modèles utilisent à la fois les identifiants des utilisateurs/éléments et leurs caractéristiques, la fonction qu'ils cherchent à modéliser devient :

$$f(\text{utilisateur_id}, \text{utilisateur_cara.}, \text{élément_id}, \text{élément_cara.})$$

2.5 Problèmes

Il existe des problèmes non résolus dans le domaine des recommandations qui pourraient s'appliquer aux architectures précédemment décrites.

2.5.1 Problème du démarrage à froid

Le démarrage à froid est un problème qui survient lorsque le système doit faire des prédictions pour des utilisateurs, des articles ou des communautés pour lesquels il n'y a pas encore assez d'informations (par exemple, des utilisateurs qui n'ont évalué que quelques articles, ou des articles qui n'ont pas encore été achetés), ce qui rend difficile pour le système de proposer des recommandations pertinentes. Une façon de contourner ce problème est d'utiliser une recommandation hybride, qui combine le filtrage collaboratif pour les prédictions "chaudes" (celles avec suffisamment d'interactions) et le filtrage basé sur le contenu pour les prédictions "froides" (celles qui n'en ont pas encore assez). Le filtrage basé sur le contenu est efficace dans ce cas car il ne nécessite pas d'informations d'interaction préalables pour recommander des articles.

2.5.2 Problème de la rareté des données

Le problème de la rareté des données se pose lorsque les utilisateurs n'interagissent qu'avec une petite fraction des articles disponibles, ce qui signifie que le système manque de données d'interaction pour former des clusters, compromettant ainsi la qualité globale des recommandations.

2.5.3 Taille des tables d'embedding

Les systèmes de recommandation collaboratifs se caractérisent par des tables d'embedding très volumineuses, utilisées pour encoder les valeurs des variables catégorielles (comme les identifiants des utilisateurs et des articles) dans l'espace latent.

2.5.4 Problème de surspécialisation

La surspécialisation se produit lorsque les utilisateurs sont empêchés de découvrir de nouveaux articles parce que les recommandations sont trop similaires à ce qu'ils connaissent déjà ou à ce qui est défini dans leur profil (par exemple, un utilisateur a acheté un smartphone et se voit recommander le même modèle ou un modèle qu'il connaît déjà). Les approches pour résoudre ce problème peuvent être un peu risquées, car elles tentent de sacrifier la pertinence des recommandations au profit de nouvelles découvertes [10].

Un peu de théorie de graphe

3.1 Introduction

L'histoire des graphes remonte au problème des Sept Ponts de Königsberg en 1736 (Biggs et al, 1986). Ces structures de données flexibles sont conçues pour représenter les relations complexes entre les individus, les rendant ainsi omniprésentes et largement utilisées dans de multiples domaines tels que la biologie, la finance, les transports, les réseaux sociaux et les systèmes de recommandation.

Alors que la théorie des graphes traditionnelle aborde l'extraction d'informations déterministes comme les plus courts chemins, les composants connectés, les regroupements locaux, l'isomorphisme des graphes, etc.. Les applications d'apprentissage automatique sur les données de graphe se concentrent davantage sur la prédiction de parties manquantes ou de dynamiques futures. Les graphes sont constitués de deux composants principaux : des noeuds (ou sommets) représentant des entités individuelles, et des arêtes (ou liens) qui connectent ces entités, décrivant ainsi les relations entre elles.

Les concepts clés de la théorie des graphes comprennent les graphes orientés et non orientés, les cycles, les chemins, les arbres, les graphes bipartis et bien plus encore. Ces concepts fondamentaux sont utilisés pour résoudre une variété de problèmes, de la recherche du chemin le plus court à la détection de structures importantes au sein des réseaux.

3.2 Définitions et Représentation des Graphes

Un **graphe** est une structure mathématique utilisée pour modéliser les relations entre les objets. Un graphe est composé de deux ensembles : un ensemble de *sommets* (ou nœuds), souvent noté V , et un ensemble d'*arêtes*, noté E , qui relient ces sommets. Mathématiquement, un graphe G est défini comme un couple $G = (V, E)$, où :

- V est l'ensemble des sommets, représentant les entités du graphe ;
- E est l'ensemble des arêtes, représentant les connexions ou relations entre les entités.

Les graphes peuvent être représentés de plusieurs façons en fonction du contexte et de l'application :

- **Représentation par Liste d'Adjacence** : Chaque sommet est associé à une liste de sommets adjacents, c'est-à-dire les sommets avec lesquels il partage une arête. Cette représentation est souvent utilisée pour des graphes avec peu d'arêtes, car elle est économiquement en mémoire.
- **Représentation par Matrice d'Adjacence** : C'est une matrice carrée A de taille $|V| \times |V|$, où $a_{ij} = 1$ si une arête existe entre les sommets i et j , et $a_{ij} = 0$ sinon. Cette représentation est pratique pour manipuler des graphes dans des contextes algébriques ou pour des calculs matriciels.
- **Représentation par Matrice d'Incidence** : Une matrice I où les lignes représentent les sommets et les colonnes représentent les arêtes. Si une arête e_k relie le sommet v_i au sommet v_j , alors la colonne k de la matrice I aura 1 en position i et -1 en position j . Cette représentation est utile pour certaines analyses de graphes, notamment en théorie des graphes orientés.

Ces différentes représentations offrent des perspectives variées sur la structure et les propriétés des graphes, facilitant leur analyse et leur utilisation dans des domaines aussi divers que l'informatique, les réseaux, ou encore les sciences sociales. Nous allons voir ces matrices en détails apres.

3.3 Types de Graphes

3.3.1 Graphes Orientés et Non Orientés

Un graphe est constitué de sommets (nœuds) et d'arêtes (lignes) qui relient ces sommets. Mathématiquement, un graphe est souvent représenté par $G = (U, V)$, où U est l'ensemble des sommets et V l'ensemble des arêtes.

Les graphes peuvent être **non orientés** ou **orientés** :

- Dans un **graphe non orienté**, les arêtes n'ont pas de direction. Par exemple, si deux sommets L et M sont reliés par une arête, alors la relation est bidirectionnelle : L est lié à M et M est lié à L . Un exemple typique de graphe non orienté est un réseau social où les sommets représentent des personnes et les arêtes leurs amitiés, voir la figure3.1(a).
- Dans un **graphe orienté**, les arêtes possèdent une direction, représentée par une flèche. Cela signifie qu'une relation entre deux sommets n'est pas nécessairement réciproque. Par exemple, dans un réseau social, une personne A peut suivre une autre personne B , mais cela ne signifie pas que B suit A en retour. Les applications typiques des graphes orientés incluent les réseaux de pages web et leurs hyperliens, les connexions dans les réseaux sociaux comme les suiveurs sur Twitter, et les graphes de communication où l'influence entre utilisateurs est étudiée, voir la figure3.1(b).

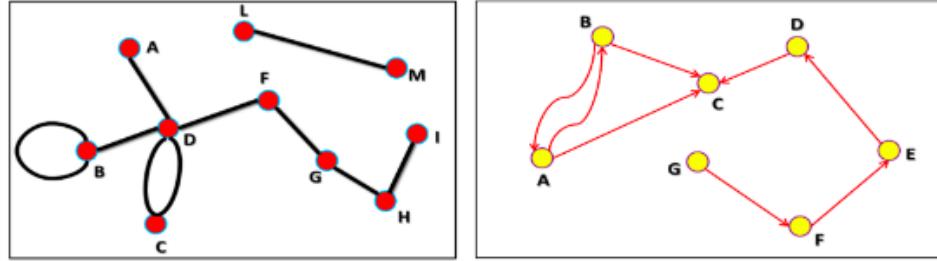


FIGURE 3.1 – (a) Exemple d'un graphe non orienté montrant les connexions bidirectionnelles entre les sommets. (b) Exemple d'un graphe orienté où les arêtes ont une direction spécifique, indiquant la relation directionnelle entre les sommets.

3.3.1.1 Graphes Pondérés

Dans un **graphé pondéré**, chaque arête est associée à un poids ou une valeur numérique, représentant la force, le coût ou toute autre caractéristique pertinente de la relation entre les sommets. Par exemple, dans un réseau de transport, les arêtes peuvent représenter des routes, et les poids peuvent indiquer la distance ou le temps de trajet entre deux points. Les graphes pondérés sont souvent utilisés dans des problèmes d'optimisation, tels que le calcul du chemin le plus court.

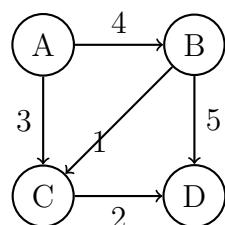


FIGURE 3.2 – Exemple d'un graphe pondéré avec quatre nœuds et des arêtes pondérées.

3.3.2 Graphes Étiquetés

Les **graphes étiquetés** ajoutent une dimension supplémentaire en associant des informations spécifiques aux nœuds et/ou aux arêtes. Chaque nœud ou arête possède une étiquette ou un attribut qui apporte un contexte supplémentaire aux relations dans le graphe. (Voir la figure 3.3)

- Les nœuds peuvent être étiquetés pour représenter des entités telles que des utilisateurs, des produits ou des concepts, avec des informations pertinentes comme l'âge, la localisation ou les intérêts dans le cadre d'un réseau social.
- Les arêtes, quant à elles, peuvent porter des étiquettes reflétant des caractéristiques de la relation, telles que la distance entre deux points dans un réseau de transport ou le type de relation entre deux utilisateurs dans un réseau social.

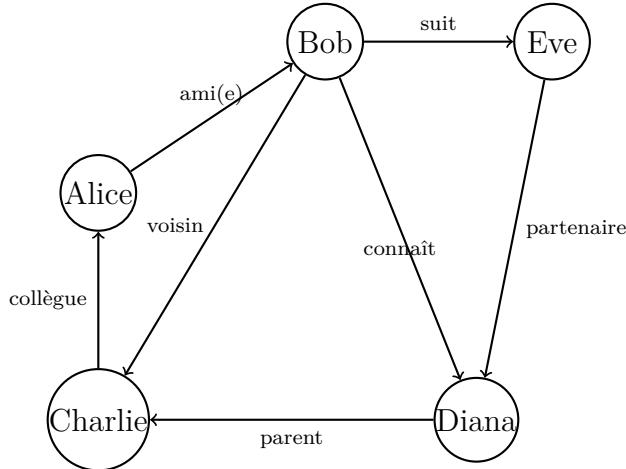


FIGURE 3.3 – Exemple d'un graphe orienté étiqueté représentant un réseau social fictif, où chaque nœud représente une personne et chaque arête décrit la nature de la relation entre les nœuds connectés.

Ce graphe orienté 3.3 représente un réseau social fictif où chaque nœud correspond à une personne, et chaque arête, avec une direction spécifique, indique la nature de la relation entre ces personnes.

- Alice est amie avec Bob, tandis que Charlie est un collègue d’Alice.
- Bob suit Eve sur un réseau social, mais il entretient aussi une relation de voisinage avec Charlie.
- Diana est la mère ou le père de Charlie, indiquant une relation familiale directe.
- Bob connaît également Diana, bien que la nature de cette relation soit plus informelle.
- Enfin, Eve et Diana sont partenaires, peut-être dans un contexte professionnel ou collaboratif.

Les flèches indiquent la direction de ces relations, montrant, par exemple, que bien que Bob suive Eve, il n’y a pas nécessairement de réciprocité. Les étiquettes sur les arêtes décrivent clairement ces relations, offrant une vue d’ensemble du réseau social représenté par ce graphe.

3.3.3 Concepts Variés : Définitions

Les graphes peuvent également être caractérisés par différents concepts qui décrivent leurs propriétés et la nature des relations entre les nœuds :

(a) Graphes Homogènes et Hétérogènes :

- Un **graphe homogène** représente des relations uniformes entre les nœuds, où chaque connexion suit un modèle similaire. Par exemple, dans un réseau social homogène, tous les utilisateurs interagissent de manière similaire avec d’autres utilisateurs, formant ainsi un graphe où chaque nœud est du même type.
- Un **graphe hétérogène**, en revanche, capture des relations diversifiées entre différents types de nœuds. Par exemple, dans un système de recommandation, les interactions entre utilisateurs et divers types de produits créent un graphe hétérogène, reflétant la diversité des en-

tités et des relations présentes.

(b) **Graphes Statiques et Dynamiques :**

- Un **graphe statique** représente un ensemble de relations figées entre les entités, sans prise en compte de l'évolution temporelle. Par exemple, un graphe social statique peut modéliser des amitiés qui ne changent pas au fil du temps.
- À l'inverse, un **graphe dynamique** modélise des relations qui évoluent avec le temps. Par exemple, dans un réseau social dynamique, les liens d'amitié peuvent apparaître, se renforcer ou disparaître, capturant ainsi les changements et les interactions au fil du temps.

(c) **Graphes Denses et Dispersés :**

- Un **graphe dense** est caractérisé par une connectivité élevée, où la majorité des noeuds sont reliés entre eux. Par exemple, dans un réseau où presque tous les utilisateurs sont interconnectés, le graphe sera dense, indiquant une forte interaction entre les noeuds.
- À l'opposé, un **graphe dispersé** présente une connectivité faible, où les connexions entre noeuds sont limitées. Par exemple, dans un réseau social où chaque utilisateur n'interagit qu'avec un petit nombre d'autres utilisateurs, le graphe résultant est dispersé, avec des connexions plus rares.

(d) **Graphes Planaires et Non Planaires :**

- Un **graphe planaire** peut être dessiné sur un plan sans que ses arêtes ne se croisent. Un exemple typique est un schéma de connexion entre villes sur une carte, où les routes peuvent être tracées sans intersection.
- Un **graphe non planaire**, en revanche, nécessite des croisements

d'arêtes lorsqu'il est représenté sur un plan. Un exemple serait un graphe représentant les liaisons aériennes entre plusieurs villes, où certaines routes se croisent dans la représentation plane, en raison des nombreuses connexions complexes entre les villes.

3.4 E-graphs - graphes sur ordinateurs

Il existe plusieurs façons de transformer un graphe en un format que l'ordinateur peut comprendre ; toutes ces méthodes se présentent sous forme de différents types de matrices.

3.4.1 Matrice d'Incidence

La matrice d'incidence, couramment notée par une majuscule I dans les articles scientifiques, est une matrice qui relie les sommets d'un graphe à ses arêtes. Les éléments de cette matrice sont généralement composés de 1, 0 et -1, où :

- 1 indique que le sommet est le point de départ de l'arête,
- -1 indique que le sommet est le point d'arrivée de l'arête,
- 0 indique qu'il n'y a pas d'incidence directe entre le sommet et l'arête.

La construction de la matrice d'incidence peut être résumée par le schéma suivant :

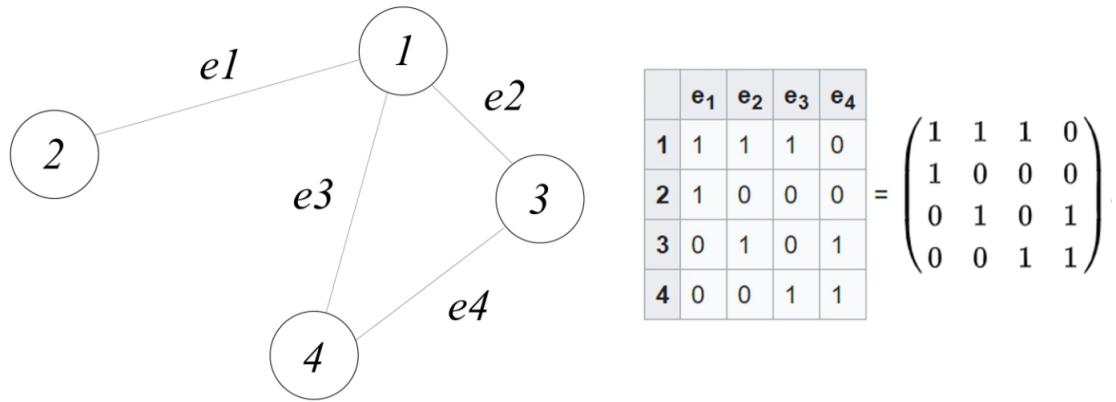


FIGURE 3.4 – Représentation de la transformation d'un graphe en sa matrice d'incidence, où chaque colonne de la matrice correspond à une arête du graphe et chaque ligne à un sommet.

Dans cet exemple, chaque colonne de la matrice correspond à une arête, et chaque ligne correspond à un sommet. La matrice est remplie selon les connexions entre les sommets et les arêtes du graphe, permettant ainsi de capturer la structure du graphe de manière concise.

3.4.2 Matrice d'Adjacence

La **matrice d'adjacence** d'un graphe est une matrice carrée utilisée pour représenter les relations entre les noeuds du graphe. Si un graphe possède n noeuds, la matrice d'adjacence A sera de dimension $n \times n$, où chaque élément a_{ij} indique la présence ou l'absence d'une arête entre le noeud v_i et le noeud v_j .

La construction de la matrice d'adjacence A suit la règle suivante :

$$a_{ij} = \begin{cases} 1, & \text{si } (v_i, v_j) \in E, \\ 0, & \text{sinon.} \end{cases}$$

Dans un graphe non orienté, la matrice d'adjacence est symétrique par rapport à sa diagonale, c'est-à-dire du coin supérieur gauche au coin inférieur droit. En revanche, pour les graphes orientés, la matrice d'adjacence n'est pas symétrique, car les arêtes ne vont que dans une seule direction, ce qui entraîne une asymétrie le long de la diagonale, Voir 3.5.

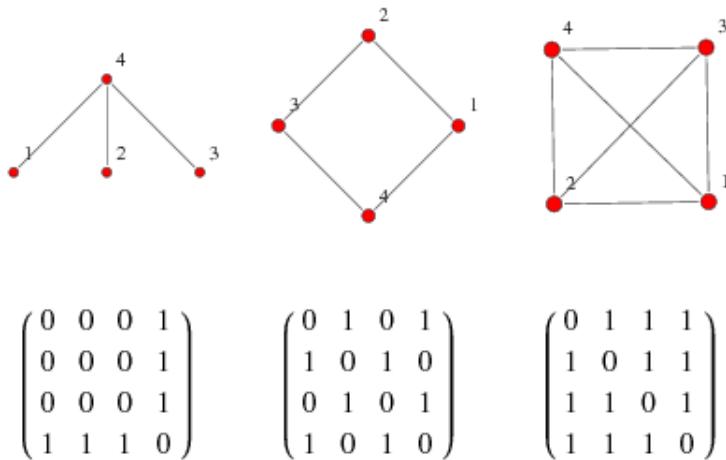


FIGURE 3.5 – Exemples de graphes simples et leurs matrices d'adjacence correspondantes : (à gauche) un graphe en étoile avec quatre sommets et sa matrice d'adjacence ; (au centre) un graphe en forme de carré avec diagonale et sa matrice d'adjacence ; (à droite) un graphe complet avec quatre sommets et sa matrice d'adjacence.

