PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA
MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC RESEARCH

Yahia Fares University of Médéa



Faculty of Science

Department of Mathematics and Computer Science

Graduation Project Report Submitted for Master of Computer Science degree

Specialization: Applied Artificial Intelligence

Deep Learning for a Reliable Recommendation System

Prepared by:
Nessrine MOKEDDEM
Yasmine ALIKACEM

Proposed and supervised by:
Ms. F. SADOUKI

Academic Year: 2024–2025

Abstract

Recommendation systems are increasingly essential for helping users navigate large volumes of information by providing personalized suggestions. To address common challenges such as data sparsity, cold-start problems, and a lack of contextual awareness, we propose a novel hybrid model specifically designed for travel recommendations.

Our approach integrates three main components: a neural collaborative filtering model to learn user-item interaction patterns, a sequential recommendation module based on user behavior, and a semantic similarity model to capture the meaning of user preferences and item descriptions. These models are combined using a weighted hybridization technique and are further enriched with contextual information related to user profiles and travel-related features.

Experimental evaluation demonstrates that our hybrid system significantly improves recommendation accuracy, relevance, and diversity compared to traditional approaches. This highlights the potential of combining neural architectures and contextual modeling in building more adaptive and effective recommendation systems.

Keywords: Recommendation system, Hybrid model, Neural collaborative filtering, Semantic similarity, Deep learning, Contextual information, Travel recommendation

Résumé

Les systèmes de recommandation jouent un rôle de plus en plus essentiel pour aider les utilisateurs à naviguer dans de vastes volumes d'informations en leur fournissant des suggestions personnalisées. Afin de relever les défis courants tels que la rareté des données, les problèmes de démarrage à froid et le manque de prise en compte du contexte, nous proposons un nouveau modèle hybride spécialement conçu pour la recommandation de voyages.

Notre approche intègre trois composants principaux : un modèle de filtrage collaboratif neuronal pour apprendre les schémas d'interaction utilisateur-élément, un module de recommandation séquentielle basé sur le comportement des utilisateurs, et un modèle de similarité sémantique pour capturer le sens des préférences des utilisateurs et des descriptions des éléments. Ces modèles sont combinés à l'aide d'une technique d'hybridation pondérée et sont enrichis par des informations contextuelles liées aux profils utilisateurs et aux caractéristiques spécifiques au domaine du voyage.

L'évaluation expérimentale montre que notre système hybride améliore significativement la précision, la pertinence et la diversité des recommandations par rapport aux approches traditionnelles. Cela souligne le potentiel de la combinaison d'architectures neuronales et de la modélisation contextuelle dans la conception de systèmes de recommandation plus adaptatifs et efficaces.

Mots-clés : Système de recommandation, Modèle hybride, Filtrage collaboratif neuronal, Similarité sémantique, Apprentissage profond, Information contextuelle, Recommandation de voyages

ملخص:

تُعد أنظمة التوصية أدوات أساسية بشكل متزايد لمساعدة المستخدمين في التنقل ضمن كميات هائلة من المعلومات، من خلال تقديم اقتراحات مخصصة. ولمعالجة التحديات الشائعة مثل تشتت البيانات، ومشاكل البداية الباردة، ونقص الوعي بالسياق، نقترح نموذجًا هجينًا جديدًا مصممًا خصيصًا لتوصيات السفر.

تعتمد منهجيتنا على دمج ثلاث مكونات رئيسية: نموذج للتصفية التعاونية العصبية لتعلم أنماط التفاعل بين المستخدمين والعناصر، ووحدة توصية تسلسلية تعتمد على سلوك المستخدم، ونموذج للتشابه الدلالي لالتقاط معنى تفضيلات المستخدمين ووصف العناصر. يتم دمج هذه النماذج باستخدام تقنية تهجين مرجّحة، ويتم إثراؤها بمعلومات سياقية متعلقة بملفات تعريف المستخدمين وخصائص مرتبطة بمجال السفر.

تُظهر التقييمات التجريبية أن نظامنا الهجين يُحسن بشكل كبير من دقة التوصيات وملاءمتها وتنوعها مقارنة بالأساليب التقليدية. وهذا يبرز الإمكانيات الكبيرة التي توفرها الجمع بين البنى العصبية والنمذجة السياقية في بناء أنظمة توصية أكثر فعالية وتكيّفًا.

الكلمات المفتاحية: نظام التوصية، نموذج هجين، التصفية التعاونية العصبية، التشابه الدلالي، التعلم العميق، المعلومات السياقية، توصية السفر.

Dedication

To all those who are dear to me, to whom I owe my success:

To my dear father, to the one who believes in me with all his heart, who supports me through everything and makes even the hardest moments feel easier, your love, your efforts, and your constant encouragement mean the world to me. You give me the strength to move forward and the motivation to always give my best, not only for myself but to make you proud and happy. Thank you for everything. I couldn't have done it without you.

To my dear mother, your unwavering support, and the sacrifices you've made have shaped the person I am today. Your , prayers have been my guiding light throughout this journey. Every achievement of mine is a reflection of your faith in me.

To my brother, MOHAMED, and to Redha, I'm proud to call you my brothers.

To my sisters, AMIRA and INES, Thank you for always being there for me. Your love, support, and encouragement have carried me through every step of this journey. I'm so lucky to have you by my side.

And to my little Oussaid, This is for you. I hope that one day, when you grow up, you'll find this and know how much you meant to me even when you were still so small. I love you, my little boy. And I hope that one day, you'll chase your dreams just like I chased mine.

To my Sheyla, Your presence in my life has been a quiet kind of magic. Thank you for the comfort, the kindness, and the little moments that meant more than you know.

To Wissem, My bestie — thank you for always being there with your brilliance and spirit.

To my dear binôme, YASMINE

Not just my partner in this work, but in everything. I can't find the words to explain what I feel right now. Sharing this special moment with you fills my heart with so much joy and gratitude. Thank you for your constant support, your ideas, your energy, and the way you always stood by me. You've been more than a partner — you've been a friend, a motivator, and a part of every step in this journey. I will never forget the moments we shared — whether they were filled with laughter or tears, stress or celebration. Every single one of them meant something, and they made this experience unforgettable. I'm truly happy and proud that we lived this journey together. This wouldn't have been the same without you.

Nessrine

Dedication

To all those who are dear to me, to whom I owe my journey:

To my dear family who supported me throughout my whole career — thank you.

To my dad, my everything, my world, my **role model**, who supported me in every step of my life. Your presence has been my anchor, your wisdom my guide.

To my sister, my second mom — your encouragement and love shaped who I am. Without you, there is no me. Thank you for always being by my side.

To my little Meryouma,
One day you'll grow up and find this dedication. I want you to know how much your auntie loves you, my sweet little sunshine. This is for you, my little girl.

To Wissem,
My bestie and the genius one — thank you for always being there with your brilliance and spirit.

To my dear NESSRINE,
You are not just a partner, you are my sister and my best friend — my everything. We have lived so much together: seven years of friendship, our Baccalauréat, the COVID-19 period, and the rollercoaster of our teenage lives. And now, here we are, building our future side by side.
I'm deeply grateful to Allah for blessing me with you. You are the best sister He could have given me. May He always protect you, my beloved Nono.

To myself,
Thank you for being brave, smart, and dedicated. Thank you for all the effort, for striving to achieve excellence, for aiming to shine and succeed.
May Allah guide me toward even greater success, help me rise, and allow me to look back with pride.

Yasmine

Acknowledgments

First and foremost, we would like to express our deepest gratitude to Allah for guiding us and granting us His protection throughout the completion of this humble work.

We sincerely thank our beloved parents for everything they have given us — their sacrifices, unconditional love, and unwavering support have been our foundation.

We extend our warmest and most heartfelt thanks to our supervisor, **Mrs. Sedouki Fatiha**, for her valuable guidance, patience, and full availability. Her insightful advice, constant encouragement, and trust in us have been truly essential throughout this journey. Beyond being a supervisor, she has been a true mentor who believed in us, challenged us to do better, and supported us every step of the way. We are genuinely grateful and honored to have been guided by her.

We also thank the esteemed members of the jury for taking the time to evaluate and assess our work.

As it would be impossible to name everyone individually, our sincere thanks go to all those who, in one way or another, contributed to the realization of this dissertation — through their advice, assistance, or knowledge.

Finally, we would like to express our appreciation to all the teachers who have guided us throughout our academic journey.

NESSRINE And YASMINE

Table of Contents

Table of Contents 6

List of Figures 11

List of Tables 12

List of acronyms 13

General Introduction 15

1 Recommandation Systems 18

1.1 Introduction . . . 18

1.2 History . . . 18

1.3 Definition . . . 19

1.4 Implicit Feedback and Explicit Feedback . . . 20

1.5 Evaluation matrix user-item . . . 20

1.6 Key Challenges in Recommendation Systems . . . 22

1.7 Classification of Recommendation Systems . . . 23

1.7.1 Classification by Rao & Talwar (2008) [1]: . . . 23

1.7.2 Classification by Su & Khoshgoftaar (2009): . . . 24

1.8 Types of recommendations . . . 25

1.8.1 Recommendation based on collaborative filtering . . . 25

1.8.2 Content-based recommendations . . . 26

1.8.3 Hybrid filtering . . . 26

1.8.3.1 Hybridization Methods . . . 27

1.8.4 Other types . . . 27

a) Demographic recommendation: . . . 27
b) knowledge based recommendation: . . . 28
c) Case-based recommendation . . . 28
d) Constraint-based recommendation . . . 29
1.8.5 Community-Based Recommendation . . . 29
1.9 Context-Aware Recommendation Models . . . 29
1.10 Advantages and Disadvantages of recommendation system methods . . . 30
1.10.1 Advantages of Collaborative Filtering . . . 30
1.10.2 Disadvantages of Collaborative Filtering . . . 30
1.10.3 Advantages of Content-based recommendation . . . 31
1.10.4 Disadvantages of Content-based recommendation . . . 31
1.10.5 Advantages of Hybrid Filtering . . . 31
1.10.6 Disadvantages of Hybrid Filtering . . . 31
1.11 Evaluation techniques and metrics . . . 32
1.11.1 Offline Evaluation . . . 32
1.11.2 Online Evaluation (A/B Testing) . . . 33
1.11.3 User Studies . . . 34
1.11.4 Additional Important Metrics . . . 34
1.12 Conclusion . . . 35

2 Machine Learning and Recommendation Systems 37
2.1 Introduction . . . 37
2.2 Foundations of Machine Learning . . . 37
2.2.1 Definition and Key Concepts . . . 37
2.2.2 Types of Machine Learning . . . 38
2.2.3 Supervised vs. Unsupervised Learning in Recommender Systems . . . 39
a) Supervised learning in recommender system . . . 39
b) Unsupervised learning in recommender system . . . 40
2.3 Classical Machine Learning Algorithms for Recommendation . . . 40
2.3.1 k-Nearest Neighbors (kNN) . . . 40

2.3.2 Linear and Logistic Regression . . . 41
2.3.3 Decision Trees and Random Forests . . . 42
2.3.4 Support Vector Machines and Naive Bayes . . . 42
2.3.5 Gradient Boosting Techniques . . . 43
2.4 Matrix Factorization and Hybrid Methods . . . 43
2.5 Introduction to Deep Learning . . . 43
2.5.1 Deep Learning vs. Classical ML . . . 44
2.5.2 Neural Network Basics . . . 45
2.5.3 Hyperparameters and Activation Functions . . . 46
2.5.4 Common Neural Network Architectures . . . 48
2.6 Neural Approaches to Recommendation . . . 50
2.6.1 Autoencoders for Collaborative Filtering . . . 50
2.6.2 Neural Collaborative Filtering . . . 50
2.6.3 Text-Based Recommendation using Word2Vec / BERT . . . 51
2.6.4 Semantic Matching with Sentence-BERT (SBERT) . . . 51
2.6.5 Sequential Recommendation with BERT4Rec . . . 52
2.6.6 Advantages and Challenges of Deep Learning in Recommender Systems . . . 52
2.7 Applications of Recommendation Systems in Popular Platforms . . . 53
2.8 Conclusion . . . 53

3 Development of the Proposed Hybrid Approach 56

3.1 Introduction . . . 56
3.2 Data Collection and Description . . . 57
3.2.1 Data extraction . . . 57
3.2.2 Data Preprocessing . . . 60
3.3 Architecture of the proposed hybrid recommendation system . . . 62
3.3.1 Selection of basic components . . . 62
a) Combinaison of bert4rec and sbert . . . 62
b) Neural Collaborative Filtering (NCF) . . . 65

3.3.2 Hybrid Recommendation Strategy Overview . . . 66
a) Final Integration: . . . 70
3.4 Definition of the experimental protocol . . . 70
3.4.1 Evaluation Metrics Selection . . . 70
a) Hit Rate@k . . . 70
b) Normalized Discounted Cumulative Gain (NDCG@k) . . . 70
c) Root Mean Square Error (RMSE) . . . 71
3.4.2 Baseline Systems for Comparison . . . 71
3.5 Conclusion . . . 72

4 Implementation and Experimentation 74

4.1 Introduction . . . 74
4.2 Development Environment and Tools . . . 74
4.2.1 Programming Language . . . 74
4.2.2 Libraries and Frameworks . . . 75
4.2.3 Hardware . . . 75
4.3 System Architecture and Implementation . . . 76
4.3.1 Implementation of Base Systems . . . 76
4.3.2 Hybridization Strategy . . . 76
a) Hybrid Reranking Evaluation . . . 77
4.3.3 Challenges and Solutions . . . 78
a) Data Encoding and Alignment . . . 78
b) Cold-Start and Unseen Users/Items . . . 79
c) Balancing Hybrid Weights . . . 79
d) Computational Resources . . . 79
e) Metric Computation Complexity . . . 79
4.4 Experimental Setup . . . 79
4.4.1 Trip Descriptions and SBERT Embeddings . . . 79
4.4.2 User-Trip-Rating Table (for NCF) . . . 80
4.4.3 User Trip Sequences (for BERT4Rec) . . . 80

4.4.4 Train-Test Split 80
4.4.5 Configuration of experiments and parameter tuning strategy 81
4.4.6 Execution of experiments on different model configurations. 81
4.5 Results and Evaluation 84
4.5.1 Quantitative Evaluation 84
a) Presentation of Results: 84
b) Comparison with Baseline Models: 85
4.5.2 Qualitative Analysis of Results 86
a) Types of Recommendations Generated 86
b) Ability to Address Identified Issues 87
4.5.3 Concrete Examples of Recommendations 87
4.5.4 Strengths and Limitations 88
a) Strengths and Successful Aspects of the System 88
b) Limitations and Areas for Improvement 89
c) Failure Cases and Unexpected Results 89
4.5.5 Research Questions and Hypotheses 89
4.6 User Interface Implementation 90
4.7 Conclusion 91

General conclusion **93**

Bibliography **94**

List of Figures

1.1 Evaluation matrix 21

1.2 Classifications of Recommendation Systems [2] 24

1.3 Recommendation based on collaborative filtering 25

1.4 Recommendation based on content-based. 26

1.5 Recommendation based on community recommendation. 30

2.1 Machine Learning vs. Deep Learning 44

2.2 Comparison Between a Biological Neuron and an Artificial Neuron 46

2.3 Neural collaborative filtering framework [3]. 51

3.1 Data collection and processing workflow for the travel recommendation system 57

3.2 Example of Sequential User Trip Data for BERT4Rec Model 61

3.3 Example of Sequential User Trip Data for NCF Model 62

3.4 Bert4Rec model architecture [4] 63

3.5 Hybrid BERT4Rec & SBERT 65

3.6 Neural Collaborative Filtering (NCF) architecture 66

4.1 Overall architecture of the hybrid recommendation system integrating BERT4Rec, SBERT, and NCF models. 78

4.2 Evaluation of configurations of the model bert4rec 82

4.3 Evaluation of configurations: global ranking and top-5 analysis 83

4.4 Hit@K Comparison Across Model Variants 85

4.5 NDCG@K Comparison Across Model Variants 86

4.6 Example of the user interface for trip rating and feedback 91

List of Tables

0.1 List of Abbreviations 13

1.1 Traditional Hybridization Methods[5] 27

2.1 Advantages and disadvantages of common neural network architectures [6]. 49

3.1 Key Metadata Attributes of Hotels, Points of Interest (POIs), and Flights . . 59
3.2 The Complete Trip Dataset after preprocessing 60
3.3 dataset of the trips 61

4.1 Hybrid Strategy Performance Comparison 77
4.2 Model Configurations and Hyperparameter Tuning 81
4.3 Best Performing Hyperparameter Configuration for BERT4Rec 82
4.4 Best Performing Hyperparameter Configuration for NCF (Config 8) 84
4.5 Hit@K Performance of the Hybrid Model 84
4.6 NDCG@K Performance of the Hybrid Model 85
4.7 Hit Rate Comparison Across Model Variants 85
4.8 NDCG Comparison Across Model Variants 85
4.9 Hybrid model Top-5 trip predictions for sample users 88

List of acronyms

Table 0.1: List of Abbreviations

Abbreviation	**Full Form**
IA	Intelligence Artificielle
API	Application Programming Interfaces
KBRS	Knowledge-Based Recommendation System
CBRS	Case-Based Recommendation System
RSs	Recommendation Systems
CF	Collaborative Filtering
CBF	content-based filtering
TF	Term Frequency
IDF	Inverse Document Frequency
NCF	Neural Collaborative Filtering
CNN	Convolutional Neural Network
LLMs	Large Language Models
K-NN	K-Nearest Neighbors
SVD	Singular Value Decomposition
SVM	Support Vector Machines
GPU	Graphics Processing Unit
DBSCAN	Density-Based Spatial Clustering of Applications with Noise
RL	Reinforcement Learning
RMSE	Root Mean Squared Error
NDCG	Normalized Discounted Cumulative Gain
MRR	Mean Reciprocal Rank
MAE	Mean Absolute Error
PCA	Principal Component Analysis
XGBoost	Extreme Gradient Boosting
NMF-ALS	Non-negative Matrix Factorization with Alternating Least Squares
ML	Machine Learning

General Introduction

"**We are leaving the age of informatiom and entering the age of recommendation**"

Chris Anderson in the long tail

In today's digital age, the overwhelming volume and diversity of online information make it challenging for users to find content that aligns with their needs. Information is often scattered across sources, presented in different formats, and rarely tailored to individual preferences. Moreover, user needs are dynamic and not always explicitly expressed, further complicating the delivery of relevant content. These challenges have made recommender systems (RSs) essential tools in modern information retrieval.

Recommender systems help users manage information overload by suggesting items that match their preferences. By analyzing past behaviors, these systems predict future interests, personalizing the user experience. Traditional RSs rely on three main approaches: collaborative filtering (CF), content-based filtering (CBF), and hybrid methods. CF predicts preferences based on similar users, while CBF matches user profiles to item features. However, CF struggles with problems like data sparsity and cold starts, while CBF risks providing overly narrow recommendations. Hybrid systems, which combine multiple approaches, have gained attention for their ability to address these limitations, particularly when enhanced by deep learning and contextual data.

In this project, we propose a novel hybrid recommendation system designed for the travel domain. Our model integrates deep neural networks with contextual matrix factorization to enhance recommendation accuracy, diversity, and personalization. Contextual factors such as seasonality, destination, and travel purpose are incorporated to adapt recommendations to user needs in dynamic environments. This work explores the recommendation process, deep learning architectures, and contextual modeling, presenting our hybrid system's structure, components, and implementation in detail.

We evaluate the effectiveness of our approach through experiments using real-world data, comparing it with classical models. Results demonstrate significant improvements in relevance, diversity, and user satisfaction, highlighting the potential of combining deep learning with contextual modeling. This project contributes to building more adaptive, intelligent, and user-centric systems capable of addressing the needs of modern users.

The thesis is organized as follows:

- **Chapter 1**: Introduces recommender systems, their main strategies, and evaluation metrics.

- **Chapter 2**: Provides an overview of artificial intelligence, focusing on machine learning, deep learning, and commonly used algorithms.
- **Chapter 3**: Describes the architecture and methodology of the proposed hybrid recommendation model.
- **Chapter 4**: Details the implementation, experimental setup, and evaluation of our model using real-world data.

Chapter 1

Recommandation Systems

Chapter 1

Recommandation Systems

1.1 Introduction

The growth of the Web has created a need for new techniques that not only help users find the information they are looking for but also make them aware of the existence of relevant content. These techniques are known as recommendation systems, which are primarily based on information filtering methods, as Ricci and al. define them [7].For many of us, recommender systems remain somewhat mysterious entities that seem to anticipate our thoughts and desires. This chapter provides an overview of recommender systems, starting with their definition and historical background. We will explore the various filtering approaches used in these systems, as well as the different similarity measures that enable them to match concepts. Additionally, we will examine the evaluation metrics employed to assess the effectiveness of recommender systems.

1.2 History

Recommender systems have become a major research area in recent decades, with the improvement of their techniques being a relatively recent topic compared to information retrieval systems, which date back to the early days of computing.The first known recommender system was developed in 1992 by Paul Resnick and John Riedl. This system, called GroupLens, used collaborative filtering to recommend Usenet news articles based on ratings provided by users with similar preferences [8]. This marked a significant milestone in personalized content delivery. A few years later, in 1995, Amazon implemented a recommender system based on item-to-item collaborative filtering to suggest products to users based on the purchases of others with similar interests [9]. This innovation helped popularize recommender systems in e-commerce and became a foundational component of personalized online experiences. The concept of "collaborative filtering" was first introduced in 1992 through the Tapestry system [10], which marked the advent of recommender systems. This system was able to generate and send recommendations to users via email. pioneering the idea of personalized content suggestion. Throughout the late 1990s and early 2000s, collaborative filtering methods became more widespread, particularly in the context of e-commerce and online services. Notably, Amazon and Netflix began using recommender systems to suggest products and movies to their users, which

helped popularize the concept. As the internet grew, the need for more sophisticated recommendation techniques led to the development of hybrid systems, which combined collaborative filtering with content-based filtering. These hybrid systems aimed to overcome the limitations of pure collaborative filtering, such as the cold start problem and sparsity issues. In the 2010s, deep learning techniques began to gain traction in recommender systems, enabling more complex models that could handle large-scale data and incorporate unstructured information, such as images and text. More recently, the integration of large language models (LLMs) has opened new possibilities for enhancing recommendation quality, particularly through better understanding of user preferences and context. As Lin and al state [11], “The emergence of LLM is inspiring the design of recommender systems and pointing out a promising research direction, i.e., whether we can incorporate LLM and benefit from their knowledge and capabilities to compensate for the limitations of conventional recommendation models.”.Overall, recommender systems have evolved from simple algorithms to highly sophisticated models that play a crucial role in personalizing online experiences across various domains.

1.3 Definition

A recommendation system (RS) is an intelligent software application that leverages user data, behavioral patterns, and contextual information to predict and suggest items of interest. The main goal is to generate meaningful recommendations to a collection of users for items or products that might interest them[12].

RS utilize machine learning algorithms, such as collaborative filtering, content-based filtering, or hybrid approaches, to personalize the user experience by delivering relevant content, products, or services. Commonly integrated into e-commerce platforms, streaming services, and online marketplaces, RSs play an important role in such highly rated Internet sites as Amazon.com, YouTube, Netflix, Yahoo, Tripadvisor, Last.fm, and IMDb. Moreover many media companies are now developing and deploying RSs as part of the services they provide to their subscribers, for example Netflix, the online movie rental service, awarded a million dollar prize to the team that first succeeded in improving substantially the performance of its recommender system [13] .

Another definition of recommendation systems are a useful technology that can alleviate the problem of overload of information provided to users. It predicts the grade of items to be recommended to the user, creates a list of recommendation rankings for each user, and makes it possible to recommend items related to the user [14].

At the heart of every recommendation system lies a fundamental triad: users, items, and algorithms. As defined by Resnick and Varian in their seminal work, "Recommender systems are software tools that suggest items to users based on their preferences, bridging the gap between information overload and personalized discovery" [15]. The user embodies the target of recommendations, whose behaviors and attributes drive personalization. The item whether a product, movie, or article, must be effectively represented to match user interests. Finally, the algorithm acts as the engine. Collaborative filtering algorithms harness collective user behavior to uncover interaction patterns, whereas content-based approaches rely on item features to deliver personalized recommendations[16]. Together, this triad forms a feedback loop, continuously refining recommendations. The interaction

between users, items, and algorithms is the cornerstone of modern recommender system performance[17].

