PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA
الجمهـــوريـــــة الجزائـريـــــة الديـمقـراطيـــــة الشـعـبيـــــة
MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC RESEARCH
وزارة التعليـم العالـــي والبحـــث العلمـي
HIGHER SCHOOL OF COMPUTER SCIENCE AND DIGITAL TECHNOLOGIES-BEJAIA
المـــدرسة العـــليا في علـــوم وتكنـولوجيـات الاعـــلام الالـــي بجـــاية

Dissertation Submitted to the Department Of Computer Science in Partial Fulfillment of the Requirements for Master's Degree in Computer Science

Specialty: Artificial Intelligence and Data Sciences

Submitted By:

TOUATI Yanis

State-of-the-Art Analysis : Exploring the Integration of Visual Data for Enhancing the Accuracy and Reliability of Recommender Systems

Supervised by:

Dr. BERBAGUE Chemseddine
LITAN Laboratory
HIGHER SCHOOL OF COMPUTER SCIENCE AND DIGITAL TECHNOLOGIES-BEJAIA (ESTIN)

Members of jury:

▪ **Dr.** ISSAADI Badredine	President	LITAN Laboratory - ESTIN
▪ **Mrs.** BOUGLIMINA Ouahiba	Examiner	LITAN Laboratory - ESTIN
▪ **Mr.** KADRI Belkacem	Examiner	LITAN Laboratory - ESTIN
▪ **Mrs.** KHENNOUCHE Feriel	Examiner	LITAN Laboratory - ESTIN

Academic year: 2023/2024

Acknowledgements

I express my warmest gratitude and love to every person that brought support and guidance to me throughout my academic journey and made this thesis possible.

First and foremost, I thank Allah for providing me with the strength, patience, and wisdom to pursue my studies and complete this thesis. Without His guidance and blessings, none of this would have been possible.

I am extremely thankful to my parents, whose unconditional support, encouragement, and love have been the pillar of my education. Their belief in me has been a constant source of motivation.

To my sister and my dear friend Mohammed, I extend my heartfelt thanks for always being there for me, offering advice, and cheering me on through every step of this journey.

I am deeply indebted to my supervisor, whose guidance, insights, and patience have been invaluable. Your mentorship has significantly shaped my academic path and contributed greatly to the completion of this thesis.

I also wish to acknowledge every teacher and supervisor I have had since the beginning of my university education. This work is a testament to the knowledge and skills I have acquired under your tutelage. Your dedication to teaching and your commitment to my learning have been instrumental in my academic growth.

Yanis

Abstract

This dissertation presents a comprehensive analysis of recommender systems, focusing on the integration of visual data to enhance recommendation accuracy and reliability. Chapter One provides a General introduction to recommender systems, emphasizing their importance in our nowadays technologies. Chapter Two explores the state-of-the-art in recommender systems, discussing various methodologies and advancements, with a particular focus on the incorporation of visual data. The chapter highlights the transformative potential of visual data integration in shaping personalized recommendation experiences across different platforms and domains and then we conclude the study by synthesizing the the summarizing of the State-of-the-Art and the challenges associated with visual data integration.

Key-words: *Artficial Intelligence, Collaborative Filtering, Visual Data Integration, Recommender Systems, Content-Based Recommendation, Feature Extraction.*

Résumé

Cette dissertation présente une analyse complète des systèmes de recommandation, en se concentrant sur l'intégration de données visuelles pour améliorer la précision et la fiabilité des recommandations. Le premier chapitre fournit une introduction générale aux systèmes de recommandation, soulignant leur importance dans nos technologies actuelles. Le second chapitre se base sur l'état de l'art des systèmes de recommandation, et des méthodologies récentes et traditionnelles, avec un accent particulier sur l'incorporation de données visuelles. Le chapitre met en évidence le potentiel transformateur de l'intégration des données visuelles dans l'élaboration d'expériences de recommandation personnalisées sur différentes plates-formes et domaines, puis nous concluons l'étude en synthétisant les résultats et en discutant des orientations de recherche futures dans le domaine de la technologie de recommandation, en mettant l'accent sur les défis et les opportunités associés à l'intégration de données visuelles.

Mots-clés : *Intelligence Artificielle, Filtrage Collaboratif, Intégration de Données multimédias(images), Systèmes de Recommendation, recommandation basée sur le contenu, extraction de charactéristiques.*

ملخص

تقدم هذا المذكرة تحليلاً شاملاً لأنظمة التوصية، مع التركيز على تكامل البيانات المرئية لتعزيز دقة التوصية وموثوقيتها. يقدم الفصل الأول مقدمة عامة لأنظمة التوصية، مع التركيز على أهميتها في تقنياتنا الحالية. يستكشف الفصل الثاني أحدث ما توصلت إليه أنظمة التوصية، ويناقش مختلف المنهجيات والتطورات، مع التركيز بشكل خاص على دمج البيانات المرئية. يسلط الفصل الضوء على الإمكانات التحويلية لتكامل البيانات المرئية في تشكيل تجارب التوصية الشخصية عبر منصات ومجالات مختلفة، ثم نختتم الدراسة من خلال تجميع النتائج ومناقشة اتجاهات البحث المستقبلية في مجال تكنولوجيا التوصية، مع التركيز على التحديات والفرص المرتبطة بالبصرية.

الكلمات المفتاحية:الذكاء الاصطناعي، التصفية التعاونية، تكامل البيانات المرئية، أنظمة التوصية، التوصية المستندة إلى المحتوى، استخلاص الميزات، التعلم الآلي.

Contents

List of Figures

List of Tables

List of Acronyms

AI Artificial Intelligence

ML Machine Learning

DL Deep Learning

RS Recommender System

CF Collaborative Filtering

CBF Content-based Filtering

SVD Singular Value Decomposition

IbCF Item-based Collaborative Filtering

UbCF User-based Collaborative Filtering

MBM Memory-Based Methods

SVM Support Vector Machines

DT Decision Trees

GMM Gaussian Mixture Model

SOM Self Organizing Map

SGAN Semi-Supervised Variational Autoencoders

SSVA Semi-Supervised Generative Adversarial Network

RL Reinforcement Learning

RF Random Forests

NLP Natural Language Processing

kNN k-Nearest Neighbors

GCN Graph Convolutional Network

ANN Artificial Neural Networks

MLP Multilayer Perceptron

GAN Generative Adversarial Network

GAE Graph Autoencoders

VGAE Variational Graph Autoencoder

VAE Variational Autoencoder

AE Autoencoder

WGAN Wasserstein Generative Adversarial Networks

AAE Adversarial Autoencoders

ANN Artificial Neural Networks

CNN Convolutional Neural Networks

RNN Recurrent Neural Networks

GNN Graph Neural Network

LSTM Long Short-Term Memory

SIFT Scale-Invariant Feature Transform

HOG Histogram of Oriented Gradients

PCA Principal Component Analysis

RMSE Root Mean Squared Error

MAE Mean Absolute Error

IoT Internet of Things

TL Transfer Learning

General Introduction

Amid the gigantic wave of digital advancements that define our current times, the depth of our technological abilities has resulted in an era of unprecedented connectivity and data propagation. In this era of digital transformation, the complexity of discerning and meeting changing needs has reached significant dimensions, posing profound challenges for effectively navigating this digital area. Every day, millions of people around the world use a wide range of digital tools, platforms, and applications, collectively generating an astronomical volume of data. Within this vast sea of digital information, each user appears as a distinct entity, characterized by a variety of interactions, preferences, and behaviors. These individual digital footprints, collectively constituting the complex fabric of the digital universe, highlight the unique needs and interests that propel users along their digital journey.

At the heart of orchestrating this digital ecosystem are recommendation systems, sophisticated IT tools that play a central role in guiding users through digital content. Evolving through years of technological innovation and refinement, recommendation systems harness the power of advanced algorithms to analyze large data sets, discern user preferences, and provide personalized content recommendations. As the digital landscape continues to evolve, fueled by unstoppable technological advancements, the imperative to improve and adapt recommendation systems remains crucial.

Visual data, such as images and videos, has become increasingly integral in enhancing the accuracy and reliability of these systems. By incorporating visual data, recommendation systems can capture nuanced user preferences and product characteristics that textual data alone cannot provide. For instance, visual elements can reveal aesthetic preferences, style choices, and detailed product attributes, enriching the personalization process. This integration not only refines the recommendations but also aligns them more closely with user expectations shaped by contemporary digital culture, where visual content is predominant.

In popular culture, visual data significantly impacts how users interact with digital content. Platforms like Instagram, Pinterest, and YouTube demonstrate the power of visual information in driving user engagement and influencing preferences. As users increasingly consume and create visual content, recommendation systems must adapt by leveraging visual data to provide more contextually relevant and appealing suggestions. This evolution underscores the necessity of integrating diverse data modalities to enhance user experience.

Recommender systems are designed to navigate through vast amounts of data and identify the most relevant information and services for individual users. This personalization is achieved by predicting user preferences through analysis of past behaviors and interactions. Modern recommendation systems utilize a variety of data types, including visual data, to generate suggestions that align with a user's interests and needs.