Une matrice d'adjacence peut également être *pondérée* (3.7), ce qui signifie que chaque arête se voit attribuer une valeur spécifique au lieu d'un simple 1. Ces valeurs sont placées dans les éléments correspondants de la matrice. Les poids peuvent représenter diverses significations en fonction du contexte. Par exemple, dans le cas des molécules, les poids peuvent indiquer le type de liaison entre deux noeuds (atomes). Dans un réseau social comme LinkedIn, ils peuvent représenter la force des connexions de premier,

deuxième ou troisième degré entre deux nœuds (personnes).

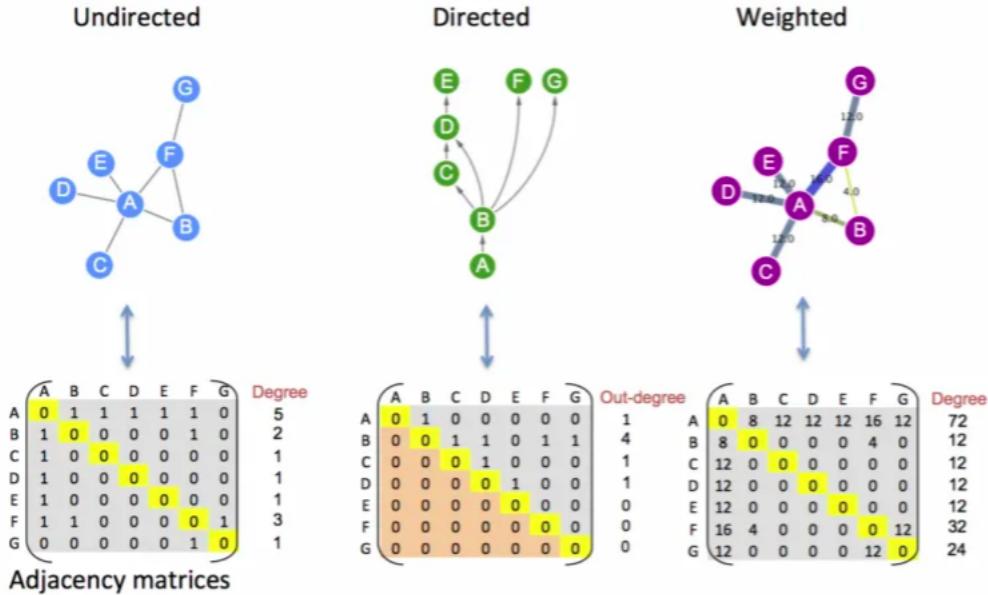


FIGURE 3.6 – Représentation des matrices d'adjacence pour différents types de graphes : (à gauche) graphe non orienté avec sa matrice d'adjacence montrant les degrés des sommets; (au centre) graphe orienté avec sa matrice d'adjacence indiquant les degrés sortants des sommets; (à droite) graphe pondéré avec sa matrice d'adjacence illustrant les degrés pondérés des sommets.

3.4.3 Matrice des degrés (D)

Le degré d'un nœud dans un graphe est le nombre d'arêtes connectées à ce nœud. Il existe deux types de degrés principaux :

- Degré entrant (in-degree) :** Dans un graphe dirigé, le degré entrant d'un nœud est le nombre d'arêtes entrantes connectées à ce nœud, c'est-à-dire le nombre d'arêtes se terminant à ce nœud.
- Degré sortant (out-degree) :** Dans un graphe dirigé, le degré sortant d'un nœud est le nombre d'arêtes sortantes connectées à ce nœud, c'est-à-dire le nombre d'arêtes partant de ce nœud vers d'autres nœuds.

Dans un graphe non dirigé, tous les noeuds ont un seul type de degré car les arêtes ne sont pas directionnelles.

La matrice des degrés d'un graphe peut être trouvée en utilisant le concept des degrés mentionné précédemment. D est essentiellement une matrice diagonale, où chaque valeur de la diagonale est le degré de son noeud correspondant.

Remarquez que le degré est simplement la somme de chaque ligne de la matrice d'adjacence. Ces degrés sont ensuite placés sur la diagonale de la matrice (la ligne de symétrie pour la matrice d'adjacence). Cela mène de manière cohérente à la matrice finale :

3.4.4 Matrice Laplacienne (L) :

La matrice laplacienne d'un graphe est le résultat de la soustraction de la matrice d'adjacence par la matrice des degrés.

$$L = D - A$$

Chaque valeur dans la matrice des degrés est soustraite de sa valeur respective dans la matrice d'adjacence comme suit :

Labeled graph	Degree matrix	Adjacency matrix	Laplacian matrix
	$\begin{pmatrix} 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$	$\begin{pmatrix} 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$	$\begin{pmatrix} 2 & -1 & 0 & 0 & -1 & 0 \\ -1 & 3 & -1 & 0 & -1 & 0 \\ 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & 0 & -1 & 3 & -1 & -1 \\ -1 & -1 & 0 & -1 & 3 & 0 \\ 0 & 0 & 0 & -1 & 0 & 1 \end{pmatrix}$

FIGURE 3.7 – Exemple d'un graphe étiqueté avec ses matrices associées : (à gauche) le graphe étiqueté ; (au centre gauche) la matrice des degrés montrant les degrés des sommets ; (au centre) la matrice d'adjacence indiquant les connexions entre les sommets ; (à droite) la matrice laplacienne résultant de la différence entre la matrice des degrés et la matrice d'adjacence

Il existe d'autres représentations matricielles de graphes telles que la matrice d'incidence, mais la grande majorité des applications de GNN sur des données de type graphe utilisent une, deux ou toutes les trois de ces matrices. C'est parce qu'elles, et la matrice laplacienne en particulier, fournissent des informations substantielles sur les entités (un élément avec des attributs) et les relations (une connexion entre les entités).

Réseaux Neuronaux Graphiques

“Data has shape, and shape has a meaning.”

INCONNU

La forme des données, bien plus qu'un simple ensemble de points isolés, révèle des connexions riches de sens. Ces connexions cachées, lorsqu'elles sont correctement exploitées, peuvent transformer notre compréhension des systèmes complexes et des interactions.

4.1 Pourquoi penser aux réseaux neuronaux graphiques ?

Cette question nous plonge au cœur d'une réflexion sur la manière dont les données interconnectées façonnent notre perception du monde moderne. L'évolution de la science des données et de l'apprentissage automatique est marquée par une quête incessante de capturer et de modéliser la complexité du monde réel. En effet, le monde qui nous entoure est un vaste réseau d'interconnexions et de relations, des interactions sociales aux structures moléculaires, en passant par les réseaux de transport et les systèmes économiques. Comprendre et exploiter ces interconnexions est essentiel pour avancer dans notre compréhension et nos capacités prédictives. C'est dans ce contexte que les réseaux neuronaux graphiques (GNNs) ont vu le jour.

Traditionnellement, les algorithmes de machine learning et les réseaux de neurones ont été conçus pour fonctionner sur des données euclidiennes, comme des images, des textes et des tableaux. Ces données sont structurées de manière régulière et se prêtent bien aux techniques classiques de traitement et d'analyse. Cependant, cette approche montre ses limites lorsque les données à analyser sont intrinsèquement non euclidiennes (voir la partie 4.3.1) et complexes, comme les graphes. Les graphes sont omniprésents : ils modélisent les relations entre amis sur les réseaux sociaux, les interactions entre utilisateurs et articles dans les systèmes de recommandation, les liaisons chimiques dans les molécules, et bien d'autres. Les méthodes traditionnelles ne peuvent pas capturer efficacement la topologie et les relations complexes de ces graphes, entraînant une perte significative d'informations essentielles.

Le besoin d'une nouvelle famille de réseaux de neurones découle du principal défi auquel nous pourrions être confrontés lors de l'application de méthodes d'apprentissage profond sur des données structurées en graphe, à savoir que les approches typiques sont définies pour des structures d'entrée précises. Par exemple, les réseaux de neurones récurrents (RNNs) supposent travailler avec des données organisées en séquences (par exemple, texte, phénomènes temporels) et les réseaux de neurones convolutionnels (CNNs) supposent travailler avec des données structurées en grille (par exemple, images, vidéos). Les deux types de données mentionnés peuvent être considérés comme un cas particulier de structure de graphe (voir la figure 4.2). Cependant, pour que les réseaux neuronaux profonds fonctionnent sur des structures de graphe généralisées, nous devons utiliser différents types d'architectures d'apprentissage profond.

Les GNNs ont été inventés pour combler cette lacune et permettre une

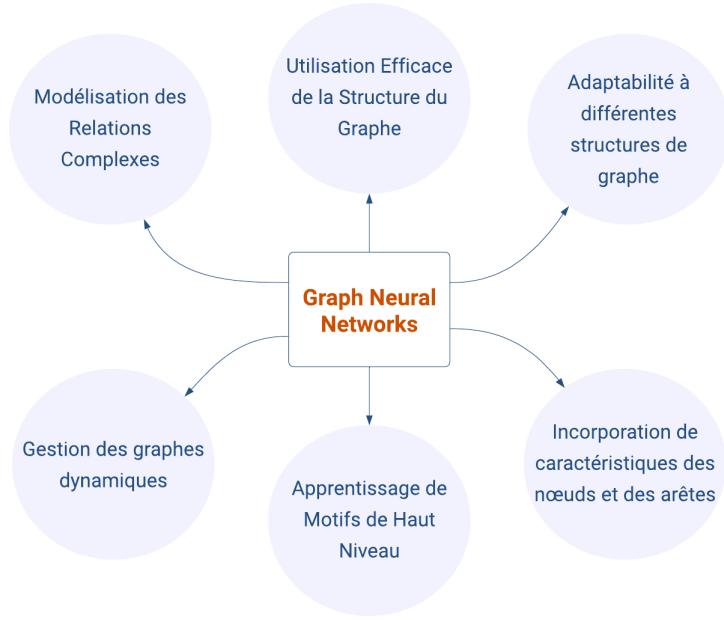


FIGURE 4.1 – Avantages des réseaux neuronaux graphiques

modélisation plus naturelle et expressive des données de graphe. L’essor de cette nouvelle architecture provient principalement des avancées des réseaux de neurones convolutionnels (CNN) et de l’apprentissage de la représentation des graphes (GRL)[21]. Les GNNs offrent une capacité unique à propager et à agréger l’information à travers les nœuds connectés, capturant ainsi les dépendances locales et globales (voir la partie 4.3.4). En utilisant des mécanismes comme la propagation de messages et l’attention, les GNNs peuvent apprendre des représentations riches et contextuelles qui sont bien plus informatives que celles obtenues par les méthodes traditionnelles. C’est pourquoi l’utilisation des réseaux de neurones sur les graphes présente de nombreux avantages. (Voir la figure 4.1)

4.2 Apprentissage Automatique sur les Graphes : Une Vue d'Ensemble

4.2.1 Méthodes Traditionnelles pour l'Apprentissage sur les Graphes

4.2.1.1 Extraction de Caractéristiques Manuelles

Les premières approches pour utiliser les graphes dans l'apprentissage automatique reposaient souvent sur l'extraction manuelle de caractéristiques à partir de la structure du graphe. Ces caractéristiques pouvaient inclure des mesures comme le degré des nœuds, la centralité, ou encore le coefficient de clustering. Ces informations étaient ensuite utilisées comme entrée pour des algorithmes d'apprentissage traditionnels, tels que les arbres de décision ou les machines à vecteurs de support (SVM).

Bien que ces approches aient offert des résultats intéressants, elles étaient limitées par leur dépendance à l'expertise humaine pour identifier les caractéristiques pertinentes. De plus, elles peinaient à capturer les relations complexes et les dépendances entre les nœuds.

4.2.2 Avènement des Méthodes d'Apprentissage de Représentation sur les Graphes

L'apprentissage de la représentation des graphes a émergé pour surmonter les limitations des méthodes traditionnelles qui peinent à capturer la complexité des structures de graphe. Il s'agit de transformer les noeuds, arêtes, ou graphes en vecteurs continus (embeddings) qui reflètent leurs propriétés structurelles et sémantiques, utilisables ensuite pour diverses tâches d'apprentissage.

DeepWalk[23] a été l'une des premières méthodes à appliquer des parcours aléatoires pour générer des séquences de noeuds, permettant ainsi de créer des embeddings. Bien que cette approche capture efficacement les relations locales, elle est limitée par la nature aléatoire des parcours, ce qui peut nuire à la compréhension des structures globales.

Node2vec[22] améliore DeepWalk en introduisant des parcours biaisés, permettant de mieux explorer à la fois les structures locales et globales du graphe. Cependant, il reste dépendant de la qualité des parcours et n'intègre pas explicitement les attributs des nœuds.

Des méthodes comme LINE et HOPE ont aussi cherché à améliorer la capture des relations globales, mais elles partagent les mêmes limites que leurs prédecesseurs. Face à ces défis, les Graph Neural Networks (GNNs) ont émergé comme une solution plus flexible et puissante, capable d'intégrer à la fois les structures et les attributs des graphes, surpassant ainsi les méthodes traditionnelles.

4.2.3 Transition vers les réseaux neuronaux graphiques (GNN)

La transition vers les Graph Neural Networks (GNNs) répond efficacement aux limitations des méthodes traditionnelles d'apprentissage de graphes. Ces approches classiques présentent plusieurs contraintes notables : elles ne peuvent pas générer des embeddings pour les noeuds absents du jeu d'entraînement, ne capturent pas toujours la similarité structurelle des graphes, et ne tirent pas pleinement parti des caractéristiques des noeuds, des arêtes et des graphes dans leur ensemble. Cette incapacité à traiter les nouveaux nœuds et à intégrer des informations contextuelles réduit considérablement leur flexibilité et leur performance dans des scénarios dynamiques et complexes.

Les GNNs offrent une solution élégante à ces problèmes en permettant une intégration complète des structures et des attributs des graphes. Ils surmontent les limitations des méthodes antérieures en générant des embeddings pour de nouveaux noeuds, en capturant des similarités structurelles profondes et en exploitant pleinement les caractéristiques des noeuds et des arêtes. Cette capacité à traiter les graphes de manière plus holistique et adaptative fait des GNNs un choix puissant pour une variété d'applications modernes, offrant ainsi une avancée significative par rapport aux techniques précédentes.

À travers les sections à venir, nous explorerons en profondeur les GNNs : de leurs principaux concepts qui déterminent les propriétés et les architectures des réseaux de neurones sur les graphes et leur structure générale à ses types différentes, ainsi que la résolution de différents problèmes sur les graphes.

4.3 Les concepts principaux

Dans la section suivante, nous allons décrire les principaux concepts qui permettent les réseaux de neurones graphiques : la nature des données non euclidiennes, l'équivariance et l'invariance par permutation, les voisinages de graphes, ainsi que les concepts de propagation de messages neuronaux.

4.3.1 Données euclidiennes et Données non euclidiennes

Alors que les modèles d'apprentissage en profondeur ont rencontré un succès particulier dans le traitement de signaux tels que la parole, les images ou la vidéo, dans lesquels il existe une structure euclidienne sous-jacente, il y a récemment eu un intérêt croissant à essayer d'appliquer l'apprentissage

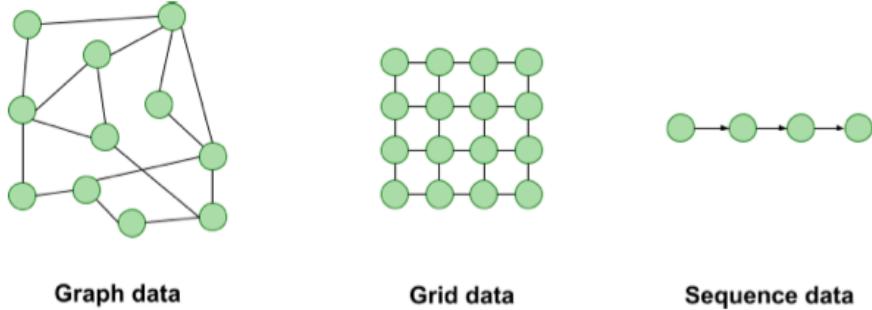


FIGURE 4.2 – Différents types de données couramment utilisées. Les données de graphe peuvent être considérées comme une généralisation des données en grille et en séquence, où la structure préalable peut être définie de manière arbitraire.

sur des données géométriques non euclidiennes. La nature non euclidienne de ces données ici signifie généralement qu'il n'existe pas de systèmes de coordonnées communs, de priorisations de données ou de structures communes pour représenter de telles données [17]. Ainsi, les approches de base qui fonctionnent sur des données euclidiennes échouent lorsqu'il s'agit de leur généralisation, le cas non euclidien, où la structure préalable peut être représentée de manière arbitraire (voir la figure 4.2). En effet, les données que nous utilisons pour entraîner les modèles d'apprentissage profond appartiennent à deux domaines principaux :

- (a) **Données euclidiennes** : données représentées dans des espaces linéaires multidimensionnels, elles obéissent aux postulats euclidiens. Les exemples incluent les données textuelles ou tabulaires (par exemple, le prix du Dogecoin au fil du temps).
- (b) **Données non euclidiennes** : dans des termes encore plus larges, des données qui ne suivent pas les postulats euclidiens (par exemple, dans ces domaines, la distance entre deux points n'est pas une ligne droite). Les exemples incluent les structures moléculaires, les interactions dans les réseaux sociaux, ou les surfaces tridimensionnelles maillées. (Voir la figure 4.3)

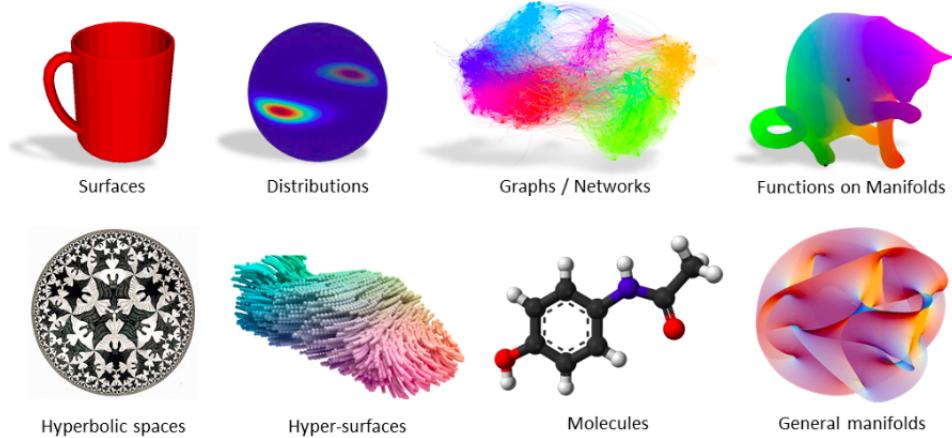


FIGURE 4.3 – Exemples des données non eulidiennes

De tels types de données se rencontrent dans de nombreuses applications. Par exemple, dans les réseaux sociaux, les caractéristiques des utilisateurs peuvent être modélisées comme des signaux sur les sommets du graphe social. Les réseaux de capteurs sont des modèles de graphe de capteurs distribués interconnectés, dont les lectures sont modélisées comme des signaux dépendant du temps sur les sommets. En génétique, les données d'expression génique sont modélisées comme des signaux définis sur le réseau de régulation. En neurosciences, des modèles de graphe sont utilisés pour représenter les structures anatomiques et fonctionnelles du cerveau. En info-graphie et vision par ordinateur, les objets 3D sont modélisés comme des variétés riemanniennes (surfaces) dotées de propriétés telles que la texture des couleurs.