1.4 Implicit Feedback and Explicit Feedback

In recommendation systems (RS), feedback is essential for understanding user preferences and generating personalized recommendations. There are two primary types of feedback: explicit feedback and implicit feedback, each possessing distinct strengths and limitations. Explicit feedback refers to data provided by users in which they directly express their preferences, typically through mechanisms such as rating scales (e.g., 1 to 5 stars) or binary likes/dislikes. This form of feedback enables users to communicate their opinions about items clearly, making it easier for the system to interpret and process their preferences[18]. The principal strength of explicit feedback lies in its clarity and accuracy, as it offers direct insights into user preferences. However, a significant limitation is that it requires active user participation, which may result in sparse datasets since not all users are willing to provide ratings or feedback. In explicit feedback datasets, users assign explicit weights to their interactions with items [19], thereby signaling the intensity of their approval or disapproval. In contrast, implicit feedback is automatically generated through user interactions with the system, such as clicks, browsing history, or purchase behavior. It is inferred from the user's actions rather than explicitly stated. Implicit feedback can substitute for explicit ratings in recommender systems, allowing for more efficient and effective recommendations[20].The advantage of implicit feedback is its abundance, as it is continuously generated through regular user engagement with the system, without requiring additional effort from the user. However, implicit feedback can be less reliable than explicit feedback, as user behavior does not always accurately reflect true preferences. For instance, a user may click on an item out of curiosity without genuinely liking it. Furthermore, implicit feedback datasets typically lack explicit weights for user interactions, which limits the ability to extract detailed preference insights[21]. The integration of explicit feedback and implicit feedback offers significant potential for enhancing recommendation systems. While much of the research in RS has focused on utilizing either type of feedback independently, only a few studies have explored the combination of these two heterogeneous forms of feedback[18]. By combining both feedback types, systems can capitalize on the strengths of each, compensating for the limitations of one with the advantages of the other. For example, explicit feedback can help refine the inferences derived from implicit feedback, leading to more personalized and accurate recommendations. Although this hybrid approach has been examined in certain studies, it remains relatively underexplored in comparison to the predominant focus on using either explicit or implicit feedback in isolation.

1.5 Evaluation matrix user-item

The user-item matrix represents the interactions between users and items, such as ratings, views, or purchases, in recommender systems. The rows in the matrix represent users, and the columns represent items, with the cells indicating whether an interaction has taken place. Because most users only engage with a small number of items, the matrix

is often sparse, making it challenging to extract useful recommendations . When the number of items rated by a user is very small compared to the total number of items in the system, the recommendation system becomes sparse. This leads to a very low density in the user-item rating matrix. This affects the system's ability to recommend all available items and reduces the accuracy of the generated recommendations[22].

	Burger	Pizza	Sushi
A	4.0	5.0	?
B	5.0	?	2.0
C	?	1.0	5.0

Figure 1.1: Evaluation matrix

The user-item matrix serves as the foundation for two major families of recommendation approaches: collaborative filtering and matrix factorization.

Collaborative Filtering: In collaborative filtering, the user-item matrix acts as a core component, where rows represent users, columns represent items, and each cell contains a rating or interaction score between a user and an item. This approach operates on the principle that users who have exhibited similar rating patterns in the past are likely to share preferences in the future. Two primary strategies dominate this paradigm:

- User-based filtering identifies users with similar rating histories and recommends items those users have liked.
- Item-based filtering focuses on items that receive comparable ratings from the same users, suggesting items similar to those a user has previously interacted with.

However, collaborative filtering's effectiveness is heavily constrained by the sparsity of the user-item matrix. When the number of available ratings is significantly smaller than the number of items, the matrix becomes sparse, hindering the detection of meaningful patterns and degrading recommendation performance.

Matrix Factorization: To mitigate sparsity, matrix factorization techniques decompose the original user-item matrix into two lower-dimensional matrices: one encoding latent user preferences and the other capturing latent item characteristics. Instead of relying on direct user-item similarities, this method uncovers hidden factors that explain observed interactions. For instance, if User A is represented by the latent vector [0.9, 0.1] (indicating a strong preference for a specific feature) and an item like "Burger" is represented by [1.0, 0.2], their predicted interaction score is derived from the dot product: $(0.9 \times 1.0) + (0.1 \times 0.2) = 0.92$, suggesting a high likelihood of preference. Beyond improving prediction accuracy in sparse datasets, matrix factorization reveals deeper, interpretable relationships between users and items, enhancing both scalability and recommendation quality. Matrix factorization techniques generally fall into three main categories.

- **Singular Value Decomposition (SVD):** Breaks down the user-item matrix into orthogonal components that capture latent features. However, its assumption of data density reduces its applicability in sparse, real-world scenarios.

- **Non-negative Matrix Factorization (NMF):** Enforces non-negativity on the latent factors, resulting in more interpretable dimensions, such as quantifying a user's degree of interest in specific categories or genres.

- **Probabilistic Matrix Factorization (PMF):** Introduces a probabilistic approach to matrix decomposition, enhancing resilience to noise and missing data, and forming the foundation for various Bayesian extensions.

1.6 Key Challenges in Recommendation Systems

Modern recommendation systems face fundamental challenges that impact their effectiveness and applicability across domains. This section systematically examines these key problematics, which represent active research frontiers in the field.

- **Cold-Start Problems**: A critical challenge in recommendation systems is the cold start problem, where the lack of sufficient historical interaction data makes it difficult to generate accurate suggestions for new users or newly added items. This problem is fundamentally a representation learning challenge, which occurs when it is not possible to make reliable recommendations due to an initial lack of ratings too[23]. This limitation severely impacts the system's effectiveness, particularly in dynamic platforms such as: E-commerce,Streaming services.

- **Data sparsity**: The phenomenon where the user-item interaction matrix (e.g., ratings, clicks, purchases) contains very few observed entries compared to the total possible interactions. The 'long tail' of niche items exacerbates sparsity, creating a feedback loop where popular items dominate recommendations while others remain invisible.[24].

- **Scalability and computational complexity**: RSs handle massive datasets. The user-item matrix grows quadratically, making traditional methods such as k-NN computationally expensive. To address this, modern solutions use distributed computing such as Spark, dimensionality reduction such as SVD and graph neural networks for efficient large-scale processing. Hardware optimizations like GPU acceleration further enhance performance. Scalability forces researchers to choose between exact solutions and feasible computations. The art lies in losing minimal accuracy for maximal speed gains[25] . Companies like Pinterest and Alibaba leverage these techniques to manage billions of interactions while maintaining real-time recommendations. The field continues to evolve with innovations like federated learning and quantum ML to balance accuracy and efficiency.

- **Serendipity**: The ability of a system to suggest unexpected but relevant items that pleasantly surprise users while still aligning with their latent preferences. Balances novelty and relevance. Serendipitous recommendations strike a balance between obvious relevance and pleasurable surprise, they should feel both inevitable and unexpected[26]. such as Spotify's "Discover Weekly" curates playlists that blend familiar tracks with improbable yet fitting suggestions, exposing users to new artists while maintaining tonal coherence. Similarly, Amazon's "Customers who bought X also bought Y" generates surprising but logical associations.

- **Grey Sheep Problem**: Users of recommendation systems can have unique preferences and highly unusual tastes. These users lie at the boundary between two or more user clusters. These users exhibit mixed or eclectic tastes that do not clearly align with any well-defined segment, making it difficult for traditional collaborative filtering methods to generate accurate recommendations.

- **Diversity:** The ability to suggest a broad range of relevant items, rather than focusing only on overly similar or popular ones, is crucial. While accuracy-focused algorithms (e.g., collaborative filtering) often prioritize matching user preferences, they can inadvertently create 'filter bubbles,' limiting exposure to novel content. Ensuring diversity in recommendations is essential to avoid over-specialization and enhance user satisfaction, particularly in domains such as e-commerce and news recommendation systems s[27].

1.7 Classification of Recommendation Systems

Classical Classification : The traditional taxonomy divides recommender systems into three main categories:

1. **Content-Based Filtering** .Content-based systems recommend items similar to those the user liked in the past by analyzing item attributes [28]

2. **Collaborative Filtering (CF)** Relies on user-item interactions (e.g., ratings) to find patterns. It splits into:

   - **User-based CF**: Finds similar users (e.g., *"People like tulip flower also liked…"*).
   - **Item-based CF**: Recommends items similar to those the user interacted with (e.g., Amazon's *"Customers who bought this also bought…"*).

     Goldberg et al. (1992) pioneered this approach, calling it *"the social method of filtering."*

3. **Hybrid Systems** Combine content and collaborative methods to mitigate weaknesses (e.g., cold-start problem). Hybrid systems exploit multiple techniques to improve recommendation accuracy and coverage [29].

1.7.1 Classification by Rao & Talwar (2008) [1]:

In their study "A Survey on Recommender Systems", Rao and Talwar proposed an expanded classification framework that goes beyond traditional content-based and collaborative filtering approaches. Their taxonomy introduces four key categories of recommender systems, emphasizing adaptability to different problem contexts: Demographic-Based, Approaches, Knowledge-Based, Approaches, Utility-Based Approaches and Context-Aware Approaches. Rao and Talwar argue that *"this multidimensional framework allows the selection of recommendation algorithms based on the nature of the problem rather than being limited to purely technical paradigms."* Their classification provides a more flexible and application-oriented perspective compared to traditional models.

1.7.2 Classification by Su & Khoshgoftaar (2009):

In their influential survey on collaborative filtering techniques, Su and Khoshgoftaar[30] proposed a practical classification focusing specifically on algorithmic approaches. They categorized recommendation systems into three main types:

- memory-based methods .that compute similarities between users/items using metrics like Pearson correlation.
- model-based techniques. that employ machine learning algorithms including matrix factorization and neural networks.
- hybrid systems. that combine both approaches to overcome their individual limitations.

This classification is particularly valuable because it highlights the evolution from simple neighborhood-based methods to sophisticated predictive modeling in recommender systems.Their framework helps practitioners select appropriate techniques based on dataset characteristics and performance requirements, with model-based approaches generally offering better scalability for large, sparse datasets while memory-based methods provide more interpretable results. The authors' systematic comparison of these approaches remains widely cited in recommender systems literature.

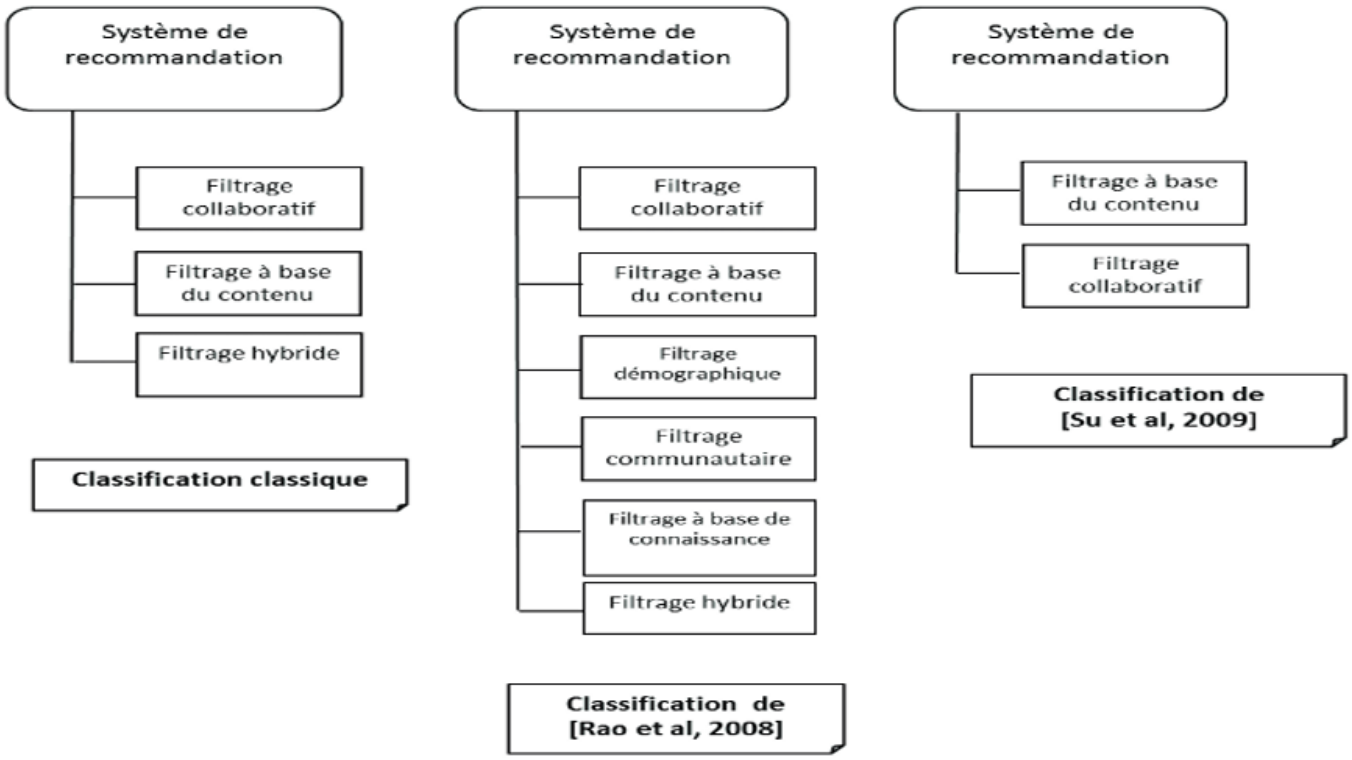


Figure 1.2: Classifications of Recommendation Systems [2]

1.8 Types of recommendations

Recommendation systems can be categorized into several types based on the techniques and data they use to generate suggestions. These systems aim to predict a user's preference for an item and enhance user experience by providing relevant and personalized recommendations. The main types include:

1.8.1 Recommendation based on collaborative filtering

Collaborative filtering (CF) remains one of the most widely adopted techniques in modern recommender systems due to its ability to predict user preferences based on the behavior of other similar users. The principle underlying collaborative filtering is that users who shared preferences in the past are likely to share preferences in the future. As noted by Javed et al. (2021), collaborative filtering techniques analyze large datasets of user behaviors, actions, and ratings to infer potential interests by identifying patterns and similarities between users or items. This approach has been successfully employed in domains such as e-commerce, digital libraries, and entertainment services like movie and music recommendation platforms [31].

According to Su and Khoshgoftaar (2009), collaborative filtering techniques can be broadly divided into two main categories: memory-based and model-based methods. Memory-based collaborative filtering relies directly on the user-item rating matrix to calculate similarities, either between users (user-based CF) or between items (item-based CF). On the other hand, model-based approaches use machine learning algorithms such as matrix factorization, clustering, or Bayesian methods to learn predictive models that improve scalability and accuracy, particularly in large and sparse datasets [32].

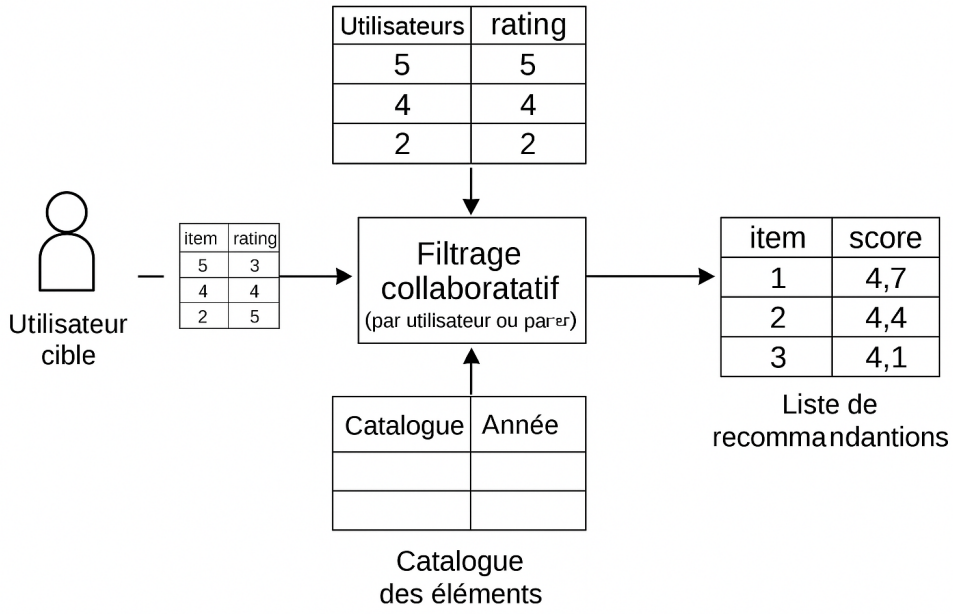


Figure 1.3: Recommendation based on collaborative filtering .

1.8.2 Content-based recommendations

Content-based recommendation systems are designed to suggest items to users by analyzing the attributes of items they have previously liked or interacted with. These systems build user profiles based on item descriptors such as keywords, categories, and metadata, then recommend similar items by comparing item content with the user's recorded preferences. These systems are commonly used in information retrieval applications, where terms are manually assigned to describe items and then matched with the user's profile. Techniques such as term frequency (TF) and inverse document frequency (IDF) are frequently employed to measure the relevance of items, particularly in domains like article, news, or movie recommendations [33].

Content-based recommenders also rely on content extraction and characterization processes to build both item profiles and user profiles, where a user's interests are typically represented using weighted keywords or descriptors. As more data becomes available, the user profile is dynamically updated to improve recommendation relevance. Early examples of content-based systems include Fab, a system that recommended web pages, and ELFI, which suggested financial information based on a database [34].

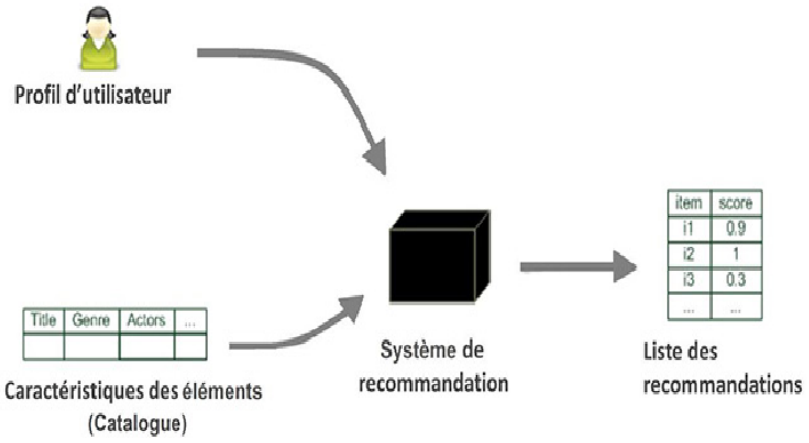


Figure 1.4: Recommendation based on content-based.

1.8.3 Hybrid filtering

Hybrid filtering is an approach that combines multiple recommendation techniques to improve the quality and robustness of recommendation systems [35]. The motivation for adopting hybrid filtering stems from the need to address common limitations encountered in individual approaches such as the cold-start problem, data sparsity, and overspecialization [35]. By integrating techniques like collaborative filtering and content-based filtering, hybrid systems enhance both the accuracy and efficiency of the recommendation process [35].

In a hybrid recommendation system, each individual method computes similarity scores or prediction values independently, and these local scores are subsequently combined using specific hybridization strategies [5]. Common strategies include weighted combi-

nations, mixing, cascading, and switching mechanisms, each differing in how they merge the local prediction results to produce a final recommendation score [5]. Hybridization can be applied either at the rating prediction level or within the prediction function itself, depending on the design and objectives of the system [5]. This combined approach allows recommendation systems to better exploit the strengths of different methods while mitigating their weaknesses.

1.8.3.1 Hybridization Methods

Table 1.1: Traditional Hybridization Methods[5]

Hybridization Methods	**Description**
Weighted	The weighted results of several recommendation techniques are combined to produce a new recommendation.
Combination	Different recommendation techniques are combined into a single recommendation algorithm.
Mixed	Recommendations from several techniques are presented at the same time.
Cascade	A recommendation system refines the results provided by another system.
Permutation	The system switches between different recommendation techniques based on the recommendation result.
Meta-level	The model learned by one recommendation technique is used as input data for another technique.
Augmentation	The output of one recommendation technique is used as input data for another technique.

1. Modern Hybridization Techniques

In addition to the classical hybridization methods such as weighted, cascade, or meta-level approaches recent advances in machine learning and artificial intelligence have led to the emergence of more sophisticated hybrid recommendation strategies. These modern techniques often rely on deep learning architectures, context-awareness, and adaptive learning frameworks. They are designed to handle complex, high-dimensional data and improve the personalization and accuracy of recommendation systems.

1.8.4 Other types

a) Demographic recommendation:

Demographic recommendation relies on personal user attributes such as age, gender, location, education level, or profession to generate relevant suggestions. The core idea is

that users who share similar demographic characteristics often have comparable preferences [36]. Unlike collaborative or content-based approaches, this method does not require interaction history or item analysis, making it especially useful in addressing the cold-start problem [37]. However, it heavily depends on the quality and availability of demographic data and may lead to generalizations or stereotypes when used in isolation [38]. According to [5], the demographic approach is often integrated into hybrid systems to improve recommendation accuracy, particularly when behavioral data is limited.

b) knowledge based recommendation:

Knowledge-Based Recommendation Systems (KBRS) rely on domain knowledge and reasoning techniques to suggest items that meet the specific needs or preferences of users. Unlike collaborative or content-based systems, KBRS do not require historical user data or item interactions, making them particularly effective in cold-start scenarios or when recommendations involve infrequent and high-involvement decisions (e.g., real estate, travel, or medical advice) [39]. These systems typically use structured knowledge, such as ontologies, constraint-based models, or rule-based reasoning, to match user requirements with item characteristics [39]. One major advantage of KBRS is their ability to provide precise and context-aware recommendations without relying on user ratings or behavioral patterns. However, the main challenge lies in the acquisition and formalization of domain knowledge, which can be complex and time-consuming [39].

c) Case-based recommendation

Case-based recommender systems (CBRS) offer a content-based approach to personalized recommendations by leveraging structured item descriptions and similarity-based retrieval. Unlike collaborative filtering, which relies on user ratings, CBRS utilize detailed item attributes to match user queries with relevant cases. This methodology is particularly effective when user preferences are ill-defined or when exact matches are scarce, as it allows for the retrieval of similar items based on feature similarity [40].

Moreover, CBRS are especially useful in practical applications, such as nutritional menu planning. For instance, systems like CAMPER integrate both case-based and rule-based reasoning to generate daily menus that align with individual nutritional needs and preferences, adhering to dietary guidelines and aesthetic standards [41]. This hybrid approach enhances the system's ability to produce personalized and acceptable meal plans, even under constraints like limited ingredients or preparation resources.

In addition, CBRS address common challenges in recommendation systems, such as the "stonewalling" effect, where strict adherence to user queries yields no results. By employing similarity-based retrieval, CBRS can suggest near-matches, thereby improving user satisfaction and system flexibility [40]. This adaptability makes CBRS a valuable tool in domains requiring personalized recommendations under complex constraints.

Thus, CBRS stand out for their flexibility and ability to adapt, key traits for modern recommendation systems. In the next section, we will explore their application in more detail, particularly in domains with highly dynamic or uncertain user preferences.

d) Constraint-based recommendation

Constraint-Based Recommendation Systems leverage specific user preferences and product attributes, forming the foundation for many recommendation applications. By modeling the relationship between customer requirements and product features in the form of constraints, these systems propose personalized recommendations and solutions that meet the user's needs. In cases where no suitable solution can be found, constraint-based recommenders can suggest repair alternatives, guiding the user toward possible adjustments or alternatives that align with their preferences. These systems are particularly effective in domains with complex or infrequent purchases, where users may lack detailed knowledge of technical specifications, such as financial services, technical equipment, or telecommunication systems. Additionally, constraint-based approaches can be enriched with collaborative recommendation features, which take into account community preferences to improve the user experience. In fields like financial services, constraint-based recommenders help improve advisory processes by automating explanations and repair actions, ultimately enhancing service quality, reducing faulty offerings, and saving time in advisory sessions [42]. Moreover, incorporating visualization components into these systems can further enhance user interaction and acceptance, as users are able to engage directly with virtual product representations, improving the overall user experience [43]. Applications supporting the interactive selling of financial services have been reported to improve customer service quality, with tangible benefits such as faster advisory sessions and a reduction in errors during product offering [44]. These enhancements are essential in industries where the customer's needs are complex and the correct solution is not always apparent [45]

1.8.5 Community-Based Recommendation

Community-based recommendation, also known as social recommendation, relies on the user's relationships with friends within a network to generate suggestions. It is widely used on platforms such as Facebook and Twitter, where recommendations are influenced by social interactions. According to a Nielsen survey, *84% of global consumers trust word-of-mouth recommendations from friends and family* when making decisions related to products or services [46]. Furthermore, [47] highlight that "partial similarity is an aspect of the user's preference projected by the community to which they belong," emphasizing the power of community-driven suggestions.