By leveraging visual data, recommender systems can better understand and anticipate user preferences, resulting in more accurate and engaging recommendations. This approach highlights the ongoing evolution of recommendation technologies and the importance of adapting to the changing digital landscape. Integrating visual data represents a pivotal advancement, promising to further elevate the efficacy and utility of recommendation systems in catering to the diverse and evolving needs of users. Figure 1 illustrates a basic recommender system architecture. It consists of three primary modules: data collection and preprocessing, recommender algorithm, and user interface. The data collection and preprocessing module gathers user-item interaction data, preparing it for analysis by the recommender algorithm. This algorithm leverages the processed data to identify patterns and predict user preferences. Finally, the user interface component presents these recommendations to the user.

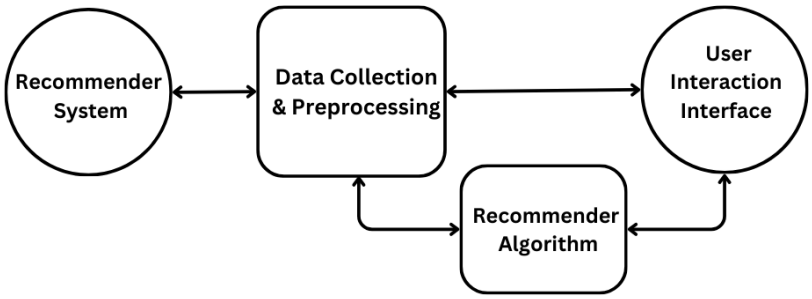

Figure 1: Diagram illustrating the operation of a basic recommendation system

As precised previously, the landscape of recommender systems has witnessed a significant evolution, with emerging technologies enabling the integration of diverse data modalities to enhance recommendation accuracy and user satisfaction. Among these modalities, visual data has emerged as a particularly potent resource for enriching the recommendation process. Visual information, like images, videos, and other multimedia content, offers a nuanced understanding of user preferences and product characteristics that other data sources alone cannot capture. By exploiting visual data, recommender systems can gather subtle insights such as aesthetic preferences, style preferences, and product attributes, thereby facilitating more personalized and contextually relevant recommendations. This integration of visual data represents a pivotal advancement and challenge in recommendation technology, promising to further elevate the efficacy and utility of recommendation systems in catering to the diverse and evolving needs of users.

All that prompts us to inquire what are the primary technologies employed in recommender systems, and how can visual data be integrated to develop a scalable recommender system using techniques such as hybridization? Furthermore, what are the challenges and methodologies to consider in this process?

This dissertation's purpose is to explore the modern environment of recommender systems, examining their techniques and innovations. In the second Chapter we will dive into the state-of-the-art of RSs, studying multiple methodologies. A focal point of this exploration lies in the integration of visual data. By incorporating visual cues such as images and videos, recommender systems stand to refine their understanding of user preferences and product characteristics, ultimately delivering more tailored and engaging recommendations and then we will colcude by synthesizing the findings and implications of the study, particularly focusing on the potential of visual data integration in enhancing recommendation accuracy and user satisfaction and the challenges this field faces.

Chapter 1

Recommender Systems : State of the Art

1.1 Introduction

Recommendation systems are now part of our lives, and they have brought about a great impact ranging from the online products we buy to films and news we follow. There is a lot of information which is used by these systems to give advice on various things used in making the user experience more enjoyable thus enhancing user engagement across multiple platforms. Recent times have seen significant leaps made in the area of recommendation algorithms leading to their development and increased complexity.

This chapter seeks to give a full account of the current state of recommendation technologies. In the historical perspective, we will trace back to how recommendation systems have been developing over a period of time before diving into the most recent methodologies and approaches in the discipline then outline some difficulties as far as tomorrow challenges surrounding recommendation literature.

The subsequent section, "Formal Definition, Core Principles, and Notations," will teach some basic ideas we need to know about recommendation systems and help illustrate the words of key importance, as well as explain the main ideas behind them and introduce symbols used within our document. This laid down concept will ensure that we understand better other topics on advanced materials that will be discussed further in our document and hence develop a complete foundation for studying techniques that are currently being applied in this field.

1.2 Formal Definition, Core Principles and Notations

We introduce the reader to the notation used in this report, thus, we define what RSs are:

A recommender system can be described as a program that strive to recommend the most appropriate articles to specific users. Thus, Recommendation systems have been characterized in various ways, but a widely accepted and comprehensive definition, as proposed by Robin Burke [13], describes them as systems designed to offer personalized recommendations or direct users to interesting or useful resources within a vast data space. In any recommendation system, two core entities represent the driving forces behind these systems: ***users*** and ***items***. In the following, we introduce the notation we adopt throughout our report.

- **User :** is someone who accesses the system and registers by entering their demographic informations or other personal details that constitues its profile and interests. The set of all users in the system is represented by U with an individual user denoted by u, where $u \in U$.
- **Item** : in an entity that satisfies the user's interests(needs). This includes all potentially marketable products (books, merchandise, etc., on e-commerce sites like Amazon.com), viewable content (movies on streaming platforms like Netflix), listened-to content (music), or read content (such as news articles on online newspapers, magazines in digital libraries), as well as vacation destinations, restaurants, etc. The set of items in the system is denoted by I, where $i \in I$.
- **User-Item Interactions (Matrix)** : refers to a matrix of interactions between users and items within a recommendation system. In mathematical terms, it is a $U \times I$ matrix R, where rows represent users, columns represent items, and cells indicate user-item interactions. This matrix models user preferences and is essential for implementing recommendation algorithms.

The user-item interaction matrix in recommender systems can represent various types of interactions between users and items [38]. Here are some common types:

- **Explicit Ratings**: This is the most straightforward type of interaction. Users explicitly express their preference for an item by assigning a rating, typically on a scale (e.g., 1-5 stars, thumbs up/down).
- **Implicit Ratings**: These are interactions that indirectly reflect a user's preference for an item. Examples include:
  - Purchase history: Buying an item indicates a positive interaction.
  - Browsing history: Time spent viewing an item suggests potential interest.
  - Click-through rates: Clicking on an item recommendation shows some level of interest.
  - Adding items to a wishlist: This implies potential purchase intent.
- **Count**: This records the number of times a user interacts with an item. A higher count might suggest a stronger preference, but it doesn't necessarily indicate the nature of the preference (positive or negative).
- **Binary**: This is a simpler version of the count, indicating only whether there was an interaction (1) or no interaction (0).

Interactions types might be depending on on the specific RS and the data we have in our system. Explicit ratings provide the most direct information about user preferences, but they can be rare. Implicit ratings offer a richer data source but require meticulous interpretation to infer user preferences accurately [33]. All recommendation techniques revolve around these entities: users interact with the system, which recommends items to them, and in turn, they provide feedback on those items.

In addition to the conceptual understanding of recommendation systems, they can also be defined formally using mathematical equations. One common formalization is through matrix factorization techniques.

Let U be the ensemble of users in the RS, I as the ensemble of items and R be the user-item interaction matrix, where each entry R_{ui} represents the user-item evaluation(interaction) of user u with item i:

$$R : U \times I \rightarrow R_{u,i} \tag{1}$$

A RS's purpose is to calculate the unknown ratings in the user-item interaction matrix. By estimating these unknown ratings, the system can suggest items that users are likely to appreciate based on their preferences and behavior patterns [7].

User ID	**Feature 1**	**Feature 2**	$\cdots$	**Feature** n
a	0.1	0.2	$\cdots$	0.3
b	0.2	0.3	$\cdots$	0.4
c	0.3	0.4	$\cdots$	0.5
d	0.4	0.5	$\cdots$	0.6

Table 1.1: Example of a User Matrix

Item ID	**Feature 1**	**Feature 2**	...	**Feature** k
a	0.5	0.6	...	0.7
b	0.6	0.7	...	0.8
c	0.7	0.8	...	0.9
d	0.857	0.945	...	1.20

Table 1.2: Item-Matrix Illustrating Example

	Item a	**Item b**	**Item c**	**Item d**
User a	4.45	4.34	5	-
User b	1.5	2	1.7	-
User c	2.45	-	4	2.5
User d	1.5	-	3	2

Table 1.3: Example of User-Item Matrix

Tables 1.1, 1.2, 1.3, show user-item interactions matrix, the user, and item matrices, they represent the format in which the users/items are saved in the RS, the features columns, populated with randomly generated numerical values for the sake of our example, represent attributes related to the users/items in the dataset, they might be demographic, or system-generated attributes related to the items/users, demonstrating how user and item data might be structured. To contextualize this, we can take for instance the following :

User Matrix

$U =$

	Age	Gender	Location
User a	30	M	*Annaba*
User b	31	F	*Setif*
User c	45	M	*Bejaia*

In this matrix, User a is 30 years old, Male (denoted by M), and from Annaba. User b is 31 years old, Female (denoted by F), and from Setif. User c is 45 years old, Male (denoted by M), and from Bejaia. Age, Gender and location represent the features related to the user.

Item Matrix

$I =$

	Genre	Length	Release Year
Movie 1	1	120	2018
Movie 2	2	150	2019
Movie 3	1	90	2017
Movie 4	3	110	2020

In this matrix, Movie 1 is of Genre 1, 120 minutes long, and was released in 2018. Movie 2 is of Genre 2, 150 minutes long, and was released in 2019. Movie 3 is of Genre 1, 90 minutes long, and was released in 2017. Movie 4 is of Genre 3, 110 minutes long, and was released in 2020. Genre, Length and Release Year represent the features related to the item.