4.3.2 Équivariance et invariance par permutation

Lorsque l'on applique des méthodes d'apprentissage profond aux données de graphe, une idée courante et raisonnable serait d'utiliser un réseau de

neurones, comme un perceptron multicouche (MLP), sur la matrice d'adjacence du graphe en tant qu'entrée. Avec une telle approche, nous pourrions tenter de générer des représentations de graphe e_G à partir de la concaténation des lignes d'une matrice d'adjacence aplatie A :

$$\mathbf{e}_G = \text{MLP}([\mathbf{A}[1] \oplus \mathbf{A}[2]\mathbf{A} \oplus \dots \oplus \mathbf{A}[|V|]]) \quad (4.1)$$

Cette approche serait dépendante de l'ordre des noeuds utilisé dans la matrice d'adjacence. En d'autres termes, si nous devions fournir la même structure de graphe mais avec les noeuds dans un ordre différent, alors l'encodage du graphe serait différent. En fait, une telle approche est dite non invariante par permutation.

La propriété clé pour concevoir des réseaux de neurones qui fonctionnent sur des graphes est qu'ils doivent être invariants ou équivariants par permutation. Plus formellement, une fonction f qui opère sur une matrice d'adjacence A en tant qu'entrée doit satisfaire l'une des deux propriétés suivantes :

$$f(\mathbf{P}\mathbf{A}\mathbf{P}^\top) = f(\mathbf{A}) \quad (\text{Invariance par Permutation}) \quad (4.2)$$

$$f(\mathbf{P}\mathbf{A}\mathbf{P}^\top) = \mathbf{P}f(\mathbf{A}) \quad (\text{Equivariance par Permutation}) \quad (4.3)$$

Où P est une matrice de permutation. Intuitivement, l'invariance par permutation (*permuter l'entrée, la sortie reste la même et en mappant un graphe à un vecteur*) signifie que la fonction f ne dépend pas de l'ordre des lignes/colonnes de la matrice d'adjacence. L'équivariance par permutation, de manière similaire (*permuter l'entrée, la sortie se permute également en conséquence et en mappant le graphe à une matrice*), signifie que la sortie

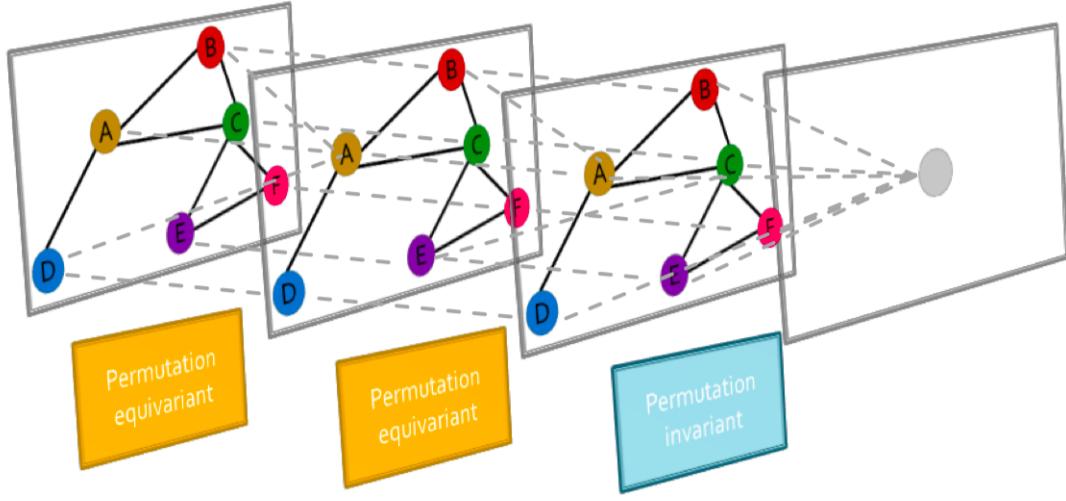


FIGURE 4.4 – Aperçu d'un exemple des couches d'un GNN : Les réseaux de neurones sur les graphes sont constitués de plusieurs fonctions équivariantes ou invariantes par permutation.

de f est permutée de manière cohérente lorsque la matrice d'adjacence est permutée.

Pourquoi voulons-nous étudier ces deux propriétés, l'invariance et l'équivariance ? C'est parce que ces propriétés sont très importantes et constituent, en essence, les éléments de base pour les méthodes d'apprentissage sur les graphes. La figure 4.4 présente ici un exemple de méthode d'apprentissage sur les graphes. Vous pouvez voir que cela implique généralement plusieurs couches équivariantes par permutation et des couches invariantes par permutation. C'est ainsi que nous nous assurons que les méthodes d'apprentissage profond sur les graphes peuvent représenter fidèlement le graphe.

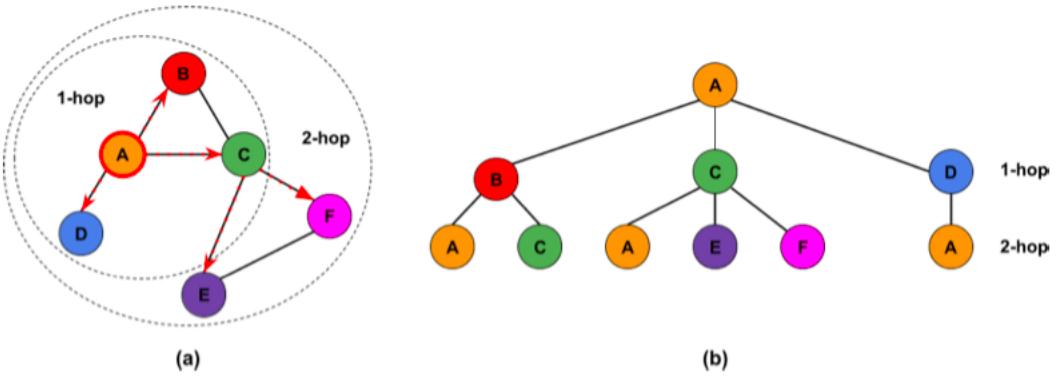
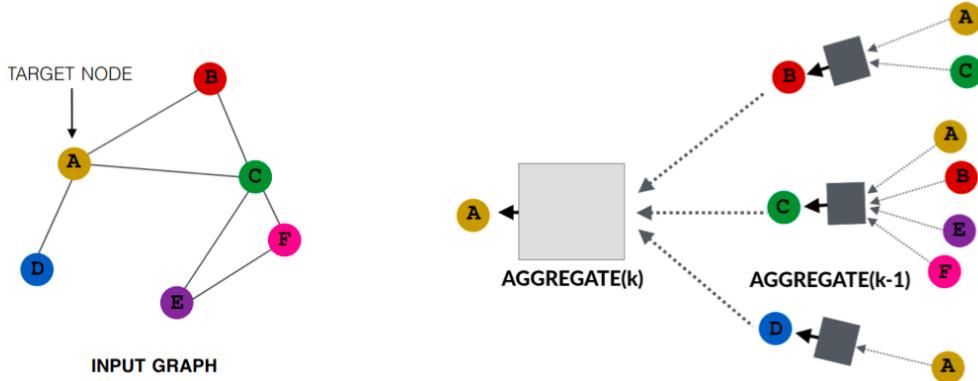


FIGURE 4.5 – (a) Voisinages à 1 saut et à 2 sauts d'un nœud cible donné A .
(b) Structure arborescente correspondant au voisinage à 2 sauts du nœud A . Les structures arborescentes des voisinages représentent implicitement des graphes de calcul qui peuvent être exploités lors de la conception de différents types de couches GNN.

4.3.3 Voisinage dans un graphe

Étant donné un graphe G , le voisinage d'un nœud donné u est l'ensemble contenant tous les nœuds du graphe qui sont adjacents à u , noté dans ce travail comme $N(u)$ ou N_u . Bien que $N(u)$ soit un ensemble de voisins directs de u , certains auteurs incluent également u lui-même dans l'ensemble de voisinage. Une extension de ce concept est le voisinage à k -sauts, qui désigne un ensemble contenant tous les nœuds distants d'au plus k connexions d'un nœud donné u , comme visible dans la Figure 4.5 (a).

Une autre propriété des voisinages est qu'ils définissent implicitement des structures arborescentes avec le nœud donné comme racine, comme on peut le voir dans la Figure 4.5 (b). Ce fait sera exploité dans les sections suivantes lors de la définition de la propagation de messages, ainsi que pour différents types de couches GNN.



Source : "Jure Leskovec, Stanford CS224W : Machine Learning with Graphs, <http://cs224w.stanford.edu>"

FIGURE 4.6 – Pour un graphe d’entrée donné et un nœud cible A , les nouvelles représentations de caractéristiques peuvent être calculées en prenant les représentations agrégées des voisins de A B , C , D , dont les caractéristiques sont à leur tour calculées en prenant les représentations de leurs propres voisins. Cette visualisation représente deux couches d’un modèle de propagation de messages et peut être représentée à l’aide d’une structure arborescente en dépliant les voisinages autour du nœud cible.

4.3.4 Propagation de Messages Neuronaux

Lorsqu’on travaille avec des graphes, nous traitons plus spécifiquement des informations associées à leurs nœuds, en termes de caractéristiques des nœuds, et à la connectivité du graphe, en termes d’arêtes. Afin que les modèles de réseaux neuronaux de graphiques apprennent des caractéristiques/représentations de noeuds efficaces, nous devons introduire le concept de propagation de messages neuronaux. (Voir la figure 4.6).

Au cœur de la propagation de messages neuronaux, on suppose que chaque nœud $u \in V$ dans un graphe donné $G = (V, E)$, est associé à un vecteur de caractéristiques $x_u \in \mathbb{R}^d$, où d est une dimension de caractéristiques. Pour mettre à jour le vecteur de caractéristiques d’un nœud u , produisant un nouveau vecteur de caractéristiques h_u , nous devons être capables de collecter les informations de caractéristiques provenant des voisins du nœud, ainsi que d’intégrer ces informations pour produire une nouvelle représentation pour le nœud. En termes de propagation de messages, nous avons

besoin de deux fonctions AGGREGATE et UPDATE qui peuvent être utilisées de la manière suivante pour produire une nouvelle représentation :

$$\mathbf{h}_u^{(k+1)} = \text{UPDATE}^{(k)} \left(\mathbf{h}_u^{(k)}, \text{AGGREGATE}^{(k)} \left(\left\{ \mathbf{h}_v^{(k)}, \forall v \in \mathcal{N}(u) \right\} \right) \right) \quad (4.4)$$

$$\mathbf{h}_u^{(k+1)} = \text{UPDATE}^{(k)} \left(\mathbf{h}_u^{(k)}, \mathbf{m}_{\mathcal{N}(u)}^{(k)} \right) \quad (4.5)$$

où AGGREGATE et UPDATE sont deux fonctions différentiables arbitraires (k) (par exemple, des réseaux neuronaux) et $m_{\mathcal{N}(u)}$ est dit être un « message » qui est agrégé à partir du voisinage de u , $\mathcal{N}(u)$.

Remarque : Étant donné que l'opération AGGREGATE agit sur des ensembles en entrée, les GNN définis de cette manière sont, par conception, équivariants par permutation.

L'indice (k) est utilisé pour indiquer les embeddings et les fonctions à différentes couches/itérations de la propagation de messages. En effet, à chaque couche/itération k de cette procédure, la fonction AGGREGATE prend en entrée un ensemble d'embeddings $e^{(k)}$ des voisins du noeud u , $\mathcal{N}(u)$, $\{h_v, \forall v \in \mathcal{N}(u)\}$, et construit un message $m_{\mathcal{N}(u)}^{(k)}$ basé sur les informations agrégées des voisins. La fonction UPDATE^(k) combine ensuite ce message $m_{\mathcal{N}(u)}^{(k)}$ avec l'embedding précédent du noeud u , $h_u^{(k-1)}$, pour générer un embedding mis à jour $h_u^{(k)}$.

Après avoir effectué K itérations de propagation de messages, nous pouvons utiliser la représentation finale à l'étape K pour définir les embeddings de chaque noeud comme suit :

$$\mathbf{z}_u = \mathbf{h}_u^{(K)}, \forall u \in \mathcal{V} \quad (4.6)$$

La représentation finale des nœuds calculée de la manière précédemment décrite permet au modèle d'encoder à la fois des informations basées sur les caractéristiques et des informations structurelles. Les informations structurelles sont les informations qui proviennent implicitement de la connectivité du graphe, telles que les degrés de tous les nœuds dans un voisinage à k sauts, ce qui trouve des usages dans des domaines comme la prédiction des propriétés moléculaires ou les systèmes de recommandation, pour utiliser la structure implicite du graphe que d'autres modèles non basés sur les graphes ignoreraient ou devraient utiliser explicitement.

Les informations basées sur les caractéristiques sont les informations capturées à partir des embeddings réels des nœuds. Grâce à la manière dont la propagation de messages est définie, les nœuds agrègent les caractéristiques provenant de leurs voisins, leur permettant d'encoder les caractéristiques de l'ensemble du voisinage à k sauts, de manière similaire à ce que font les réseaux neuronaux convolutionnels avec des informations spatiales locales.

Comme indiqué précédemment, étant donné que les GNN effectuent une agrégation locale des caractéristiques sur la structure du graphe, le nombre de couches dans un GNN est un hyperparamètre qui influence le nombre de sauts de voisinage que le réseau effectuera à partir d'un seul nœud central. Définir cet hyperparamètre à une valeur élevée (c'est-à-dire plus de 6) peut en fait entraîner une sur-lissage des caractéristiques, en raison de l'empilement de trop nombreuses couches.

Dans ce qui suit, nous allons découvrir la structure générale d'un réseau de neurones graphiques (GNN) mais avant, examinons d'abord la structure de base d'un réseau neuronal traditionnel.

4.4 Structure générale d'un GNN

Un GNN est une forme de réseau neuronal qui partage une structure générale similaire. Tout d'abord, nous explorerons la structure générale d'un réseau neuronal standard. Ensuite, nous examinerons celle d'un GNN, qui diffère principalement dans sa phase d'extraction des caractéristiques.

4.4.1 La structure générale d'un réseau neuronal

La structure générale d'un réseau neuronal quelconque peut être décomposée en plusieurs parties principales, qui suivent souvent un schéma similaire (voir le figure 4.7). Voici une vue d'ensemble de ces composants :

- (a) **Input (Entrée)** : Les données brutes sont introduites dans le réseau neuronal. Selon le type de problème, l'entrée peut être sous forme de vecteurs (pour des données structurées comme des nombres), de matrices (pour des images ou des données audio), ou de séquences (pour des données séquentielles comme du texte).
- (b) **Phase d'Extraction des Caractéristiques** : Cette phase comprend les couches cachées du réseau neuronal, où les caractéristiques significatives sont extraites des données d'entrée. Cela se fait par l'application de transformations linéaires et non linéaires par les neurones et les fonctions d'activation au sein de chaque couche
- (c) **Tâches Spécifiques** : Les différentes tâches spécifiques sont effectuées par des couches particulières du réseau neuronal en fonction de l'architecture et du type de réseau utilisé : Classification, Régression, Génération..
- (d) **Output (Sortie)** : Les résultats finaux sont générés par la couche de sortie du réseau neuronal, qui convertit les représentations internes apprises par le réseau en une forme appropriée pour la tâche spécifique

(par exemple, des probabilités pour la classification, des valeurs réelles pour la régression, etc.).

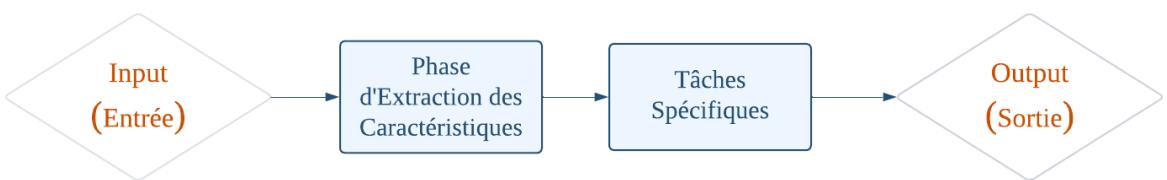


FIGURE 4.7 – Structure générale d'un réseau neuronal

Après avoir générée les sorties du réseau neuronal et définir la fonction objective, l'étape d'optimisation ajuste les paramètres du modèle pour minimiser cette fonction et pour affiner les performances du modèle. Cela se fait en deux étapes clés :

- **Calcul du Gradient** : Utilisation de la rétropropagation pour calculer les gradients de la fonction de perte par rapport aux poids du modèle.
- **Mise à Jour des Poids** : Les poids sont ajustés en fonction des gradients, généralement avec des algorithmes d'optimisation comme SGD ou Adam, pour minimiser la fonction de perte.