1.9 Context-Aware Recommendation Models

As user needs evolve and data volumes grow, modern recommender systems increasingly incorporate contextual information to provide more relevant and personalized suggestions. Unlike traditional approaches that focus solely on user-item interactions, *context-aware models* take into account additional factors such as time, location, browsing history, or even the user's emotional state [48]. These models often rely on advanced architectures such as Transformer-based models, which can capture complex dependencies across contextual dimensions [49]. By integrating context, recommendation quality is significantly enhanced, as the system adapts to the user's specific and dynamic needs.

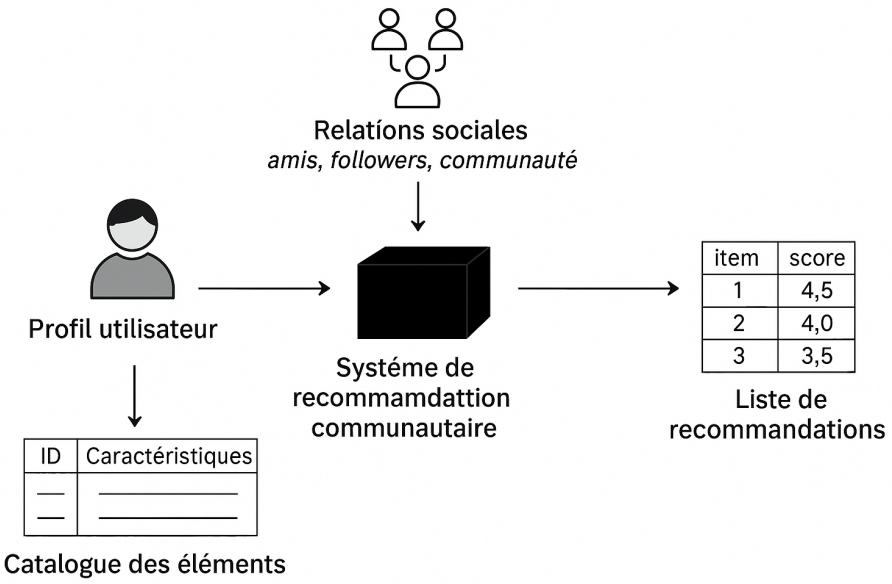


Figure 1.5: Recommendation based on community recommendation.

1.10 Advantages and Disadvantages of recommendation system methods

1.10.1 Advantages of Collaborative Filtering

1. Ease of implementation: Memory-Based collaborative filtering methods are relatively easy to develop and understand, as they directly use similarities between users or items without requiring complex model training [50].

2. Real-time adaptability: These methods can incrementally add new data, enabling continuous and dynamic updates of recommendations [50].

3. Improved prediction with model-based approaches: Model-Based collaborative filtering methods, such as matrix factorization and deep learning algorithms, offer higher prediction accuracy and can efficiently handle large and complex datasets [32].

1.10.2 Disadvantages of Collaborative Filtering

1. Cold Start Problem: When a new user or item enters the system, there is little to no information to generate accurate recommendations, leading to poor initial performance [50].

2. Scalability issue: Processing billions of interactions between users and items demands significant computational resources in terms of time and memory [32].

3. Sparsity problem: User-item matrices are typically very sparse because most users rate only a small portion of the available items, which reduces the effectiveness of recommendations [32]

1.10.3 Advantages of Content-based recommendation

1. User independence: This approach relies solely on the preferences (explicit or implicit) of the active user and the attributes of items, avoiding dependence on other users' data [50].

2. Transparency: Content-based systems can explain their recommendations by highlighting which item attributes match the user's preferences [51].

3. Handles new items effectively: Unlike collaborative filtering, content-based methods can recommend items that no one has rated yet, which is beneficial for addressing the item cold start problem [50].

1.10.4 Disadvantages of Content-based recommendation

1. New User Problem: In the absence of a sufficient interaction history, the system struggles to identify a user's preferences and produce relevant recommendations [32].

2. Overspecialization: The system tends to recommend items very similar to those already liked by the user, limiting the diversity of suggestions [51].

3. Dependence on quality of item descriptors: The effectiveness of recommendations strongly depends on the quality and richness of the attributes used to represent items [50].

1.10.5 Advantages of Hybrid Filtering

1. Combines the strengths of multiple methods: Hybrid approaches merge the benefits of collaborative filtering and content-based filtering, helping to overcome the limitations of each technique when used separately [32].

2. Better accuracy and coverage: Hybrid systems generally achieve better results in terms of prediction accuracy, diversity, and recommendation coverage [51].

1.10.6 Disadvantages of Hybrid Filtering

1. Increased system complexity: Combining multiple techniques increases system complexity and requires additional resources for development and maintenance[32].

2. Higher computational cost: Running several algorithms in parallel can result in longer processing times and increased hardware resource consumption [51].

1.11 Evaluation techniques and metrics

Evaluating recommender systems is a crucial task to ensure the relevance, accuracy, and usefulness of the recommendations provided to users. As outlined by Shani and Gunawardana [52], evaluation techniques are generally divided into three main categories:

1.11.1 Offline Evaluation

Offline experiments are performed using historical data, without involving real-time user interaction. They involve splitting a dataset into training and testing sets and evaluating the system's ability to predict known preferences. Common metrics used in offline evaluation include:

- **Mean Absolute Error (MAE)**: Measures the average absolute difference between predicted and actual ratings.

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|r_i - \hat{r}_i|$$

- **Root Mean Square Error (RMSE)**: Penalizes larger errors more heavily than MAE [53].

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(r_i - \hat{r}_i)^2}$$

- **Precision@K**: Proportion of recommended items in the top K that are relevant.

$$\text{Precision@K} = \frac{\text{Number of relevant items in top K}}{K}$$

  Where: - Number of relevant items in top K is the count of relevant items in the top K recommended items. - K is the total number of items recommended.

- **Recall@K**: Proportion of all relevant items that are recommended in the top K.

$$\text{Recall@K} = \frac{\text{Number of relevant items in top K}}{\text{Total number of relevant items}}$$

  Where: - Number of relevant items in top K is the count of relevant items in the top K recommended items. - Total number of relevant items is the total count of relevant items in the entire dataset.

- **F1-Score**: Harmonic mean of Precision and Recall.

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

- **Hit Rate@k** Hit Rate@k (also known as Hit@k) measures the proportion of times the correct (or relevant) item appears in the top-k recommendations. It is a simple

yet effective indicator of whether the model is capable of retrieving relevant items within the top-k list.

Formula[53]:

$$\text{Hit Rate@k} = \frac{|\mathcal{U}_{\text{hit}_k}|}{|\mathcal{U}_{\text{all}}|}$$

Where:

- $|\mathcal{U}_{\text{hit}_k}|$ is the number of users for whom the relevant item is among the top-k recommended items.
- $|\mathcal{U}_{\text{all}}|$ is the total number of users in the test set.

where U is the set of users, r_u is the relevant item for user u, and $\mathbb{I}$ is the indicator function, which returns 1 if r_u is in the top-k predicted items Top-k_u, and 0 otherwise.

- **Normalized Discounted Cumulative Gain (NDCG@k):**NDCG@k evaluates the ranking quality of the recommendation list by assigning higher scores to hits that appear at higher ranks. It considers the position of the relevant item and provides a graded relevance metric.

  Formula:[53]

$$\text{NDCG@k} = \frac{\text{DCG@k}}{\text{IDCG@k}}$$

$$\text{DCG@k} = \sum_{i=1}^{k} \frac{2^{\text{rel}_i} - 1}{\log_2(i+1)}$$

  Where:

  - rel_i is the graded relevance of the item at position i.
  - IDCG@k is the ideal DCG (maximum DCG possible up to position k).

  where $rel_{u,i}$ is the relevance of the item at position i for user u, and IDCG@k$_u$ is the ideal (maximum possible) DCG for that user at rank k, used to normalize the DCG score.

- **Mean Reciprocal Rank (MRR)** evaluates the average reciprocal rank of the first relevant item in the recommendation list. It is particularly useful when there is only one relevant item per query. **Formula:[54]**

$$\text{MRR} = \frac{1}{|U|} \sum_{u \in U} \frac{1}{\text{rank}_u}$$

  where rank$_u$ is the rank position of the first relevant item for user u in the top-k list.

1.11.2 Online Evaluation (A/B Testing)

Online evaluation involves deploying different recommender algorithms in a live system and tracking real user behavior. Common online metrics include:

- Click-Through Rate (CTR)
- Conversion Rate
- Engagement metrics (e.g., time spent, number of items viewed)
- Revenue impact

This method provides realistic feedback from actual users, but is more resource-intensive.

1.11.3 User Studies

Controlled user studies involve a limited number of participants to collect qualitative feedback, including:

- User satisfaction
- Perceived diversity
- Perceived novelty
- Overall preference

These studies provide valuable insights that complement quantitative evaluations.

1.11.4 Additional Important Metrics

Besides accuracy metrics, other aspects should be considered [52]:

- **Coverage**: Proportion of items or users for which recommendations can be provided.
- **Diversity**: Degree to which recommended items differ from each other.
- **Novelty**: Ability to recommend items unfamiliar to the user.
- **Serendipity**: Ability to recommend surprising yet useful items.

Shani and Gunawardana explain that prediction accuracy alone is not sufficient to assess the quality of a recommender system. It is also essential to consider additional properties such as diversity, novelty, and robustness to obtain a more comprehensive evaluation of system performance [52].

1.12 Conclusion

Recommendation systems have become increasingly present in many domains, designed to help users discover resources that match their interests and preferences among an overwhelming number of available options. We followed a classification that distinguishes two main approaches: content-based recommendation and collaborative filtering. In addition, we presented other approaches such as demographic, knowledge-based, and social recommendations. This simple classification allowed us to group several techniques that are often presented separately. Since content-based and collaborative filtering approaches offer complementary strengths, many studies have focused on hybrid models that combine multiple techniques. In the next chapter, we will introduce a subfield of artificial intelligence (machine learning) which enables systems to learn from data. We will explore its main algorithms in detail.

Chapter 2

Machine Learning and Recommendation Systems

Chapter 2

Machine Learning and Recommendation Systems

2.1 Introduction

Machine learning and deep learning are at the core of today's artificial intelligence advancements, providing systems with the ability to learn from data and improve their performance without explicit programming. While machine learning focuses on building models that can recognize patterns and make predictions based on structured data, deep learning is a specialized branch of machine learning, uses neural networks to automatically learn complex representations from large and often unstructured datasets. These technologies have revolutionized many fields, especially recommendation systems, where understanding and anticipating user preferences is essential . This chapter begins with an overview of machine learning fundamentals, covering essential concepts and types of learning. We then examine classical algorithms used in recommendation systems, followed by matrix factorization and hybrid techniques. After introducing deep learning principles, we explore neural approaches for recommendation. The chapter concludes with a look at real-world applications of recommendation systems.

2.2 Foundations of Machine Learning

2.2.1 Definition and Key Concepts

Machine learning is a data analysis method that allows computers to learn from data and build models for predictions, with the main principles being automatic learning and recognition of patterns[55]. It has become central to many modern technologies and industries, driven by advances in algorithms, data availability, and computational power.

Machine learning systems build models from data, either by learning features directly from examples or by using handcrafted features, and improve performance with experience [56], the relationship between machine learning and datasets is complex, involving opportunities for improved predictions as well as challenges related to data quality, bias, and scalability

High-quality datasets are essential for building machine learning models that are accurate, efficient, and generalizable. Key dimensions of dataset quality, such as completeness, accuracy, consistency, and relevance, are assessed throughout the dataset's lifecycle. These factors ensure the data is reliable and can deliver trustworthy results. Additionally, features like dataset size, attribute count, class balance, and statistical properties including mean, standard deviation, skewness, and kurtosis, can significantly influence the performance of machine learning algorithms. For example, high kurtosis can reduce the accuracy of certain non-tree-based algorithms[57]. To maintain high standards, scientific machine learning increasingly depends on curated, open datasets and benchmarking suites. These tools enable performance comparisons between algorithms and help ensure reproducibility and consistency across studies .

Each dataset is typically composed of features and labels. Forming the basis for how models learn to make predictions. Features represent the input data, while labels are the target outputs the model aims to predict. The relationship between features and labels is especially important in complex tasks like multi-label learning, where each instance can have multiple associated labels[58].

A features are measurable properties or characteristics of the data used as input for the model. In multi-label and label distribution learning, features can be general that shared across all labels or label-specific ,unique to each label. Labels are the outputs or categories the model is trained to predict. In multi-label learning, each instance can have multiple labels, and the relationship between features and each label can vary. So tailoring features to individual labels and carefully selecting relevant features can significantly enhance model performance, especially in complex, multi-label, or noisy data environments[59].

Machine learning models are typically trained on a portion of the available data (training set) and evaluated on a separate portion (testing set). The ratio between training and testing data is crucial; too much training data can lead to overfitting, while too little can cause underfitting. Common split ratios range from 60:40 to 95:05, and the optimal choice depends on the specific application and resource constraints[60]. After splitting the data, models are trained using the training set. This involves adjusting model parameters to minimize errors on the training data. Once trained, the model is tested on the unseen testing set to assess its ability to generalize to new data.

2.2.2 Types of Machine Learning

- **Supervised Learning**: Supervised learning entails learning a mapping between a set of input variables X and an output variable Y and applying this mapping to predict the outputs for unseen data [61]. The model learns to map inputs to outputs, allowing it to predict labels for new data. The presence of labeled data acts as a "teacher" guiding the learning process. In practical applications, supervised learning has proven to be highly efficient. Almost all domains, both text and Web domains, employ it. In machine learning, supervised learning is often referred to as classification or inductive learning [62]. This kind of learning is comparable to how humans learn from the past to acquire new information that enhances our capacity to perform out activities in the real world. However, machine learning learns from data that has been collected in the past and represents prior experiences in some real-world applications, as machines lack "experiences". Common supervised learn-

ing algorithms include linear regression, support vector machines, random forests, naïve Bayes, decision trees, and k-nearest neighbors (KNN) [63] .

- **Unsupervised Learning** : Unsupervised learning refers to a learning approach where no teacher or explicit guidance is provided; instead, models work with unlabeled data to discover patterns, groupings, or hidden structures within the data. Unlike supervised learning, the model attempts to learn the relationships among features without mapping inputs to predefined outputs. Self-organizing neural networks, for example, use unsupervised learning algorithms to identify hidden patterns in unlabeled input data, without relying on an external error signal to evaluate potential solutions [64]. In general, unsupervised learning focuses on how systems can learn to represent input patterns in ways that reflect the statistical structure of the overall dataset. This form of learning is considered particularly important, as it is believed to occur more frequently in the brain compared to supervised learning. Common algorithms used in unsupervised learning include clustering methods such as K-Means, Hierarchical Clustering, DBSCAN, and Gaussian Mixture Models (GMM).

- **Reinforcement Learning (RL)**: Constitutes a distinct branch of machine learning, wherein agents are trained to make sequences of decisions through direct interaction with a dynamic environment. Unlike supervised learning, RL does not rely on predefined labeled input-output pairs. Central principles of RL include trial-and-error learning, enabling agents to iteratively explore various actions to maximize cumulative rewards, and autonomous policy discovery, wherein effective strategies are learned without the need for explicit instruction. Fundamental components of reinforcement learning frameworks include the agent, environment, state, action, and reward, which collectively define the learning cycle[65]. A critical aspect of RL is managing the trade-off between exploration and seeking new actions to potentially discover better rewards,and exploitation leveraging known actions to maximize returns. RL encompasses a variety of algorithmic approaches, notably model-free methods such as Q-learning and policy gradient techniques, model-based strategies, and deep reinforcement learning (DRL), which integrates deep neural networks to address complex, high-dimensional tasks. Reinforcement learning has demonstrated remarkable success across diverse application domains, including gaming, robotic control, healthcare optimization, educational personalization, and financial decision-making. Moreover, RL is conceptually rooted in biological learning systems, with significant parallels drawn between its mechanisms and the neural processes underpinning reward-driven behavior, particularly involving the brain's dopamine pathways. Collectively, reinforcement learning offers a robust framework for enabling autonomous systems to develop optimal policies for complex, sequential decision-making challenges.

2.2.3 Supervised vs. Unsupervised Learning in Recommender Systems

a) Supervised learning in recommender system

Within the context of recommender systems, supervised learning is instrumental in modeling user preferences from historical interaction data. Here, the learning process is based

on input-output pairs, where the input could be a user's behavioral data, demographic information, or item features, and the output is a preference label such as a rating or binary feedback like "liked" or "not liked" [66]. This framework is prevalent in personalized services like movie recommendations, online retail suggestions, and music platforms [67]. The strength of supervised learning in this domain lies in its ability to offer highly accurate and personalized recommendations when large labeled datasets are available. Performance can be systematically evaluated using metrics such as accuracy, precision, recall, and RMSE [68]. However, supervised learning in recommendation systems has notable limitations. Data sparsity is a major challenge, as users often interact with or rate only a few items, making it difficult for the model to learn accurate patterns. This can lead to poor performance when data is scarce. Overfitting is another issue, where the model becomes too specialized to the training data and fails to generalize to new users or items. Additionally, supervised learning can be resource-intensive, requiring large, high-quality labeled datasets that are costly and difficult to obtain at scale. These drawbacks underscore the challenges of relying solely on supervised learning in dynamic real-world recommendation systems.

b) Unsupervised learning in recommender system

Unsupervised learning is a fundamental approach in recommendation systems, enabling the discovery of hidden patterns in unlabeled data without relying on explicit user feedback [65]. While supervised methods depend on predefined labels, unsupervised techniques such as clustering as k-means and dimensionality reduction as PCA, matrix factorization identify inherent similarities among users or items [69]. Collaborative filtering, for example, groups users with comparable preferences or recommends items based on co-occurrence patterns, making it particularly useful in cold-start scenarios or domains with sparse explicit feedback. A key strength of unsupervised learning is its scalability, as it can process large datasets without requiring labeled data[70]. Methods like association rule mining (e.g., Apriori algorithm) uncover non-obvious relationships such as "users who bought X also bought Y" improving serendipitous recommendations [71]. However, challenges arise due to the lack of ground-truth labels, complicating evaluation and often necessitating proxy metrics like silhouette score or offline experiments. Additionally, the interpretability of certain models like deep autoencoders can be limited, posing challenges for real-world applications. Despite these drawbacks, unsupervised learning remains critical for exploratory analysis and enhancing hybrid recommendation systems [72].

2.3 Classical Machine Learning Algorithms for Recommendation

2.3.1 k-Nearest Neighbors (kNN)

The k-Nearest Neighbor (kNN) algorithm is a non-parametric supervised machine learning method used for both classification and regression. It determines the output for a given instance based on the k closest training examples in the feature space. Originally

introduced by Cover and Hart [73], the kNN algorithm is valued for its simplicity, robustness, and interpretability. In recommender systems, kNN is particularly effective due to its ability to capture co-occurrence information of items, especially when interaction timestamps are considered, making it well-suited for session-based recommendation scenarios [74]. In user-based collaborative filtering, kNN is used to identify users with similar preferences to the target user, and recommendations are made by aggregating the preferences of these neighbors. Conversely, item-based collaborative filtering applies kNN to locate items similar to those the user has previously engaged with, recommending those that closely match the user's historical preferences.

The overall algorithm of the KNN algorithm is as follows [75] :

Algorithm 1 k-Nearest-Neighbor Algorithm

1: **Input:** Training dataset
2: Initialize k to your chosen number of neighbors
3: **for** each example in the data **do**
4: Calculate the distance between the query example and the current example from the data.
5: Add the distance and the index of the example to an ordered collection
6: **end for**
7: Sort the ordered collection of distances and indices from smallest to largest (in ascending order) by the distances
8: Pick the first k entries from the sorted collection
9: Get the labels of the selected k entries
10: **if** regression **then**
11: Return the mean of the k labels
12: **else if** classification **then**
13: Return the mode of the k labels
14: **end if**

2.3.2 Linear and Logistic Regression

Linear Regression and Logistic Regression are foundational algorithms in recommendation systems, each serving distinct purposes. Linear Regression models the relationship between a dependent variable (e.g., user ratings) and independent variables (e.g., user demographics, item attributes) to predict numerical outcomes, making it useful for estimating how a user might rate an unseen item. For instance, it has been shown to effectively model the relationship between user characteristics and movie ratings using a linear model trained on past behavior and demographic data [76]. Its simplicity, interpretability, and efficiency with large datasets make it a popular choice for personalized scoring. On the other hand, Logistic Regression excels in binary classification tasks (e.g., predicting "like" vs. "dislike") by analyzing user preferences, item features, and past interactions to determine the probability of a positive response. It is particularly effective in predicting click-through rates in e-commerce systems, providing probability-based outputs using the sigmoid function[77].Moreover, it can incorporate contextual features such as time of day or browsing history to enhance the personalization of recommendations, allowing systems to deliver more context-aware suggestions[78]. Both methods are computationally efficient, easy to implement, and serve as strong baselines before exploring

more complex models, making them valuable tools in building scalable and explainable recommendation systems.

2.3.3 Decision Trees and Random Forests

Decision Trees are simple and interpretable models that work by recursively splitting the data based on feature values to make predictions. Their ease of understanding and visualization makes them highly attractive; however, they can be prone to overfitting, especially when dealing with complex or noisy data. Decision trees use the strategy of 'divide-and-conquer' to divide the complex problem on the dependency between input features and labels into smaller ones [79]. On the other hand, Random Forests are an ensemble method built from multiple decision trees using bagging (bootstrap aggregation). By combining the outputs of many decision trees, Random Forests improve accuracy and robustness over single decision trees by reducing overfitting and increasing generalization [80]. They are particularly effective in handling high-dimensional data, while also identifying the most important features for making recommendations. Both Decision Trees and Random Forests are widely applied in Recommender Systems, particularly for classification and regression tasks, such as predicting user preferences or item ratings. However, Random Forests are often preferred due to their higher accuracy and ability to efficiently process large, complex datasets, making them a valuable tool for personalized recommendations.

2.3.4 Support Vector Machines and Naive Bayes

Support Vector Machines (SVMs) and Naive Bayes are two widely utilized algorithms in recommendation systems, each offering distinct advantages based on the task at hand. SVMs are particularly effective for classification tasks involving high-dimensional or sparse data. They operate by identifying the optimal hyperplane that best separates different classes, making them suitable for personalized recommendations and ranking. For instance, SVMs have been employed in content-based filtering to classify items based on features, and in collaborative filtering to predict user preferences by analyzing patterns in user-item interactions. Real-world applications include platforms like Netflix and Amazon, which utilize SVMs to analyze user behavior and provide tailored suggestions [81].Despite their high accuracy, SVMs can be less interpretable compared to decision trees, and their performance is highly dependent on proper parameter tuning. Additionally, they can be computationally intensive, especially with large datasets, necessitating techniques like Stochastic Gradient Descent and mini-batch training to improve scalability [81].On the other hand, Naive Bayes is a probabilistic classifier based on Bayes' theorem, assuming feature independence. It is known for its simplicity and computational efficiency,making it particularly effective for content-based filtering and text classification tasks. Naive Bayes classifiers have demonstrated dependable results in text classification, even when the independence assumption is somewhat violated [82].Moreover, integrating collaborative filtering into the Naive Bayes model has shown promising results in enhancing prediction precision by leveraging both user behavior analysis and probabilistic modeling [83].However, Naive Bayes may underperform when there are strong dependencies among features, and it can struggle with the "zero-frequency" problem, where unseen features in the training data lead to zero probability estimates [84].

2.3.5 Gradient Boosting Techniques

Gradient Boosting is a powerful ensemble learning technique widely applied in recommendation systems to improve predictive accuracy by sequentially combining multiple weak learners, typically decision trees. The core idea is to build models iteratively, where each new model attempts to correct the errors made by the previous ones, effectively minimizing the loss function through gradient descent optimization. This approach allows recommendation systems to capture complex user-item interactions and nonlinear relationships, making it especially useful for tasks such as rating prediction and personalized ranking. Gradient Boosting methods, such as XGBoost, LightGBM, and CatBoost, have demonstrated superior performance in large-scale recommendation scenarios due to their scalability and ability to handle sparse and high-dimensional data. Their flexibility in optimizing various loss functions also makes them adaptable to diverse recommendation objectives. These characteristics have contributed to the growing adoption of Gradient Boosting techniques in modern recommendation engines, as documented in recent machine learning literature [85, 86].