User-Item Matrix

$$R = \begin{array}{c|cccc} & \text{Movie 1} & \text{Movie 2} & \text{Movie 3} & \text{Movie 4} \\ \hline \text{User a} & 4.5 & 2.2 & 3.1 & - \\ \text{User b} & 3.3 & - & 4.1 & - \\ \text{User c} & - & 4 & - & 2.1 \end{array}$$

In this example, User a rated Movie 1 with 4.5 stars, Movie 2 with 2.2 stars, and Movie 3 with 3.1 stars. User 2 rated Movie 1 with 3.3 stars and Movie 3 with 4.1 stars. User 3 rated Movie 2 with 4 stars and Movie 4 with 2.1 stars. The '-' means that the user has not given a rating for that specific movie.

These matrices provide a structured way to store and manage user and item data within a recommender system, facilitating the application of various recommendation algorithms to predict user preferences and generate recommendations.

1.3 Recommender Systems : a Brief History

There have been six major phases in the evolution of recommender systems throughout which there have been significant steps and innovations. These systems have always changed to improve personal user experience as early as simple information retrieval algorithms up to AI driven modern systems in use today. This part sheds light into the major events in the development of recommendation engines.

1970s – 1980s : Early Beginnings with the Information Lens System

The 1980s saw the creation of the Information Lens System (ILS) [26], which became the first recommender engine that was based on offline knowledge and know-how. To propose a profile-based set of items for users, there were guidelines set out by experts themselves on these items. For example, if someone habitually borrowed detective stories then, on the basis of established norms, the system would suggest to them other novels in the same genre with bestsellers by the author they were looking for. Despite their innovation, these systems struggled to scale due to the manual effort required to define rules for an ever-increasing volume of informations.

Beginning of 1990s : Arrival of Recommendation Systems

The initial RS arrived in the 90s, with techniques such as collaborative filtering and content-based filtering. Pioneering work by Resnick and Varian laid the groundwork for these systems [58]. the concept of "filtering" was introduced by Malone [49], In the early 90s, to the realm of recommender systems. He defined filtering as the process of "selecting items from a larger set of possibilities," encompassing both positive and negative filteringto either include or exclude items.

Mid to late 1990s : Collaborative Filtering

In the context of an online movie store in the 1990s, a Collaborative Filtering algorithm would analyze user ratings for movies and recommend films that other users with similar preferences had enjoyed. This period witnessed the ascendancy of Collaborative Filtering as a predominant approach for recommendation systems, illustrated by techniques such as matrix factorization and neighbor-based algorithms. The recommendations were based on the preferences of similar users, notable works during that era are The Tapestry system [26] which actually introduced the term "collaborative filtering", GroupLens [58], and research papers like Breese, J. S., et al. (1998) [11]. These systems, rooted in collaborative filtering techniques, demonstrated versatility in recommending a wide array of items, ranging from books and videos to web pages, articles, and internet links.

2000s : Hybrid Systems Merging Collaborative and Content-Based Filtering Techniques

During that era, we saw the development of hybrid systems that used CF approaches with CBF techniques. CBF recommends items that are similar to those a user has liked in the past, using item features for comparison, at the time, researchers began combining different recommendation techniques, such as CF and CBF, to improve recommendation accuracy. Work by Linden et al. explored this approach [47].

2010s : Emergence of Deep Learning in RSs

The application of deep learning in RSs has become widely impactful. due to their ability to capture complex patterns in data [1]. Widely used nowadays, a streaming service in the 2010s might utilize deep learning to analyze a user's watch history, movie ratings, and actors they like. Based on this information, the DL model could recommend similar movies or shows the user might enjoy.

2020s : Personalization and Interpretability

Today, recommender systems are widely used in a huge variety of web applications, including e-commerce sites, social media platforms, and music streaming services, as a matter of a fact, Researchers are increasingly focusing on personalizing recommendations and making recommendation systems more interpretable for users, Recent work by Adomavicius and Kwon has addressed these challenges [28]

Figure 1.1, Illustrate those main stations in Recommender Systems History on a Timeline :

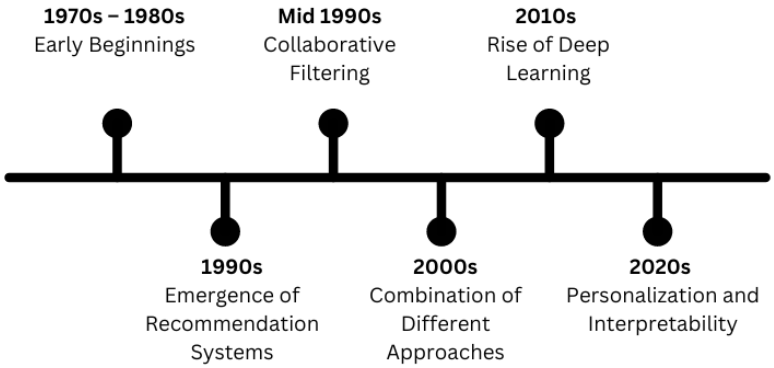

Figure 1.1: Timeline of the Evolution of RSs

1.4 Classification of Recommender Systems

The standard Recommendation Systems classification categorizes them based on their techniques. Here's a breakdown of the classical classification : [2]:

- **Content-Based Filtering**: This technique recommends items to users by taking into consideration the similarity between items and the users' past preferences or interactions. It examines the content or attributes of items and suggests those that are similar to what the user has liked previously. [5].
- **Collaborative Filtering**: CF recommends items to users based on the preferences of other users who have a similar profile. It does not require knowledge of item attributes but instead depends on the historical interactions of users to recommend.

- **Hybrid Systems:** They merge diverse techniques to overcome the challenges of individual methods and provide more accurate and diverse recommendations.

On the other hand, Burke's [13] classification provides a comprehensive overview for existing recommendation techniques by identifying the input data of each method and its associated algorithm. He defines five main types of Recommendation Techniques, where he adds 3 more layers to the classical classification cited above:

- **Demographic Filtering**: These techniques take into account users' demographic characteristics to personalize recommendations. Input data includes information such as age, gender, geographic location, etc., and the algorithm used may involve matching rules between demographic characteristics and user preferences.
- **Knowledge-Based Systems**: In these systems, recommendations are generated based on explicit knowledge about user preferences, domain-specific rules, or expert knowledge. They are often used in domains where there is limited historical data available, such as in specialized or niche markets.
- **Utility-Based Systems (UBS)** : break the mold by focusing on what truly matters: user satisfaction. Instead of just suggesting similar items, UBRS predict how much a user would enjoy an item. They achieve this by modeling user preferences through ratings, purchase history, and even demographics.

The classification proposed by Su and al. in 2009 [71] specifically focuses on collaborative recommendation systems and offers a detailed approach to classify them into different categories. Here's a detailed explanation of the two categories of collaborative filtering :

- **Memory-Based CF Approaches:** These approaches rely on user-user or item-item similarity to generate recommendations. A common example is the K-nearest neighbors approach, where recommendations for a user are based on the preferences of users similar to them.
- **Model-Based CF Approaches:** These approaches use statistical or machine learning models to predict user preferences for items. They encompass a variety of techniques such as clustering, Bayesian networks, matrix factorization, Markov decision processes, etc.
- **Hybrid CF**: Combines memory-based and model-based approaches for improved performance and addressing limitations of individual techniques

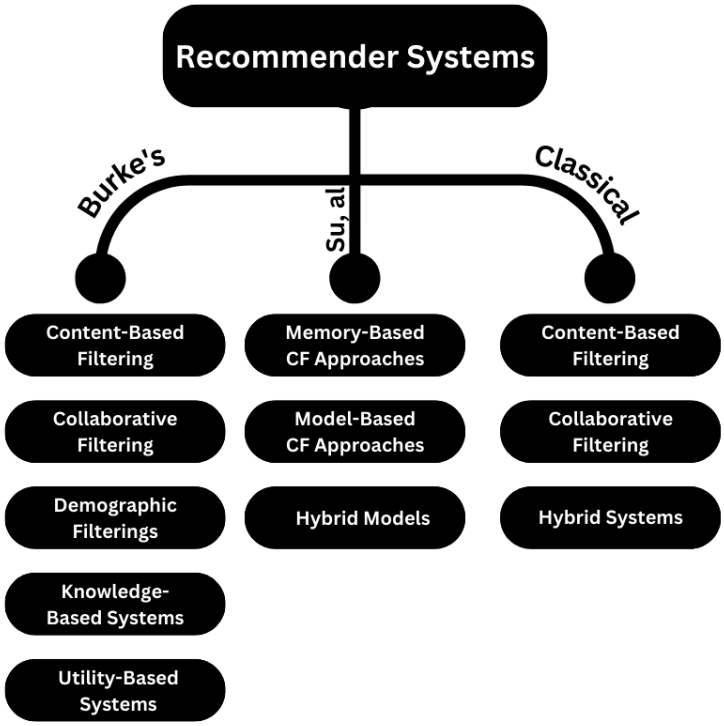

Figure 1.2: RSs Different Classifications

Figure 1.2 summarizes the classification of RSs, and showcases the classifications explained at the beginning.

While both the classical approach and Burke's classification offer frameworks for understanding recommender systems, they primarily focus on technical aspects. They clearly categorize methods based on data sources (content, collaboration) and algorithms (similarity, ratings), but lack a deeper discussion on real-world application considerations and techniques, their contextualization is not fully described or studied.