Cette phase d'optimisation est essentielle pour la performance du modèle, car elle permet d'ajuster les paramètres du réseau neuronal afin qu'il puisse accomplir efficacement les tâches spécifiques pour lesquelles il a été conçu.

4.4.2 Pipeline d'un réseau de neurones graphiques

(a) Entrée du Graphe

- **Données du Graphe :** Le graphe est décrit par ses nœuds, ses arêtes, ainsi que par les caractéristiques associées à chacun. Ces données peuvent être formatées sous forme de matrices d'adjacence et de caractéristiques des nœuds, ou bien présentées sous forme de listes d'arêtes ou dans une représentation hétérogène.

(b) Propagation des Messages avec les Couches de GNN

- **Couches de GNN :** Les couches du GNN effectuent des opérations de propagation des messages, d'agrégation et de mise à jour des caractéristiques des nœuds ou arêtes (voir la partie 4.3.4).
- **Génération des Embeddings :** Après plusieurs couches de GNN, chaque nœud (ou arête) est représenté par un vecteur d'embedding qui encode les informations de structure et de caractéristiques du graphe.

(c) Transformation des Embeddings pour Tâches Spécifiques

- **Transformation des Embeddings :** Les embeddings des nœuds sont transformés en embeddings adaptés à la tâche spécifique : Tâches au niveau des nœuds, des liens ou du graphe entier (voir la partie 4.6).
- **Couche de Modèle Supplémentaire :** Les embeddings générés sont ensuite passés à une couche ou à un modèle supplémentaire (comme un MLP) qui transforme ces embeddings en prédictions ou en résultats spécifiques à la tâche (par exemple, classification, prédiction de liens, régression).

(d) Définition d'une Fonction Objectif

- **Fonction de Perte :** Une fonction objectif (ou fonction de perte) est définie pour évaluer la performance du modèle. Cela peut inclure :
 - **Perte Point-Wise :** Évalue chaque point de données de manière indépendante (Par exemple, la cross-entropy pour les tâches de classification, l'erreur quadratique moyenne pour la régression..)
 - **Perte Pair-Wise :** Évalue les relations entre paires d'exemples. Cela est utilisé dans des tâches où la comparaison entre paires est importante où l'on cherche à maximiser la similarité entre des paires positives et minimiser la similarité entre des paires négatives.

(e) Optimisation

- **Calcul du Gradient :** Les gradients de la fonction de perte par rapport aux paramètres du modèle sont calculés en utilisant des méthodes telles que la rétropropagation.
- **Mise à Jour des Poids :** Les paramètres du modèle sont mis à jour en fonction des gradients calculés, généralement en utilisant des optimisateurs comme Stochastic Gradient Descent (SGD) ou Adam.

Ce pipeline décrit le processus complet de traitement et d'apprentissage dans un modèle de GNN, depuis l'entrée des données jusqu'à l'optimisation des paramètres du modèle pour accomplir des tâches spécifiques sur les graphes.

4.5 Les types des GNNs

Ayant décrit certaines des motivations théoriques derrière les réseaux de neurones graphiques dans la section précédente, y compris une définition

abstraite du passage de messages neuronaux et la pipeline d'un réseau de neurones graphiques, nous décrirons dans cette section comment différents types de couches de GNN concrètes peuvent être définis en termes de fonctions d'agrégation de voisinage et de mise à jour.

Lorsqu'on parle des différents types de couches de GNN, ce qui change principalement entre les variantes est la manière dont les fonctions UPDATE et AGGREGATE sont définies. Parfois, le choix du type d'échantillonnage de voisins ou le choix des opérations de non-linéarité affectent également la couche. Les exemples typiques de fonctions de mise à jour incluent la moyenne, le max, les réseaux neuronaux et les réseaux neuronaux récurrents, à appliquer au message entrant et à l'état précédent du nœud. Pour les fonctions d'agrégation, il y a généralement la moyenne, le max pooling, le pooling de somme normalisée et le réseau neuronal, appliqués à toutes les informations entrantes provenant des nœuds voisins.

Un travail précoce, par exemple, les réseaux convolutionnels graphiques (GCN) [18] (décrits plus tard) utilisent une somme normalisée des embeddings des voisins comme fonction d'agrégation et intègrent la fonction de mise à jour dans l'agrégation en ajoutant une boucle sur soi-même. Les réseaux d'attention sur graphes (GAT)[19] utilisent des poids d'attention à l'intérieur de la fonction d'agrégation pour pondérer la somme en fonction d'un score d'attention associé à chaque paire de nœuds. Dans ce qui suit, nous présenterons plusieurs modèles de GNN révolutionnaires pour expliquer comment les réseaux neuronaux sont implémentés sur les graphes.

4.5.1 Agrégation Simple de Voisinage

Étant donné un graphe $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ avec N nœuds ($|\mathcal{V}| = N$) et sa matrice d'adjacence $A \in \mathbb{B}^{N \times N}$, le type le plus simple de couche GNN qui utilise l'agrégation de voisinage pour agréger les caractéristiques associées à un noeud i pourrait être défini comme suit :

$$\mathbf{h}_i^{(l+1)} = \sigma \left(\sum_{j \in \mathcal{N}(i)} \mathbf{W} h_j^{(l)} \right) \quad (4.7)$$

Ici, nous supposons que $\mathbf{h}_i \in \mathbb{R}^K$ est un vecteur de caractéristiques associé au i -ème nœud du graphe et que $\mathbf{W} \in \mathbb{R}^{K \times D}$ est une matrice de transformation apprenable qui mappe les dimensions des caractéristiques de K à D , produisant une nouvelle représentation de nœud $\mathbf{h}_i^* \in \mathbb{R}^D$.

En notant qu'une matrice d'adjacence A d'un graphe représente la situation de voisinage pour chaque nœud, nous pourrions définir $\tilde{A} = A + I_N$ pour étendre la matrice d'adjacence avec des boucles de soi et réécrire (4.7) sous une forme vectorielle plus compacte et efficacement calculable :

$$H^{(l+1)} = \sigma \left(\tilde{A} \mathbf{W} H^{(l)} \right) \quad (4.8)$$

Cette formule matricielle est une généralisation de la formule scalaire-Cette forme vectorielle. Elle permet de calculer efficacement les nouvelles représentations des nœuds en utilisant des opérations matricielles et traiter efficacement les graphes entiers plutôt que de traiter les nœuds un par un. Elle est particulièrement utile pour les implémentations dans les frameworks de deep learning comme PyTorch¹, où les opérations matricielles sont très optimisées.

1. Pour plus d'informations, consultez le site officiel de PyTorch : <https://pytorch.org>.

4.5.2 Réseaux convolutionnels sur graphes

GCN est inspiré par les réseaux de neurones convolutifs (CNN) appliqués aux graphes. L'idée principale est d'exploiter les informations locales des voisins d'un noeud pour mettre à jour ses caractéristiques. Par ailleurs, la convolution de graphe (GCN) [18] étend le concept d'agrégation de voisinage décrit dans la section précédente en incorporant un facteur de normalisation, qui est calculé en fonction des degrés des noeuds. En particulier, elle utilise une matrice de normalisation $S = \tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}$, où \tilde{D} est la matrice des degrés avec la diagonale $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$. Le rôle de cette nouvelle matrice des degrés est de pondérer les embeddings entrants des voisins v_j par le facteur $\frac{1}{\sqrt{\deg(v_i)\deg(v_j)}}$, lors de l'exécution du processus d'agrégation pour le noeud v_i . Nous pouvons écrire la formule pour calculer l'agrégation d'un GCN, basée sur l'équation (4.7) :

$$h_i^{(l+1)} = \sigma \left(\sum_{j \in N_i} \frac{1}{S_{ij}} \mathbf{W}^{(l)} h_j^{(l)} \right) \quad (4.9)$$

La normalisation induite par S est appelée normalisation symétrique dans la littérature sur les GNN. Elle permet d'éviter les problèmes de mise à l'échelle des caractéristiques des noeuds. Il existe également une normalisation "gauche", qui correspond à la normalisation des messages par les degrés entrants de chaque noeud, ce qui équivaut à la moyenne des messages reçus :

$$\frac{1}{\deg(v_1)}.$$

4.5.3 Réseaux d'attention sur graphes

Les réseaux d'attention sur graphes (GAT) [19] sont une autre généralisation des réseaux convolutionnels sur graphes, où l'influence des différents noeuds voisins est pondérée par un score d'attention calculé dynamiquement, plutôt que par un facteur constant lors de l'opération d'agrégation.

L'idée est que tous les voisins ne sont pas également importants, donc il faut apprendre un poids d'attention pour chaque voisin. La formule de mise à jour des caractéristiques peut être décrite par l'équation suivante :

$$\mathbf{h}_i^{(l+1)} = \sigma \left(\sum_{j \in \mathcal{N}_i} \alpha_{ij}^{(l)} \mathbf{z}_j^{(l)} \right) \quad (4.10)$$

Cette formule peut être décomposée en plusieurs étapes clés :

- (a) **Transformation des Caractéristiques** : Chaque nœud i dans le graphe est associé à un vecteur de caractéristiques $\mathbf{h}_i^{(l)}$. Ce vecteur est transformé en un vecteur intermédiaire $\mathbf{z}_i^{(l)}$ en utilisant une matrice de transformation apprenable $\mathbf{W}^{(l)}$, définie comme suit :

$$\mathbf{z}_i^{(l)} = \mathbf{W}^{(l)} \mathbf{h}_i^{(l)} \quad (4.11)$$

où $\mathbf{W}^{(l)} \in \mathbb{R}^{D \times D'}$ est la matrice qui adapte les caractéristiques de dimension D à une nouvelle dimension D' .

- (b) **Calcul des Scores d'Attention** : Pour chaque paire de nœuds i et j , un score d'attention $e_{ij}^{(l)}$ est calculé afin de mesurer l'importance relative du nœud j dans l'agrégation des caractéristiques du nœud i . Ce score est obtenu en appliquant une fonction d'attention $a^{(l)}$ à la concaténation des vecteurs $\mathbf{z}_i^{(l)}$ et $\mathbf{z}_j^{(l)}$, suivie d'une activation LReLU :

$$e_{ij}^{(l)} = \text{LReLU} \left(a^{(l)} \left(\mathbf{z}_i^{(l)} \| \mathbf{z}_j^{(l)} \right) \right) \quad (4.12)$$

Ici, LReLU est une fonction d'activation qui ajoute une non-linéarité au calcul du score d'attention.

- (c) **Normalisation des Scores d'Attention** : Les scores d'attention $e_{ij}^{(l)}$ sont normalisés en utilisant la fonction softmax pour obtenir des coefficients d'attention $\alpha_{ij}^{(l)}$. Ces coefficients indiquent l'importance des voisins

dans l'agrégation des caractéristiques :

$$\alpha_{ij}^{(l)} = \frac{\exp(e_{ij}^{(l)})}{\sum_{k \in \mathcal{N}_i} \exp(e_{ik}^{(l)})} \quad (4.13)$$

Cette normalisation assure que les coefficients sont compris entre 0 et 1 et que leur somme est égale à 1.

- (d) **Agrégation Pondérée des Caractéristiques** : Les caractéristiques des voisins sont agrégées en utilisant les coefficients d'attention $\alpha_{ij}^{(l)}$. La somme pondérée des caractéristiques est ensuite transformée en une nouvelle représentation $\mathbf{h}_i^{(l+1)}$ du nœud i en appliquant une fonction d'activation σ :

$$\mathbf{h}_i^{(l+1)} = \sigma \left(\sum_{j \in \mathcal{N}_i} \alpha_{ij}^{(l)} \mathbf{z}_j^{(l)} \right) \quad (4.14)$$

où σ est généralement une fonction d'activation non linéaire, telle que ReLU.

4.5.4 Échantillonnage et Agrégation sur Graphes

Dans le cas des réseaux de neurones sur graphes, les approches existantes peuvent être classifiées en approches inductives et transductives :

- **Transductive** : Tous les nœuds évalués au moment du test étaient également observés au moment de l'entraînement.
- **Inductive** : Les nœuds présents au moment du test n'étaient pas nécessairement présents dans les données d'entraînement.

Le travail de GraphSAGE[19] (Graph Sample and Aggregation) peut être considéré comme une extension inductive des Réseaux Convolutionnels sur Graphes (GCN) et des Réseaux d'Attention sur Graphes (GAT). L'idée de leur approche est de ne considérer qu'un voisinage de taille fixe

en échantillonnant le voisinage complet d'un nœud donné lors de l'étape d'agrégation, ce qui permet à la fois de rendre l'étape d'agrégation plus efficace et de travailler sur des nœuds non vus dans l'ensemble d'entraînement, car la matrice de Laplace complète du graphe n'est plus nécessaire pour effectuer l'agrégation.

En plus de cette extension inductive, le travail introduit également une méthode pour apprendre des représentations de nœuds dans un cadre non supervisé en définissant la fonction de perte suivante :

$$J_{\mathcal{G}}(\mathbf{z}_u) = -\log \left(\sigma \left(\mathbf{z}_u^\top \mathbf{z}_v \right) \right) - Q \cdot \mathbb{E}_{v_n \sim P_n(v)} \log \left(\sigma \left(-\mathbf{z}_u^\top \mathbf{z}_{v_n} \right) \right) \quad (4.15)$$

où :

- Les vecteurs de caractéristiques pour les nœuds u et v sont respectivement notés \mathbf{z}_u et \mathbf{z}_v .
- Le premier terme de la fonction de perte,

$$-\log \left(\sigma \left(\mathbf{z}_u^\top \mathbf{z}_v \right) \right),$$

encourage les nœuds voisins à avoir des représentations similaires, où σ est la fonction sigmoïde qui donne la probabilité que les nœuds u et v soient voisins.

- Le deuxième terme de la fonction de perte,

$$-Q \cdot \mathbb{E}_{v_n \sim P_n(v)} \log \left(\sigma \left(-\mathbf{z}_u^\top \mathbf{z}_{v_n} \right) \right),$$

utilise l'échantillonnage négatif pour encourager des représentations distinctes pour les nœuds qui ne sont pas voisins. $P_n(v)$ est une distribution d'échantillonnage négatif avec Q échantillons.

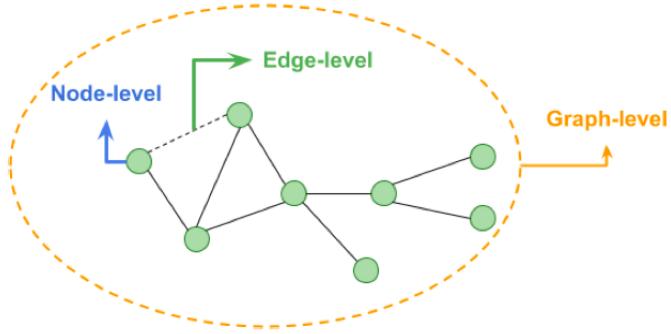


FIGURE 4.8 – Divers niveaux courants de tâches peuvent être résolus sur des structures de graphes par des approches d'apprentissage automatique qui utilisent des embeddings de nœuds. Les tâches au niveau des nœuds consistent à prédire une propriété d'un nœud individuel (par exemple, l'âge d'un utilisateur dans un graphe de réseau social). Les tâches au niveau des arêtes consistent à prédire une propriété ou l'existence d'une arête (par exemple, la note qu'une personne a donnée à un produit, ou si une transaction a eu lieu ou non). Les tâches au niveau du graphe consistent à prédire une propriété d'un graphe entier (par exemple, le type d'une molécule donné un graphe moléculaire).

Cette fonction de perte (Perte Pair-Wise) est utilisée pour entraîner des modèles comme GraphSAGE en apprenant à distinguer les nœuds voisins des nœuds non voisins dans l'espace des caractéristiques. Bien que GraphSAGE puisse utiliser diverses fonctions d'agrégation et d'apprentissage, la fonction de perte de type *Perte par Échantillonnage Négatif*¹ est typique pour l'entraînement dans le cadre des représentations non supervisées des graphes.

4.6 Problèmes sur les graphes

Ayant décrit comment les types de couches de base permettent de calculer de nouvelles représentations des embeddings de nœuds en exploitant la structure du graphe, cette section décrira brièvement les techniques génér-

1. Représentations non supervisées, souvent utilisée avec des modèles comme GraphSAGE.

rales qui peuvent être appliquées aux embeddings de nœuds pour résoudre différentes tâches au niveau des nœuds, des arêtes et des graphes, comme illustré dans la Figure 4.8.

4.6.1 Tâches au niveau des nœuds :

Pour aborder les tâches au niveau des nœuds avec un réseau de neurones graphiques (GNN), une approche courante consiste à appliquer une tête de tâche spécifique, telle qu'une tête de classification, sur les caractéristiques des nœuds apprises. L'idée derrière cette approche est qu'après avoir traversé les couches du GNN, les représentations des nœuds, ayant intégré les informations de la structure du graphe, peuvent être traitées comme des vecteurs de données. Ces vecteurs peuvent ensuite être utilisés avec des modèles d'apprentissage automatique classiques pour des tâches telles que la classification.

4.6.1.1 Application d'un Classificateur Linéaire

Voici un exemple d'application d'un classificateur linéaire aux embeddings des nœuds $\{e_i \mid i \in \mathcal{V}_{\text{train}}\}$ obtenus par le GNN :

Calcul des Scores de Classification : Les scores de classification pour chaque nœud i sont calculés par :

$$\mathbf{s}_i = \mathbf{W}^{(clf)} e_i + \mathbf{b}^{(clf)} \quad (4.16)$$

où :

- \mathbf{s}_i est le vecteur des scores de classification pour le nœud i .
- $\mathbf{W}^{(clf)}$ est la matrice de poids du classificateur linéaire.
- e_i est l'embedding du nœud i .
- $\mathbf{b}^{(clf)}$ est le biais du classificateur.

Calcul de la Fonction de Perte : La fonction de perte est ensuite calculée pour évaluer l'écart entre les scores de prédiction et les véritables étiquettes de lien. La cross-entropy est couramment utilisée comme mesure de perte :

$$\mathcal{L}_{\text{Cross-Entropy}} = \sum_{i \in \mathcal{V}_{\text{train}}} -\log(\text{softmax}(\mathbf{s}_i)) \quad (4.17)$$

où :

- \mathcal{L} est la perte totale pour tous les nœuds dans l'ensemble d'entraînement $\mathcal{V}_{\text{train}}$.
- $\text{softmax}(\mathbf{s}_i)$ transforme les scores de classification en probabilités.