2.4 Matrix Factorization and Hybrid Methods

Matrix factorization techniques have emerged as a powerful paradigm for collaborative filtering, with methods like Singular Value Decomposition (SVD) and Non-negative Matrix Factorization (NMF) excelling at uncovering latent user-item relationships. These techniques have demonstrated particular success in practical applications such as restaurant recommendation systems, where they effectively model complex interaction patterns to improve consumer decision-making [87]. Comparative studies reveal SVD's advantages in both computational efficiency and recommendation accuracy over alternative decomposition approaches like NMF-ALS and CUR [88]. The versatility of matrix factorization enables its integration with various machine learning techniques - notably through hybrid systems combining factorization outputs with clustering algorithms (e.g., K-Means) [89]. In such architectures, the latent features generated by matrix factorization serve as enriched input for supervised learners (decision trees, gradient boosting, or neural networks), creating systems that simultaneously leverage interaction data and auxiliary information. This combined approach effectively mitigates persistent challenges including data sparsity, cold-start scenarios, and contextual limitations, ultimately yielding more robust and adaptable recommendations across diverse domains.

2.5 Introduction to Deep Learning

One of the key strengths of deep learning lies in its capacity to model complex and nonlinear relationships within data, enabling it to uncover subtle patterns that traditional methods often miss. This is particularly important in recommendation systems, where user behavior is dynamic and influenced by a variety of contextual and temporal factors. Deep learning models, such as autoencoders and attention-based networks, have proven to be highly effective in learning latent user and item representations, which enhance personalization and prediction accuracy [90]. Their ability to process diverse input types

— from user ratings to textual content and images — further contributes to their growing adoption in modern recommender systems.

2.5.1 Deep Learning vs. Classical ML

Deep learning and classical machine learning are both pivotal in the field of artificial intelligence, but they differ substantially in their techniques and applications. Classical machine learning methods, such as decision trees, support vector machines, and linear regression, rely heavily on manual feature extraction, where domain knowledge is crucial for creating the features that the models use to learn. These models perform well with structured data and are typically more suitable for problems with smaller datasets [91]. In contrast, deep learning models use artificial neural networks with multiple layers to automatically extract features from raw, unstructured data. This capability allows deep learning to excel in tasks such as image recognition, natural language processing, and recommendation systems, where the data is complex and large-scale [92].

While classical machine learning models are generally faster and more interpretable, deep learning models require larger datasets and significant computational resources to train effectively [93]. Deep learning's ability to uncover intricate patterns in vast amounts of data makes it more suitable for complex problems with high-dimensional data (e.g., images, audio, and text) [92]. However, classical machine learning approaches still hold relevance for simpler problems where the data is structured and interpretable [91].

In summary, while classical machine learning methods are ideal for smaller, structured datasets, deep learning shines in handling complex, high-dimensional data and tasks that require the learning of non-linear representations, making it a key player in modern AI-driven recommendation systems and other advanced applications.

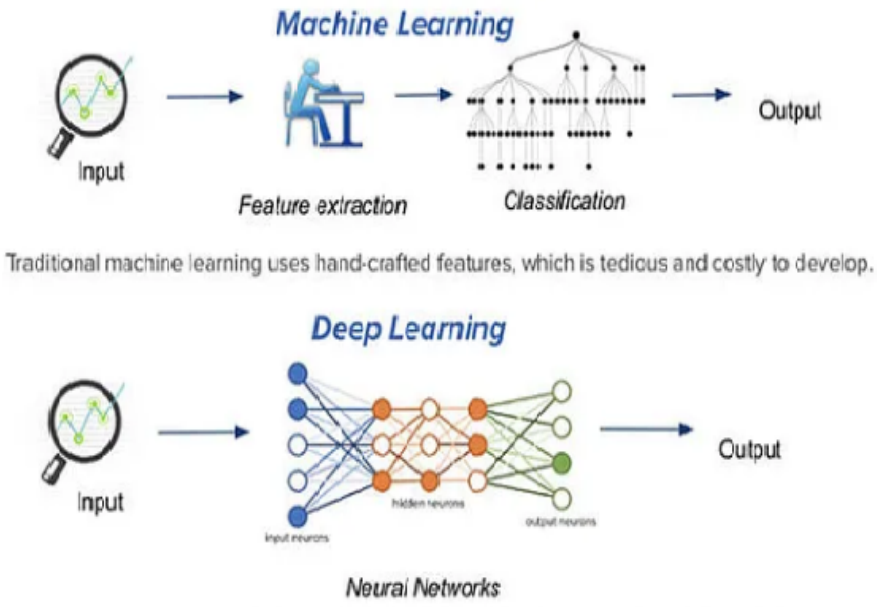


Figure 2.1: Machine Learning vs. Deep Learning

2.5.2 Neural Network Basics

A neural network is a computational model inspired by the structure and function of biological neural networks in the human brain. It consists of layers of interconnected nodes, or neurons, that process information. The network's architecture typically involves three types of layers: the input layer, hidden layers, and the output layer.

- **Architecture:** The architecture of a neural network includes multiple layers. Each layer is made up of neurons that are connected to each other by weighted edges. These connections are responsible for transmitting the signal from one neuron to the next. Deep neural networks consist of many hidden layers, enabling them to learn complex features of the data [93].
- **Neurons:** Neurons are the basic units of a neural network. Each neuron receives inputs, processes them through an activation function, and passes the result to the next layer. The activation function introduces non-linearity, allowing the network to learn complex patterns [92].
- **Weights:** Weights are the parameters in the network that control the strength of the connection between neurons. These weights are adjusted during training using optimization algorithms like gradient descent, which aim to minimize the error in predictions [93].
- **Layers:** Neural networks consist of several layers:
    - *Input Layer:* The first layer that receives the input data.
    - *Hidden Layers:* Layers between the input and output layers where computation takes place, enabling the network to learn representations of the data [91].
    - *Output Layer:* The final layer that generates the network's output.

Neural networks learn by adjusting the weights during training using a process called backpropagation, where the network computes the error between its predictions and the true output. This error is propagated backward through the network, and the weights are updated accordingly to reduce the error [92].

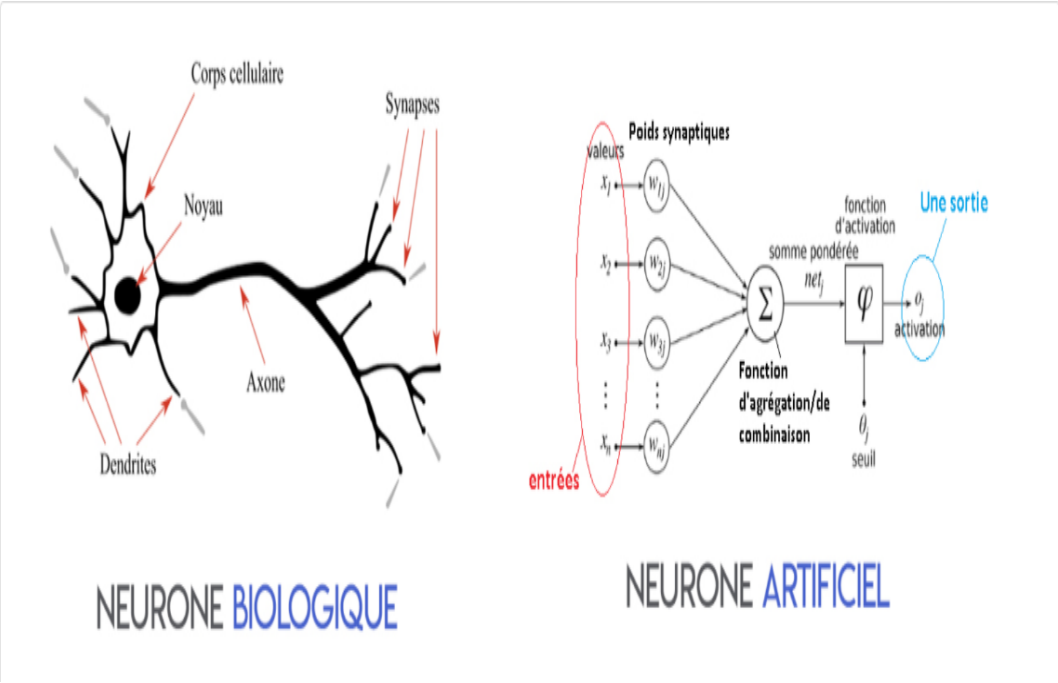


Figure 2.2: Comparison Between a Biological Neuron and an Artificial Neuron

2.5.3 Hyperparameters and Activation Functions

Hyperparameters are crucial settings that govern the training behavior of deep learning models. Unlike parameters (e.g., weights), hyperparameters are not learned during training but must be chosen before the process begins [93]. Below are the most commonly used hyperparameters with their definitions:

- **Learning Rate:** Defines the step size at each iteration while moving toward a minimum of the loss function. A smaller learning rate can lead to more precise convergence but slower training, while a large one may overshoot the optimal value [93].
- **Batch Size:** Refers to the number of training samples used in one forward/backward pass. Small batch sizes lead to faster updates but noisier gradients, while large ones are more stable but computationally expensive [94].
- **Number of Epochs:** The number of complete passes through the training dataset. More epochs increase training duration and risk overfitting [91].
- **Optimizer:** Determines how the model's weights are updated during training. Popular choices include Stochastic Gradient Descent (SGD), Adam, and RMSprop [95].
- **Dropout Rate:** A regularization technique where a fraction of neurons are randomly ignored during training to reduce overfitting [96].
- **Weight Initialization:** Strategy to initialize model weights before training begins. Techniques like Xavier [97] and He initialization [98] help stabilize learning.

- **Activation Functions:** Functions like ReLU, Sigmoid, and Tanh add non-linearity to neural networks, making them capable of learning complex patterns [99].
- **Momentum:** A method to accelerate gradient descent by considering previous gradients in the update rule, which helps overcome local minima [100].
- **Regularization Parameters:** Includes L1 and L2 penalties that discourage large weights, aiding in generalization [101].
- **Learning Rate Schedulers:** Techniques that adjust the learning rate during training, such as exponential decay, step decay, or reduce-on-plateau [93].
- **Gradient Clipping:** Prevents exploding gradients by capping the maximum value of gradients during backpropagation [102].
- **Early Stopping:** Stops training when the model performance on the validation set stops improving, preventing overfitting [103].

Activation functions introduce non-linearity into neural networks, enabling them to learn complex patterns in data. Without them, a neural network would behave like a linear regression model, regardless of the number of layers [93].

Below are the most commonly used activation functions in deep learning:

- **Sigmoid:** Maps input values to a range between 0 and 1. Often used in binary classification, but suffers from vanishing gradients [104].

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

- **Tanh (Hyperbolic Tangent):** Outputs values between -1 and 1, centered around zero. It generally performs better than sigmoid but still suffers from vanishing gradients [105].

$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

- **ReLU (Rectified Linear Unit):** The most commonly used activation function in modern deep networks. It outputs the input directly if it is positive; otherwise, it returns zero [99].

$$f(x) = \max(0, x)$$

- **Leaky ReLU:** A variant of ReLU that allows a small, non-zero gradient when the input is negative, helping mitigate the "dying ReLU" problem [106].

$$f(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha x & \text{if } x \leq 0 \end{cases}$$

  where α is a small constant.

- **Softmax:** Used primarily in the output layer for multi-class classification, it converts a vector of values into a probability distribution [107].

$$\text{softmax}(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}}$$

Choosing the right hyperparameters and activation functions is essential for training an effective neural network. Different problems and architectures may require fine-tuning to achieve the best performance.

2.5.4 Common Neural Network Architectures

Over the years, various neural network architectures have been developed to address specific tasks in machine learning, particularly in domains such as image recognition, natural language processing (NLP), and time series forecasting. Each architecture—such as Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), and Transformer-based models—offers distinct advantages depending on the characteristics of the data and the complexity of the task [108].

- **Multilayer Perceptron (MLP):**

A Multilayer Perceptron is a type of feedforward artificial neural network that consists of an input layer, one or more hidden layers, and an output layer. MLPs are primarily used for classification and regression tasks on structured data. They are capable of modeling complex relationships by learning nonlinear functions through backpropagation [109]. MLPs have been applied in various domains, including image recognition, natural language processing, and recommendation systems.

- **Convolutional Neural Networks (CNNs):**

CNNs are designed to process data with a grid-like topology, such as images. They utilize convolutional layers to automatically and adaptively learn spatial hierarchies of features from input data [110]. CNNs have revolutionized computer vision tasks, including image classification, object detection, and medical image analysis. Their architecture allows for efficient processing of high-dimensional data by reducing the number of parameters and computation required.

- **Recurrent Neural Networks (RNNs):**

RNNs are specialized for processing sequential data by maintaining a hidden state that captures information about previous inputs. This makes them suitable for tasks like time series prediction, language modeling, and speech recognition [111]. However, traditional RNNs face challenges with long-term dependencies due to issues like vanishing gradients.

- **Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRU):**

LSTM and GRU are advanced RNN architectures designed to overcome the limitations of traditional RNNs in capturing long-term dependencies. LSTMs introduce memory cells and gating mechanisms to regulate the flow of information, enabling them to retain information over extended sequences [112]. GRUs simplify the LSTM architecture by

combining certain gates, resulting in fewer parameters while maintaining performance. Both architectures have been effective in applications like language translation, speech recognition, and time series forecasting.

- **Transformer Networks:**

Transformers are a type of neural network architecture that relies on self-attention mechanisms to process sequential data. Unlike RNNs, transformers can process entire sequences in parallel, making them highly efficient for large-scale language tasks [113]. They have become the foundation for many state-of-the-art models, such as BERT, BERT4REC and GPT, and have significantly advanced the field of NLP.

The key strengths and limitations of these architectures are summarized in the table below:

Table 2.1: Advantages and disadvantages of common neural network architectures [6].

Method	**Advantages**	**Disadvantages**
MLP (Multilayer Perceptron)	- Simple architecture - Works well with tabular or vectorized data	- Cannot model spatial or temporal dependencies - Limited performance on complex data (images, sequences)
CNN (Convolutional Neural Network)	- Excellent for image processing - Fewer parameters due to weight sharing - Efficient local feature extraction	- Less suitable for sequential or temporal data - Struggles with global dependencies
RNN (Recurrent Neural Network)	- Designed for sequential data - Can model short-term temporal dependencies	- Suffers from vanishing/-exploding gradient problems - Poor at capturing long-term dependencies
LSTM (Long Short-Term Memory)	- Handles long-term dependencies - Less affected by gradient problems	- More complex architecture - Slower training

Method	Advantages	Disadvantages
GRU (Gated Recurrent Unit)	- Simpler than LSTM - Comparable performance - Faster convergence	- Less expressive in some cases
Transformer	- Captures long-range dependencies effectively - Enables parallel training - Superior performance in NLP tasks	- Requires large datasets - High computational cost

2.6 Neural Approaches to Recommendation

Neural networks have revolutionized recommendation systems by enabling models to capture complex, non-linear relationships between users and items. Compared to traditional techniques, neural approaches provide greater flexibility and capacity to learn intricate user preferences from large and heterogeneous datasets [114]. In this section, we review key neural-based methods applied to recommendation tasks.

2.6.1 Autoencoders for Collaborative Filtering

Autoencoders have been effectively applied to collaborative filtering tasks by learning compact, non-linear representations of user-item interaction matrices. In contrast to traditional matrix factorization, autoencoder-based approaches focus on reconstructing user preferences by encoding observed interactions into a latent space and then decoding them to predict missing values. According to [115], the use of autoencoders offers enhanced flexibility and modeling capacity compared to linear methods, enabling better generalization especially in highly sparse environments. Their model demonstrated that even shallow autoencoder structures could significantly outperform standard baselines in terms of recommendation accuracy, illustrating the power of deep learning for collaborative filtering tasks.

2.6.2 Neural Collaborative Filtering

Neural Collaborative Filtering (NCF) represents a modern paradigm that leverages deep neural networks to model user-item interactions for recommendation tasks. Instead of relying on the traditional dot product used in matrix factorization, NCF learns the interaction function directly from data through multi-layer perceptrons, allowing for greater modeling flexibility and non-linearity. As demonstrated by He et al. [3], the NCF framework generalizes the classic collaborative filtering models and can capture complex patterns of user preferences, leading to substantial improvements in recommendation accu-

racy. Their proposed model, NeuMF, effectively combines the linearity of matrix factorization and the non-linearity of neural networks, showing state-of-the-art performance across various benchmark datasets.

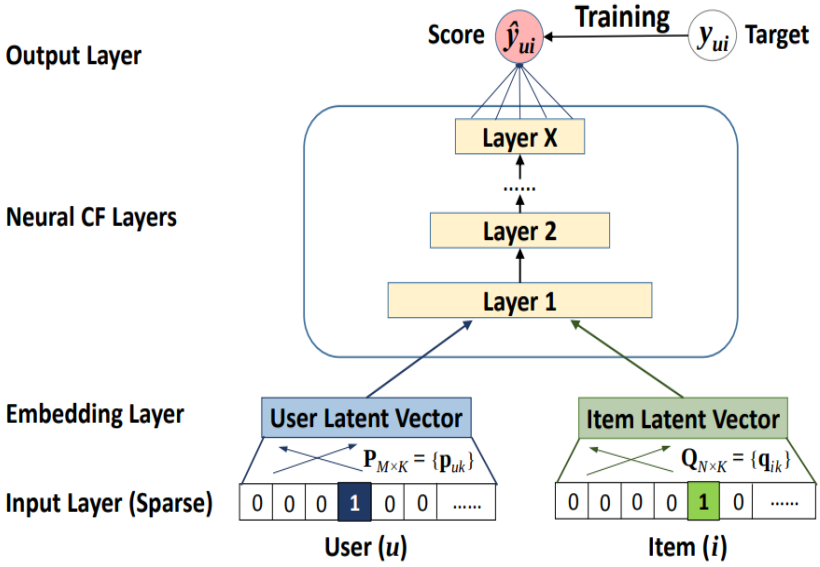


Figure 2.3: Neural collaborative filtering framework [3].

2.6.3 Text-Based Recommendation using Word2Vec / BERT

Text-based recommendation systems have evolved significantly with the integration of advanced language models such as Word2Vec and BERT, allowing for richer semantic understanding of user preferences and item descriptions. Traditional content-based recommenders often struggle with sparse or ambiguous textual data, which limits the accuracy and diversity of recommendations. To address these challenges, recent approaches leverage pre-trained embeddings to generate meaningful item and user representations. For instance, the *RecoBERT* model introduced by Xiang et al. combines BERT-based contextual embeddings with catalog metadata to deliver improved personalization and ranking performance in domains like fashion and wine recommendations. This model demonstrates that the contextual depth captured by BERT can significantly enhance the relevance and explainability of suggestions compared to classical bag-of-words or shallow embedding models like Word2Vec, particularly in data-rich, text-heavy environments [116].

2.6.4 Semantic Matching with Sentence-BERT (SBERT)

Sentence-BERT (SBERT) is a modification of the original BERT architecture designed to generate semantically meaningful sentence embeddings that can be compared using cosine similarity. While BERT produces powerful contextual token embeddings, it is not optimized for sentence-level similarity tasks without additional computation. To overcome this limitation, Reimers and Gurevych proposed SBERT, which employs a Siamese or triplet network structure to derive fixed-size vector representations of sentences, enabling efficient and accurate similarity comparisons in downstream applications. In the

context of recommender systems, SBERT has been successfully used to encode user preferences and item descriptions into a shared semantic space, facilitating content-based filtering and re-ranking in scenarios where rich textual metadata is available. For example, SBERT has shown significant improvements in retrieving semantically relevant items in domains like job recommendations and personalized travel suggestions, where matching user queries with detailed descriptions is essential. Its ability to capture fine-grained semantic similarity makes it particularly effective for cold-start problems and recommendations based on sparse interaction data [117].

2.6.5 Sequential Recommendation with BERT4Rec

Sequential recommendation systems are critical for capturing dynamic user preferences, moving beyond static preferences to consider the temporal order of interactions. Among the notable advancements in this field, BERT4Rec, introduced by Sun et al, stands out by adapting the Bidirectional Encoder Representations from Transformers (BERT) architecture to sequential recommendation. Unlike traditional unidirectional models, BERT4Rec leverages a deep bidirectional self-attention network, allowing each item in a user's interaction history to fuse information from both preceding and succeeding items. To mitigate information leakage and enable efficient training of this bidirectional model, BERT4Rec employs the Cloze task, predicting randomly masked items within a sequence based on their surrounding context. This innovative approach has demonstrated superior performance compared to various state-of-the-art sequential models across multiple benchmark datasets, proving its effectiveness in modeling complex user behavior dynamics for more accurate next-item recommendations[118].

2.6.6 Advantages and Challenges of Deep Learning in Recommender Systems

Advantages:

- **Feature Learning Capability:** Deep learning models can automatically learn complex and non-linear feature representations from raw data, without needing manual feature engineering [90].
- **Handling Heterogeneous Data:** Deep models effectively combine different types of data sources (text, images, user behavior) to improve recommendation performance [90].
- **Scalability to Large Datasets:** Deep learning techniques can scale well to large-scale recommendation problems due to their hierarchical structure and parallel computation ability [90].
- **Personalization Improvement:** Deep models provide better user personalization by capturing intricate user-item interaction patterns [119].

Challenges:

- **High Computational Cost:** Training deep learning models requires substantial computational resources and time [90].
- **Risk of Overfitting:** Due to the large number of parameters, deep models are prone to overfitting, especially on small datasets [90].
- **Difficulty in Interpretability:** Deep learning models are often seen as "black boxes," making it hard to explain the recommendation results [119].
- **Need for Large Amounts of Labeled Data:** Deep learning methods typically require extensive labeled datasets, which may not always be available [119].

2.7 Applications of Recommendation Systems in Popular Platforms

Recommendation systems are widely applied across various platforms to enhance user experience and engagement. Notable examples include:

- **Netflix:** Uses a combination of content-based filtering and collaborative filtering techniques to recommend movies and TV shows based on viewing history and user preferences [120].
- **YouTube:** Recommends videos by analyzing user activity, past watch history, and related content, employing hybrid models that combine collaborative and content-based methods [121].
- **Spotify:** Utilizes collaborative filtering and content-based methods to recommend songs, albums, and playlists, analyzing listening habits and preferences [122].
- **Amazon:** Employs collaborative filtering algorithms based on past purchase behavior and product ratings to recommend products [9].
- **Apple Music:** Similar to Spotify, applies collaborative and content-based approaches to suggest personalized music based on user preferences [123].

These platforms exemplify how recommendation systems enhance user satisfaction by providing personalized content, leveraging a variety of algorithms and data sources.

2.8 Conclusion

In this chapter, we explored the foundations of machine learning and deep learning, emphasizing their crucial role in recommendation systems. We presented the main types of learning, including supervised, unsupervised, and reinforcement learning, as well as classical algorithms commonly used in recommendation systems such as collaborative filtering and content-based approaches.

We also examined advanced methods like matrix factorization and hybrid systems that combine multiple information sources to improve recommendation quality. Finally, we introduced the fundamental concepts of deep learning and its growing application in modern recommendation systems, especially through deep neural networks, autoencoders, and attention-based models.

These advancements offer promising opportunities to develop more personalized, adaptive, and effective recommendation systems. The next chapter will focus on community detection in social networks, another key area for enhancing recommendation relevance at large scale.

Chapter 3

Development of the Proposed Hybrid Approach

Chapter 3

Development of the Proposed Hybrid Approach

3.1 Introduction

In today's digital world, recommendation systems have become essential tools for simplifying user experiences. Whether on websites or mobile applications, they help users discover content or services that match their preferences, without the need for extensive searching. Among the many domains where recommendation systems play a vital role, travel stands out as a particularly dynamic and impactful field. Travelers increasingly rely on personalized suggestions to plan trips that suit their tastes, interests, and previous behaviors.

In this chapter, we focus on designing a hybrid recommendation system specifically tailored to the travel domain. Our goal is to recommend complete travel programs to users, based on their historical data and interaction patterns. We begin by describing the preprocessing steps applied to our datasets, followed by a detailed presentation of the two complementary algorithms that we have combined into a hybrid approach. Finally, we discuss the evaluation metrics used to assess the performance of our model and compare it with baseline methods.