The classical classification, for instance, isn't able to address certain issues within Collaborative Filtering as the the processes behind CF are not implicitly included in the classification. Similarly, Burke's classification, although more comprehensive, overlooks potential challenges like maintaining domain knowledge in knowledge-based systems.

Su & al.'s approach in 2009 categorized recommender systems into model-based, memory-based, and hybrid models. While this classification provides a comprehensive framework for understanding different recommendation techniques, it has been critiqued for its rigidity and lack of adaptability to the rapid advancements in machine learning and data availability. It doesn't fully capture the evolving landscape of recommender systems, particularly in terms of scalability and the integration of new data types and sources.

As illustrated in Figure 1.3, Isinkaye & al.'s approach in 2015 offers a thorough and well-rounded classification of recommender systems [37], effectively addressing both technical and practical aspects. Their classification integrates content-based, collaborative, and hybrid methods, as a matter of fact, this proposition brings together the advantages of the last three techniques without having the same limitations. And Our objective is to rely on the most recognized classification for this study. Due to its elevance, we will adopt that classification as the foundational framework for our study. This approach provides the most complete and adaptable classification.

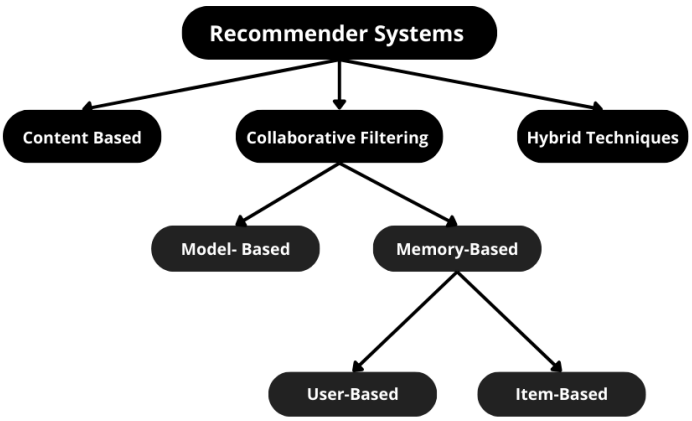

Figure 1.3: Isinkaye et al.'s Classification of RSs

1.5 Classical algorithms

1.5.1 Collaborative Filtering

Collaborative filtering (CF) is a method used in RSs to recommend items to users based on the preferences of similar users, it contains 2 techniques; memory-based or model-based. In the context of memory-based collaborative filtering, the recommendations are made through the use of existing data which may be grouped based on similarity between users or items. Contrary to this is model-based collaborative filtering which utilizes machine learning algorithms to forecast user standards after finding the patterns derived from the relevant dataset [64], thus, it's evident that CF is a pilar technique in RSs.

1.5.1.1 Memory-Based Collaborative Filtering Methods

Memory-Based Methods (MBMs), represent a class of recommendation algorithms that leverage past user-item interactions to generate recommendations. These models operate on the basis that users' preferences can be estimated by analyzing their past behaviors and the similarity between their interactions and those of similar users. These methods utilize similarity metrics to compare users or items based on their historical ratings, essentially finding those with similar preferences.

In essence, MBMs analyze a dataset containing user-item interactions to identify patterns or trends that can be used to predict users' preferences for new items.

One of the key advantages of memory-based models lies in their simplicity and transparency. They do not require complex training phases and are easily interpretable [25]. Additionally, these models can capture complex relationships between users and items, including popularity effects and emerging trends.

As an example, let's consider a movie recommendation system. A memory-based model could analyze past user ratings for various movies and identify users with similar tastes. If two users have consistently rated movies in a similar manner, the model may recommend movies highly rated by one user to the other.

However, memory-based models also have limitations. They may face scalability issues, particularly in systems with a large number of users and items [17].

Figure 1.4 illustrates the workflow of a memory-based recommendation system. It outlines the steps

involved in transforming user interaction data into personalized recommendations for each user.

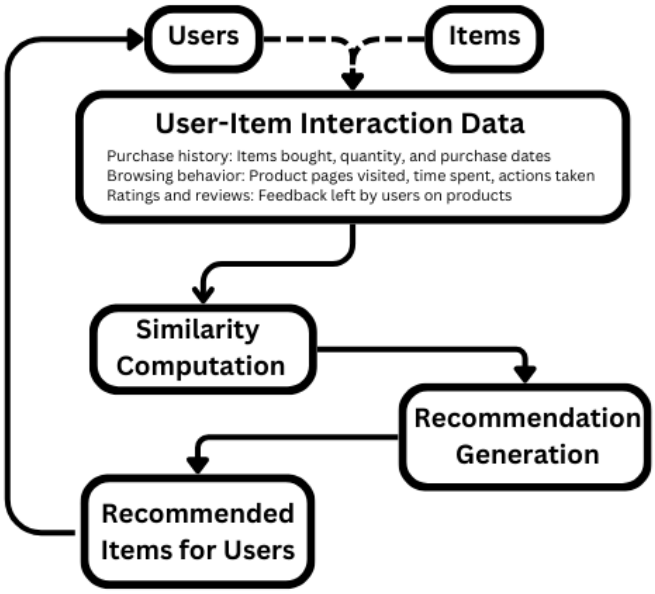

Figure 1.4: Memory-Based Recommendation System Process

Memory-based models are widely used in online environments, such as e-commerce platforms and content streaming services, due to their simplicity, transparency, and adaptability to dynamic user interactions. They are effective in recommending products, movies, music, and other items based on users' past behavior and preferences. These models offer a straightforward approach to recommendation, making them easy to implement and interpret in real-world applications.

1.5.1.1.1 Neighborhood Constitution

The K-Nearest Neighbors (KNN) algorithm is employed in machine learning for both diverse tasks like regression and classification, and it is commonly employed in collaborative filtering for recommendation systems, in particular UbCF agorithms. We will discover how KNN employs similarity measures that can help locate neighbouring points, and understand the implications they have on some important features and how such an algorithm can be deployed within real-life recommendation situations.

1. **Principal of KNN:**

The core idea behind KNN is to predict the value of a target variable (in this case, a user's preference for an item) by averaging or voting among the values of the k neighbors.

CF based on the KNN algorithm is simple, with a straightforward implementation; it also generally produces good quality predictions and recommendations [64].

KNN is used to find the k most similar users to a target user using their anterior interactions with items. Similarity measures such as cosine similarity or Pearson correlation coefficient are used to calculate the similarity between users' interests and when the k-nearest neighbors are identified, their preferences are aggregated to generate recommendations for the target user [72].

a) **In UbCF :**

Recommendations are made by identifying users who exhibit similar preferences to the target user [36].

- Similarity measures in UbCF :

Similarity measures like cosine similarity or Pearson correlation can be used to compute user similarity.

Cosine similarity measures the cosine of the angle between two vectors and is calculated as:

$$\text{similarity}_{\text{Cosine}}(u_1, u_2) = \frac{\sum_{i \in I_{u_1} \cap I_{u_2}} r_{u_1,i} \cdot r_{u_2,i}}{\sqrt{\sum_{i \in I_{u_1}} r_{u_1,i}^2} \cdot \sqrt{\sum_{i \in I_{u_2}} r_{u_2,i}^2}} \tag{2}$$

Where:

- $\text{similarity}_{\text{Cosine}}(u_1, u_2)$ represents the cosine similarity between users u_1 and u_2.
- I_{u_1} denotes the set of items rated by user u_1.
- I_{u_2} denotes the set of items rated by user u_2.
- $r_{u_1,i}$ indicates the rating given by user u_1 for item i within I_{u_1}.
- $r_{u_2,i}$ indicates the rating given by user u_2 for item i within I_{u_2}.
- $\sum_{i \in I_{u_1} \cap I_{u_2}} r_{u_1,i} \cdot r_{u_2,i}$ calculates the dot product of the ratings provided by users u_1 and u_2 for the items both users have rated.
- $\sqrt{\sum_{i \in I_{u_1}} r_{u_1,i}^2}$ and $\sqrt{\sum_{i \in I_{u_2}} r_{u_2,i}^2}$ compute the Euclidean norms (magnitudes) of the rating vectors for users u_1 and u_2, respectively.

Pearson correlation measures the linear correlation between two variables and is calculated as:

$$\text{similarity}_{\text{Pearson}}(u_1, u_2) = \frac{\sum_{i \in I} (r_{u_1,i} - \bar{r}_{u_1})(r_{u_2,i} - \bar{r}_{u_2})}{\sqrt{\sum_{i \in I} (r_{u_1,i} - \bar{r}_{u_1})^2} \cdot \sqrt{\sum_{i \in I} (r_{u_2,i} - \bar{r}_{u_2})^2}} \tag{3}$$

Where:

- $\text{similarity}_{\text{Pearson}}(u_1, u_2)$: Pearson similarity between users u_1 and u_2.
- $r_{u_1,i}$: Rating of user u_1 for item i.
- $\bar{r}_{u_1}$: Average rating given by user u_1.
- $r_{u_2,i}$: Rating of user u_2 for item i.
- $\bar{r}_{u_2}$: Average rating given by user u_2.
- I : Set of items for which both users u_1 and u_2 have given ratings.

Once the nearest neighbors are identified, UbCF recommends items that these neighbors have rated highly but the target user has not yet interacted with.

b) **In IbCF :**

Focuses on the similarity between items and It recommends items to a user based on the similarities between the target item and other items the user has interacted with in the past. The relationships between items and their co-occurrences in user interactions, enable IbCF's ability to predict the user's interest in the target item [36].