En résumé, après avoir obtenu les représentations des nœuds à l'aide des couches du GNN, un classificateur linéaire est utilisé pour prédire des étiquettes pour chaque nœud. Les scores de classification sont calculés en appliquant une transformation linéaire aux embeddings des nœuds, suivie de l'ajout d'un biais. La perte est ensuite calculée en utilisant la cross-entropy, qui évalue la précision des prédictions en comparant les probabilités prédites aux véritables étiquettes.

4.6.2 Tâches au niveau des Liens

Pour aborder les tâches au niveau des liens, il est essentiel de calculer des embeddings pour les paires de nœuds considérées, puis d'appliquer une fonction d'interaction pour estimer la probabilité de l'existence d'un lien entre ces nœuds. L'objectif est de transformer ces représentations apprises en vecteurs d'interaction qui capturent la relation entre les nœuds concernés, ce qui permet de générer des embeddings pour les liens eux-mêmes.

4.6.2.1 Application d'une Fonction d'Interaction

Une fois les embeddings des noeuds \mathbf{h}_u et \mathbf{h}_v obtenus, une fonction d'interaction est appliquée pour générer un embedding d'arête \mathbf{e}_{uv} entre les noeuds u et v :

$$\mathbf{e}_{uv} = \text{interaction}(\mathbf{h}_u, \mathbf{h}_v) \quad (4.18)$$

Les fonctions d'interaction couramment utilisées sont :

- **Produit Scalaire :**

$$\mathbf{e}_{uv} = \mathbf{h}_u^\top \mathbf{h}_v \quad (4.19)$$

Ici, l'embedding d'arête est calculé comme le produit scalaire des embeddings des noeuds, ce qui est particulièrement utile dans les tâches de recommandation.

- **Concaténation :**

$$\mathbf{e}_{uv} = \mathbf{h}_u \| \mathbf{h}_v \quad (4.20)$$

Cette méthode combine les embeddings des noeuds u et v en un seul vecteur, permettant ainsi de capturer plus de détails sur la relation entre les noeuds.

- **Perceptron Multicouche (MLP) :**

$$\mathbf{e}_{uv} = \text{MLP}(\mathbf{h}_u \| \mathbf{h}_v) \quad (4.21)$$

Un perceptron multicouche est utilisé ici pour modéliser des interactions non linéaires entre les noeuds, offrant ainsi une flexibilité accrue pour la prédiction des liens.

L'embedding \mathbf{e}_{uv} peut ensuite être utilisé pour réaliser des tâches au niveau des arêtes, telles que la classification des arêtes, en appliquant à

celui-ci le module de tâche approprié.

4.6.3 Tâches au Niveau du Graphe

Les tâches au niveau du graphe consistent à faire des prédictions qui concernent l'ensemble du graphe plutôt que des nœuds ou des arêtes individuels[23]. Pour ce faire, il est nécessaire d'agréger les embeddings de tous les nœuds du graphe afin de générer une représentation globale.

4.6.3.1 Opération de Lecture (Readout)

Afin de combiner les caractéristiques des nœuds à travers tout le graphe, une opération de "readout" est utilisée :

$$\hat{\mathbf{y}} = \text{READOUT}(\{\mathbf{h}_v \mid v \in \mathcal{V}\}) \quad (4.22)$$

où chaque \mathbf{h}_v représente l'embedding du nœud v après la dernière couche du GNN. La fonction READOUT peut être simple, comme une moyenne, ou plus complexe, telle qu'un réseau de neurones. Une version simple de l'opération de readout est donnée par :

$$\hat{\mathbf{y}} = \frac{1}{|\mathcal{V}|} \sum_{v \in \mathcal{V}} \mathbf{h}_v^{(K)} \quad (4.23)$$

L'embedding global $\hat{\mathbf{y}}$ ainsi obtenu peut ensuite être utilisé dans une tête de tâche spécifique, telle qu'une tête de classification, qui sera entraînée pour réaliser la tâche souhaitée au niveau du graphe.

En somme, les tâches au niveau du graphe nécessitent la création d'une représentation globale en agrégeant les embeddings de tous les nœuds. Cette représentation globale est ensuite utilisée pour effectuer des prédictions concernant l'ensemble du graphe, à l'aide d'une tête de tâche appropriée.

GNNs pour les systèmes de recommandation

Dans les chapitres précédents, nous avons présenté un aperçu des systèmes de recommandation en tant que domaine, ainsi que des réseaux neuronaux sur graphes comme une approche pour travailler sur des données structurées sous forme de graphes. Dans ce chapitre, nous allons d'abord voir

5.1 Pourquoi les GNNs sont-ils nécessaires pour les systèmes de recommandation ?

Récemment, les progrès dans les réseaux neuronaux sur les graphes offrent une opportunité solide et fondamentale pour répondre aux problèmes mentionnés dans les systèmes de recommandation. Plus précisément, les réseaux neuronaux sur les graphes adoptent une propagation d'incorporation pour agréger itérativement les incorporations des voisins, comme nous avons vu dans le chapitre précédent. En empilant les couches de propagation, chaque noeud peut accéder à l'information des voisins de haut niveau, plutôt que seulement aux voisins de premier niveau comme le font les méthodes traditionnelles, (voir le [chapitre 4](#)).

En d'autres termes, les réseaux neuronaux sur les graphes sont capables de capturer des informations structurales de haut niveau dans les données, en explorant les relations entre les différents éléments de manière plus pro-

fondé que les méthodes traditionnelles. Ainsi qu'ils offrent plusieurs autres avantages distincts qui les rendent particulièrement adaptés à des tâches impliquant des données structurées sous forme de graphes, de manière générale.

Avec ses avantages dans le traitement des données structurelles et l'exploration des informations structurales, les méthodes basées sur les réseaux neuronaux sur les graphes sont devenues les nouvelles approches de pointe dans les systèmes de recommandation. En résumé, les réseaux neuronaux sur les graphes permettent une modélisation plus sophistiquée des relations entre les utilisateurs, les articles et d'autres éléments, ce qui améliore la précision et la pertinence des recommandations.

5.2 Représentation des données

Lorsqu'on travaille avec des ensembles de données de recommandation, on peut généralement observer quatre types principaux de structures :

- **Interactions utilisateur-élément** : généralement représentées sous forme de matrice, ces interactions décrivent les interactions implicites (par exemple, les clics) ou explicites (par exemple, les évaluations) entre les utilisateurs et les éléments.
- **Séquence de consommation** : décrit une série d'événements de consommation consécutifs.
- **Graphe de relations sociales** : illustre les relations d'interaction entre les utilisateurs.
- **Graphe de connaissances d'un élément** : présente les propriétés des élément.

Comme le montre la figure 5.1, les données de recommandation s'intègrent naturellement dans des structures de graphes. En effet, les inter-

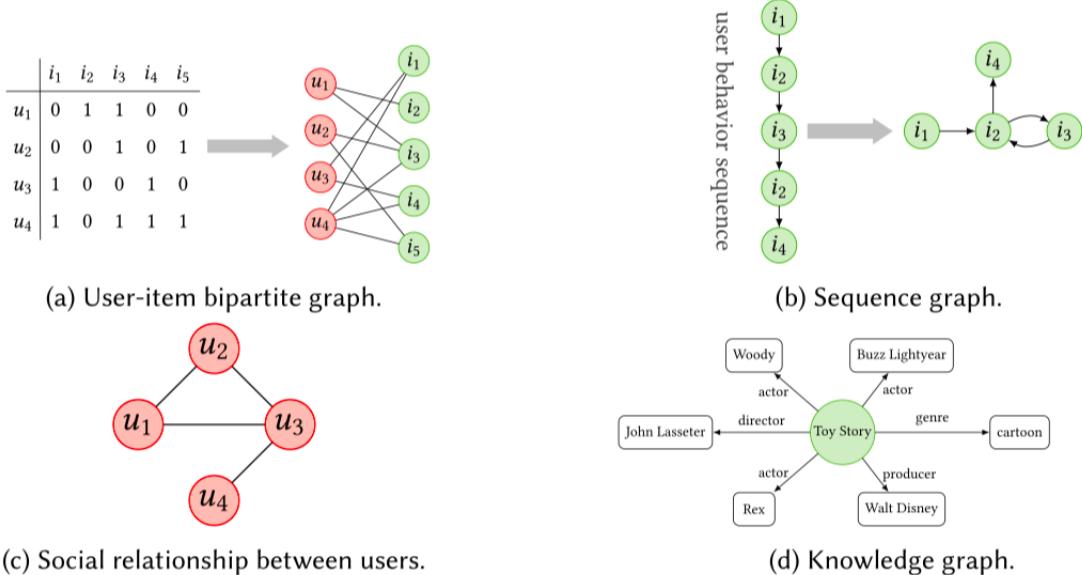


FIGURE 5.1 – Les types de données couramment utilisés dans les systèmes de recommandation peuvent être naturellement représentés à l'aide de structures de graphes.

actions utilisateur-élément peuvent être représentées sous la forme d'un graphe bipartite utilisateur-élément, où d'un côté nous avons les utilisateurs et de l'autre les éléments, tandis que les arêtes représentent l'événement de consommation/ranking d'un élément par un utilisateur. Les séquences de consommation peuvent être modélisées comme un graphe de séquences, où les noeuds représentent les éléments et les arêtes représentent l'ordre entre les événements de consommation relatifs. Les relations sociales sont représentées par un graphe d'interaction sociale, où les noeuds représentent les utilisateurs et les arêtes représentent une interaction entre eux. Enfin, le graphe de connaissances représente déjà les propriétés des éléments via un graphe.

Le domaine de la recommandation basée sur les graphes est riche en littérature et propose une large gamme de modèles adaptés aux différents types de données mentionnés [26, 27, 28]. Les auteurs de l'enquête [25] résument

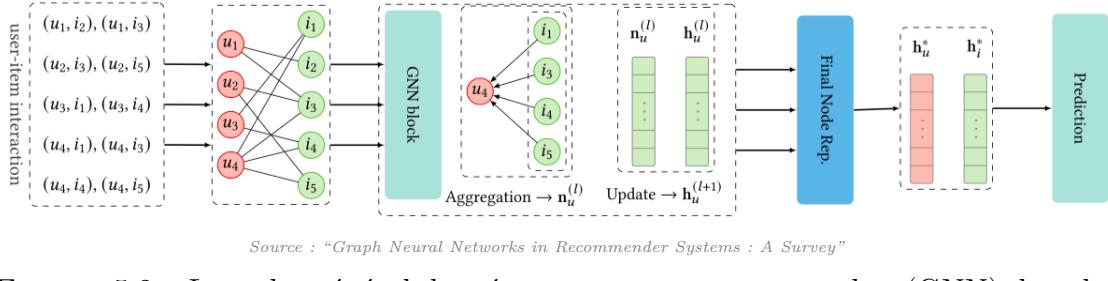


FIGURE 5.2 – Le cadre général des réseaux neuronaux sur graphes (GNN) dans le filtrage collaboratif utilisateur-élément

les principales familles de modèles et offrent une taxonomie pour la majorité d’entre eux. En nous appuyant sur cette classification, les sections suivantes de ce chapitre exploreront l’applicabilité des réseaux neuronaux sur graphes pour les tâches de recommandation. Nous mettrons particulièrement l’accent sur la recommandation basée sur les interactions utilisateur-élément, également connue sous le nom de Filtrage Collaboratif Utilisateur-Élément, ainsi que sur la prédiction des préférences des utilisateurs.

5.2.1 Filtrage collaboratif utilisateur-élément

Étant donné les données d’interaction utilisateur-élément, le principe fondamental du filtrage collaboratif utilisateur-élément est d’utiliser les éléments avec lesquels les utilisateurs ont interagi pour améliorer les représentations des utilisateurs et d’enrichir les représentations des éléments en utilisant les éléments précédemment interactés par les utilisateurs. Les techniques de réseaux neuronaux sur les graphes (GNN) apportent une approche sophistiquée à ce processus, en simulant le processus de diffusion d’informations au sein des graphes pour exploiter plus efficacement la connectivité d’ordre supérieur entre utilisateurs et éléments. La Figure 5.2 illustre le processus d’application des GNN aux informations d’interaction utilisateur-élément.

5.2.1.1 Structure générale

Les architectures de recommandation utilisateur-élément basées sur les graphes sont composées des éléments suivants :

- **Embeddings initiaux** : Les identifiants associés aux utilisateurs et aux éléments dans le graphe sont d'abord transformés en représentations denses à l'aide d'une couche d'embedding. Ces représentations initiales capturent les caractéristiques de base des nœuds.
- **Modèle de graphe** : Un réseau neuronal sur graphes (GNN) est appliqué aux représentations initiales des nœuds pour affiner ces représentations. Le modèle de graphe utilise la structure du graphe pour agréger et propager l'information entre les nœuds, améliorant ainsi les représentations des utilisateurs et des éléments en fonction de leurs relations et interactions dans le graphe.
- **Tête de tâche** : Cette composante prend les représentations des nœuds apprises (utilisateurs et éléments) produites par le modèle de graphe et les utilise pour effectuer des tâches spécifiques, telles que la prédiction de notation ou la recommandation. Une fonction de perte appropriée (comme la perte de classement BPR) est utilisée pour entraîner le modèle en fonction des objectifs de la tâche.

Ces trois éléments travaillent ensemble pour créer un système de recommandation efficace basé sur les graphes, où les interactions et relations entre utilisateurs et éléments sont exploitées pour améliorer la qualité des recommandations.

5.2.1.2 Architectures

Par la suite, cette section présentera une liste d'architectures sélectionnées utilisées pour la recommandation utilisateur-élément. Les modèles pour les graphes bipartis incluent : GC-MC, SpectralCF, STAR-GCN, NGCF, LightGCN.

Graph Convolutional Matrix Completion (GC-MC)

Graph Convolutional Matrix Completion (GC-MC) [29] est l'un des premiers modèles pour la tâche de recommandation ayant utilisé des réseaux neuronaux sur graphes. L'idée de ce travail était de résoudre la tâche de complétion de matrice à partir des caractéristiques extraites via un module d'auto-encodeur de graphe basé sur le passage de messages différentiable sur un graphe biparti d'interaction utilisateur-élément. Comme mentionné précédemment, la matrice d'interaction est transformée en un graphe biparti, sur lequel les embeddings des noeuds sont appris via un auto-encodeur de graphe. Ces embeddings des noeuds, pour les éléments et les utilisateurs, sont ensuite utilisés pour résoudre une tâche de prédiction de liens, entraînable de manière end-to-end.

En pratique, les embeddings initiaux des noeuds sont utilisés par GC-MC via un décodeur bilinéaire, où Q_r est une matrice entraînable pour chaque valeur de classement (c'est-à-dire de 1 à 5) de forme $E \times E$ et E est la dimensionnalité des embeddings des noeuds utilisateur et élément, pour reconstruire la matrice d'interaction M_{ij} , dans laquelle les probabilités d'un certain classement sont données par :

La probabilité prédictive pour un classement r dans la matrice d'interaction

\hat{M}_{ij} est donnée par :

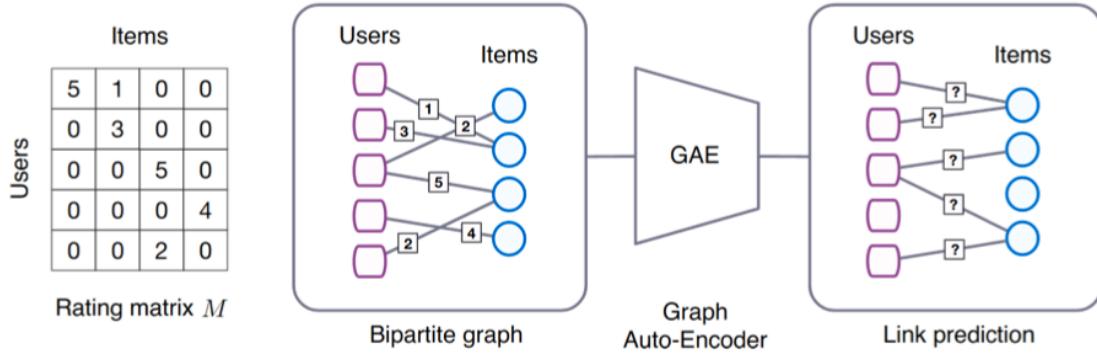
$$P(\hat{M}_{ij} = r) = \frac{\exp(u_i^\top Q_r v_j)}{\sum_{s \in R} \exp(u_i^\top Q_s v_j)} \quad (5.1)$$

La vraisemblance négative des classements prédits \hat{M}_{ij} est minimisée aux positions des classements réels :

$$\mathcal{L} = - \sum_{i,j; \Omega_{i,j}=1} \sum_{r=1}^R I[M_{ij} = r] \log P(\hat{M}_{ij} = r) \quad (5.2)$$

où Ω est une matrice binaire servant de masque pour les classements observés, avec des uns aux indices i, j des classements observés et des zéros aux indices des classements non observés.

Un schéma récapitulatif pour l'architecture GC-MC est montré dans la Figure 5.3.



"Source : Graph Convolutional Matrix Completion"

FIGURE 5.3 – La matrice de notations M des interactions utilisateur-éléments est représentée comme un graphe biparti, où les arêtes reflètent les préférences des utilisateurs. Un module d'auto-encodeur graphique est utilisé pour apprendre les représentations des noeuds, permettant la reconstruction des arêtes non observées. Ainsi, le problème est reformulé en une tâche de prédiction de liens à l'aide d'un auto-encodeur graphique entièrement entraînable de bout en bout.

SpectralCF

SpectralCF [30] est un autre travail précoce qui utilise l'opération de convolution spectrale pour apprendre directement les facteurs latents des utilisateurs et des éléments à partir du domaine spectral.