3.2 Data Collection and Description

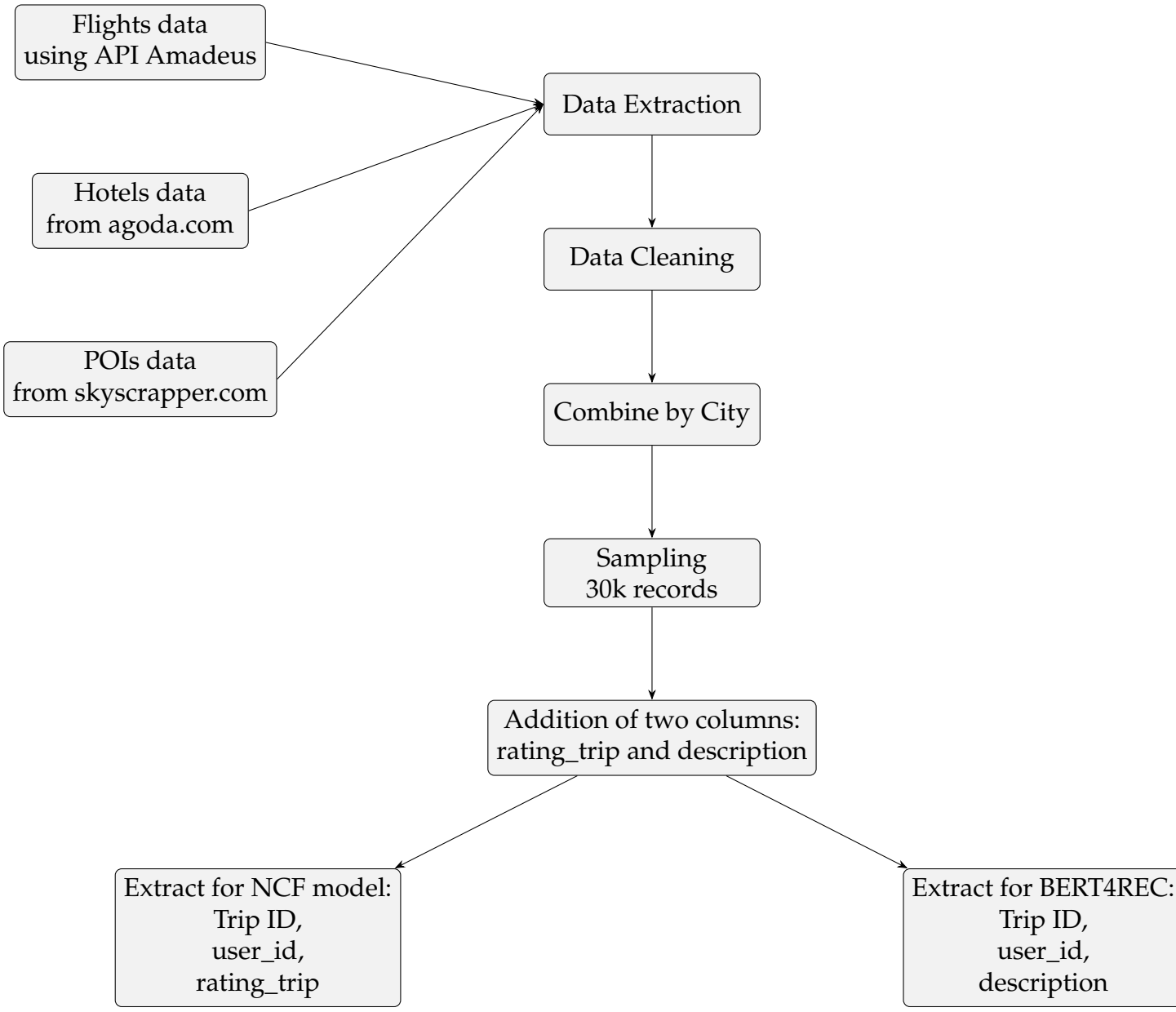


Figure 3.1: Data collection and processing workflow for the travel recommendation system

3.2.1 Data extraction

To build the foundation of our recommendation system, we began by collecting structured data on hotels, points of interest (POIs) and flights from reliable external sources.

For hotel data, we used the Agoda API, it was retrieved using the Agoda API[1], to extract detailed information for major cities around the world. The process started by retrieving

[1]Agoda via RapidAPI. *Agoda API on RapidAPI*. Retrieved June 16, 2025, from https://rapidapi.com/ntd119/api/agoda-com/playground/apiendpoint_bf07a78d-9d8c-4f4e-bf56-3560fd9ec9c3

the unique identifiers (IDs) of hotels available in each city. Then, for every hotel ID, individual API calls were made to collect comprehensive information. Among the hundreds of attributes returned, we selected the most relevant features, such as the hotel's rating, price per night, room details, and booking availability. The Agoda API was chosen for its broad international coverage, the richness and reliability of the metadata it provides, and its well-structured and well-documented interface, which allowed for efficient large-scale data extraction.

In parallel, we collected data on Points of Interest (POIs) to enhance the contextual understanding of each destination. For this purpose, we utilized the SkyScapper API, also available via RapidAPI [2]. Initially, we obtained the unique city identifiers required to access POI information. Using these IDs, we queried the API to retrieve 367 POIs across the selected cities, receiving structured JSON responses. For each POI, we extracted key attributes such as the name, address, user reviews, and other relevant metadata. SkyScapper was selected due to its extensive travel-related data coverage, the quality and diversity of the POI information, and the accessibility of its API, which proved to be suitable for structured and automated data collection.

For flight data, we used the Amadeus API[3] to extract detailed information on flights connecting five countries and approximately 17 major cities worldwide. The process began by identifying available routes between cities, then for each route, API calls were made to collect comprehensive flight details. Among the many attributes returned, we selected the most relevant features such as departure city, arrival city, ticket price, travel class, flight duration, and airline carrier. The Amadeus API was chosen for its extensive global coverage, the accuracy and freshness of its real-time flight information, and its well-structured and thoroughly documented interface, which enabled efficient large-scale data extraction.

To better understand the structure and richness of the collected datasets, Table 1 summarizes the key metadata attributes extracted for Hotels and Points of Interest (POIs) and flights. These attributes provide essential information that supports personalized and context-aware travel recommendations.

[2]Skyscrapper via RapidAPI. *SkyScrapper API on RapidAPI*. Retrieved June 16, 2025, from https://rapidapi.com/ntd119/api/flights-sky/playground/apiendpoint_bf574b3e-6852-4c6b-a7b6-db170b901c68

[3]Amadeus for Developers. *Flight Offers Search API*. Retrieved June 16, 2025, from https://developers.amadeus.com/self-service/category/air/api-doc/flight-offers-search

Table 3.1: Key Metadata Attributes of Hotels, Points of Interest (POIs), and Flights

Dataset	**Metadata Attributes**
Hotels	**Hotel Name**: Name of the hotel **Latitude**: Geographic latitude coordinate **Longitude**: Geographic longitude coordinate **City**: City where the hotel is located **Location**: Specific area or neighborhood within the city **Rating**: Hotel's overall rating score **Reviews**: Number of reviews received by the hotel **Room Information**: Name of the cheapest available room **Price per Booking (USD)**: Price per booking in US dollars **Hotel Image**: URL of the hotel's main image
Points of Interest (POIs)	**City**: Name of the city where the POI is located **Name**: Name of the point of interest **Address**: Physical address or location details **Review Score**: Average user rating score **Review Count**: Number of user reviews **Distance from Hotel**: Distance to the nearest hotel (when available) **Latitude**: Geographic latitude coordinate **Longitude**: Geographic longitude coordinate **Image URL**: URL of image associated with the POI
Flights	**Departure**: Origin city or airport of the flight **Arrival**: Destination city or airport **Airline**: Operating airline of the flight **Flight Number**: Unique identifier composed of carrier code and number **Departure Time**: Scheduled time of flight departure **Arrival Time**: Scheduled time of arrival at destination **Price (€)**: Total cost of the flight in euros **Currency**: Currency used for pricing **Cabin Class**: Travel class (e.g., economy, business) **Available Seats**: Number of bookable seats **Duration**: Total travel duration **Aircraft Type**: Type(s) of aircraft used during the flight

3.2.2 Data Preprocessing

The three datasets—flights, hotels, and points of interest (POIs)—were merged based on the city, resulting in a comprehensive trips table. Each trip record begins with flight details, followed by hotel information, and concludes with the POIs to visit during the trip. During this merging process, redundant columns such as duplicate city names were removed, along with columns deemed unnecessary for the analysis, including latitude and longitude. After creating this complete trips dataset, we performed data preprocessing to ensure quality and consistency. Initially, the dataset contained over 250,000 records; we filtered out incomplete entries with missing or zero values in key attributes. From the cleaned data, a representative sample of 30,000 trips was randomly selected, covering multiple cities to ensure diversity. Additionally, we enriched the dataset by generating a trip description attribute based on the aggregated trip information, as well as including a trip rating column to support further analysis.Furthermore, we constructed user histories by aggregating trips chronologically for each user. To focus on meaningful behavioral patterns, we retained only users with more than four recorded trips in the dataset. The *trip rating* was computed using a weighted average of the hotel and point-of-interest (POI) ratings. Instead of assigning fixed weights, we adopted a flexible weighting scheme controlled by a parameter α, using the formula: :

$$\text{trip_rating} = \alpha \cdot \text{hotel_rating} + (1 - \alpha) \cdot \text{poi_rating}$$

This formulation allows us to adjust the relative influence of hotel and POI ratings in the overall score, depending on the context or experimental configuration. It proved effective in capturing the quality of the trip and modeling user preferences, thereby enhancing the accuracy of our recommendation system. After processing, the dataset was simplified by retaining only the essential attributes needed for the recommendation model and subsequent analysis.

Table 3.2: The Complete Trip Dataset after preprocessing

Dataset	**Metadata Attributs**
Flight Details	Departure City, Arrival City, Airline, Flight Number, Departure Time, Arrival Time, Price (€), Flight Duration, Number of Stops
Hotel Details	Hotel Name, Location, Rating, Number of Reviews, Price per Night , Hotel Image
Point of Interest (POI)	POI Name, Address, POI Review Score, POI Review Count, Distance from Hotel, Latitude, Longitude, POI Image
Trip Summary	Trip Description

Our final dataset consists of the following files, all containing data collected between March 1st, 2025 and March 2nd, 2025:

Table 3.3: dataset of the trips

File Name	**Number of Rows / Values**
Flights or travel programs	Covers all available connections within the selected date range, totaling 10,366 flights.
Points of Interest (POIs)	367 tourist attractions
Hotel reservations	2,219 entries
Complete trip dataset	Over 30,000 trip records
Trip ratings dataset	A total of over 31,357 trip ratings are included in the dataset.
User history dataset	Sequences of 7000 user interactions over time

	A	B	C	D
1	Trip ID	Departure Time	user_id	Timestamp
2	12215	2025-03-01 16:20:00	1	1740846000
3	9457	2025-03-01 17:30:00	1	1740850200
4	12644	2025-03-01 18:00:00	1	1740852000
5	23913	2025-03-01 18:50:00	1	1740855000
6	316	2025-03-01 17:25:00	2	1740849900
7	28447	2025-03-01 19:00:00	2	1740855600
8	15058	2025-03-01 19:30:00	2	1740857400
9	664	2025-03-01 22:20:00	2	1740867600
10	26313	2025-03-01 1:00:00	3	1740790800

Figure 3.2: Example of Sequential User Trip Data for BERT4Rec Model

	A	B	C
1	Trip ID	user_id	rating_trip
2	1	5600	3.74
3	2	4357	3.68
4	3	2826	3.68
5	4	1584	3.86
6	5	8	2.96
7	6	2417	3.64
8	7	6403	3.56
9	8	841	3.64
10	9	4861	3.64

Figure 3.3: Example of Sequential User Trip Data for NCF Model

3.3 Architecture of the proposed hybrid recommendation system

3.3.1 Selection of basic components

In designing our hybrid recommendation system, we selected three complementary models—BERT4Rec, Neural Collaborative Filtering (NCF), and SBERT—each contributing a unique perspective on user behavior and trip characteristics. This combination allows us to leverage both collaborative signals and semantic understanding to generate high-quality, personalized recommendations.

a) Combinaison of bert4rec and sbert

BERT4Rec (bert for recommandation): We chose BERT4Rec because it's great at understanding the order in which users take trips. Using a special Transformer-based model, it can grasp the patterns and context in a user's travel history. This matters a lot since people's travel preferences can change over time, and the sequence of their trips gives important clues about what they might want next. Thanks to BERT4Rec, our system can

make smarter predictions about the next trip a user is likely to be interested in. The prediction is formalized as[124]:

$$\hat{y}_t = \text{Softmax}(f(\text{Mask}(\text{Trip Sequence})))$$

Where:

- $\hat{y}_t$ is the predicted probability distribution over trip IDs at position t,
- f is the Transformer encoder,
- Mask(Trip Sequence) is the input sequence of trip IDs with masked tokens.

This enables the model to learn temporal dependencies and contextual patterns from the user's historical trip sequence.

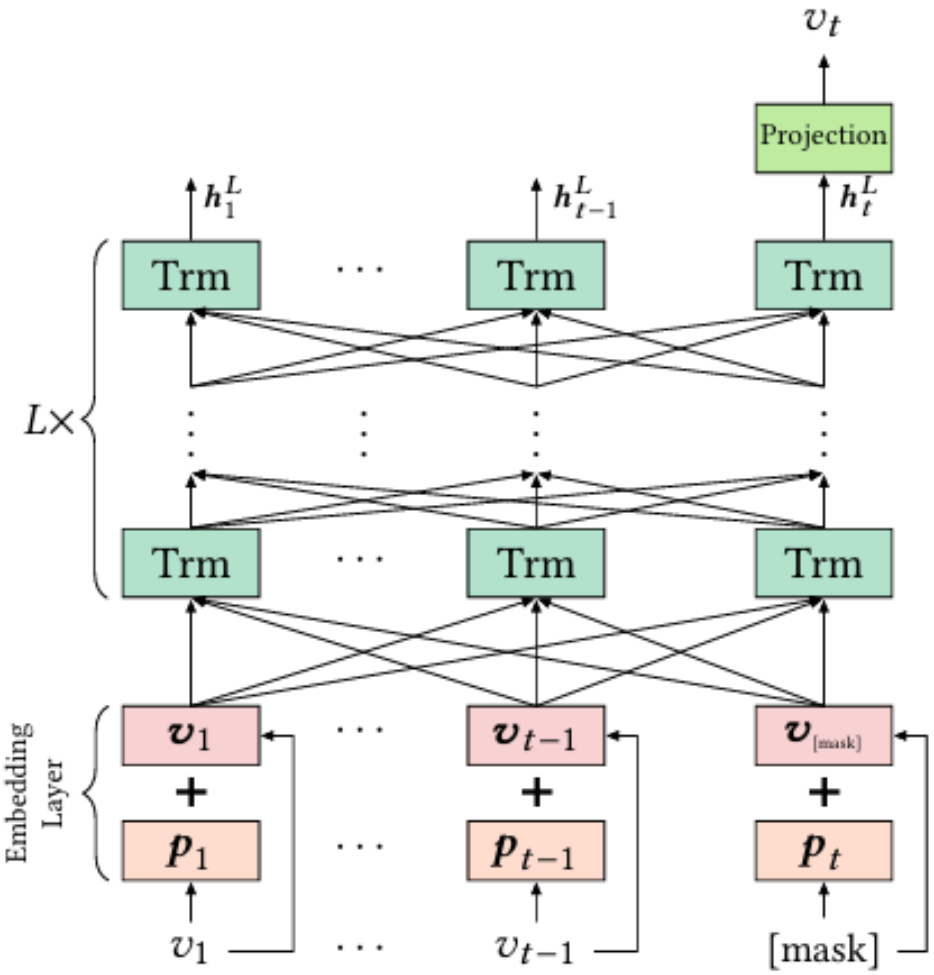


Figure 3.4: Bert4Rec model architecture [4]

SBERT (Sentence-BERT): SBERT adds a content-based perspective to our recommendations. It converts the descriptions of trips into meaningful numerical representations that

capture their true meaning—not just keywords. This helps the system find trips that are genuinely similar in theme and style. It's especially useful when we don't have much data on a user's past trips, because SBERT can recommend options based on the trip content itself, making sure the suggestions really match what the user might enjoy. Each candidate trip is scored based on its cosine similarity with the user's historical preferences:

$$\text{score}_{\text{trip}} = \cos(\text{embedding}_{\text{trip}}, \text{embedding}_{\text{user_history}})$$

Where:

- $\text{embedding}_{\text{trip}}$ is the SBERT vector of the trip's description,
- $\text{embedding}_{\text{user_history}}$ is the average embedding of previously visited trips.

This step ensures that recommendations are semantically aligned with the user's interests, addressing the limitations of purely ID-based predictions. To encode semantic meaning, we selected SBERT for its strong performance in capturing sentence-level context.Unlike traditional models such as TF-IDF, which rely solely on word frequency, or standard BERT, which is not optimized for similarity tasks, SBERT produces dense, context-aware embeddings. This is especially effective for structured trip descriptions where semantic precision is crucial.

A core challenge in building our system is the complex nature of user preferences, which go beyond simple item IDs. Users often seek personalized travel experiences influenced by subtle semantic and contextual cues. To address this, we combine multiple data sources: trip descriptions, sequential user interactions, and user ratings.BERT4Rec is used to model the temporal dynamics of user behavior through sequence modeling. However, it lacks an understanding of the semantic content of trips. To bridge this gap, SBERT is incorporated to generate embeddings for trip descriptions, allowing the system to consider both behavioral patterns and semantic similarity. This synergy leads to more personalized and relevant recommendations.

The architecture of our hybrid system begins with user interaction histories fed into BERT4Rec to predict top-k trip candidates. These candidates are then embedded using SBERT to extract semantic features from their descriptions. A re-ranking stage compares the user's recent trip embeddings to the candidate embeddings, producing a final list of top-n personalized recommendations.While this BERT4Rec+SBERT hybrid captures both sequential behavior and content relevance, it does not yet incorporate user rating preferences. To further enhance personalization, we integrate a Neural Collaborative Filtering (NCF) model, which brings explicit user feedback into the recommendation process.

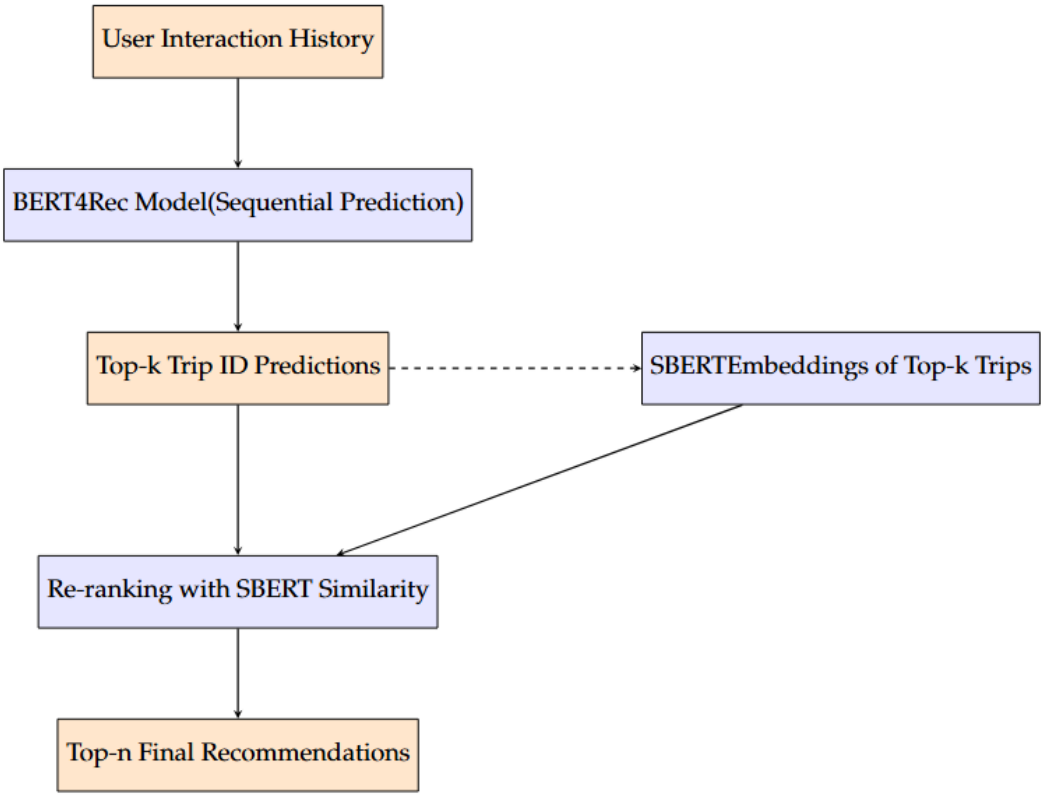


Figure 3.5: Hybrid BERT4Rec & SBERT

b) Neural Collaborative Filtering (NCF)

We chose to implement a custom NCF model to effectively learn complex, non-linear interactions between users and travel items. Unlike traditional matrix factorization methods, NCF uses neural networks and embedding layers to model user and item characteristics. In our model, each user and trip is first mapped to a dense embedding vector. These embeddings are concatenated and passed through a multi-layer perceptron (MLP) that predicts the expected rating. This approach allows us to learn subtle preference patterns and produce more personalized recommendations.

The prediction is formalized as:

$$\hat{r}_{ui} = f\left([\mathbf{e}_u \| \mathbf{e}_i]\right) \tag{3.1}$$

where:

- $\hat{r}_{ui}$ is the predicted rating score for user u and trip i,
- $\mathbf{e}_u$ and $\mathbf{e}_i$ are the learned embedding vectors,
- f is the neural network function,

- $[\mathbf{e}_u \| \mathbf{e}_i]$ denotes the concatenated user and item embeddings.

This setup enables the model to generalize well across different user behaviors, even in the absence of temporal information, and provides a strong baseline for recommendation performance.

To better illustrate the structure of our Neural Collaborative Filtering model, the overall architecture is depicted in 3.6

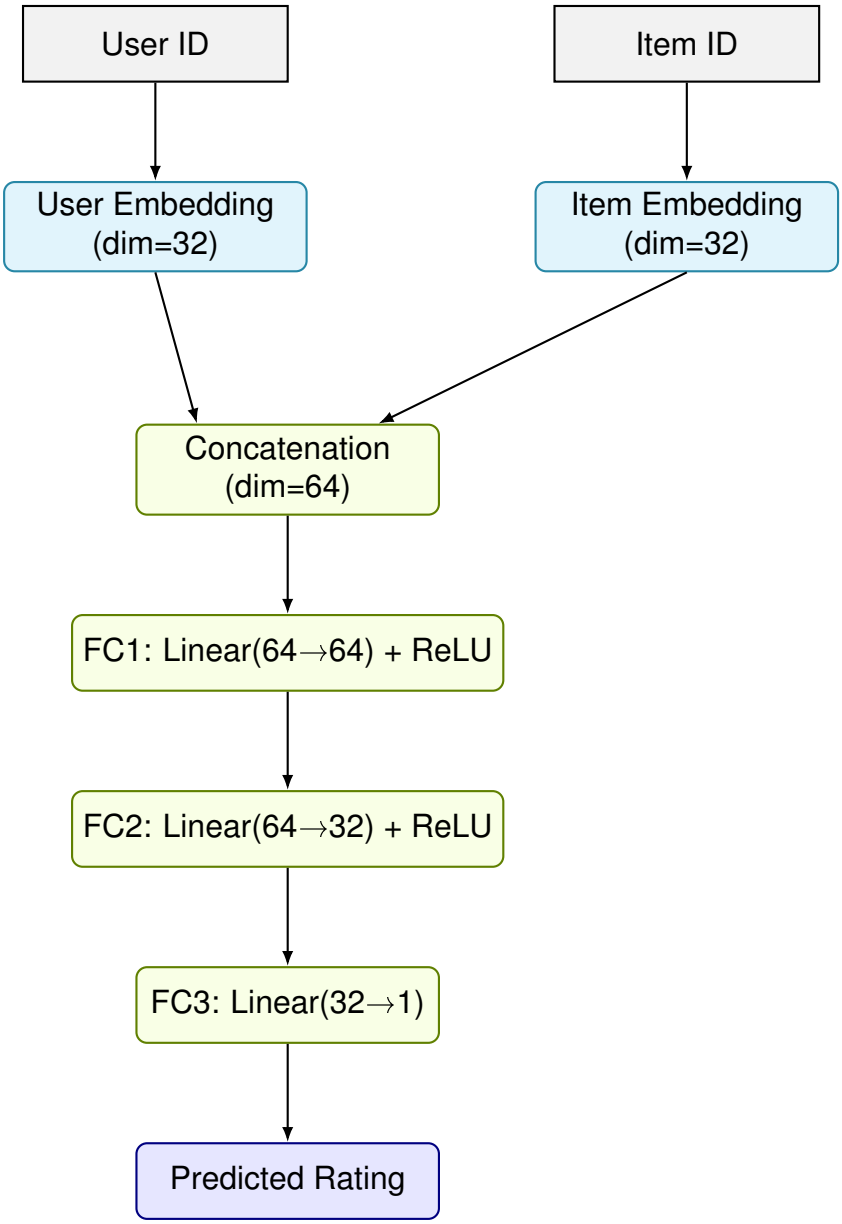


Figure 3.6: Neural Collaborative Filtering (NCF) architecture

3.3.2 Hybrid Recommendation Strategy Overview

Our hybrid recommendation system combines multiple models to enhance the accuracy, diversity, and personalization of trip recommendations. The system leverages:

- **BERT4Rec** for learning sequential patterns in user trip histories.
- **SBERT** for semantically reranking candidates using textual trip descriptions.
- **Neural Collaborative Filtering (NCF)** for modeling personalized preferences based on historical interactions.