- Similarity measures in IbCF :

Cosine similarity quantifies how often users interact with similar items, measuring the cosine of the angle between item vectors, it is calculated as :

$$\text{similarity}_{\text{Cosine}}(i_1, i_2) = \frac{\sum\limits_{u \in U_{i_1} \cap U_{i_2}} r_{u,i_1} \cdot r_{u,i_2}}{\sqrt{\sum\limits_{u \in U_{i_1}} r_{u,i_1}^2} \cdot \sqrt{\sum\limits_{u \in U_{i_2}} r_{u,i_2}^2}} \tag{4}$$

Where:

- $\text{similarity}_{\text{Cosine}}(i_1, i_2)$: Cosine similarity between items i_1 and i_2.
- U_{i_1} : Set of users who have rated item i_1.
- U_{i_2} : Set of users who have rated item i_2.
- r_{u,i_1} : Rating given by user u to item i_1.
- r_{u,i_2} : Rating given by user u to item i_2.

On the other hand, *Pearson correlation* considers both the ratings given to items and the deviations from the average rating for each item, and it's calculated as :

$$\text{similarity}_{\text{Pearson}}(i_1, i_2) = \frac{\sum\limits_{u \in U_{i_1} \cap U_{i_2}} (r_{u,i_1} - \bar{r}_{i_1})(r_{u,i_2} - \bar{r}_{i_2})}{\sqrt{\sum\limits_{u \in U_{i_1}} (r_{u,i_1} - \bar{r}_{i_1})^2} \cdot \sqrt{\sum\limits_{u \in U_{i_2}} (r_{u,i_2} - \bar{r}_{i_2})^2}} \tag{5}$$

Where:

- $\text{similarity}_{\text{Pearson}}(i_1, i_2)$: Pearson similarity between items i_1 and i_2.
- U_{i_1} : Set of users who have rated item i_1.
- U_{i_2} : Set of users who have rated item i_2.
- r_{u,i_1} : Rating given by user u to item i_1.
- r_{u,i_2} : Rating given by user u to item i_2.
- $\bar{r}_{i_1}$: Average rating of item i_1 across all users who have rated it.
- $\bar{r}_{i_2}$: Average rating of item i_2 across all users who have rated it.

Figure 1.5 is a schema representing the process of both IbCF and UbCF, and on which entity the similarity is computed.

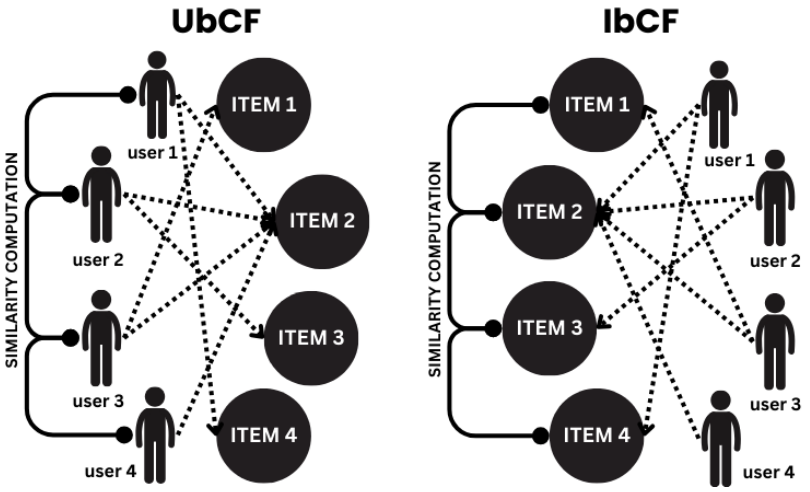

Figure 1.5: UbCF vs. IbCF

These collaborative filtering techniques enhance recommendation systems by providing personalized suggestions aligned with individual users' tastes and preferences.

2. **Prediction Aggregation (Prediction Form) :**

In this phase we will predict the target user's interaction with unseen items. This prediction forms the basis for recommendation generation in KNN-based Collaborative Filtering (UBCF). Here, we explore two common approaches for prediction aggregation:

- Average Rating:

This method is based on the idea that the combined views of users with similar tastes can accurately predict the target user's preference for an item. Mathematically, for a target user u and an item i that hasn't been rated by u, the estimated rating $\hat{r}_{ui}$ can be represented as:

$$\hat{r}_{ui} = \frac{1}{k} \sum_{j=1}^{k} r_{nj,i} \tag{6}$$

Where:

- $\hat{r}_{ui}$ represents the estimated rating for user u and item i.
- k denotes the number of closest neighbors.
- $r_{nj,i}$ is the rating that the j-th closest neighbor assigned to item i.

This method assumes that users with similar historical interaction patterns are likely to have similar preferences for unrated items [36].

- Weighted Average Rating:

A more nuanced approach involves the weighted average rating. This method assigns higher weights to ratings from closer neighbors (more similar users) in the k-neighborhood. In this context, the similarity score $Sim(u, nj)$ between the target user and each neighbor is included in the prediction formula:

$$\hat{r}_{ui} = \frac{\sum_{j=1}^{k} \text{Sim}(u, n_j) \cdot r_{n_j,i}}{\sum_{j=1}^{k} |\text{Sim}(u, n_j)|} \tag{7}$$

Where:

- $\hat{r}_{ui}$ denotes the predicted rating for user u and item i.
- k represents the number of closest neighbors considered.
- $\text{Sim}(u, n_j)$ stands for the similarity score between the target user u and the j-th neighbor n_j.
- $r_{n_j i}$ denotes the rating provided by the j-th nearest neighbor for item i.

This approach leverages the varying degrees of similarity among the k neighbors. Ratings from neighbors with higher similarity scores (stronger taste alignment) contribute more significantly to the prediction, potentially leading to a more accurate reflection of the target user's preferences [36].

- Choice of aggregation formula:

The choice between average rating and weighted average rating depends on various factors, As explained by (Herlocker & al., 1999) [34] The simple average rating is often criticized for its excessive simplicity. This method does not account for the variation in the quality of evaluations, while The weighted average rating is generally considered more sophisticated because it can incorporate various factors to adjust the importance of each rating. However, this method is not without its drawbacks. The main challenge is determining the appropriate criteria and weights to assign to different ratings. Additionally, this approach can be more complex to implement and justify, generally we follow these following criterias to choose :

- Data Characteristics: For sparse datasets with limited ratings, the average rating might be sufficient. Weighted average rating can be more effective in denser datasets where neighbor similarity provides valuable information for refining predictions.
- Recommendation Goals: The average rating prioritizes items endorsed by many neighbors, regardless of the specific rating strength. Weighted average rating focuses on recommendations that align with the target user's typical rating behavior (e.g., high or low ratings).

3. **Recommendation Generation – Top N recommendation :**

This process involves identifying the N items with the highest predicted scores or ratings and presenting them as recommendations to the user and we have do the following:

a) **Selection of Predicted Scores**

After computing predicted scores or ratings for unrated items, the next step is to rank these items based on their scores. The items are sorted in descending order of predicted scores. This ranking reflects the system's prediction of the user's potential interest in each item.

- **Selection Strategies :** Within UBCF, various strategies can be employed to select the Top-N recommendations, As illustrated in (J. Bobadilla & al., 2013)'s work, and they are :
  - **Thresholding**: This method sets a minimum predicted rating threshold. Only items exceeding this threshold are included in the recommendation list. This approach ensures a certain level of predicted user preference for the recommended items.
  - **Ranking**: All estimated ratings are taken into account, and items are sorted in descending order according to their predicted scores. The top-N items with the highest predicted ratings are then recommended. This method offers a wider range of choices, which may include items with slightly lower predicted preferences but still of potential interest.
  - **Hybrid Approaches**: Combining thresholding and ranking can be beneficial. A threshold can be used to filter out items with very low predicted ratings, followed by ranking the remaining items for final recommendation selection.

b) **Determination of N**

The parameter N indicates the number of items to recommend to the user. This value is usually predetermined by the recommendation system or chosen by the user. It determines the size of the recommendation list shown to the user and directly affects the diversity and relevance of the recommendations[54]. Setting an optimal value for N is important because it balances the need for diversity and relevance in the recommendations. If N is too small, the list may lack diversity and fail to capture the user's varied interests. Conversely, if N is too large, the recommendations may become less relevant, overwhelming the user with too many choices. Therefore, carefully selecting or allowing users to specify N can significantly enhance the user experience by presenting a well-rounded and pertinent set of recommendations [36].

c) **Recommendation**

After selecting the top-N recommended items, they are displayed to the user through the interface of the recommendation system. The presentation format may vary depending on the application context, with options including lists, grids, carousels, or personalized feeds. Clear and intuitive presentation enhances user engagement and facilitates the exploration of recommended items.

1.5.1.1.2 Impact of the Neighborhood Size on the Performance of Recommendation

The choice of K impacts the quality of predictions in top-N recommendation. A smaller value of K may lead to more localized predictions, where recommendations are based on a restricted subset of similar users or items. Inversely, a larger K value may result in more generalized predictions, incorporating a broader range of user or item preferences. [45], thus, it also suggests that the diversity of recommendations is influenced by the K value in top-N recommendation. A higher K value tends to yield more diverse recommendations by considering a larger pool of similar users or items. This diversity enhances the variety of items presented to the user. Metrics such as catalog coverage or intra-list diversity can be used to measure the diversity of recommendations [21].