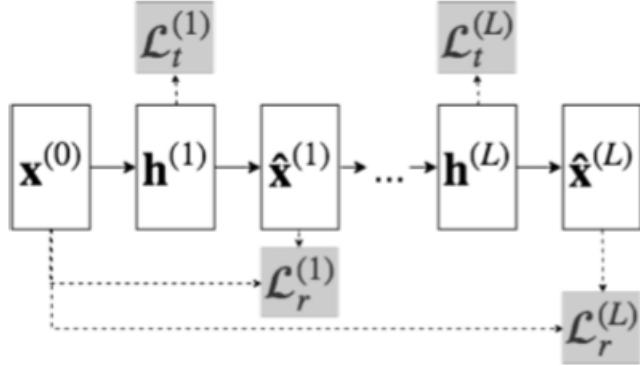
Étant donné X_u et X_i , les représentations des utilisateurs et des éléments respectivement, une opération de convolution spectrale pour produire de nouvelles représentations pourrait être définie comme suit :

$$\begin{bmatrix} X_u^K \\ X_i^K \end{bmatrix} = \sigma \left((UU^\top + U\Lambda U^\top) \begin{bmatrix} X_u^{K-1} \\ X_i^{K-1} \end{bmatrix} \Theta_{K-1} \right)$$

où Θ est une matrice de paramètres de l'opération, U est une matrice des vecteurs propres et Λ est un vecteur des valeurs propres de la matrice de Laplacien de graphe, respectivement. Un bloc de telles convolutions spectrales serait empilé pour former un réseau plus profond.

STAR-GCN

STAR-GCN [8] est une architecture multi-blocs utilisant une pile de L modules GCN encodeur-décodeur, comme montré à la figure 5.4. L'encodeur crée de nouvelles représentations latentes des noeuds en intégrant les informations du voisinage avec les caractéristiques des noeuds d'entrée. Le décodeur, quant à lui, récupère les embeddings des noeuds d'entrée à partir des représentations latentes fournies par l'encodeur. Les représentations générées par l'encodeur sont optimisées avec une perte spécifique à la tâche, tandis que les reconstructions du décodeur sont optimisées avec une perte de reconstruction.



“Source : STAR-GCN : Stacked and Reconstructed Graph Convolutional Networks for Recommender Systems”

FIGURE 5.4 – Résumé de l’architecture du modèle STAR-GCN.

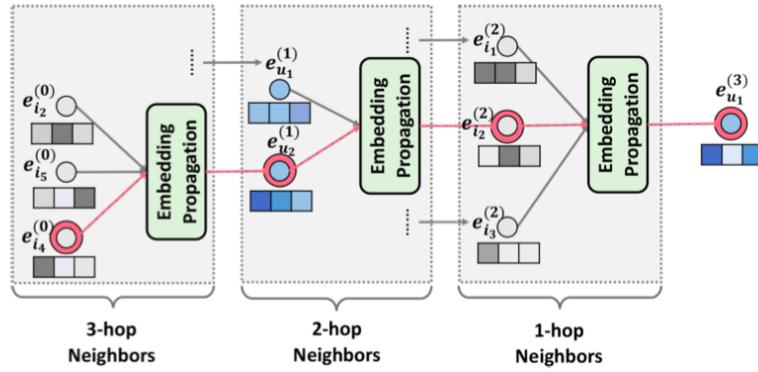
Pour résoudre le problème du démarrage à froid et créer des embeddings généralisables à de nouveaux noeuds au-delà du jeu d’entraînement, les auteurs proposent d’utiliser une table d’embeddings initiale où un pourcentage (par exemple, 20%) des embeddings d’entrée est masqué aléatoirement en les fixant à zéro. L’entraînement apprend alors à reconstruire ces embeddings masqués à partir des informations du voisinage. En phase de test, les embeddings des noeuds inconnus sont initialisés à zéro et affinés itérativement par les L modules du réseau.

Les auteurs ont également découvert que les modèles basés sur GCN pouvaient révéler des étiquettes pendant l’entraînement, car l’agrégation des voisins est effectuée sur un graphe biparti où les évaluations utilisateur-élément (connues à partir des données d’entraînement) sont utilisées pour construire les liens, introduisant ainsi les étiquettes dans la structure du graphe et faisant en sorte que le modèle se comporte comme $r = f_\theta(x, r)$ au lieu de $r = f_\theta(x)$.

Pour éviter ce problème, les auteurs proposent une stratégie d’entraînement *sample-and-remove*, où une partie fixe des liens est retirée du graphe d’entraînement lors de l’échantillonnage, avant l’entraînement du modèle.

NGCF

Le Neural Graph Collaborative Filtering (NGCF) [9] peut être considéré comme une généralisation des algorithmes de filtrage collaboratif typiques, basés sur une couche d'embedding et la modélisation des interactions, tels que SVD++. Il utilise une architecture similaire à GC-MC, avec une profondeur de 3 pour le modèle de graphe, permettant d'agréger des informations sur un voisinage à 3 sauts, également connu sous le nom de propagation d'ordre trois (voir Figure 5.5 pour une meilleure illustration).



“Source : “Neural Graph Collaborative Filtering”

FIGURE 5.5 – Illustration de la propagation d'un embedding de troisième ordre pour un utilisateur u_1

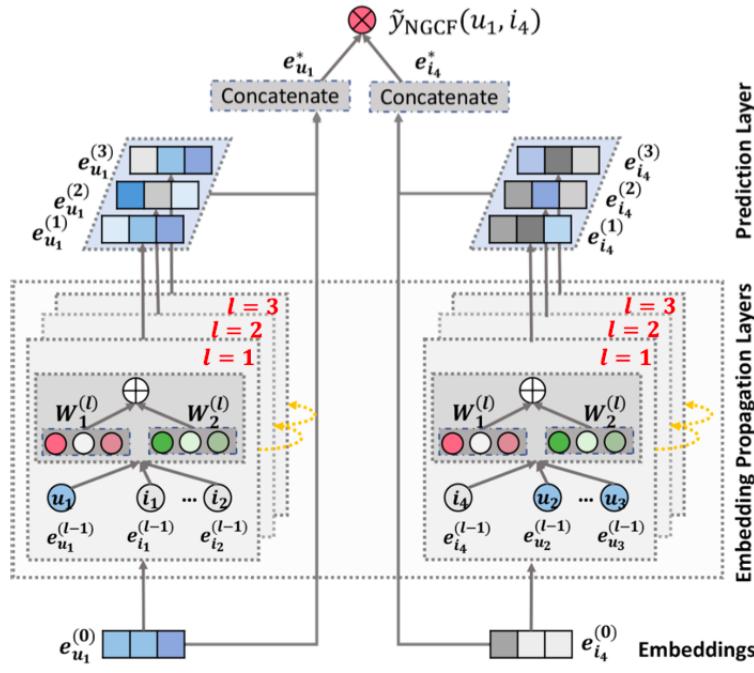
Les embeddings des utilisateurs et des articles sont affinés entre les blocs de saut par un produit élémentaire entre les embeddings des articles et des utilisateurs dans les messages :

$$m_{u \rightarrow i} = \frac{1}{\sqrt{|\mathcal{N}_u|} \sqrt{|\mathcal{N}_i|}} (W_1 e_i + W_2 (e_i \odot e_u))$$

cela rend le message dépendant de l'affinité entre e_i et e_u , augmentant

ainsi l'efficacité du passage de messages entre articles similaires.

Une pile de trois blocs de ce type est utilisée, et les embeddings finaux de chaque bloc sont concaténés en un seul embedding avant d'être transmis à la tête de tâche, comme illustré dans le résumé de la Figure 5.6.



"Source : "Neural Graph Collaborative Filtering"

FIGURE 5.6 – Résumé de l'architecture du modèle NGCF. Les représentations initiales $e_{u_1}^{(0)}$ and $e_{i_4}^{(0)}$ sont affinées via plusieurs couches de propagation d'embeddings, correspondant à une propagation sur un voisinage de 3 sauts. Les embeddings obtenus sont ensuite concaténés et transmis à la tête de tâche.

LightGCN

L'architecture de LightGCN [14] repose sur la simplification du design du GCN, un composant clé du filtrage collaboratif. Les auteurs simplifient l'opération de convolution sur le graphe en éliminant la fonction d'activation non linéaire, réduisant ainsi le modèle à un seul ensemble de paramètres,

tout en conservant l'agrégation de voisinage, ce qui aboutit à la formule suivante pour la mise à jour des embeddings :

$$e_u^K = \sum_{i \in \mathcal{N}_u} \frac{1}{\sqrt{|\mathcal{N}_u|} \sqrt{|\mathcal{N}_i|}} e_i^{K-1}$$

$$e_i^K = \sum_{u \in \mathcal{N}_i} \frac{1}{\sqrt{|\mathcal{N}_u|} \sqrt{|\mathcal{N}_i|}} e_u^{K-1}$$

où les seuls paramètres entraînables du modèle sont les embeddings initiaux e_i^0 pour tous les utilisateurs u et e_u^0 pour tous les items i .

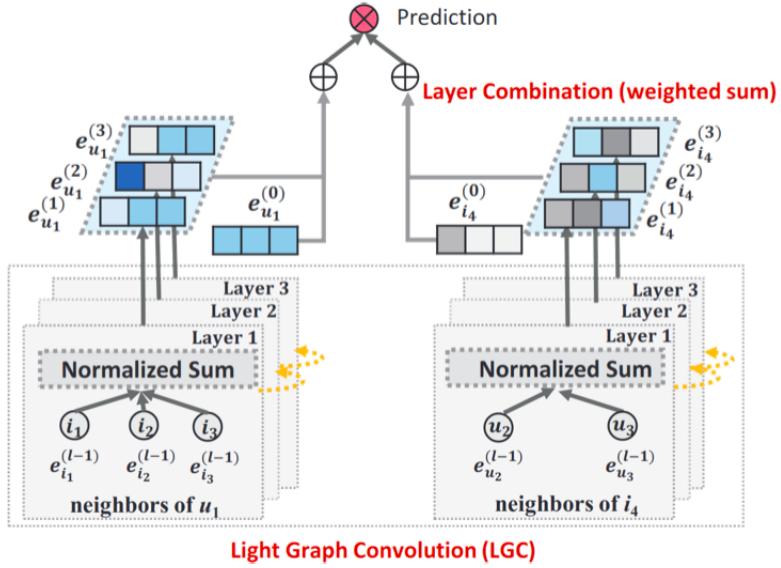
Il est important de noter que LightGCN n'agrège que les noeuds voisins, sans intégrer les informations du noeud lui-même. Ce choix a été fait parce que les auto-connections nécessitent généralement un traitement particulier lors de l'agrégation. De plus, les auteurs démontrent que leur stratégie de concaténation des caractéristiques avant la tête de tâche a le même effet que l'ajout d'auto-connections.

Avant d'être transmises à la tête de tâche, les caractéristiques provenant des différentes couches, agissant comme un simple agrégateur de voisinage, sont combinées en une combinaison linéaire :

$$e_u = \sum_{k=0}^K \alpha_k e_u^k$$

$$e_i = \sum_{k=0}^K \alpha_k e_i^k$$

où α_k est un paramètre spécifique à chaque couche, que les auteurs fixent à $1/(K + 1)$ pour simplifier. Un résumé de l'architecture est présenté à la Figure 5.7.



"Source : LightGCN : Simplifying and Powering Graph Convolution Network for Recommendation"

FIGURE 5.7 – Résumé de l'architecture du modèle LightGCN.

En résumé, les GNN représentent une avancée significative dans le domaine des systèmes de recommandation. Leur capacité à exploiter les interactions complexes entre utilisateurs et éléments permet de mieux capturer les relations sous-jacentes, surpassant les approches traditionnelles de filtrage collaboratif et de factorisation de matrices. Avec des architectures spécifiques adaptées aux graphes, ces modèles offrent une approche puissante et flexible. Passons maintenant à la phase de Simulations et Résultats, où nous mettrons en œuvre ces modèles dans un cadre expérimental afin d'évaluer leurs performances sur des jeux de données concrets.

Simulations et Résultats

Dans ce chapitre, nous allons explorer et comparer les performances de trois approches différentes pour la recommandation de films, en utilisant le célèbre dataset MovieLens 100k. Notre but est de démontrer comment un modèle de réseaux de neurones sur les graphes, conçu spécifiquement pour exploiter les relations complexes entre utilisateurs et films, se mesure face à des techniques plus classiques et bien établies dans le domaine des systèmes de recommandation.

6.1 Objectifs de la Simulation

Notre objectif dans ce chapitre est de démontrer comment un modèle de réseaux de neurones sur les graphes (GNN), spécialement conçu pour exploiter les relations complexes entre utilisateurs et films, se compare à des techniques plus traditionnelles et bien établies dans le domaine des systèmes de recommandation. Pour ce faire, nous décrivons en détail les différentes approches que nous avons mises en œuvre pour modéliser les préférences des utilisateurs et prédire leurs évaluations de films qu'ils n'ont pas encore vus. Les méthodes explorées incluent une approche basée sur les graphes, une méthode collaborative basée sur la mémoire (Memory-based Collaborative Filtering), et une technique de factorisation de matrice (Matrix Factorization).

Afin de déterminer le modèle le plus efficace pour la tâche de recommandation, nous comparons ensuite ces approches en termes de précision des

prédictions, mesurée par l'erreur quadratique moyenne (RMSE).

Nous commencerons par une explication du choix de ces modèles, en mettant en lumière les forces et les faiblesses potentielles de chacun dans le contexte des systèmes de recommandation. Ensuite, nous détaillerons les étapes de préparation des données, y compris le prétraitement et la division des données en ensembles d'entraînement et de test, pour garantir une évaluation rigoureuse des performances des modèles.

Chaque modèle sera ensuite présenté avec une description de son implémentation, y compris les choix algorithmiques, les hyperparamètres, et les techniques spécifiques utilisées pour optimiser leur performance. Nous discuterons ensuite des résultats obtenus, mesurés en termes de Root Mean Squared Error (RMSE), une métrique couramment utilisée pour évaluer la qualité des prédictions dans les systèmes de recommandation.

Enfin, nous comparerons les performances des différents modèles et discuterons des résultats obtenus. Cette comparaison nous permettra de mieux comprendre les capacités de chaque approche et de tirer des conclusions sur leur applicabilité dans des contextes réels de recommandation.

Ce chapitre vise non seulement à présenter les résultats de nos simulations, mais aussi à offrir une analyse critique des méthodes utilisées, afin de fournir des recommandations pour des recherches futures et des implementations pratiques.

6.2 Configuration des Modèles

6.2.1 Modèle GNN

6.2.1.1 Construction du Dataset et Représentation

Pour la représentation du dataset, nous utiliserons un graphe biparti, un graphe pondéré non orienté composé de deux types différents de noeuds (utilisateurs et films). Dans ce graphe, des arêtes pondérées relient les noeuds de type utilisateur aux noeuds de type film. Une arête entre un utilisateur et un film signifie que l'utilisateur a vu et évalué le film. La note que l'utilisateur a attribuée au film est encodée en tant que poids de l'arête reliant les deux entités. Aucune connexion de type utilisateur-utilisateur ou film-film ne sera prise en compte dans cette implémentation.

La tâche à résoudre consiste à prédire, en supposant qu'une arête existe entre deux noeuds, le poids de cette arête. Par conséquent, notre objectif est de construire un modèle qui prédit les poids des liens cibles. Le modèle apprendra les relations à partir des notes attribuées par les utilisateurs aux films en utilisant à la fois les caractéristiques initiales des noeuds et un ensemble d'arêtes dont nous connaissons déjà les poids.

La première étape consiste à représenter l'information sous forme d'un graphe biparti, composé de deux types de noeuds : les utilisateurs et les films. Chaque type de noeud sera associé à des informations différentes (vecteurs de caractéristiques), mais tous les noeuds d'un même type partageront les mêmes données.

- **Nœuds de type film :** Le genre et le titre constituent l'encodage des vecteurs de caractéristiques initiaux pour les noeuds de type film. Les genres seront représentés par des vecteurs encodés en one-hot, et le titre

sera converti en vecteur par un transformeur pré-entraîné. Le vecteur de caractéristiques final d'un film sera la concaténation de ses genres encodés en one-hot et de l'embedding de la phrase correspondant à son titre.

- **Nœuds de type utilisateur :** Par défaut, nous ne considérerons aucune information sur les utilisateurs, hormis un identifiant numérique. Le vecteur de caractéristiques initial d'un utilisateur est un encodage one-hot de ce numéro.

Après avoir défini les nœuds et leurs vecteurs de caractéristiques initiaux, il faut déterminer les liens entre les nœuds. Un lien existera entre un utilisateur et un film si l'utilisateur a regardé et évalué le film. L'évaluation sera reflétée par le poids de l'arête. La figure 6.1 illustre l'ensemble des aspects abordés.

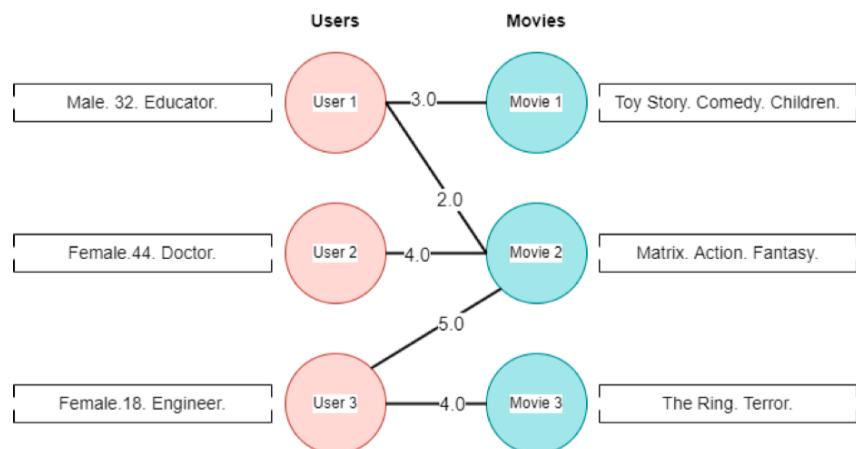


FIGURE 6.1 – Graphe biparti présentant les informations disponibles

Les utilisateurs et les films sont reliés par des liens, et chaque nœud possède un vecteur de caractéristiques initiales qui lui est associé. Il est important de noter que la figure 6.1 présente les vecteurs de caractéristiques

initiales en langage naturel pour en faciliter la compréhension, mais en réalité, il s'agirait d'un encodage vectoriel représentant ces informations.