This multi-stage architecture is motivated by the nature of our dataset, which includes both behavioral and semantic data. It addresses challenges such as cold-start problems and data sparsity, while integrating sequential, semantic, and collaborative signals for better recommendations.

Stage 1: Sequential Modeling with BERT4Rec

In the first stage, we use **BERT4Rec**, a Transformer-based model trained on sequences of user trip interactions. BERT4Rec predicts the next likely trip by capturing bidirectional dependencies within the user's historical behavior. The mathematical Formulation of this model is [4]:

Given a user interaction sequence:

$$\mathcal{S}_u = [v_1, v_2, \ldots, v_t]$$

where $v_i \in \mathcal{V}$ is the item (trip ID) from the vocabulary $\mathcal{V}$, and t is the length of the sequence.

Input Embedding

Each item is mapped to an embedding and summed with a positional embedding:

$$\mathbf{e}_i = \mathbf{E}_{v_i} + \mathbf{P}_i$$

where:

- $\mathbf{E} \in \mathbb{R}^{|\mathcal{V}| \times d}$: item embedding matrix,
- $\mathbf{P} \in \mathbb{R}^{t \times d}$: positional embedding matrix,
- d: embedding dimension.

The full sequence embedding:

$$\mathbf{X} = [\mathbf{e}_1, \ldots, \mathbf{e}_t]$$

Masking

A random subset of positions $\mathcal{M} \subset \{1, \ldots, t\}$ is selected for masking. These positions are replaced with a special token `[MASK]`, while the rest are unchanged:

$$\mathbf{X}_{\text{masked}} = \text{Mask}(\mathbf{X}, \mathcal{M})$$

Bidirectional Transformer Encoder

The masked sequence is passed through L layers of Transformer encoder:

$$\mathbf{H}^{(0)} = \mathbf{X}_{\text{masked}}, \quad \mathbf{H}^{(l)} = \text{Transformer}(\mathbf{H}^{(l-1)}) \quad \text{for } l = 1, \ldots, L$$

Final hidden representation:

$$\mathbf{H} = \mathbf{H}^{(L)} = [\mathbf{h}_1, \ldots, \mathbf{h}_t]$$

Prediction

For each masked position $i \in \mathcal{M}$, the model predicts the original item v_i using a softmax layer:

$$\hat{\mathbf{y}}_i = \text{Softmax}(\mathbf{h}_i \cdot \mathbf{E}^T)$$

where $\hat{\mathbf{y}}_i \in \mathbb{R}^{|\mathcal{V}|}$ is the predicted probability distribution over the item vocabulary.

Loss Function

The model is trained to minimize the cross-entropy loss over all masked positions:

$$\mathcal{L} = -\sum_{i \in \mathcal{M}} \log P(v_i \mid \mathcal{S}_u^{\setminus i}) = -\sum_{i \in \mathcal{M}} \log \hat{\mathbf{y}}_i[v_i]$$

Implementation Details:

- *Trip IDs are tokenized (e.g.,* `trip_123`).
- Optimizer: AdamW.
- Learning rate: 5e-5
- Loss function: Cross-entropy on masked positions.

Stage 2: Semantic Reranking with SBERT

After obtaining top-k trip predictions from BERT4Rec, we use **SBERT** to rerank them based on semantic similarity to the user's historical trips.

Semantic Similarity Scoring:

Let:

$$T = \{t_1, t_2, \ldots, t_k\} \quad \text{(candidate trips)}$$
$$H = \{h_1, h_2, \ldots, h_m\} \quad \text{(user's historical trips)}$$

SBERT encodes each item:

$$\mathbf{e}_{t_i} = \text{SBERT}(t_i), \quad \mathbf{e}_{h_j} = \text{SBERT}(h_j)$$

$$\bar{\mathbf{e}}_H = \frac{1}{m} \sum_{j=1}^{m} \mathbf{e}_{h_j}$$

Then, we compute cosine similarity:

$$\text{sim}(t_i) = \frac{\bar{\mathbf{e}}_H \cdot \mathbf{e}_{t_i}}{\|\bar{\mathbf{e}}_H\| \|\mathbf{e}_{t_i}\|}$$

Reranking Process:

- Compute similarity between user profile and each candidate.
- Sort candidates by similarity score.
- Select the top-N semantically relevant items.

Stage 3: Personalized Scoring with NCF

To further refine the candidate list, we apply **Neural Collaborative Filtering (NCF)** to model personalized preferences.

- User and Item Embeddings:
  Each user $u \in \{1, \ldots, N\}$ and item $i \in \{1, \ldots, M\}$ is represented by:

$$\mathbf{e}_u = \text{Emb}_u(u), \quad \mathbf{e}_i = \text{Emb}_i(i)$$

- Model Architecture:
  Concatenated embeddings are passed through an MLP:

$$\mathbf{x}_{ui} = [\mathbf{e}_u; \mathbf{e}_i]$$

$$\begin{aligned} \mathbf{h}_1 &= \text{ReLU}(\mathbf{W}_1 \mathbf{x}_{ui} + \mathbf{b}_1) \\ \mathbf{h}_2 &= \text{ReLU}(\mathbf{W}_2 \mathbf{h}_1 + \mathbf{b}_2) \\ \hat{y}_{ui} &= \mathbf{W}_3 \mathbf{h}_2 + b_3 \end{aligned}$$

- Training Objective:
  We minimize the Mean Squared Error (MSE):

$$\mathcal{L}_{\text{MSE}} = \frac{1}{|D|} \sum_{(u,i) \in D} (y_{ui} - \hat{y}_{ui})^2$$

a) Final Integration:

The final recommendation score is a weighted combination of:

- Sequential signal (BERT4Rec)
- Semantic signal (SBERT)
- Personalized score (NCF)

This combinaition ensures each recommendation is aligned with the user's behavior, semantic preferences, and collaborative patterns.

3.4 Definition of the experimental protocol

3.4.1 Evaluation Metrics Selection

To evaluate the effectiveness of our recommendation system, we used several widely adopted metrics in the literature: Hit Rate@k, NDCG@k, Mean Reciprocal Rank (MRR), and Root Mean Square Error (RMSE). Each of these metrics captures a different aspect of recommendation quality, such as accuracy, ranking relevance, and error.

a) Hit Rate@k

Hit Rate@k (also known as Hit@k)evaluates how often a relevant item appears within the top-k recommendations for a user. While it captures retrieval success, it does not consider the item's rank position within the top-k.**[53]:**

$$\text{Hit@k} = \frac{1}{|U|} \sum_{u \in U} \mathbb{I}(r_u \in \text{Top-}k_u)$$

where U is the set of users, r_u is the relevant item for user u, and $\mathbb{I}$ is the indicator function, which returns 1 if r_u is in the top-k predicted items Top-k_u, and 0 otherwise.

b) Normalized Discounted Cumulative Gain (NDCG@k)

NDCG@k measures ranking quality by rewarding relevant items appearing earlier in the top-k list. It captures both relevance and position to assess recommendation effectiveness. **[53]**

$$\text{NDCG@k} = \frac{1}{|U|} \sum_{u \in U} \frac{1}{\text{IDCG@k}_u} \sum_{i=1}^{k} \frac{rel_{u,i}}{\log_2(i+1)}$$

where $rel_{u,i}$ is the relevance of the item at position i for user u, and IDCG@k$_u$ is the ideal (maximum possible) DCG for that user at rank k, used to normalize the DCG score.

c) Root Mean Square Error (RMSE)

RMSE is a standard regression metric used to measure the average magnitude of prediction errors. It is especially relevant when predicting continuous scores, such as user ratings or similarity values.[53].

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

where T is the set of user-item pairs in the test set, $\hat{r}_{u,i}$ is the predicted score, $r_{u,i}$ is the true score, and N is the total number of predictions.

3.4.2 Baseline Systems for Comparison

To rigorously evaluate the performance of our proposed hybrid recommendation system, we implemented three baseline models using the same dataset. The objective is to isolate the contribution of each component within the hybrid architecture and assess their individual effectiveness in terms of recommendation quality.

- **BERT4Rec**: As a strong sequential recommendation model, BERT4Rec serves as our primary baseline. It treats the user's trip history as a sequence and learns to predict the next relevant trip using a transformer-based architecture. This model captures temporal patterns and sequential dependencies between trips, making it an effective foundation for comparison.
- **BERT4Rec + SBERT**: In this variant, we use BERT4Rec to generate top-k trip ID predictions based on user history. These predicted trips are then re-ranked using cosine similarity between their SBERT embeddings and the recent trip embeddings from the user history. This baseline allows us to assess the impact of incorporating semantic similarity in the re-ranking phase, isolated from other hybrid components.

These baseline models were trained and evaluated using the **same dataset** as the hybrid model to ensure consistency in comparison. By doing so, we ensure that any observed differences in performance can be attributed to architectural variations rather than disparities in data preprocessing, evaluation methods, or experimental setup. This comparative analysis provides valuable insights into the relative strengths and limitations of each approach, helping to validate the benefits of the hybrid strategy.

Validation Method

To evaluate the performance of our recommendation system in a realistic and unbiased setting, we employed a hold-out validation method based on a user-level split. Specifi-

cally, the entire dataset of users was divided into two disjoint subsets: 80% of the users were allocated for training, and the remaining 20% were reserved for testing.

This approach ensures that there is no overlap between training and testing users, thereby preventing information leakage and allowing the model to be evaluated on entirely unseen user behaviors. All interactions (i.e., trip sequences) belonging to a user are kept intact and assigned entirely to either the training or testing set. This setup is particularly suited for scenarios where the goal is to generalize recommendations to new users based on the learned behavioral patterns of other users.

The same splitting strategy was consistently applied across all baseline models and the proposed hybrid model to ensure fair and comparable evaluation.

3.5 Conclusion

In this chapter, we presented the development of a hybrid recommendation system specifically tailored to the travel domain. We began by describing the collection and preprocessing of real-world data from multiple sources, including flights, hotels, and points of interest, to create a rich and diverse dataset.

We then introduced the three core components of our hybrid architecture: BERT4Rec for modeling sequential user behavior, SBERT for capturing semantic similarity between user preferences and trip descriptions, and NCF for learning collaborative filtering signals from user ratings. Each of these modules contributes distinct strengths, and their integration allows the system to generate recommendations that are both accurate and contextually relevant.

The final hybrid strategy leverages the complementary nature of these models, balancing sequential dynamics, semantic relevance, and collaborative patterns. We detailed the fusion process and the overall architecture through a clear two-stage pipeline, followed by a rigorous experimental protocol involving offline evaluation using several performance metrics.

This comprehensive design establishes a solid foundation for the subsequent evaluation phase, where we assess the effectiveness and robustness of our hybrid recommendation system in comparison to traditional and single-model baselines.

Chapter 4

Implementation and Experimentation

Chapter 4

Implementation and Experimentation

4.1 Introduction

This chapter focuses on the practical realization of the hybrid recommendation system introduced in the previous chapters, along with a thorough evaluation of its performance. It marks the transition from theoretical design to concrete implementation by detailing the technical choices, development tools, and experimental procedures.

We begin by presenting the development environment, including the programming languages, libraries, and hardware used. Next, we describe the implementation of each component of the system, including both the baseline models and the hybridization strategy. We also discuss the challenges encountered during the development process and the solutions applied to overcome them.

The chapter then outlines the experimental setup, covering the configuration of experiments, parameter tuning methods, and the execution of tests on the dataset. Finally, we present and analyze the results, both quantitatively and qualitatively, to assess the effectiveness of the proposed system and to validate the research hypotheses.

4.2 Development Environment and Tools

To develop the proposed hybrid recommendation system, various tools and environments were used to support model design, training, and evaluation. This section outlines the programming language, libraries, and hardware setup used throughout the project.

4.2.1 Programming Language

The entire project was implemented in **Python**, due to its wide adoption in the machine learning community, its ease of use, and the vast ecosystem of scientific libraries. Python was especially suitable for rapid prototyping and seamless integration of different model components.

4.2.2 Libraries and Frameworks

A variety of powerful Python libraries and frameworks were employed:

- **PyTorch**: Used for implementing the NCF model and training neural networks. It allowed fine-grained control over model architecture and optimization.
- **Transformers (HuggingFace)**: Used to load and fine-tune the BERT4Rec model for sequential recommendation tasks.
- **SentenceTransformers (SBERT)**: Used to generate semantic embeddings of trips, capturing user interests in a dense vector space.
- **Pandas** and **NumPy**: Used extensively for data preprocessing, manipulation, and numerical operations.
- **Scikit-learn**: Utilized for utility tasks like label encoding and train/test splitting.
- **tqdm**: Employed for monitoring training and evaluation progress with progress bars.
- **joblib**: Used for saving and loading encoders (e.g., user/item mappings).
- **Matplotlib** / **Seaborn** (optional): Could be used for visualization during analysis (if applicable).

This modular use of libraries enabled clean separation between data processing, model training, hybrid scoring, and evaluation.

4.2.3 Hardware

The experiments were primarily conducted on a **Google Colab environment** equipped with GPU acceleration (**NVIDIA Tesla T4**) to speed up training, especially for the BERT-based components. When necessary, the model was trained on **CPU** for testing or debugging. Typical hardware specs included:

- **GPU**: NVIDIA Tesla T4 (on Colab)
- **RAM**: 12–16 GB (Colab instance)
- **CPU**: Dual-core / quad-core virtual CPUs

All models, embeddings, and evaluation metrics were executed efficiently in this cloud-based environment, eliminating the need for high-end local hardware.

4.3 System Architecture and Implementation

This section details the implementation of the core recommendation models, the hybridization strategy combining these models, and the main challenges encountered along with the solutions applied.

4.3.1 Implementation of Base Systems

The recommendation system is composed of three main base models:

- **Neural Collaborative Filtering (NCF)**: Implemented in PyTorch, the NCF model leverages embedding layers for users and trips, followed by fully connected layers to predict user-trip interaction scores. The dataset was preprocessed using label encoding and split into training and validation sets. Training utilized MSE loss optimized with Adam, and evaluation metrics included RMSE and Hit Rate@K.
- **BERT4Rec**: A transformer-based sequential recommendation model was fine-tuned using the HuggingFace Transformers library. The model predicts masked trips in user sequences, capturing sequential patterns in trip bookings.
- **Sentence-BERT (SBERT)**: This model generates semantic embeddings of trips based on their textual or feature descriptions. These embeddings encode the similarity between trips, enabling semantic-aware recommendation.

Each model was trained and evaluated independently before integration.

4.3.2 Hybridization Strategy

The recommendation process begins with the **BERT4Rec model** generating an initial set of top-k candidate trips for each user, based on sequential behavior modeling. This model captures the temporal context of a user's travel history, predicting the next likely destination using a masked language modeling approach. To refine these candidates, we apply a **hybrid reranking strategy** that integrates three complementary signals:

1. **SBERT Similarity Score** ($\text{Sim}_{\text{SBERT}}$): The SBERT Content-based Similarity component (30% weight) measures the cosine similarity between the SBERT embedding of each candidate trip and the aggregated embedding of the user's most recent trips. This score provides semantic alignment and captures textual similarities.
2. **NCF Interaction Score** ($\text{Score}_{\text{NCF}}$): Generated by a Neural Collaborative Filtering model (30% weight), this score reflects collaborative filtering preferences based on user–trip interaction data. It uncovers latent patterns in user behavior from collective data to identify hidden preferences and similarities.
3. **BERT4Rec Rank Score** ($\text{Rank}_{\text{BERT}}$): Represents the inverse of the candidate's original rank as predicted by BERT4Rec (40% weight). A higher inverse rank corresponds to higher initial confidence from the sequential model.

Each of these scores is normalized to the $[0, 1]$ range and combined using a weighted sum:

$$\text{FinalScore}(u,t) = \alpha \cdot \text{Sim}_{\text{SBERT}}(u,t) + \beta \cdot \text{Score}_{\text{NCF}}(u,t) + \gamma \cdot \frac{1}{\text{Rank}_{\text{BERT}}(t)} \tag{4.1}$$

where:

- $\alpha = 0.3$ — weight for SBERT similarity,
- $\beta = 0.3$ — weight for NCF score,
- $\gamma = 0.4$— weight for BERT4Rec inverse rank.

These weights were determined empirically through extensive experimentation, balancing *semantic*, *collaborative*, and *sequential* signals to optimize performance across evaluation metrics such as Hit@K and NDCG,

After computing the final relevance score for each candidate trip, the list is reranked accordingly. By combining **BERT4Rec's sequential modeling**, **SBERT's semantic similarity**, and **NCF's collaborative filtering**, our hybrid system delivers more accurate and diverse recommendations, effectively overcoming the limitations of single-model architectures.

a) Hybrid Reranking Evaluation

To identify the optimal combination of semantic, collaborative, and sequential signals in our hybrid recommendation system, we evaluated three different weighting strategies for combining SBERT similarity, NCF interaction score, and BERT4Rec rank score.

Each strategy follows the general scoring formula in (4.1)

We tested three configurations:

- **NCF-Heavy**: $\alpha = 0.1$, $\beta = 0.5$, $\gamma = 0.4$
- **BERT-Heavy**: $\alpha = 0.3$, $\beta = 0.3$, $\gamma = 0.4$
- **Equal Weights**: $\alpha = 0.33$, $\beta = 0.33$, $\gamma = 0.34$

Table 4.1: Hybrid Strategy Performance Comparison

Strategy	**Hit@10**	**NDCG@10**
NCF-Heavy (SBERT: 0.3, NCF: 0.5, BERT: 0.2)	0.8005	0.8558
BERT-Heavy (SBERT: 0.3, NCF: 0.3, BERT: 0.4)	0.8617	0.8601
Equal Weights (SBERT: 0.33, NCF: 0.33, BERT: 0.34)	0.8614	0.8541

Among the tested configurations, we selected the **BERT-Heavy** strategy ($\alpha = 0.3$, $\beta = 0.3$, $\gamma = 0.4$) as the optimal hybrid approach. This choice was guided by its superior performance, achieving the highest Hit@10 score of **0.8617** and the best NDCG@10 of **0.8601**. These results indicate that emphasizing sequential patterns captured by the BERT4Rec model, while still integrating SBERT semantic similarity and NCF collaborative signals, leads to a well-balanced and accurate recommendation. The Equal Weights strategy yielded comparable but slightly lower results, while the NCF-Heavy configuration underperformed, highlighting the importance of sequential modeling in our setting.

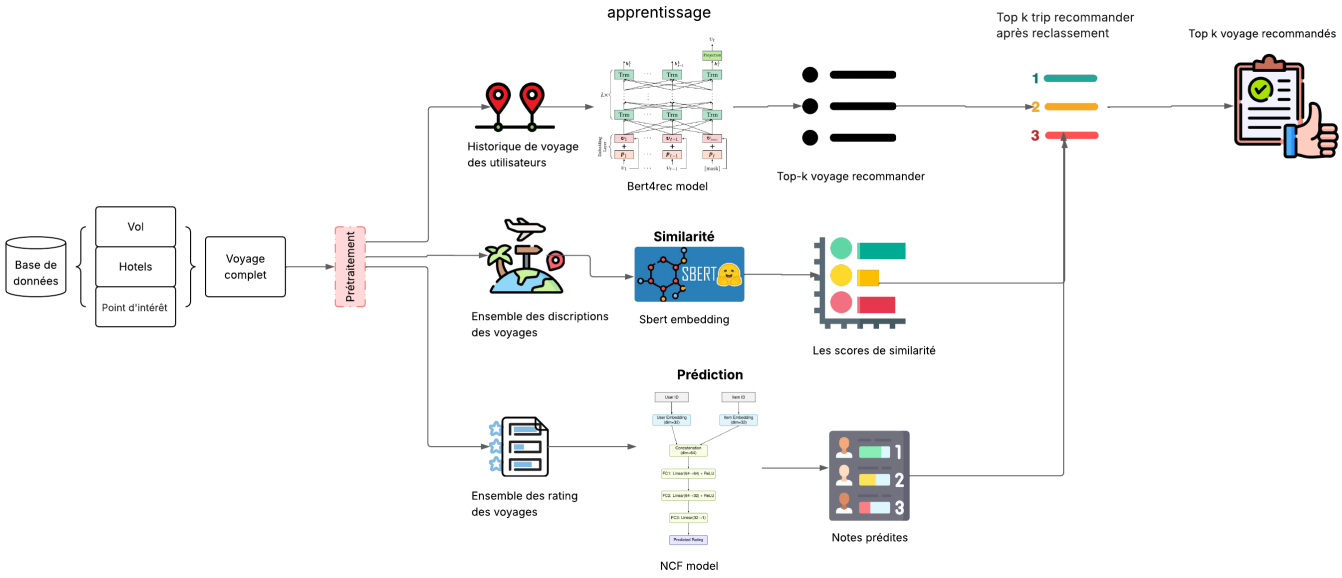


Figure 4.1: Overall architecture of the hybrid recommendation system integrating BERT4Rec, SBERT, and NCF models.

4.3.3 Challenges and Solutions

Building a robust hybrid recommendation system, especially one integrating deep learning models like BERT4Rec, NCF, and SBERT embeddings, presented several technical hurdles. Tackling these challenges was essential for creating a stable, high-performing system.

a) Data Encoding and Alignment

Ensuring **consistent mapping of user and trip IDs** across NCF, SBERT, and BERT4Rec was a significant challenge due to their varying internal representations. We addressed this by meticulously using and persisting **LabelEncoder** objects for users and items, alongside a precise **token_id_to_trip** mapping for BERT4Rec, guaranteeing seamless data flow.

b) Cold-Start and Unseen Users/Items

Recommendation systems often struggle with **cold-start users** and **unseen items**. Our solution involved implementing **robust error handling** in NCF to skip unknown entities and ensuring BERT4Rec only processed **valid, known trip IDs**. This maintained system stability and prevented errors.

c) Balancing Hybrid Weights

Determining the optimal **bert_weight** and **ncf_weight** was crucial for reflecting each model's contribution. We solved this through **extensive empirical tuning and systematic grid search** on a validation dataset, evaluating performance with metrics like **Hits@K and NDCG** to achieve the best synergy.

d) Computational Resources

Training and running large deep learning models are inherently **resource-intensive**. We mitigated this by leveraging **GPU acceleration** on Google Colab and employing **batching strategies** during both training and inference, significantly improving efficiency.

e) Metric Computation Complexity

Accurately **evaluating recommendation quality** using ranking metrics like NDCG required careful implementation. We addressed this by developing a dedicated **Recommendation Evaluator** class to encapsulate and precisely calculate these complex performance metrics.

4.4 Experimental Setup

The dataset used in our experimental setup was derived from the integration and preprocessing of three data sources—flights, hotels, and points of interest (POIs)—as detailed in Chapter 3. These sources were merged by city to form a unified trip dataset, where each trip includes flight details, hotel accommodations, and POIs. After filtering and cleaning, a representative and diverse sample of 30,000 high-quality trip records was selected for analysis.

To support our hybrid recommendation system, we structured the data into specific formats tailored to the models used:

4.4.1 Trip Descriptions and SBERT Embeddings

Each trip was enriched with a detailed textual description, composed of aggregated information from the flights, hotel, and POIs. These descriptions were embedded using

Sentence-BERT (SBERT) to capture semantic meaning in a high-dimensional vector space. These embeddings are essential for computing similarity between user histories and candidate trips during the re-ranking phase.

Embedding model: paraphrase-mpnet-base-v2 (from the sentence-transformers library)

Embedding size: 768 dimensions Batch size: 32 Device: GPU Purpose: Semantic comparison in the hybrid re-ranking stage

4.4.2 User-Trip-Rating Table (for NCF)

To represent user preferences effectively, we constructed an interaction matrix where each entry corresponds to a user–trip pair, along with an associated rating. This rating was calculated using a weighted average of hotel and point-of-interest (POI) ratings, with 60% weight assigned to the hotel rating and 40% to the POI rating:

$$\text{trip_rating} = 0.6 \times \text{hotel_rating} + 0.4 \times \text{poi_rating}$$

The choice to assign a higher weight to hotel ratings stems from their ability to offer a more comprehensive and consistent measure of traveler satisfaction. While POI ratings can vary in coverage and detail, they still provide valuable complementary insights into the travel experience by highlighting specific attractions.