Another thing that's relevant in KNN use in the context of RSs, is that results have shown that the choice of K value in top-N recommendation can mitigate the risk of overfitting or underfitting the recommendation model to the data. An excessively small K value may lead to overfitting, where recommendations are overly tailored to the training data, potentially compromising generalization to unseen users or items. On the same rail, an excessively large K value may result in underfitting, where recommendations lack specificity and fail to capture user preferences effectively. Techniques such as cross-validation or grid search can be employed to identify an optimal value of K that balances model complexity and generalization performance [59]. As a brief conclusion, we can say that the K value represents a crucial parameter in the recommendation process and balancing personalization with diversity is crucial for ensuring a satisfying recommendation experience for users [2].

1.5.1.2 Model Based Filtering Algorithms

In model-based filtering, clustering and classification techniques are commonly employed, such as Bayesian methods or neural networks for collaborative filtering (CF). This strategy involves condensing vast databases into a single model and executing recommendation tasks by referencing this model [18]. model-based filtering transcends the limitations of rigid rule-based systems. Instead, it uses the power of machine learning to discern intricate patterns and latent correlations within vast datasets. Cooperative Modelling Instead of using a predetermined set of rules, filters use a statistical or machine learning model to identify and exploit hidden links and patterns in the data. These models are then used to estimate users' preferences for unseen objects based on their training data of past interactions between users and items.

Memory-based recommendation systems often lack the desired speed and scalability, particularly in real-world scenarios where real-time recommendations are generated from extensive datasets. To overcome these challenges, model-based recommendation systems are employed.

In Model-based recommendation systems, a model is constructed based on the dataset of ratings. Essentially, information is extracted from the dataset and utilized as a "model" to generate recommendations without the need to access the entire dataset each time. This approach potentially combines the advantages of both speed and scalability. However, while model-based recommendation systems hold immense promise, they are not without their challenges [17]. Developing and fine-tuning these models can be a complex and resource consuming, requiring substantial expertise and computational resources. Additionally, the inherent black-box nature of machine learning models can pose challenges in terms of transparency and interpretability, raising concerns about trust and accountability.

Despite these challenges, the benefits of model-based recommendation systems far outweigh the drawbacks. By using the power of advanced machine learning techniques, these systems unlock new possibilities for personalized recommendation experiences. Moreover, their scalability and adaptability make them well-suited for addressing the evolving needs of modern recommender systems.

1.5.1.2.1 Understanding Machine Learning Techniques: Overview

In recent years, artificial intelligence (AI), especially machine learning (ML), has seen rapid growth in the realm of data analysis and computing. This advancement allows applications to function intelligently. Machine learning endows systems with the capability to learn and improve from experience automatically, without explicit programming[64].

1. **The Machine Learning Process :**

The machine learning process encompasses a series of interconnected steps aimed at developing and deploying predictive models from data. Beginning with data collection and preprocessing, where raw data

is gathered and refined to ensure quality and relevance, the process progresses to feature engineering. Here, meaningful attributes are extracted or constructed from the data to facilitate model learning. Following feature engineering, the model selection phase involves choosing an appropriate algorithm or ensemble of algorithms based on the problem domain, data characteristics, and desired outcomes. Once a model is selected, it undergoes training, where it learns patterns and relationships from labeled data in supervised learning or discovers intrinsic structures in unlabeled data in unsupervised learning. Following training, model evaluation assesses its performance on unseen data, ensuring generalizability and robustness, Figure 1.6 Illustrates the workflow of those primary steps.

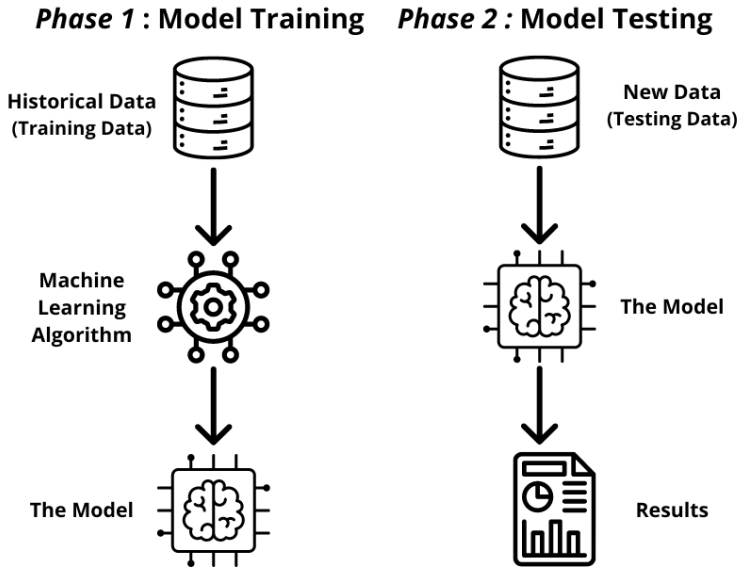

Figure 1.6: Workflow of the Model Training and Testing Phases

Finally, successful models are deployed into operational environments, where they make predictions or decisions in real-world applications

2. **Types of Machine Learning Techniques**

Machine Learning algorithms are broadly categorized into four types: *Supervised learning, Unsupervised learning, Semi-supervised learning, and Reinforcement learning* [51], depicted in Fig. 1.13. This section provides a brief overview of each learning technique, discussing their characteristics and applications in solving real-world problems.

a) **Supervised :**

It involves training a machine learning model to map inputs to outputs using labeled data. By utilizing pairs of input-output examples, the model learns to infer the relationship between them. This approach is goal-oriented, where specific objectives are defined based on the input data [51]. Common tasks in supervised learning include classification, which categorizes data, and regression, which predicts continuous values. For example, text classification, such as determining the sentiment of a tweet or a product review, is a typical application of supervised learning.

- **Linear Regression :** Linear regression is a machine learning method used to predict a continuous dependent variable y [64]. It is widely recognized as one of the most popular regression techniques.

In linear regression, the dependent variable y is continuous, while the independent variable(s) x can be continuous or discrete. The regression line takes a linear form, establishing a relationship between y and x (also referred to as the regression line) [29]. Figure 1.7 illustrates this concept visually. The technique is characterized by the following equation:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \cdots + \beta_n x_n + \epsilon \tag{9}$$

where:

- y is the dependent variable.
- $x_1, x_2, x_3, \ldots, x_n$ are the independent variables.
- β_0 is the y-intercept.
- $\beta_1, \beta_2, \beta_3, \ldots, \beta_n$ are the coefficients corresponding to each independent variable.
- ϵ is the error term.

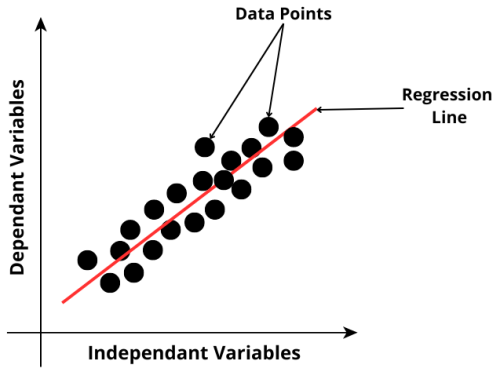

Figure 1.7: Illustration of a Regression Model

- **Decision Trees (DT) :** DT learning techniques are versatile tools applied to both classification and regression tasks [64]. Among the well-established DT algorithms are ID3, C4.5, and CART. Recent innovations such as BehavDT and IntrudTree, introduced by (Sarker et al. 2020) [65] , have shown promise in domains like user behavior analytics [64]. The classification process of DT involves traversing the tree structure from the root to leaf nodes[70], as illustrated in Figure 1.8. DTs calculations are guided by criterias like Gini impurity and information gain known as entropy, which are expressed mathematically by the following :

  Gini impurity:

$$\text{Gini}(p) = 1 - \sum_{i=1}^{n} p_i^2 \tag{11}$$

  where:

  - p is a vector of probabilities for each class,
  - n is the number of classes,
  - p_i is the probability of class i.

  Information gain:

$$\text{Gain}(D, A) = \text{Entropy}(D) - \sum_{v \in \text{Values}(A)} \frac{|D_v|}{|D|} \text{Entropy}(D_v) \tag{12}$$

  where:

- D is the dataset before the split,
- A is the attribute being considered for the split,
- Values(A) is the set of possible values for attribute A,
- D_v is the subset of D where attribute A has value v,
- Entropy(D) is the entropy of dataset D,
- Entropy(D_v) is the entropy of subset D_v.

Entropy:

$$\text{Entropy}(D) = -\sum_{i=1}^{n} p_i \log_2(p_i) \tag{13}$$

where:

- p_i is the proportion of instances in class i in dataset D,
- n is the number of classes.