Etapes de l'entraînement du modèle

Un ensemble des arêtes de ce graphe sera utilisé pour entraîner le modèle, un autre pour la validation, et un dernier pour les tests. Les étapes pour entraîner le modèle sont :

- (a) Définir les vecteurs de caractéristiques initiales des noeuds.
- (b) Diviser les arêtes en ensembles d'entraînement, de validation et de test, puis masquer leurs poids respectifs.
- (c) Entraîner le modèle. Ce processus peut être subdivisé en plusieurs étapes :
 - Le modèle reçoit les vecteurs de caractéristiques initiales des noeuds ainsi que les arêtes d'entraînement avec leurs poids masqués.
 - Le système propage les vecteurs de caractéristiques des noeuds un certain nombre de fois pour générer des embeddings de noeuds.
 - Calculer un embedding pour chaque arête en concaténant les embeddings de l'utilisateur et du film associés.
 - Transformer l'embedding de l'arête en un nombre via un MLP. Ce nombre représentera le poids prédict pour l'arête.
 - Calculer la MSELoss des poids prédicts par rapport aux poids réels et effectuer la rétropropagation. *Ce processus sera répété un certain nombre de fois (époques). À la fin de chaque époque, le RMSE sera calculé avec les arêtes de validation. Si cette métrique ne s'améliore pas pendant 20 époques, le taux d'apprentissage sera réduit pour rechercher un minimum local.*
- (d) Évaluer le système. Les liens de test seront utilisés pour évaluer le modèle avec le RMSE.

6.2.1.2 Architecture du Modèle

Le modèle de réseaux de neurones sur les graphes (GNN) que nous avons développé repose sur l'utilisation de la couche SAGEConv¹, inspirée de l'algorithme GraphSAGE. Cette architecture est spécifiquement conçue pour capturer et exploiter les relations complexes entre utilisateurs et films au sein d'un graphe biparti. L'architecture repose sur plusieurs étapes clés pour transformer les données d'entrée en prédictions d'évaluations. L'architecture du modèle est montrée à la figure 6.2.

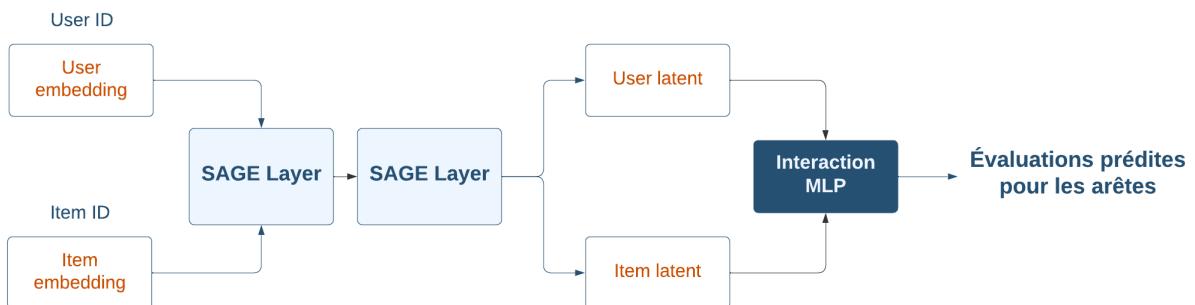


FIGURE 6.2 – Architecture du modèle proposé basé sur GraphSAGE, où les embeddings d'utilisateurs et d'items sont raffinés à travers plusieurs couches SAGE avant d'être combinés via un réseau de neurones multi-couches (MLP) pour prédire les évaluations des interactions.

Convolution sur les Graphes

Le modèle commence par appliquer deux couches successives de convolution sur les graphes en utilisant la couche SAGEConv de PyTorch Geometric.

La première couche prend en entrée les caractéristiques des noeuds, agrège ces caractéristiques avec celles de leurs voisins directs, et les transforme en

1. C'est une implémentation de la brique de base de GraphSAGE, c'est-à-dire la convolution par agrégation des caractéristiques des voisins. Pour plus d'informations, consultez le site officiel de PyTorch Geometric : <https://pytorch-geometric.readthedocs.io/>

un espace de dimension intermédiaire. Ce processus est crucial pour capturer les relations de premier ordre entre les noeuds, tels que les liens directs entre un utilisateur et les films qu'il a notés.

L'opération effectuée par la première couche de convolution peut être formulée ainsi :

$$\mathbf{h}_v^{(1)} = \text{ReLU} \left(\mathbf{W}^{(1)} \cdot \text{Aggregate} \left(\left\{ \mathbf{h}_u^{(0)} : u \in \mathcal{N}(v) \right\} \right) \right)$$

où :

- $\mathbf{h}_v^{(1)}$ est le vecteur de caractéristiques du noeud v après la première couche,
- $\mathbf{W}^{(1)}$ est la matrice de poids de la première couche,
- Aggregate représente l'agrégation par moyenne des caractéristiques des voisins,
- ReLU est la fonction d'activation rectifiée linéaire appliquée après l'agrégation.

La deuxième couche de **SAGEConv** affine davantage ces représentations. Elle suit une opération similaire mais applique la transformation sur les caractéristiques déjà transformées par la première couche. Cependant, contrairement à la première couche, aucune activation ReLU n'est appliquée après la deuxième couche de convolution :

$$\mathbf{h}_v^{(2)} = \mathbf{W}^{(2)} \cdot \text{Aggregate} \left(\left\{ \mathbf{h}_u^{(1)} : u \in \mathcal{N}(v) \right\} \right)$$

Ce choix permet de conserver une transformation linéaire, ce qui est souvent souhaité pour les dernières étapes de préparation des embeddings avant la prédiction. En ne forçant pas les valeurs négatives à zéro, on évite de perdre des informations potentiellement importantes, et on facilite la combinaison linéaire des embeddings des noeuds dans les étapes ultérieures.

Prédiction des évaluations

Une fois que les représentations finales des nœuds (utilisateurs et films) sont générées, le modèle utilise ces représentations pour prédire les interactions entre les nœuds, en particulier les évaluations que les utilisateurs pourraient attribuer aux films. Pour chaque paire de nœuds (un utilisateur u et un film m) connectés par une arête, les vecteurs de caractéristiques des nœuds correspondants sont concaténés :

$$\mathbf{z}_{(u,m)} = \text{Concat}(\mathbf{h}_u^{(2)}, \mathbf{h}_m^{(2)})$$

Cette concaténation produit un vecteur qui combine les informations des deux nœuds impliqués dans l'interaction.

Le vecteur concaténé est ensuite passé à travers un réseau de neurones entièrement connecté pour produire la prédiction finale. La première couche de ce réseau applique une transformation linéaire suivie d'une activation ReLU :

$$\mathbf{z}' = \text{ReLU}(\mathbf{W}_1 \mathbf{z}_{(u,m)})$$

Ensuite, la sortie est transformée par une deuxième couche linéaire pour produire une seule valeur correspondant à la prédiction de l'évaluation :

$$\hat{y}_{(u,m)} = \mathbf{W}_2 \mathbf{z}'$$

où :

- \mathbf{W}_1 et \mathbf{W}_2 sont des matrices de poids apprises,
- $\hat{y}_{(u,m)}$ est la prédiction de l'évaluation de l'utilisateur u pour le film m .

Modèle Hétérogène

Le modèle est configuré pour fonctionner sur des graphes hétérogènes, c'est-à-dire des graphes avec différents types de noeuds (utilisateurs et films) et de relations (évaluations). L'utilisation de `to_hetero` permet d'adapter les opérations de convolution pour ces types de noeuds diversifiés, en veillant à ce que l'agrégation et la transformation soient correctement appliquées en fonction du type de chaque noeud.

Ce modèle GNN applique deux couches de convolution SAGEConv pour apprendre les représentations des noeuds en agrégant les caractéristiques de leurs voisins, suivies par des transformations linéaires pour prédire les interactions entre utilisateurs et films. Le choix de ne pas appliquer une activation après la deuxième couche permet de maintenir une transformation linéaire, évitant ainsi de perdre des informations cruciales pour la prédiction, tout en facilitant la combinaison des embeddings des noeuds dans un contexte de graphe hétérogène.

6.2.1.3 Entrainement et Évaluation du Modèle

L'entraînement se déroule sur 200 époques. À chaque époque, le modèle passe par trois étapes principales :

Entrainement

Le modèle est mis en mode d'entraînement, où les gradients sont calculés et les poids du modèle sont ajustés pour minimiser l'erreur quadratique moyenne (MSE) entre les prédictions et les cibles réelles.

$$\text{loss} = \frac{1}{N} \sum_{i=1}^N (\hat{y}_i - y_i)^2$$

Évaluation sur l'ensemble de validation

Après l'entraînement, le modèle est évalué sur l'ensemble de validation pour suivre l'évolution de la performance. Les prédictions sont arrondies et bornées entre 1 et 5 pour correspondre à l'échelle des évaluations réelles. Le scheduler ajuste le taux d'apprentissage si la perte sur l'ensemble de validation ne s'améliore pas.

Évaluation sur l'ensemble de test

Enfin, les performances du modèle sont également suivies sur l'ensemble de test pour vérifier que le modèle généralise bien aux données non vues. Le RMSE (Root Mean Squared Error) est calculé pour quantifier la performance du modèle.

Les meilleures performances du modèle sont sauvegardées, permettant de récupérer les poids optimaux à la fin de l'entraînement.

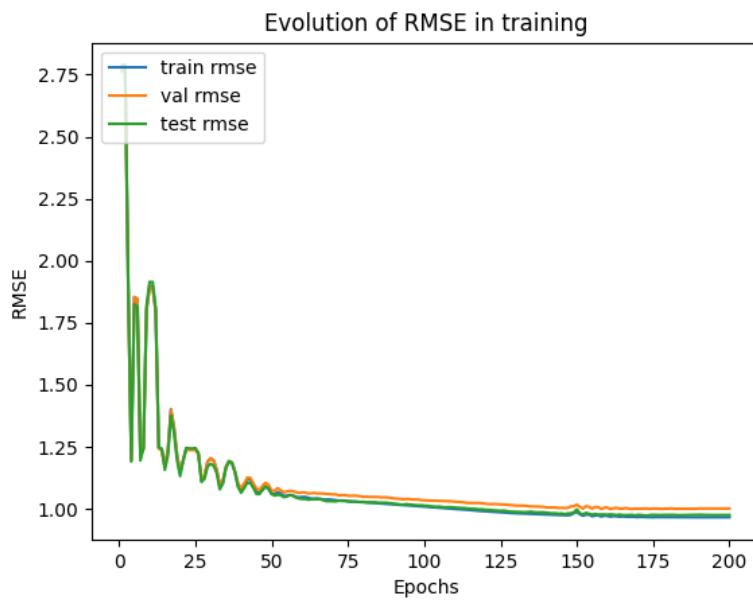


FIGURE 6.3 – L'évolution de RMSE pendant la phase d'entraînement du modèle GNN.

Résultats et Visualisation

Au cours des 200 époques, les métriques RMSE sont enregistrées pour les ensembles d'entraînement, de validation, et de test. À la fin de l'entraînement, ces résultats sont visualisés pour observer l'évolution des performances du modèle (voir la figure 6.3). Cette visualisation aide à évaluer la stabilité de l'entraînement et à identifier les époques où le modèle atteint ses meilleures performances. Voici une analyse de la visualisation :

- **Début de l'entraînement (0-25 epochs)** : Pendant les premières itérations, on observe une baisse rapide du RMSE, indiquant que le modèle apprend rapidement dans cette phase initiale. Quelques fluctuations sont présentes, ce qui est typique lors de l'ajustement des paramètres dans les premières epochs.
- **Stabilisation (après 25 epochs)** : Phase de stabilisation progressive (25-175 epochs) : Après la 25e epoch, les courbes commencent à se stabiliser autour d'une valeur proche de 1. Cependant, le modèle continue de s'améliorer légèrement jusqu'à environ l'epoch 175, où l'erreur devient pratiquement constante.
- **Convergence des trois courbes** : Les courbes d'entraînement, de validation, et de test restent proches les unes des autres tout au long du processus, ce qui est un bon indicateur que le modèle généralise bien sur les nouvelles données. Le fait que les courbes ne divergent pas suggère également que le modèle n'est pas en surapprentissage (overfitting).

Le modèle semble bien s'entraîner et généralise correctement sur les données de validation et de test. La stabilisation des courbes RMSE autour de 1 signifie que le modèle atteint un niveau d'erreur acceptable. Le fait que les courbes ne divergent pas et restent proches montre que le modèle a été bien régularisé et qu'il n'y a pas de signes évidents de surapprentissage.

6.2.2 Factorisation de Matrice

Pour un recap, la factorisation de matrice est une méthode puissante pour réduire la dimensionnalité du problème de recommandation. Elle décompose la matrice des évaluations en deux matrices de rang inférieur, capturant les interactions latentes entre utilisateurs et items. Cette méthode est particulièrement efficace pour gérer les données sparsées, où peu d'utilisateurs ont évalué un grand nombre de films (voir le chapitre 2).

Implémentation

L'implémentation de la factorisation de matrice repose sur l'utilisation de la décomposition en valeurs singulières tronquée (Truncated SVD) pour réduire la dimensionnalité de la matrice utilisateur-film. Le processus commence par le chargement des données de notation des films, qui sont ensuite pivotées pour former une matrice utilisateur-film, où chaque ligne représente un utilisateur et chaque colonne un film. Les valeurs manquantes dans cette matrice sont remplacées par des zéros pour simplifier les calculs.

La décomposition en valeurs singulières est ensuite appliquée à cette matrice pour obtenir deux matrices de rang réduit : une matrice utilisateur et une matrice film. Ces matrices, qui capturent les facteurs latents représentant les utilisateurs et les films, sont ensuite multipliées pour prédire les évaluations manquantes dans la matrice originale.

Pour évaluer la qualité des prédictions, l'erreur quadratique moyenne (RMSE) est calculée en comparant les évaluations prédictes aux évaluations réelles. Cette métrique permet de quantifier la précision du modèle.

6.2.3 Filtrage Collaboratif Basé sur la Mémoire

Principe du Filtrage Collaboratif Basé sur la Mémoire

Pour un recap, le filtrage collaboratif basé sur la mémoire(item-based) est une technique de recommandation qui exploite les similarités entre items (films) pour prédire les évaluations des utilisateurs. Contrairement aux modèles basés sur les facteurs latents, cette approche utilise directement les données d'évaluations pour calculer des similarités entre les items ou les utilisateurs. En se basant sur l'hypothèse que des items similaires seront notés de manière similaire par les utilisateurs, le filtrage collaboratif prédit les évaluations manquantes en analysant les interactions passées.

Deux méthodes courantes de calcul des similarités sont utilisées : la similarité cosinus et la similarité de Pearson. La similarité cosinus mesure l'angle entre les vecteurs d'évaluations, tandis que la similarité de Pearson évalue la corrélation linéaire entre les évaluations de deux items. Ces similarités sont ensuite utilisées pour pondérer les évaluations existantes et estimer celles qui manquent.

Implémentation

Le filtrage collaboratif basé sur la mémoire utilise les similarités entre les films pour prédire les évaluations des utilisateurs. Tout d'abord, les données d'évaluations sont chargées et converties en une matrice utilisateur-item, qui est ensuite normalisée en soustrayant la moyenne des évaluations de chaque utilisateur. Deux types de similarités entre les items sont calculés : la similarité cosinus et la similarité de Pearson. Ces similarités permettent de déterminer à quel point les films sont similaires entre eux en termes de préférences des utilisateurs.

Les prédictions d'évaluations sont ensuite générées en multipliant la matrice des évaluations normalisées par la matrice de similarité des items, et en

réintroduisant la moyenne des évaluations pour chaque utilisateur. Enfin, la précision des prédictions est évaluée en calculant l'erreur quadratique moyenne (RMSE) pour les deux méthodes de similarité, fournissant une mesure de la performance du modèle.

6.3 Résultats Comparés

Les performances des trois modèles — GNN, factorisation de matrice (MF), et filtrage collaboratif basé sur la mémoire (CF) — ont été évaluées à l'aide de la Root Mean Squared Error (RMSE). Cette métrique, qui quantifie l'écart entre les prédictions du modèle et les évaluations réelles, nous permet de comparer directement l'efficacité de chaque approche dans le contexte d'un système de recommandation. Voici le tableau qui représente les résultats de ces trois modèles :

Modèle	Train RMSE	Val RMSE	Test RMSE
GNN	0.9759	0.9883	0.9938
MF	-	-	2.1323
CF (Cosinus)	-	-	3.0027
CF (Pearson)	-	-	3.0139

FIGURE 6.4 – Comparaison de la performance des trois modèles avec RMSE

Le modèle de réseau de neurones sur les graphes (GNN) a montré une supériorité notable avec un RMSE de 0.9759 sur l'ensemble d'entraînement, 0.9883 sur l'ensemble de validation, et 0.9938 sur l'ensemble de test. Ces résultats indiquent que le GNN est capable de capturer les relations complexes entre utilisateurs et films, offrant des prédictions extrêmement précises.

En comparaison, la factorisation de matrice (MF) a obtenu un RMSE de 2.1323, ce qui est significativement plus élevé. Bien que la MF soit effi-

cace pour découvrir les interactions latentes entre utilisateurs et films, elle semble moins performante que le GNN dans ce cadre, probablement en raison de sa capacité limitée à modéliser des relations plus complexes.

Enfin, le filtrage collaboratif basé sur la mémoire (CF) a affiché des RMSE de 3.0027 pour la similarité cosinus et de 3.0139 pour la similarité de Pearson. Ces résultats sont les moins précis des trois approches, ce qui peut s'expliquer par la nature simplifiée de la méthode, qui repose directement sur les similarités entre items sans capturer les interactions latentes plus profondes.

6.4 Analyse et Discussion

La performance supérieure du GNN par rapport aux autres modèles souligne l'importance de capturer les relations complexes et les structures de graphe dans les données de recommandation. Le GNN a non seulement surpassé la MF, mais il a également démontré une capacité à généraliser efficacement aux données non vues, comme en témoigne le faible écart entre les RMSE sur les ensembles de validation et de test.

La MF, bien que performante, montre ses limites lorsque les interactions utilisateurs-films sont plus compliquées que ce que permet une simple décomposition matricielle. En revanche, le CF, malgré sa simplicité et son efficacité dans des contextes moins complexes, a peiné à fournir des prédictions précises, particulièrement lorsque les données sont sparsées ou lorsque les similarités ne sont pas suffisamment représentatives.