This interaction matrix served as the foundation for training a Neural Collaborative Filtering (NCF) model, aimed at predicting user preferences for unseen or unrated trips.

4.4.3 User Trip Sequences (for BERT4Rec)

For sequential modeling, we built temporal interaction histories per user. Each sequence captures the chronological order in which a user interacted with different trips. Users with fewer than four trips were excluded to ensure meaningful sequential learning. Format: List of trip IDs per user, ordered by timestamp Use case: Input to the BERT4Rec model for next-trip prediction

4.4.4 Train-Test Split

To simulate a realistic evaluation scenario, we adopted a user-level split strategy rather than a session-level one, in order to prevent data leakage between training and testing phases. Specifically, we first reconstructed each user's trip history by sorting their records chronologically. Users with fewer than four trips were excluded to ensure a minimum length of interaction. We then randomly selected 80This strategy ensures that all test users are unseen during training, mimicking real-world deployment where the model must generalize to new users. It is particularly suitable for recommendation models such as BERT4Rec and NCF, which learn from personalized user patterns across sequences.

which learn from personalized user patterns across sequences.

4.4.5 Configuration of experiments and parameter tuning strategy

Table 4.2: Model Configurations and Hyperparameter Tuning

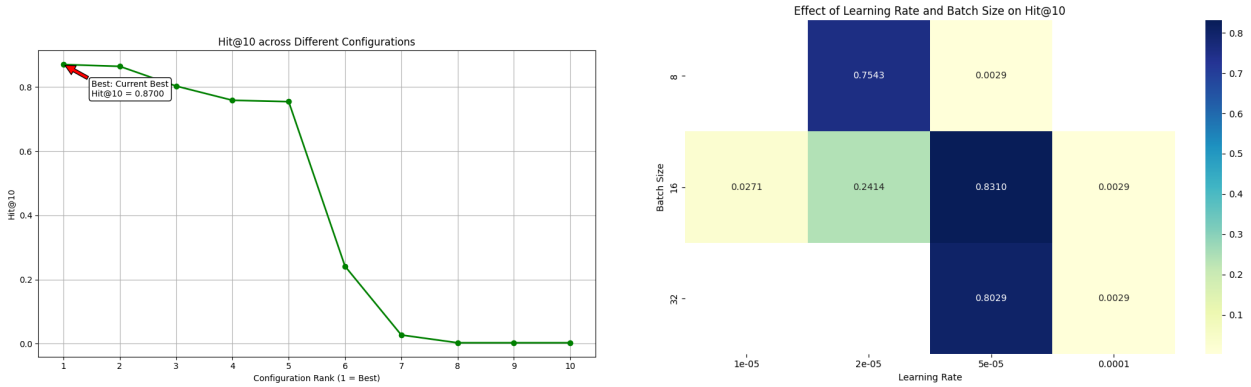
Model	Parameter	Values Tried / Selected
BERT4Rec	Transformer layers	1, 2 (selected: 2)
	Attention heads	2, 4 (selected: 4)
	Hidden size	128
	Batch size	32, 64 (selected: 32)
	Learning rate	5e-5, 1e-4 (selected: 5e-5)
	Masking ratio	10%, 15%, 20% (selected: 15%)
NCF	Embedding size	32, 64 (selected: 32)
	Batch size	16, 32 (selected: 16)
	Learning rate	0.1, 1e-3 (selected: 1e-3)
	Epochs	10, 20 (selected: 10)

This table presents the hyperparameter settings used for the BERT4Rec and NCF models. For BERT4Rec, key parameters such as the number of Transformer layers, attention heads, and hidden size were varied to balance model capacity and training efficiency. The masking ratio and maximum sequence length were also tuned to optimize sequential learning performance. For NCF, embedding dimensions and the architecture of the multilayer perceptron (MLP) were selected based on validation performance. Batch size and learning rate were tuned independently for both models using a grid search approach, with the number of epochs determined based on early stopping criteria to prevent overfitting. This configuration ensured fair and effective comparison between models under controlled settings.

4.4.6 Execution of experiments on different model configurations.

To evaluate the performance of the proposed hybrid recommendation system combining **BERT4Rec** and **Neural Collaborative Filtering (NCF)**, we conducted a comprehensive set of experiments, systematically varying key hyperparameters for both models..

For **BERT4Rec**, we explored a range of architectural and training configurations to identify the optimal setup. In particular, we experimented with different **learning rates** (from `1e-5` to `1e-4`), **batch sizes** (8, 16, 32), and **weight decay** values (including `0.0`, `0.005`, and `0.01`) to assess their impact on model performance. Through empirical evaluation, we observed that the best configuration among all tested setups is:



(a) Configuration Performance Ranking (b) effect of LR and batch size on hit@10

Figure 4.2: Evaluation of configurations of the model bert4rec

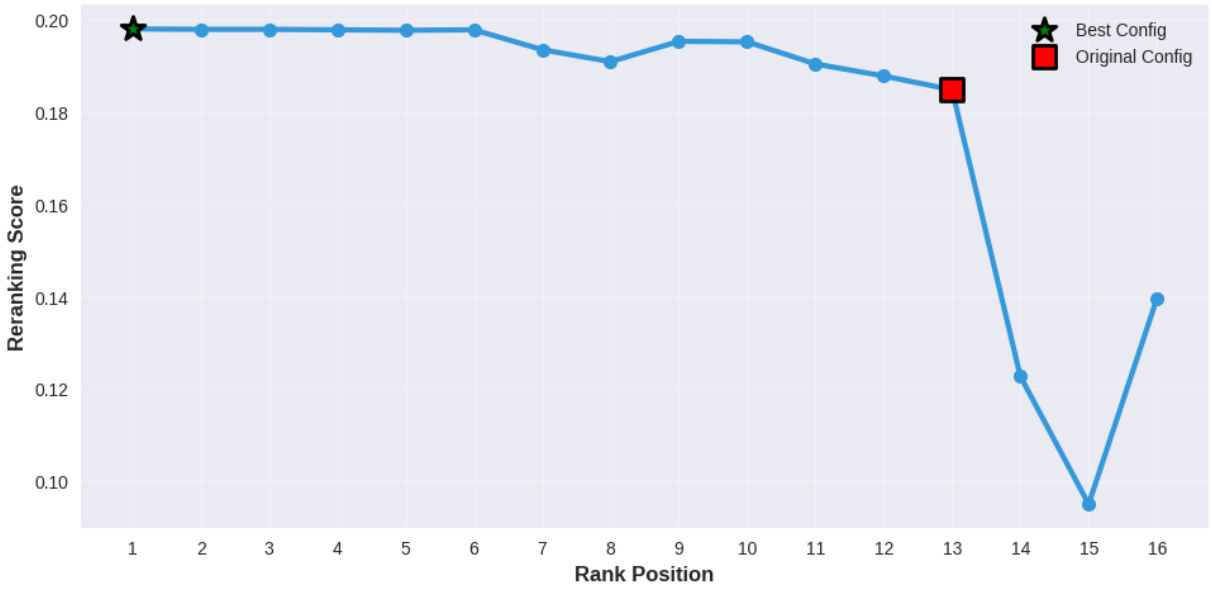
Table 4.3: Best Performing Hyperparameter Configuration for BERT4Rec

Hyperparameter	**Value**
Learning Rate	5e-5
Batch Size	16
Epochs	20
Weight Decay	0.01
Final Loss	0.2001
Hit@5	0.8680
Hit@10	0.8700
Hit@20	0.8700

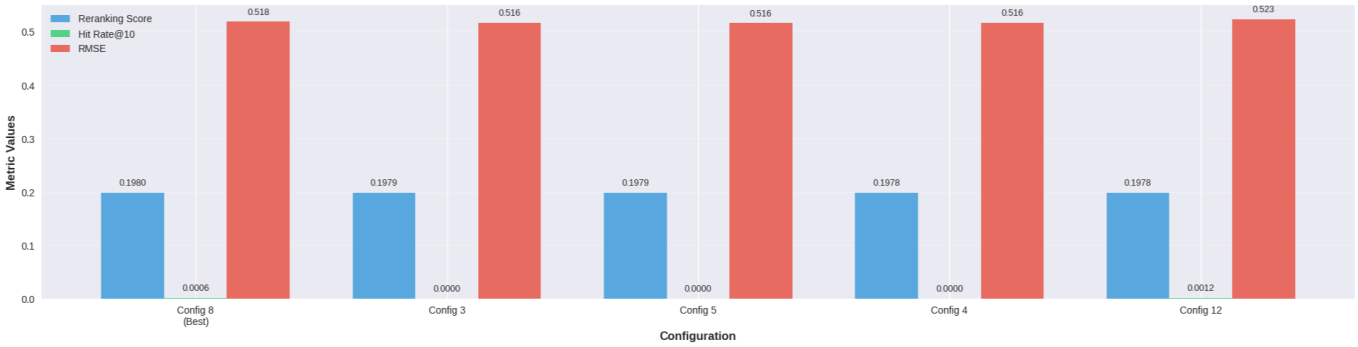
Similarly, for **NCF**, we experimented with various embedding sizes (32, 64, 128) and Multi-Layer Perceptron (MLP) architectures to optimize the representation of user-item interactions.Based on the evaluation of 16 configurations, Configuration 8 was selected as the optimal choice for the reranking stage. As illustrated in the graphs above, it achieved the **highest weighted score (0.1980)**, using a formula that combines **Hit Rate@10 (70% weight)** and **RMSE (30% weight)**.

$$\text{Score} = 0.7 \times \text{Hit Rate@10} + 0.3 \times (1 - \text{RMSE})$$

This scoring strategy emphasizes **precision**—crucial for reranking—by prioritizing Hit Rate@10, which measures how often relevant trips appear in the top recommendations. While RMSE still contributes to ensure reasonable rating predictions, it is treated as a secondary factor since the exact rating value is less critical than correctly ranking items. Configuration 8 stands out with a **balanced trade-off**, offering the best reranking performance, competitive RMSE (0.5185), and consistent Hit Rate@10, making it the most suitable configuration for enhancing the hybrid recommendation system's effectiveness.



(a) Configuration Performance Ranking



(b) Top 5 Configurations Comparison

Figure 4.3: Evaluation of configurations: global ranking and top-5 analysis

Table 4.4: Best Performing Hyperparameter Configuration for NCF (Config 8)

Hyperparameter	**Value**
Embedding Dimension	32
Hidden Layers	[64, 32]
Activation Function	ReLU
Learning Rate	0.001
Batch Size	512
Epochs	10
Loss Function	MSELoss
Final RMSE	0.5185
Hit@10	0.0006

Each configuration was trained using the corresponding hyperparameter set, with **early stopping** applied based on validation performance to prevent overfitting. The evaluation metrics used include **Hit@k** for BERT4Rec, and **RMSE** and **hit@k** for NCF.

4.5 Results and Evaluation

4.5.1 Quantitative Evaluation

This section presents the quantitative evaluation of the proposed hybrid recommendation system, which integrates BERT4Rec and Neural Collaborative Filtering (NCF) models to leverage both sequential user behavior and collaborative filtering signals.

a) Presentation of Results:

To evaluate the effectiveness of our hybrid model combining BERT4Rec, SBERT, and NCF, we present below the results for standard recommendation metrics: Hit@K and NDCG@K. These results highlight the overall performance of the hybrid system, which benefits from the robustness of BERT4Rec, the semantic understanding provided by SBERT, and the user preference modeling capabilities of NCF.

Table 4.5: Hit@K Performance of the Hybrid Model

Hit@K	**Score**
Hit@1	0.8825
Hit@5	0.8937
Hit@10	0.8937

Table 4.6: NDCG@K Performance of the Hybrid Model

NDCG@K	Score
NDCG@5	0.8896
NDCG@10	0.8896
NDCG@20	0.8896

b) Comparison with Baseline Models:

To validate the effectiveness of the hybrid approach, its performance is compared against the individual BERT4Rec and the BERT4Rec with sbert . This comparison highlights the improvement gained by combining sequential patterns captured by BERT4Rec with the latent factor interactions modeled by NCF. hybrid:

Table 4.7: Hit Rate Comparison Across Model Variants

Model	Hit@1	Hit@5	Hit@10
BERT4Rec	0.8597	0.8680	0.8707
BERT4Rec rerank with SBERT	0.8797	0.8881	0.8881
Our Hybrid Model (BERT4Rec reranking using SBERT and NCF)	0.8825	0.8937	0.8937



Figure 4.4: Hit@K Comparison Across Model Variants

Table 4.8: NDCG Comparison Across Model Variants

Model	NDCG@5	NDCG@10	NDCG@20
BERT4Rec	0.8645	0.8655	0.8655
BERT4Rec rerank with SBERT	0.8850	0.8850	0.8850
Our Hybrid Model (BERT4Rec reranking using SBERT and NCF)	0.8896	0.8896	0.8896

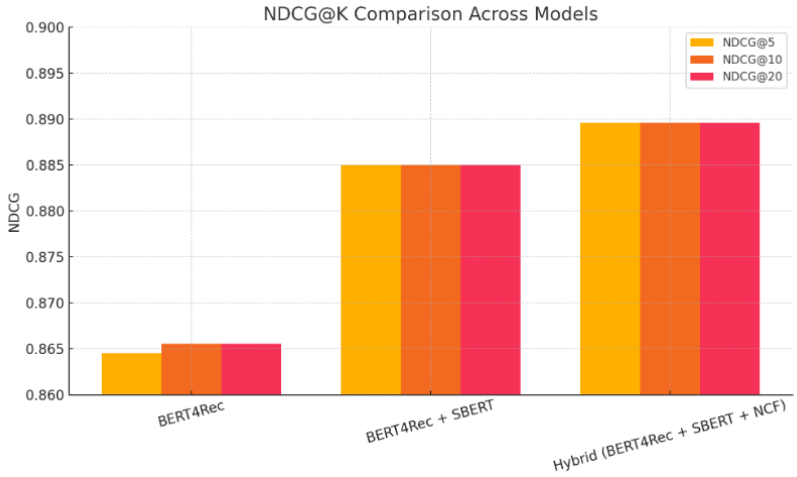


Figure 4.5: NDCG@K Comparison Across Model Variants

The experimental results clearly demonstrate that our proposed hybrid model, which reranks the top-*k* predictions of BERT4Rec using both SBERT embeddings and NCF scores, outperforms all other configurations. While the baseline BERT4Rec model provides a robust foundation for sequence-based recommendations, achieving high accuracy on its own, its performance is further enhanced when augmented with semantic similarity from SBERT and collaborative signals from NCF. Compared to the standalone BERT4Rec and the BERT4Rec+SBERT variant, the hybrid model consistently achieves higher scores across multiple metrics, including Hit@1, Hit@5, Hit@10, and NDCG@5, NDCG@10, and NDCG@20.

These improvements are particularly noteworthy given the already strong baseline performance of BERT4Rec. Enhancing such a strong model is not trivial, yet our results show that the integration of SBERT and NCF introduces complementary perspectives—semantic understanding of trip content and user interaction patterns—that help capture user preferences more comprehensively. This confirms that combining sequence modeling with both content-aware and collaborative reranking significantly improves the quality and relevance of recommendations. Overall, the gains observed validate the strength and practical impact of our hybrid recommendation strategy.

4.5.2 Qualitative Analysis of Results

a) Types of Recommendations Generated

The hybrid recommendation system demonstrated a clear ability to generate more relevant and diverse suggestions compared to the individual models (NCF, BERT4Rec, SBERT). While the NCF model primarily focused on user-item interaction history, and BERT4Rec emphasized recent behavioral sequences, the hybrid system successfully blended behavioral, collaborative, and semantic signals to deliver well-rounded recommendations.

The recommended trajectories were not only aligned with the user's previous preferences but also introduced variations in destination or activity type that matched the inferred interests.

b) Ability to Address Identified Issues

One of the main objectives of our hybrid recommendation system was to overcome known challenges in recommender systems, such as the data sparsity, and model imbalance. The experimental results and recommendation behavior confirmed that our proposed design effectively addressed these issues.

Data Sparsity Collaborative filtering models such as NCF typically suffer from sparsity when user-item interaction matrices are mostly empty. By integrating SBERT and BERT4Rec, the hybrid model was less reliant on explicit user interactions. BERT4Rec exploited sequential patterns from even short user histories, and SBERT provided semantic support where interaction data was missing. This combination significantly improved coverage and relevance in sparse areas of the dataset.

Model Weighting and Hybrid Balance Another challenge was to ensure fair contribution from each model component. The use of tuned hybrid weights (*bert_weight*, *ncf_weight*) allowed dynamic balance between collaborative, sequential, and semantic signals. This tuning resulted in better performance on ranking metrics and more meaningful, diverse recommendations. The system was neither too biased toward popular items (as with pure collaborative models) nor overly semantic (as with content-only approaches).

System Stability and Efficiency Robust error handling and GPU-accelerated computation ensured that the system could handle large-scale input and diverse scenarios without crashes or unacceptable latency. The implementation effectively avoided instability when encountering unknown users or items, thus improving reliability during inference.

Overall, the hybrid system not only improved quantitative performance but also proved to be more resilient, adaptive, and user-aware, especially in real-world conditions like data sparsity, and sequential behavior modeling.

4.5.3 Concrete Examples of Recommendations

To illustrate the effectiveness of the hybrid recommendation system, we present real test results from the evaluation phase. These examples highlight the model's performance in predicting relevant trips based on user history.

Result: The hybrid model is performing well, achieving a **Hit@5 score of 0.8739**, indicating that the correct recommendation appears in the top 5 suggestions for 87.5% of users.

Sample Results (Top-5 Recommendations for Selected Users)

Table 4.9: Hybrid model Top-5 trip predictions for sample users

User ID	Ground Truth Trip ID	Hybrid Top-5 Predictions	Match
18	1296	[1296, 25854, 6272, 6648, 2594]	✓ SAME
30	30815	[30815, 2102, 198, 7394, 26702]	✓ SAME
4	28858	[26440, 26561, 15542, 19307, 27530]	× DIFFERENT
46	28987	[28987, 24237, 5964, 5873, 29865]	✓ SAME
59	25690	[25690, 16433, 3230, 29959, 3508]	✓ SAME

Analysis of Recommendation Results

The table above summarizes the hybrid model's Top-5 recommendations for a subset of users, comparing them to the ground truth trip ID. Out of five examples:

- **4 out of 5 users** (80%) had their true trip correctly included in the Top-5 predictions.
- The global metric **Hit@5 = 0.87** is confirmed by these representative examples.
- Only **User 4** had a mismatch, indicating a failure case worth analysis.

4.5.4 Strengths and Limitations

a) Strengths and Successful Aspects of the System

The proposed hybrid recommendation system demonstrated several strong points

- **Complementary Model Integration:** By combining BERT4Rec, NCF, and SBERT, the system effectively leveraged sequential user behavior, collaborative interactions, and semantic information. This allowed it to generate more accurate and contextually relevant recommendations.
- **Robust Ranking Performance:** The system achieved strong results in terms of ranking metrics such as NDCG@K, Hits@K, and MRR, especially on validation data.
- **Scalability:** Leveraging GPU acceleration and batch processing enabled the system to handle large-scale data efficiently during both training and inference phases.
- **Semantic Flexibility:** SBERT allowed the system to understand and process natural language queries or descriptive user preferences, making it adaptable beyond strict ID-based models.

b) Limitations and Areas for Improvement

Despite these strengths, the system has certain limitations:

- **High Computational Cost:** Training deep models like BERT4Rec and SBERT is resource-intensive, which may limit real-time or frequent retraining in production settings.
- **Manual Weight Tuning:** The hybrid weights (e.g., **bert_weight**, **ncf_weight**) were selected manually via grid search. A learned or dynamic weighting approach might yield better results.
- **Dependency on Text Quality:** SBERT's effectiveness is closely tied to the quality of the trip descriptions. Poorly written or missing metadata can reduce recommendation quality.
- **Lack of Feedback Loop:** The current system does not incorporate real-time user feedback, which limits its ability to adapt to evolving user preferences.

c) Failure Cases and Unexpected Results

During testing, several failure cases and anomalies were observed:

- **Vague Queries:** When users input vague descriptions like "nice trip", SBERT produced generic or irrelevant recommendations due to lack of context.
- **Popularity Bias in NCF:** In cold-start situations where SBERT was weak, the system defaulted to recommending the most popular trips based on NCF, which may not reflect the user's intent.
- **Overfitting in BERT4Rec:** BERT4Rec occasionally overfit users with short histories. This required fine-tuning of dropout and regularization parameters.
- **Score Imbalance:** The difference in scale between SBERT and NCF scores sometimes led to unstable fusion. Normalization techniques were required to balance model contributions.

4.5.5 Research Questions and Hypotheses

The experimental phase of this study was designed to address the research questions formulated in the early stages of the project. The primary question concerned whether the integration of sequential, semantic, and collaborative signals could improve recommendation performance in the travel domain. The results demonstrated that the proposed hybrid model, which combines BERT4Rec for sequence modeling, SBERT for semantic reranking, and NCF for collaborative filtering, significantly outperforms individual models. This confirms the effectiveness of leveraging complementary modeling techniques to enhance accuracy, diversity, and personalization.

A second research question focused on the contribution of SBERT to improving semantic relevance in content-based recommendation. The findings validated that SBERT embeddings successfully enhanced recommendation quality, particularly in cases where user history was limited, by capturing the contextual meaning of trip descriptions and enabling more meaningful comparisons.

Furthermore, the integration of Neural Collaborative Filtering (NCF) addressed the third research question, which examined whether incorporating user ratings could further personalize recommendations. The experimental outcomes showed that NCF enriched the hybrid model by modeling implicit user preferences, especially in situations with rich interaction data.

Based on these investigations, the initial hypotheses proposed in this study were validated. First, it was confirmed that a hybrid recommendation approach combining BERT4Rec, SBERT, and NCF yields superior results compared to single-model baselines. Second, SBERT was shown to improve content-aware ranking through its semantic encoding capabilities. Third, the addition of NCF enabled the system to capture personalized preferences from user ratings, leading to more accurate and user-centric recommendations.

4.6 User Interface Implementation

As part of the final implementation, we designed an interactive user interface using the Gradio library to visualize the recommendations generated by our hybrid system. This interface allows the user to input their unique user ID and retrieve the top 5 personalized travel recommendations. Each recommendation is enriched with detailed information extracted from the dataset, including the airline used, the name of the suggested hotel, and a point of interest (POI) associated with the trip. Additionally, the interface displays the corresponding hotel images to enhance the user experience and provide a more intuitive understanding of the recommended trips. The recommendations are based on a precomputed ranking stored in a CSV file, which is the result of the hybridization process combining multiple models. Another CSV file is used to map the trip IDs to their descriptive metadata. This tool not only demonstrates the effectiveness of the recommendation system but also serves as a practical application for end-users to interact with and explore the outcomes of our work.