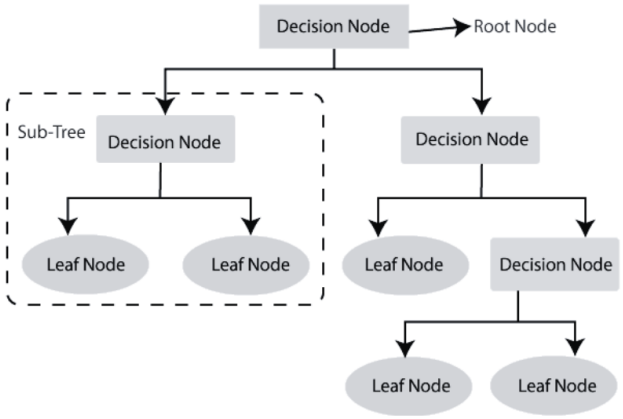

Figure 1.8: Decision Tree Structure

- **Random Forests** : Random forest classification is a powerful ensemble technique widely utilized in machine learning and data science across diverse applications. This method employs a strategy known as "parallel ensembling," wherein multiple decision tree classifiers are simultaneously trained on different subsets of the dataset, as depicted in Fig. 1.9. By employing techniques such as majority voting or averaging of outcomes, random forest mitigates the risk of overfitting while enhancing prediction accuracy and control. Consequently, a random forest model comprising multiple decision trees typically outperforms a single decision tree-based model. They are also adaptable to both classification and regression [12].

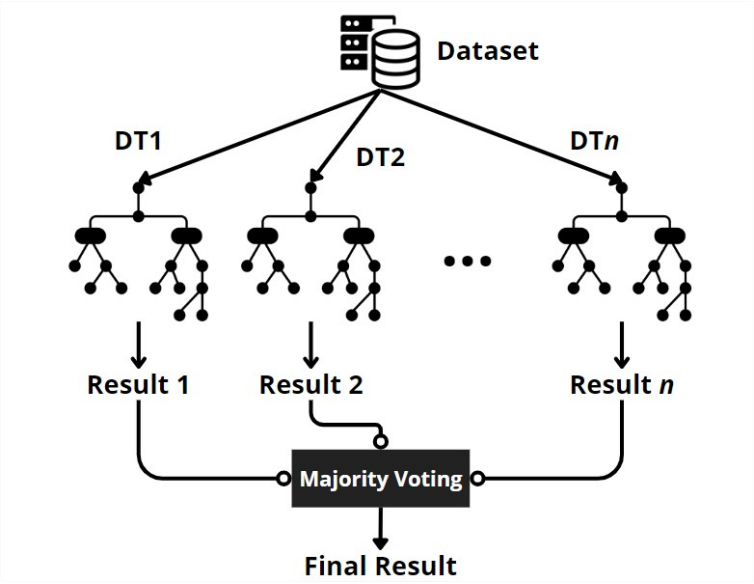

Figure 1.9: Structure of a Random Forest made of Multiple Decision Trees

- **Artficial Neural Networks (ANN):** ANNs have emerged as a foundation in modern machine learning research and applications. These versatile models consist of interconnected nodes organized into layers, with each node applying a mathematical transformation to its inputs and passing the result to subsequent layers. The proliferation of deep neural networks, characterized by multiple hidden layers, has revolutionized various domains by enabling the automatic extraction of hierarchical features from raw data. Figure 1.10 Illustrates the structure of a basic ANN.

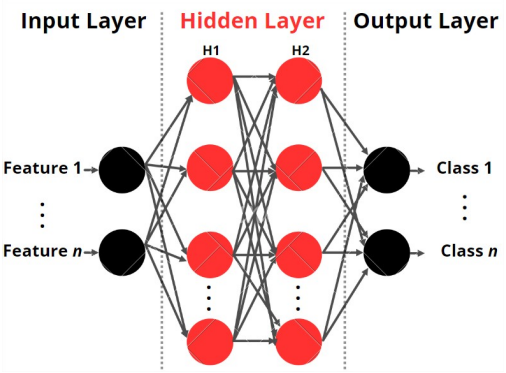

Figure 1.10: Structure of a Neural Network Modeling with Multiple Layers

b) **Unsupervised**:

Unsupervised learning operates autonomously on unlabeled datasets, devoid of human intervention, thereby embracing a purely data-centric approach. Its applications span various domains, encompassing the extraction of generative features, discernment of meaningful patterns and structures, clustering of data points, and exploratory analysis [64], In the context of RSs, the two most relevant things to dive into are :

- **Clustering :** It is an essential technique in unsupervised machine learning. Its primary purpose is to identify natural groupings (clusters) within large datasets without the need for pre-existing labels

or outcomes [22]. The core idea is to group data points in such a way that members within a cluster are more similar to each other than to those in other clusters [29]. Figure 1.11 provides a visual representation of this concept. Cluster analysis is instrumental in data exploration, allowing for the detection of underlying patterns and trends. For example, it can be used to segment customers based on their behavioral patterns [29].

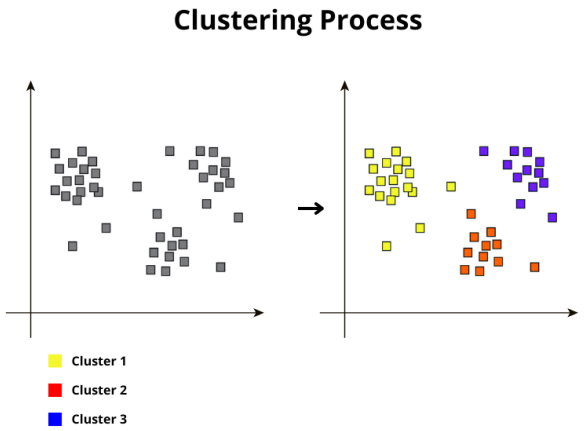

Figure 1.11: Illustration of the Clustering Process

- **Dimesinality Reduction Techniques :** Are an unsupervised learning techniques, is crucial as it enhances human interpretability, reduces redundancy by simplifying models. Both feature selection and feature extraction are employed for dimensionality reduction. The key distinction lies in feature selection retaining a subset of original features, whereas feature extraction generates new ones. We'll delve into these techniques in the next sections.

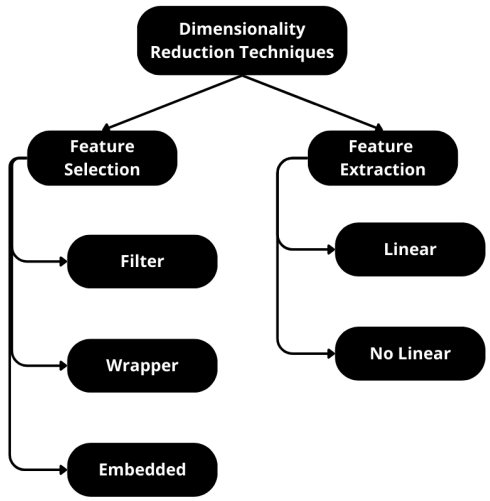

Figure 1.12: Listing of Dimensionality Reduction Techniques

c) **Semi-supervised**:

Semi-supervised learning represents a blend of supervised and unsupervised methods, leveraging both

labeled and unlabeled data [29, 63]. Positioned between supervised and unsupervised learning paradigms, semi-supervised learning becomes particularly valuable in scenarios where labeled data is limited but unlabeled data is abundant. Its primary goal is to enhance prediction accuracy by combining labeled and unlabeled data. This approach finds applications across diverse fields such as machine translation, fraud detection, and text classification, where utilizing both types of data leads to superior performance [64].

- **Graph-Based Methods:** Graph-based models are inherent structure of data represented as graphs to enhance learning from both labeled and unlabeled data. These models exploit the relationships and dependencies between data points, which are often encoded as edges in a graph, to propagate label information through the network [82]. Techniques like Label Propagation [81] rely on the assumption that nodes connected by an edge might share the same label.

  Another significant advancement is the introduction of Graph Convolutional Networks (GCNs) [6]. GCNs extend the concept of convolutional neural networks to graph-structured data, allowing for the efficient extraction of local neighborhood features. By stacking multiple graph convolutional layers, GCNs can capture both local and global structures within the graph, making them highly effective for semi-supervised learning tasks.

  Graph Attention Networks (GATs) [75] further enhance GCNs by incorporating attention mechanisms. Recent developments also include Graph Autoencoders (GAEs) and Variational Graph Autoencoders (VGAEs) [44], which use encoder-decoder architectures to learn embeddings for nodes in an unsupervised manner.

- **Generative Models :** They are models that are designed to understand and replicate the underlying distribution of the data, thereby providing a foundation to generate new data points that resemble the original dataset. In semi-supervised learning, generative models help in improving classification tasks by making use of the abundance of unlabeled data.

  The use of Gaussian Mixture Models (GMMs) represents a focal approach in this field. GMMs assume that the data is generated from a mixture of several Gaussian distributions, each representing a different class. In a semi-supervised context, GMMs utilize both labeled and unlabeled data to estimate the parameters of these distributions, allowing for improved classification by modeling the probability of each class.

  More recently, models like Semi-Supervised Variational Autoencoders (SSVAEs) [43] and Semi-Supervised Generative Adversarial Networks (SGANs) [62] have been proposed. These models build on the strengths of VAEs and GANs by explicitly incorporating semi-supervised learning objectives into their training process. SSVAEs, for instance, introduce additional loss terms to account for the labeled and unlabeled data separately, while SGANs enhance the discriminator to classify real data samples while maintaining the adversarial training dynamics.

d) **Reinforcement Learning (RL)**:

Reinforcement learning is a machine learning approach using software agents to autonomously assess optimal behaviors in specific contexts to enhance efficiency [64]. It operates on a reward or penalty system, aiming to leverage insights gained from interactions with the environment to maximize rewards or mitigate risks. This methodology is instrumental in training AI models to automate tasks or optimize operations in complex systems like robotics, autonomous driving, and supply chain logistics. However, it's not recommended for solving simple or straightforward problems.