Ces résultats suggèrent que, pour des systèmes de recommandation sophistiqués où les interactions entre utilisateurs et items sont nombreuses et

variées, le GNN est la meilleure approche. La MF peut être une solution de repli efficace dans des situations où la complexité du modèle doit être réduite, tandis que le CF est plus adapté à des scénarios simples où les données sont abondantes et bien structurées.

Application Web de Recommandation avec GNN

7.1 Introduction

Suite à la simulation de Réseaux de Neurones Graphiques (GNN) présentée dans le chapitre précédent, l'étape suivante a consisté à rendre ces résultats accessibles et utilisables via une application web. L'objectif de cette application est de permettre aux utilisateurs d'obtenir des recommandations personnalisées, en exploitant les prédictions générées par le modèle GNN. Pour ce faire, nous avons développé une interface web en utilisant **Flask**¹, un framework léger pour Python, qui permet d'intégrer directement les fonctionnalités du modèle de recommandation.

Ce chapitre se concentre sur la mise en œuvre de cette application web, en détaillant les technologies utilisées, son fonctionnement, ainsi que des exemples concrets d'utilisation.

7.2 Technologies Utilisées

Pour développer cette application web, nous avons utilisé les outils suivants :

1. Flask est un cadre d'application web WSGI léger conçu pour faciliter le démarrage rapide et simple d'un projet, tout en offrant la possibilité de s'adapter à des applications plus complexes. Pour plus d'informations, visitez : <https://flask.palletsprojects.com/en/3.0.x/>

- **Flask** : Un micro-framework en Python pour construire le serveur web et gérer les requêtes des utilisateurs.
- **PyTorch Geometric** : Utilisé pour charger le modèle GNN pré-entraîné et faire les prédictions, (*voir le chapitre 6*).
- **HTML²** : Pour créer l’interface utilisateur simple et permettre aux utilisateurs de soumettre leurs requêtes et d’afficher les résultats.

Ces technologies permettent de lier facilement la partie backend (le modèle GNN) à une interface web simple et accessible.

7.3 Structure de l’Application

L’application web est organisée en plusieurs répertoires et fichiers, chacun ayant un rôle spécifique pour assurer son bon fonctionnement. Voici un aperçu de la structure :

- **app.py** : Le fichier principal qui gère la logique de l’application web, incluant la configuration du serveur Flask, la gestion des requêtes, et l’intégration du modèle GNN pour la recommandation.
- **templates/** : Ce répertoire contient les fichiers HTML utilisés pour rendre les pages web. Flask utilise le moteur de templates **Jinja2³** pour relier les variables et les résultats du backend aux fichiers HTML.
 - **index.html** : La page d’accueil où les utilisateurs peuvent soumettre leur ID ou interagir avec l’application.

2. HTML, ou HyperText Markup Language, est le langage de balisage standard utilisé pour créer des pages web. Pour plus d’informations, visitez : <https://developer.mozilla.org/en-US/docs/Web/HTML>

3. un moteur de templates rapide et flexible pour gérer les fichiers HTML et intégrer les variables ou résultats provenant du backend Python

- `result.html` : Page qui affiche les recommandations pour les nouveaux utilisateurs.
- `existing_user_results.html` : Page qui affiche les recommandations pour les utilisateurs existants ayant déjà des interactions dans le système.
- **static/posters/** : Ce répertoire contient les images des posters des films recommandés, qui sont affichés sur l'interface utilisateur pour enrichir l'expérience visuelle.
- **models/** : Ce répertoire contient les fichiers liés au modèle GNN, incluant le modèle pré-entraîné et les scripts de prédiction. Ce modèle est chargé dans `app.py` pour faire des recommandations en temps réel.
- **data/** : Les fichiers de données nécessaires pour les interactions avec les utilisateurs et les films (données utilisateur, films, interactions, etc.) sont stockés ici.
- **download_posters.py** : Un script Python permettant de télécharger les posters des films recommandés pour les afficher dans l'application web.

Cette structure simple permet de séparer clairement les différentes fonctionnalités de l'application : interface utilisateur (templates), logique backend (Flask dans `app.py`), et gestion des données et modèles.

7.4 Fonctionnement de l'Application

L'application web offre deux principales fonctionnalités pour fournir des recommandations personnalisées, selon que l'utilisateur soit existant ou nouveau.

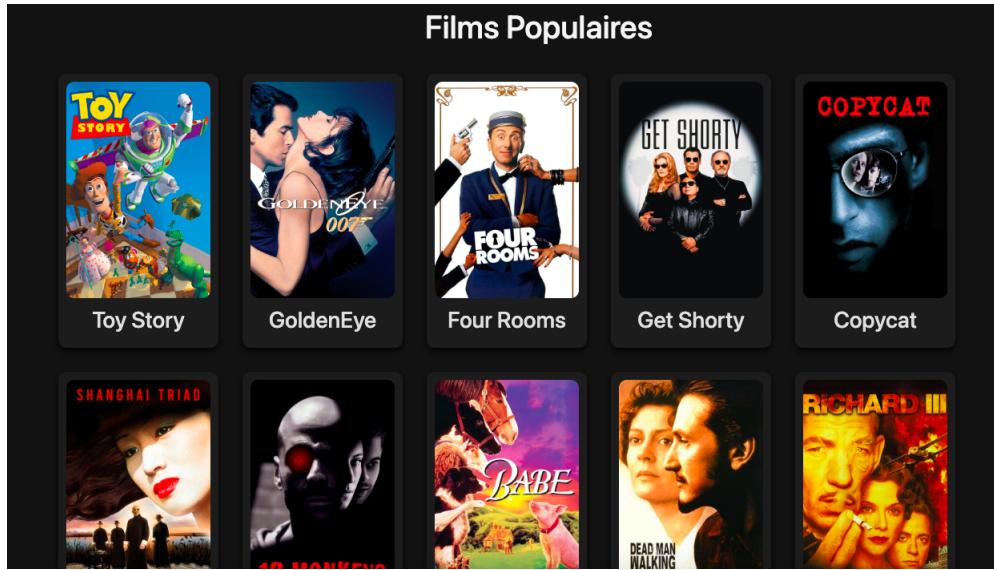


FIGURE 7.1 – La page d'accueil : Début de page

Page d'Accueil

Sur la page d'accueil, les utilisateurs voient d'abord une liste de films populaires, déterminée par les films ayant reçu le plus grand nombre d'évaluations. Ces recommandations visent à capturer les tendances actuelles et les films les plus populaires parmi les utilisateurs du système, voir la figure 7.1.

À la fin de la page, un formulaire interactif est proposé pour déterminer si l'utilisateur est un utilisateur existant, figure 7.3, ou un nouvel utilisateur, figure 7.2.

Utilisateur Existant

Si l'utilisateur a déjà interagi avec l'application, il peut entrer son ID utilisateur dans le champ prévu à cet effet, voir la figure 7.3. Une fois l'ID soumis, l'application accède à l'historique des interactions de cet utilisateur

Aidez-nous à mieux vous connaître !

Êtes-vous un nouvel utilisateur ou un utilisateur existant ?
Nouvel utilisateur

Film 1
Toy Story

Évaluation (1-5)

Film 2
Toy Story

Évaluation (1-5)

Film 3
Toy Story

Évaluation (1-5)

Voir mes recommandations

FIGURE 7.2 – La page d'accueil : Le formulaire interactif du **Nouveau Utilisateur**.

Aidez-nous à mieux vous connaître !

Êtes-vous un nouvel utilisateur ou un utilisateur existant ?
Utilisateur existant

Entrez votre ID utilisateur

Voir mes recommandations

FIGURE 7.3 – La page d'accueil : Le formulaire interactif du **Utilisateur existant**.

(les films déjà évalués, les notes données) et utilise le modèle GNN pour générer des recommandations personnalisées basée sur ses préférences passées.

Nouvel Utilisateur

Si l'utilisateur est nouveau, il doit fournir des informations sur trois films qu'il a déjà regardés, accompagnés des notes qu'il leur a attribuées, voir la figure 7.3. Ces informations sont ensuite utilisées pour initialiser un profil utilisateur et permettre au modèle GNN de générer une recommandation de film, même sans historique complet.

Le processus est fluide et rapide, permettant à tout utilisateur, nouveau ou existant, de recevoir des recommandations de films adaptées à ses goûts.

7.5 Conclusion

L'application web de recommandation que nous avons développée permet aux utilisateurs, qu'ils soient nouveaux ou existants, d'obtenir des recommandations de films personnalisées en fonction de leurs préférences. En utilisant le modèle de Graph Neural Networks (GNN), elle exploite les interactions utilisateur-film pour générer des suggestions pertinentes et adaptées aux goûts de chacun.

Les résultats montrent que cette approche basée sur les GNN est efficace pour proposer des recommandations de qualité, même pour des utilisateurs avec peu ou pas d'historique. De futures améliorations pourraient inclure l'ajout de nouvelles fonctionnalités ou une optimisation des performances pour mieux gérer un plus grand nombre d'utilisateurs.

Confusion

“Toute connaissance commence par les sentiments.”

—LÉONARD DE VINCI

C'est par l'amour de défier l'inconnu, et c'est par le même amour de la connaissance que ce sujet était décidé d'être aborder et qui, d'une part, était peu traité et récemment développé, et d'autre part, était rempli d'ambiguité et nécessitaient une recherche minutieuse, une réflexion approfondie et un travail soigneusement rigoureux. Au fur et à mesure, et avec le soutien et la patience de mon encadrant, l'image devenait plus clair et mon projet commença à tenir la route. Dés lors, j'ai pris de plus en plus conscience de la richesse du sujet, de sa beauté abstraite et de ses liens avec divers domaines scientifiques contemporains.

Conclusion

Les réseaux de neurones graphiques (GNN) pour les systèmes de recommandation nous offrent une nouvelle fenêtre sur la manière dont les relations complexes entre utilisateurs et contenus peuvent être modélisées. Cependant, comme tout domaine de pointe, ils ouvrent la voie à une multitude de questionnements encore non résolus, laissant entrevoir des horizons de recherche inexplorés.

Nous invitons le lecteur à réfléchir sur certaines de ces interrogations fondamentales, qui, telles des conjectures en mathématiques, poussent à l'approfondissement de la réflexion :

- Comment les GNN sauront-ils s'adapter à la vaste complexité des systèmes de recommandation à grande échelle, où les interactions se comptent par millions et échappent à toute modélisation traditionnelle ?
- De quelle manière ces modèles pourront-ils capter la dynamique subtile et changeante des graphes, là où chaque nouvelle interaction redessine les contours d'un réseau en perpétuelle évolution ?
- Quel cheminement théorique permettra aux GNN de surmonter les biais inhérents aux systèmes de recommandation, pour offrir des suggestions plus justes et transparentes, à l'abri des influences populaires ou récurrentes ?

Ces questions, encore à la frontière du savoir, rappellent que chaque avancée dans les systèmes de recommandation basés sur les GNN n'est qu'un jalon sur un chemin plus long. Un chemin où la beauté abstraite des modèles se mêle à la complexité du réel, et où chaque réponse ouvre la porte à de nouvelles énigmes. Il appartient maintenant à la communauté scientifique de s'engager sur cette voie, de déchiffrer ces mystères et de faire émerger les nouvelles architectures qui redéfiniront notre manière d'interagir avec l'information

“On ne va jamais si loin que lorsque l'on ne sait pas où l'on va.”

—CHRISTOPHE COLOMB

Bibliographie

- [1] Yuliana Lavryk, Yurii Kryvenchuk, *Product Recommendation System Using Graph Neural Network*, Lviv Polytechnic National University, Stepana Bandery Street 12, Lviv, 79013, Ukraine.
- [2] Zonghan Wu, Shirui Pan, Member, IEEE, Fengwen Chen, Guodong Long, Chengqi Zhang, Senior Member, IEEE, Philip S. Yu, Fellow, IEEE, *A Comprehensive Survey on Graph Neural Networks*, 2019.
- [3] Chen Gao, Yu Zheng, Nian Li, Yinfeng Li, Yingrong Gin, Jinghua Piao, Yuhan Quan, Jianxin Chang, Depeng Jin, Xiangnan He, Yong Li, *A Survey of Graph Neural Networks for Recommender Systems : Challenges, Methods, and Directions*, Beijing National Research Center for Information Science and Technology (BNRist), Department of Electronic Engineering, Tsinghua University, China and 2 School of Information Science and Technology, University of Science and Technology of China, China.
- [4] Zonghan Wu , Shirui Pan, Fengwen Chen, Guodong Long Chengqi Zhang , and Philip S. Yu, Life Fellow, *A Comprehensive Survey on Graph Neural Networks*, 2021.
- [5] Sergios Karagiannakos, Best Graph Neural Network architectures : GCN, GAT, MPNN and more, <https://theaisummer.com/gnn-architectures/>

- [5] D. Goldberg, D. Nichols, B. M. Oki, and D. Terry. Using collaborative filtering to weave an information tapestry. *Commun. ACM*, 35(12) :6170, December 1992. ISSN : 0001-0782. DOI : 10.1145/138859.138867. URL : <https://doi.org/10.1145/138859.138867>.
- [6] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems : a survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*, 17(6) :734749, 2005. DOI : 10.1109/TKDE.2005.99.
- [7] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8) :30–37, 2009. DOI : 10.1109/MC.2009.263.
- [8] J. Zhang, X. Shi, S. Zhao, and I. King. Stargcn : stacked and reconstructed graph convolutional networks for recommender systems, 2019. arXiv : 1905.13129 [cs.IR].
- [9] X. Wang, X. He, M. Wang, F. Feng, and T.S. Chua. Neural graph collaborative filtering. *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, July 2019. DOI : 10.1145/3331184.3331267. URL : <http://dx.doi.org/10.1145/3331184.3331267>.
- [10] Z. Abbassi, S. Amer-Yahia, L. V. Lakshmanan, S. Vassilvitskii, and C. Yu. Getting recommender systems to think outside the box. In *Proceedings of the Third ACM Conference on Recommender Systems, RecSys '09*, pages 285–288, New York, New York, USA. Association for Computing Machinery, 2009. ISBN : 9781605584355. DOI : 10.1145/1639714.1639769. URL : <https://doi.org/10.1145/1639714.1639769>.
- [11] Flawnson Tong, What is Geometric Deep Learning ? <https://flawnsontong.medium.com/what-is-geometric-deep-learning-b2adb662d91d>

- [12] Simplilearn, What is Graph Neural Network? | An Introduction to GNN and Its Applications, <https://www.simplilearn.com/what-is-graph-neural-network-article>
- [13] A Gentle Introduction to Graph Neural Networks, <https://distill.pub/2021/gnn-intro/>
- [14] X. He, K. Deng, X. Wang, Y. Li, Y. Zhang, and M. Wang. Lightgcn : simplifying and powering graph convolution network for recommendation, 2020. arXiv : 2002.02126 [cs.IR].
- [15] ROBIN BURKE, *Hybrid Recommender Systems : Survey and Experiments.*
- [16] KIM FALK, *Practical Recommender Systems.*
- [17] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst. Geometric deep learning : going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4) :18–42, July 2017. ISSN : 1558-0792. DOI : 10.1109/msp.2017.2693418. URL : <http://dx.doi.org/10.1109/MSP.2017.2693418>.
- [18] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks, 2017. arXiv : 1609.02907 [cs.LG].
- [19] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio. Graph attention networks, 2018. arXiv : 1710.10903 [stat.ML].
- [20] W. L. Hamilton, R. Ying, and J. Leskovec. Inductive representation learning on large graphs, 2018. arXiv : 1706.02216 [cs.SI].
- [21] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. 2020. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems* 32, 1 (2020), 4–24.
- [22] Grover et al. 2016. node2vec : Scalable Feature Learning for Networks.

- [23] Perozzi et al. 2014. DeepWalk : Online Learning of Social Representations.
- [23] C. Cangea, P. Veličković, N. Jovanović, T. Kipf, and P. Liò. Towards sparse hierarchical graph classifiers, 2018. arXiv : 1811.01287 [stat.ML]
- [24] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR : Bayesian personalized ranking from implicit feedback. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*. 452–461.
- [25] S. Wu, F. Sun, W. Zhang, and B. Cui. Graph neural networks in recommender systems : a survey, 2021. arXiv : 2011.02260 [cs.IR].
- [26] R. van den Berg, T. N. Kipf, and M. Welling. Graph convolutional matrix completion, 2017. arXiv : 1706.02263 [stat.ML].
- [27] L. Wu, P. Sun, Y. Fu, R. Hong, X. Wang, and M. Wang. A neural influence diffusion model for social recommendation, 2019. arXiv : 1904.10322 [cs.IR].
- [28] W. Song, Z. Xiao, Y. Wang, L. Charlin, M. Zhang, and J. Tang. Sessionbased social recommendation via dynamic graph attention networks. Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining, January 2019. DOI : 10.1145/3289600.
- [29] R. van den Berg, T. N. Kipf, and M. Welling. Graph convolutional matrix completion, 2017. arXiv : 1706.02263 [stat.ML].
- [30] L. Zheng, C.-T. Lu, F. Jiang, J. Zhang, and P. S. Yu. Spectral collaborative filtering. Proceedings of the 12th ACM Conference on Recommender Systems, September 2018. DOI : 10.1145/3240323.3240343. URL : <http://dx.doi.org/10.1145/3240323.3240343>.
- [31] A. van den Oord, S. Dieleman, and B. Schrauwen. Deep content-based music recommendation. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neu-*

- ral Information Processing Systems, volume 26. Curran Associates, Inc., 2013. URL : <https://proceedings.neurips.cc/paper/2013/file/b3ba8f1bee1238a2f37603d90b58898d-Paper.pdf>.
- [32] H. O. Song, Y. Xiang, S. Jegelka, and S. Savarese. Deep metric learning via lifted structured feature embedding, 2015. arXiv : 1511.06452 [cs.CV].
- [33] S. Bell, Y. Liu, S. Alsheikh, Y. Tang, E. Pizzi, M. Henning, K. Singh, O. Parkhi, and F. Borisyuk. Groknet : unified computer vision model trunk and embeddings for commerce. In Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery Data Mining. Association for Computing Machinery, New York, NY, USA, 2020, pages 2608–2616. ISBN : 9781450379984. URL : <https://doi.org/10.1145/3394486.3403311>.
- [34] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.S. Chua. Neural collaborative filtering, 2017. arXiv : 1708.05031 [cs.IR].