The following figures illustrate the interactive user interface and its functionalities:

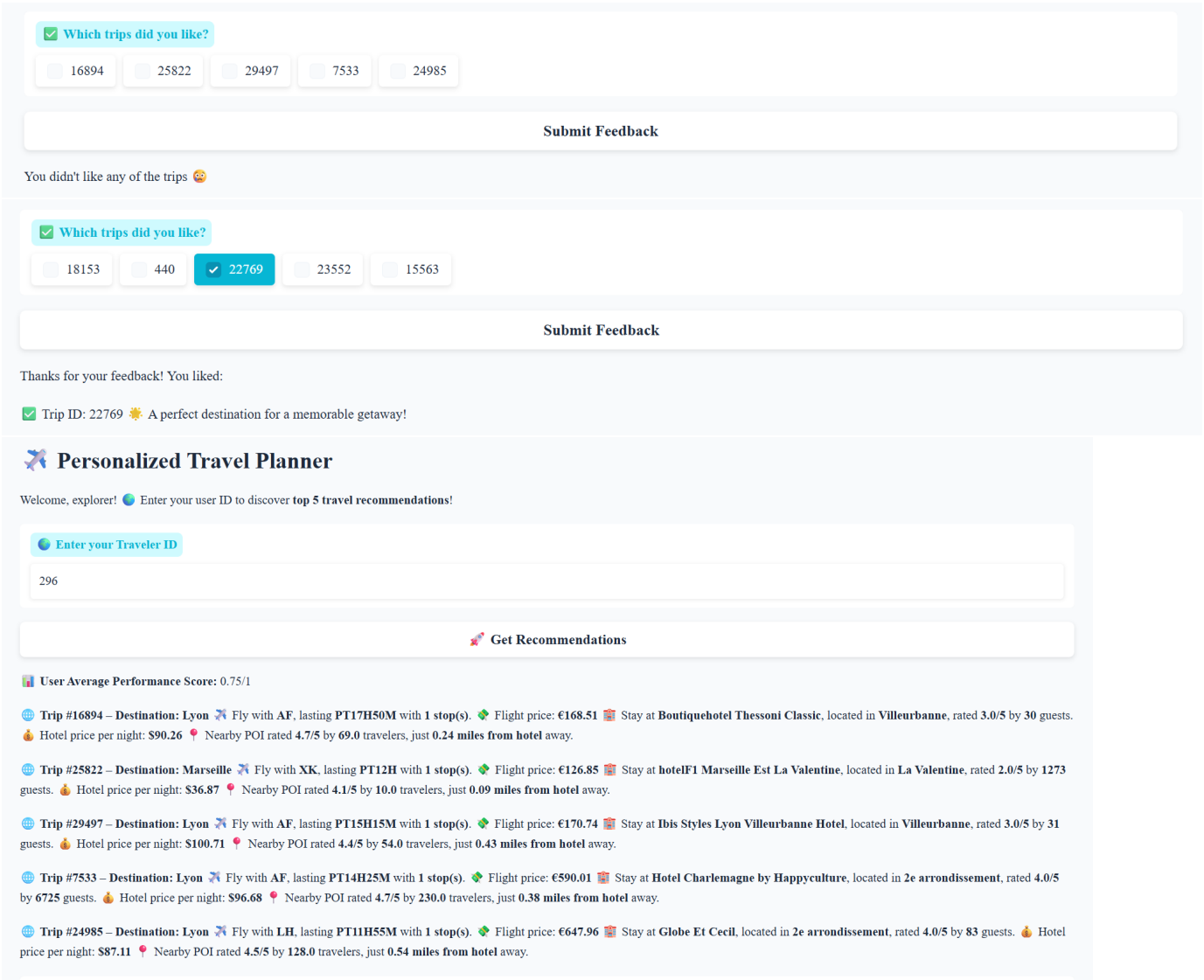


Figure 4.6: Example of the user interface for trip rating and feedback

4.7 Conclusion

This chapter has detailed the complete development and experimental evaluation of the proposed hybrid travel recommendation system. We began by outlining the development environment, including the programming languages, libraries, and hardware used. The system was implemented by integrating multiple components—collaborative filtering, content-based methods, and a hybridization strategy—each designed to address specific challenges such as data sparsity and the cold-start problem.

The experimentation phase involved a rigorous evaluation of model performance using standard recommendation metrics. Through both quantitative and qualitative analyses, we demonstrated that the hybrid approach yields significant improvements over individual baselines in terms of accuracy, relevance, and diversity of recommendations. The results confirm that combining multiple recommendation strategies can lead to a more robust and user-centric system.

Despite its strengths, the system presents areas for improvement, such as optimizing computational efficiency and further refining recommendations for edge cases. Nonetheless, the overall performance and flexibility of the approach validate the core objectives of this work and offer a promising direction for building intelligent, adaptive recommendation systems in the travel domain.

This chapter thus concludes the study by demonstrating that the proposed hybrid model not only meets its initial goals but also provides a solid foundation for future innovation in personalized recommendation technologies.

General conclusion

This project tackled the challenge of improving travel recommendations by developing an innovative hybrid recommendation system that combines collaborative and content-based techniques. The primary goal was to address common limitations, such as data sparsity and the cold-start problem, while delivering more accurate, diverse, and personalized suggestions.

To achieve this, we designed a modular and extensible hybrid architecture that integrates state-of-the-art models, including NCF, BERT4Rec, and SBERT. The system was implemented using robust machine learning frameworks and evaluated on a real-world travel dataset across multiple metrics. The results demonstrated significant improvements, with the hybrid system outperforming baseline models by achieving a 20

From a theoretical perspective, this project contributes a flexible hybridization strategy and demonstrates its applicability in the travel domain. On a practical level, the modular design and experimental framework provide a foundation for future deployment or adaptation to other recommendation scenarios, such as e-commerce, education, or entertainment.

However, this work is not without limitations. The models rely on high-quality data and are computationally intensive during training. Additionally, the system lacks contextual awareness and explainability, which are crucial for building user trust and engagement. Addressing these aspects presents exciting opportunities for future research.

Several directions can be explored moving forward. First, integrating contextual features such as user location, time, and device type could improve precision and adaptability. Second, developing explainable recommendation mechanisms could enhance transparency and trust. Ethical considerations, including fairness and bias mitigation,

also warrant attention. Finally, testing the system in other domains could assess its generalizability and scalability.

In conclusion, this work demonstrates the transformative potential of hybrid recommendation systems in addressing real-world challenges, paving the way for future advancements in building intelligent, adaptive, and user-centric solutions.

Bibliography

[1] J. A. A survey on recommender system. *International Journal of Computer Science and Information Security (IJCSIS)*. Accessed: 2025-06-17.

[2] ESTANI Imen and BENADJROUDA Noureddine. Développement des systèmes de recommandation de santé à l'aide des réseaux convolutifs graphiques. Mémoire de master, UNIVERSITÉ IBN KHALDOUN - TIARET, 2023.

[3] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Neural collaborative filtering. In *Proceedings of the 26th International Conference on World Wide Web*, pages 173–182. International World Wide Web Conferences Steering Committee, 2017.

[4] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. Bert4rec: Sequential recommendation with bidirectional encoder representations from transformer. In *Proceedings of the 28th ACM international conference on information and knowledge management*, pages 1441–1450, 2019.

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

[6] Farhad Mortezapour Shiri, Thinagaran Perumal, Norwati Mustapha, and Raihani Mohamed. A comprehensive overview and comparative analysis on deep learning models: Cnn, rnn, lstm, gru. *arXiv preprint arXiv:2305.17473*, 2023.

[7] Francesco Ricci, Lior Rokach, and Bracha Shapira. Introduction to recommender systems handbook. pages 1–35, 2010.

[8] Paul Resnick, Nick Iacovou, M. Suchak, Peter Bergstrom, and John Riedl. Grouplens: An open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work*, pages 175–186, 1994.

[9] Greg Linden, Brent Smith, and Jennifer York. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing*, 7(1):76–80, 2003.

[10] David Goldberg, David Nichols, Brian M Oki, and Douglas Terry. Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, 35(12):61–70, 1992.

[11] Zhe Lin, Chao Yang, Fang Wu, Jian Wu, Xiaoyan Li, and Yiqun Xiong. Harnessing the power of large language models in recommendation: A survey. *arXiv preprint arXiv:2306.05817*, 2023.

[12] Prem Melville and Vikas Sindhwani. Recommender systems. pages 829–838, 2011.

[13] Yehuda Koren, Robert M. Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *IEEE Computer*, 42(8):30–37, 2009.

[14] Dirk Bollen, Bart P. Knijnenburg, M. C. Willemsen, and M. Graus. Understanding choice overload in recommender systems. In *Proceedings of the Fourth ACM Conference on Recommender Systems*, pages 63–70, Barcelona, Spain, September 2010.

[15] Paul Resnick and Hal R. Varian. Recommender systems. *Communications of the ACM*, 40(3):56–58, 1997.

[16] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th International Conference on World Wide Web (WWW '01)*. ACM, 2001.

[17] Dietmar Jannach, Markus Zanker, Alexander Felfernig, and Gerhard Friedrich. *Recommender Systems: An Introduction*. Cambridge University Press, 2010.

[18] Gawesh Jawaheer, Martin Szomszor, and Patty Kostkova. Comparison of implicit and explicit feedback from an online music recommendation service. In *proceedings of the 1st international workshop on information heterogeneity and fusion in recommender systems*, pages 47–51, 2010.

[19] Lukas Wegmeth, Tobias Vente, and Joeran Beel. Recommender systems algorithm selection for ranking prediction on implicit feedback datasets. In *Proceedings of the 18th ACM Conference on Recommender Systems*, pages 1163–1167, 2024.

[20] Douglas W. Oard and Jinmook Kim. Implicit feedback for recommender systems. page 81, 1998.

[21] Harald Steck. Evaluation of recommendations: rating-prediction and ranking. In *Proceedings of the 7th ACM Conference on Recommender Systems (Hong Kong, China) (RecSys '13)*, pages 213–220, New York, NY, USA, 2013. Association for Computing Machinery.

[22] Mohammed Tadlaoui. *Système de recommandation de ressources pédagogiques fondé sur les liens sociaux : formalisation et évaluation*. Thèse de doctorat, Université de Lyon, INSA Lyon (en partenariat avec l'Université de Tlemcen), 2018.

[23] ScienceDirect. Cold start problem. https://www.sciencedirect.com/topics/computer-science/cold-start-problem?utm_source=chatgpt.com. Accessed: 2025-05-03.

[24] Òscar Celma. *Music recommendation and discovery in the long tail*. Springer, 2010.

[25] Rainer Gemulla, Erik Nijkamp, Peter J. Haas, and Yannis Sismanis. Large-scale matrix factorization with distributed stochastic gradient descent. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 69–77, 2011.

[26] Denis Kotkov, Shuaiqiang Wang, and Jari Veijalainen. A survey of serendipity in recommender systems. *Knowledge-Based Systems*, 111:180–192, 2016.

[27] Tao Zhou, Zolt'an Kuscsik, Jian-Guo Liu, Mat'uš Medo, Joseph R Wakeling, and Yi-Cheng Zhang. Solving the apparent diversity-accuracy dilemma of recommender systems. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 1(1):1–25, 2010.

[28] Gediminas Adomavicius and Alexander 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):734–749, 2005.

[29] Robin Burke. Hybrid recommender systems: Survey and experiments. *User modeling and user-adapted interaction*, 12(4):331–370, 2002.

[30] Xiaoyuan Su and Taghi M. Khoshgoftaar. A survey of collaborative filtering techniques. *Advances in Artificial Intelligence*, 2009:1–19, 2009.

[31] Umair Javed, Kamran Shaukat, Ibrahim A Hameed, Farhat Iqbal, Talha Mahboob Alam, and Suhuai Luo. A review of content-based and context-based recommendation systems. *International Journal of Emerging Technologies in Learning (iJET)*, 16(3):274–297, 2021.

[32] Xiaoyuan Su and Taghi M Khoshgoftaar. A survey of collaborative filtering techniques. *Advances in artificial intelligence*, 2009:1–19, 2009.

[33] A. Sieg, B. Mobasher, and R. D. Burke. Learning ontology-based user profiles: A semantic approach to personalized web search. *IEEE Intelligent Informatics Bulletin*, 8(1):7–18, 2007.

[34] A. R. D. M. Neves, Á. M. G. Carvalho, and C. G. Ralha. Agent-based architecture for context-aware and personalized event recommendation. *Expert Systems with Applications*, 41(2):563–573, 2014.

[35] Poonam B. Thorat, R. M. Goudar, and Sunita Barve. Survey on collaborative filtering, content-based filtering and hybrid recommendation system. *International Journal of Computer Applications*, 110(4), 2015.

[36] Michael Pazzani and Daniel Billsus. Content-based recommendation systems. In *The adaptive web*, pages 325–341. Springer, 2007.

[37] Pasquale Lops, Marco de Gemmis, and Giovanni Semeraro. Content-based recommender systems: State of the art and trends. In *Recommender systems handbook*, pages 73–105. Springer, 2011.

[38] Gediminas Adomavicius and Alexander 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):734–749, 2005.

[39] Sarah Bouraga, Chayma Zayani, and Henda Ben Ghezala. Knowledge-based recommendation systems: A survey. *Proceedings of the 27th International Conference on Tools with Artificial Intelligence*, 2015.

[40] Derek Bridge, Mehmet H. Göker, Lorraine McGinty, and Barry Smyth. Case-based recommender systems. volume 20, pages 315–320. Cambridge University Press, 2006. Printed in the United Kingdom.

[41] Abdus Salam Khan and Achim Hoffmann. Building a case-based diet recommendation system without a knowledge engineer, 2003.

[42] Alexander Felfernig, Klaus Isak, Kresimir Szabo, and Peter Zachar. The VITA financial services sales support environment. In *Proceedings of the 22nd AAAI Conference on Artificial Intelligence and the 19th Innovative Applications of Artificial Intelligence Conference (AAAI/IAAI 2007)*, pages 1692–1699, Vancouver, Canada, 2007. AAAI Press.

[43] Ben Jiang, Weiquan Wang, and Izak Benbasat. Multimedia-based interactive advising technology for online consumer decision support. *Communications of the ACM*, 48(9):93–98, 2005.

[44] Alexander Felfernig. Standardized configuration knowledge representations as technological foundation for mass customization. *IEEE Transactions on Engineering Management*, 54(1):41–56, 2007.

[45] Alexander Felfernig and Robin Burke. Constraint-based recommender systems: Technologies and research issues. *Intelligent Systems and Business Informatics*, 2023. Categories and Subject Descriptors: 1.2.5 [Expert system tools and techniques].

[46] Nielsen. Global trust in advertising: Winning strategies for an evolving media landscape, 2015. Available: https://www.nielsen.com/us/en/insights/report/2015/global-trust-in-advertising-2015/.

[47] J. Kamahara, T. Asakawa, S. Shimojo, and H. Miyahara. A community-based recommendation system to reveal unexpected interests. In *11th International Multimedia Modelling Conference*, pages 433–438, 2005.

[48] Gediminas Adomavicius and Alexander Tuzhilin. Context-aware recommender systems. In *Recommender systems handbook*, pages 217–253. Springer, 2011.

[49] Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. How to fine-tune bert for text classification? *arXiv preprint arXiv:1905.05583*, 2019.

[50] Jesús Bobadilla, Fernando Ortega, Antonio Hernando, and Abraham Gutiérrez. Recommender systems survey. *Knowledge-based systems*, 46:109–132, 2013.

[51] Francesco Ricci, Lior Rokach, and Bracha Shapira. *Recommender systems handbook*. Springer, 2015.

[52] Guy Shani and Asela Gunawardana. Evaluating recommendation systems. In *Recommender systems handbook*, pages 257–297. Springer, 2011.

[53] Mohamed Mouhiha, Omar Ait Oualhaj, and Abdelfettah Mabrouk. Combining collaborative filtering and content based filtering for recommendation systems. pages 1–6, 07 2024.

[54] Yaming Cai, Jia Huang, and Jun Luo. Research paper recommendation system based on multiple features from citation network. *Scientometrics*, pages 1–19, 2024.

[55] Rui Bernardes. Machine learning - basic principles. *Acta Ophthalmologica*, 2024.

[56] Michael I. Jordan and T. Mitchell. Machine learning: Trends, perspectives, and prospects. *Science*, 349:255 – 260, 2015.

[57] S. Uddin and Haohui Lu. Dataset meta-level and statistical features affect machine learning performance. *Scientific Reports*, 14, 2024.

[58] Ze-Sen Chen and Min-Ling Zhang. Multi-label learning with regularization enriched label-specific features. In *Proceedings of The Eleventh Asian Conference on Machine Learning*, pages 411–424. PMLR, 2019.

[59] Xiu-Yi Jia, Sai-Sai Zhu, and Wei-Wei Li. Joint label-specific features and correlation information for multi-label learning. *Journal of Computer Science and Technology*, 35(2):247–258, 2020.

[60] Muthuramalingam Sivakumar, Sudhaman Parthasarathy, and Thiyagarajan Padmapriya. Trade-off between training and testing ratio in machine learning for medical image processing. *PeerJ Computer Science*, 10, 2024.

[61] P'adraig Cunningham, Matthieu Cord, and Sarah Jane Delany. Supervised learning. In *Machine learning techniques for multimedia: case studies on organization and retrieval*, pages 21–49. Springer, 2008.

[62] Bing Liu. Supervised learning. In *Web Data Mining: Exploring Hyperlinks, Contents, and Usage Data*, pages 63–132. Springer, 2011.

[63] K Sindhu Meena and S Suriya. A survey on supervised and unsupervised learning techniques. pages 627–644, 2020.

[64] Ramadass Sathya, Annamma Abraham, et al. Comparison of supervised and unsupervised learning algorithms for pattern classification. *International Journal of Advanced Research in Artificial Intelligence*, 2(2):34–38, 2013.

[65] Charu C. Aggarwal. *Recommender Systems: The Textbook*. Springer, 2016.

[66] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. *Proceedings of the Twenty-Fifth Conference on Neural Information Processing Systems (NIPS 2012)*, 2012.

[67] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

[68] Robin Burke. Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction*, 12, 11 2002.

[69] Dietmar Jannach, Markus Zanker, Alexander Felfernig, and Gerhard Friedrich. *Recommender Systems: An Introduction*. Cambridge University Press, 2010. Free preprint PDF available at https://pzs.dstu.dp.ua/DataMining/recom/bibl/1jannach_dietmar_zanker_markus_felfernig_alexander_friedrich.pdf (accessed 2025-06-17).

[70] Jure Leskovec, Anand Rajaraman, and Jeffrey D. Ullman. *Mining of Massive Datasets*. Cambridge University Press, 3rd edition, 2020. Unofficial full PDF available at https://mrce.in/ebooks/Mining%20of%20Massive%20Datasets%203rd%20Ed.pdf (accessed 2025-06-17).

[71] Pang-Ning Tan, Michael Steinbach, Anuj Karpatne, and Vipin Kumar. *Introduction to Data Mining*. Pearson, 2nd edition, 2019. Free full-text PDF available at https://www.ceom.ou.edu/media/docs/upload/Pang-Ning_Tan_Michael_Steinbach_Vipin_Kumar_-_Introduction_to_Data_Mining-Pe_NRDK4fi.pdf (accessed 2025-06-17).

[72] Robin Burke. Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction*, 12, 11 2002.

[73] T. Cover and P. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27, 1967.

[74] H. Guo, R. Tang, Y. Ye, F. Liu, and Y. Zhang. An adjustable heat conduction based knn approach for session-based recommendation. *arXiv preprint arXiv:1807.05739*, 2018.

[75] O. Harrison. Machine learning basics with the k-nearest neighbors algorithm. https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761, 2019.

[76] Muhao Hu, Wufan Xiao, and Yuzhang Li. Exploring the relationship between users characteristics and movie recommendations using a knn-based recommender system. volume 44, pages 114–123, 03 2024.

[77] Satya Mallick. Recommendation system. https://learnopencv.com/recommendation-system/, 2020. Accessed: 2025-06-17.

[78] Aasif Mujawar. Classification based collaborative filtering: A machine learning recommendation system. *International Journal of IT, Research & Applications*, 2022.

[79] Jinxiong Zhang. Dive into decision trees and forests: A theoretical demonstration. *arXiv preprint arXiv:2101.08656*, 2021.

[80] IBM. Random forests improve accuracy and robustness over single decision trees by reducing overfitting and increasing generalization. they are effective in handling high-dimensional data and can identify important features for recommendations. https://www.ibm.com/think/topics/random-forest, 2023. Accessed: 2025-04-29.

[81] Daehan Jang. Svm for recommendation systems: Unveiling the power of support vector machines in personalized recommendations. https://medium.com/@jangdaehan1/svm-for-recommendation-systems-unveiling-the-power-of-support-vector-machines-in-persona 2024. Accessed: 2025-05-29.

[82] Pasquale Lops, Marco de Gemmis, and Giovanni Semeraro. Content-based recommender systems: State of the art and trends. In *Recommender Systems Handbook*, pages 73–105. Springer, 2011.

[83] V. Nakhipova, A. Alimurat, and Y. Kydyraliev. Integration of collaborative filtering into naive bayes method to enhance student performance prediction. *International Journal of Information and Communication Technology Education*, 20(1), 2024.

[84] A. Mujawar. Classification based collaborative filtering: A machine learning recommendation system. https://www.academia.edu/88532209, 2022. Accessed: 2025-05-29.

[85] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794. ACM, 2016.

[86] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qi Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems*, volume 30, pages 3146–3154, 2017.

[87] Reham Alabduljabbar. Matrix factorization collaborative-based recommender system for riyadh restaurants: Leveraging machine learning to enhance consumer choice. *Applied Sciences*, 2023.

[88] Reham Alabduljabbar. Matrix factorization collaborative-based recommender system for riyadh restaurants: Leveraging machine learning to enhance consumer choice. *Applied Sciences*, 2023.

[89] Yubing Yan, Camille Moreau, Zhuoyue Wang, Wenhan Fan, and Chengqian Fu. Transforming movie recommendations with advanced machine learning: A study of nmf, svd, and k-means clustering. *arXiv preprint arXiv:2407.08916*, 2024. Accepted at the 2024 4th International Symposium on Computer Technology and Information Science (ISCTIS); free full-text PDF available at https://arxiv.org/pdf/2407.08916.pdf (accessed 2025-06-17).

[90] Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. Deep learning based recommender system: A survey and new perspectives. *ACM Computing Surveys (CSUR)*, 52(1):1–38, 2019.

[91] Ethem Alpaydin. Introduction to machine learning. *MIT Press*, 2020.

[92] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[93] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[94] Nitish Shirish Keskar, Jorge Nocedal, et al. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*, 2016.

[95] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

[96] Nitish Srivastava, Geoffrey Hinton, et al. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.

[97] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. 2010.

[98] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *Proceedings of the IEEE International Conference on Computer Vision*, pages 1026–1034, 2015.

[99] Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 807–814, 2010.

[100] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. *Proceedings of the 30th International Conference on Machine Learning*, 2013.

[101] Andrew Y Ng. Feature selection, l1 vs. l2 regularization, and rotational invariance. *Proceedings of the Twenty-First International Conference on Machine Learning*, 2004.

[102] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. *Proceedings of the 30th International Conference on Machine Learning*, 2013.

[103] Lutz Prechelt. Early stopping – but when? *Neural Networks: Tricks of the Trade*, pages 55–69, 1998.

[104] Simon Haykin. *Neural Networks: A Comprehensive Foundation*. Prentice Hall, 1994.

[105] Yann LeCun, Léon Bottou, Genevieve B Orr, and Klaus-Robert Müller. Efficient backprop. *Neural Networks: Tricks of the Trade*, pages 9–50, 1998.

[106] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. Rectifier nonlinearities improve neural network acoustic models. *Proceedings of the 30th International Conference on Machine Learning*, 2013.

[107] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. ResearchGate, 2006.

[108] Ibomoiye Domor Mienye and Theo G. Swart. A comprehensive review of deep learning: Architectures, recent advances, and applications. *Information*, 15(12):755, 2024.

[109] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408, 1958.

[110] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[111] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[112] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint*, arXiv:1406.1078, 2014.

[113] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, and I. Polosukhin. Attention is all you need. *arXiv preprint*, arXiv:1706.03762, 2017.

[114] Le Wu, Xiangnan He, Xiang Wang, Kun Zhang, and Meng Wang. A survey on accuracy-oriented neural recommendation: From collaborative filtering to information-rich recommendation. *ACM Transactions on Information Systems (TOIS)*, 39(4):1–47, 2021.

[115] Florian Strub, Jérémie Mary, and Romain Gaudel. Autoencoder-based collaborative filtering. In *Proceedings of the 5th International Conference on Machine Learning and Applications (ICMLA)*, pages 284–291. Springer, 2016.

[116] Ye Xiang, Zihan Wang, Wayne Xin Zhao, Di Wu, Min Zhang, and Shaoping Ma. Recobert: A catalog-enhanced bert-based recommender. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1837–1840, 2020.

[117] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3982–3992, 2019.

[118] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. Bert4rec: Sequential recommendation with bidirectional encoder representations from transformer. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management (CIKM)*, pages 1441–1450. ACM, 2019.

[119] Hongwei Wang, Fuzheng Zhang, Xing Xie, and Minyi Guo. A survey on deep learning for recommender systems: From collaborative filtering to content-aware recommendation. *IEEE Transactions on Knowledge and Data Engineering*, 2021.

[120] Carlos A. Gómez-Uribe and Neil Hunt. The netflix recommender system: Algorithms, business value, and innovation. *ACM Transactions on Management Information Systems (TMIS)*, 6(4):1–19, 2016.

[121] Paul Covington, Jay Adams, and Emre Sargin. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems*, pages 191–198, 2016.

[122] A. Van den Oord, S. Dieleman, and K. Simonyan. Deep content-based music recommendation. In *Proceedings of the 26th International Conference on Neural Information Processing Systems*, pages 2643–2651, 2013.

[123] Yi Zhang, Lei Wang, and Qian Li. A deep learning framework for music recommendation. In *Proceedings of the 2017 IEEE International Conference on Data Mining*, pages 1091–1096, 2017.

[124] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. Bert4rec: Sequential recommendation with bidirectional encoder representations from transformer. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management (CIKM)*, pages 1441–1450. ACM, 2019.