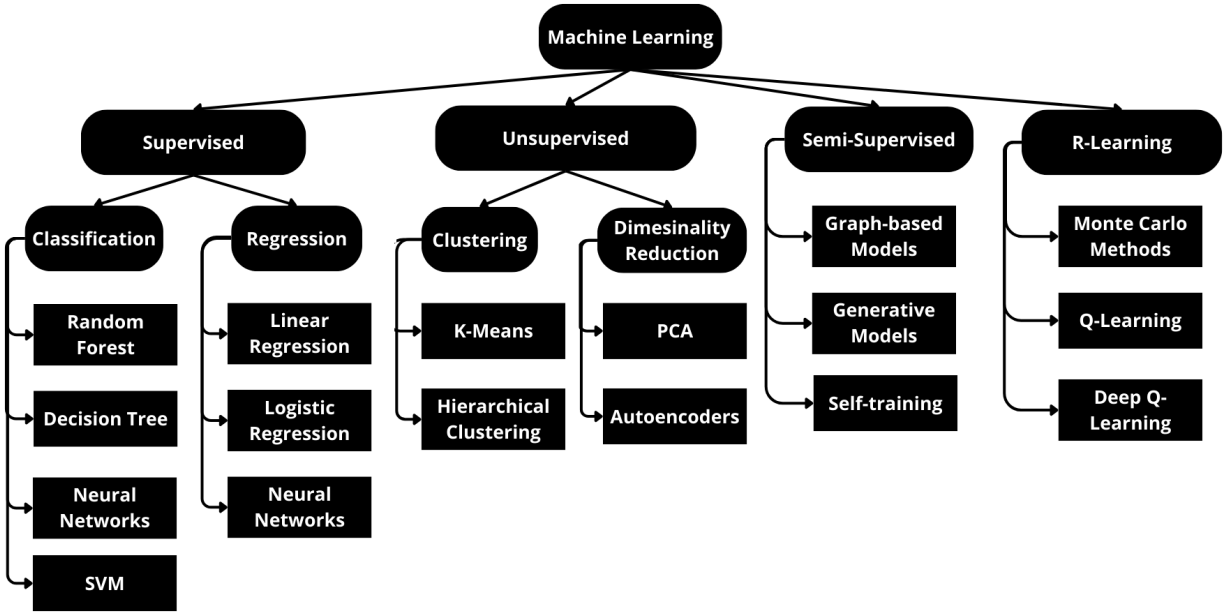

Figure 1.13: Taxonomy of Machine Learning

3. **Applications of Machine Learning in Recommender Systems** :

Nowadays, ML's a pivotal technology across multiple domains, revolutionizing how data is analyzed and decisions are made. Among those services, we have Recommender systems, which have become a cornerstone in a multitude of industries, from e-commerce to entertainment, by providing personalized content to users. The application of various machine learning algorithms has significantly advanced the efficiency and accuracy of these systems. we'll briefly see the machine learning techniques used in RSs :

- Collaborative Filtering :

  CF algorithms predict user preferences by leveraging the historical interactions of users and items. Matrix factorization techniques, such as Singular Value Decomposition (SVD) and Alternating Least Squares (ALS), have been widely used to uncover latent factors that explain observed user-item interactions [45]. Another popular approach within CF is the use of neighborhood-based methods, which recommend items based on the preferences of similar users (user-based) or similar items (item-based) [66].

- Content-Based Filtering :

  Since Content-based filtering (CBF) focuses on the attributes of items and the profiles of users, it relies heavily on Natural Language Processing (NLP) to analyze textual data such as item descriptions, reviews, and user profiles. TF-IDF and word embeddings are common NLP techniques used to represent textual data in a numerical format, and it does rely on Computer Vision techniques to extract features from different visual data. Machine learning algorithms are widely used, including decision trees and support vector machines, then utilize these representations to recommend items similar to those the user has interacted with in the past [48].

- Hybrid Models :

  These systems can be implemented through model-based or feature-combination methods. For instance, Netflix's recommendation system employs a hybrid approach that blends CF with CBF, utilizing both user-item interaction data and item metadata [27]. Ensemble learning techniques, such as boosting and

bagging, are often employed to merge the predictions from various models to improve recommendation accuracy.

- Graph-Based Methods :

  Graph Neural Networks (GNNs) have shown great promise in capturing the complex dependencies in user-item interaction graphs [78]. Techniques such as random walks and graph embeddings (e.g., Node2Vec) help in representing the graph structure in a lower-dimensional space, preserving the essential relationships for effective recommendations [56].

1.5.1.2.2 Deep Learning Paradigms and Architectures for Enhanced Recommender Systems

While traditional Machine learning (ML) techniques rely on manually crafted features and simpler models, Deep Learning (DL) represents a more advanced subset of ML that leverages multi-layered neural networks to automatically learn complex patterns from data. ML algorithms such as decision trees, linear regression, and clustering methods often require significant feature engineering and domain expertise to perform well [24]. In contrast, deep learning models can automatically extract and learn high-level features from raw data, making them particularly powerful for tasks involving large and complex datasets, as illustrated in Figure 1.14.

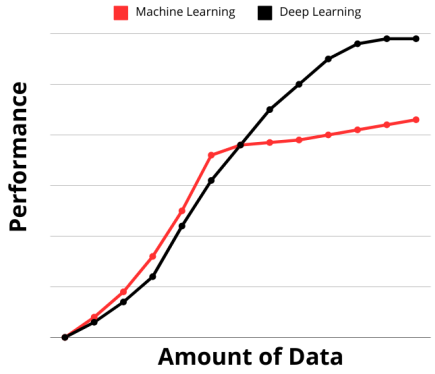

Figure 1.14: Comparison Between the Performances of DL and ML, in Relation to the Quantity of Data

Unlike traditional methods, deep learning employs computational architectures comprising multiple layers, input, hidden, and output to glean insights from data. Its primary advantage lies in its superior performance, especially when dealing with large datasets. However, its efficacy may vary based on data characteristics and experimental setups. Common deep learning algorithms include Multilayer Perceptron (MLP), Convolutional Neural Network (CNN), and Long Short-Term Memory (LSTM). In the subsequent sections, we explore different deep learning methodologies tailored for diverse applications [64].

- **Multilayer Perceptron (MLP) :**

  The multilayer perceptron (MLP) is a fundamental component of deep learning, also known as the feed-forward artificial neural network [64]. An MLP typically consists of an input layer, one or more hidden layers, and an output layer, with each layer fully connected to the next through weighted connections. The "Backpropagation" algorithm is used to adjust the internal weights during the training process. However, the MLP is sensitive to feature scaling and requires careful tuning of various hyperparameters, such as the number of hidden layers, neurons, and iterations, which can lead to increased computational complexity [60]. Figure 2.11 Illustrates the structure of a basic Multilayer Perceptron.

- **Convolutional Neural Network (CNN) :**

  It represents an advancement in the field of machine learning, particularly in tasks that involve spatial data such as images and videos. Since their introduction, CNNs have revolutionized numerous applications across different domains, leveraging their ability to automatically and adaptively learn spatial hierarchies of features through backpropagation by using multiple building blocks, such as convolution layers, pooling layers, and fully connected layers [18].

  Overview of CNN Architecture

  A CNN typically consists, as illustrated in Figure 1.15, of several key components:

  – Convolutional Layers: These layers apply a set of convolutional filters to the input data, capturing local patterns such as edges, textures, and simple shapes in images. The output of each filter is a feature map.

  – Pooling Layers: These layers perform down-sampling operations along the spatial dimensions, reducing the size of the feature maps and providing translation invariance. Common pooling operations include max pooling and average pooling.

  – Fully Connected Layers: After several convolutional and pooling layers, the high-level reasoning in the neural network is done via fully connected layers. These layers connect every neuron in one layer to every neuron in another layer, much like traditional neural networks.

  – Activation Functions: Non-linear functions such as ReLU (Rectified Linear Unit) are applied to the output of convolutional and fully connected layers to introduce non-linearity into the model, enabling it to learn more complex patterns.

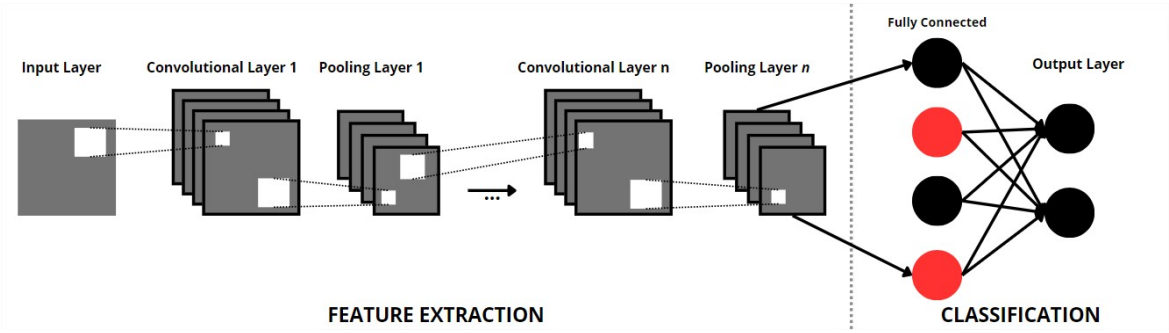

Figure 1.15: Structure of a CNN Network

In the realm of recommender systems, CNNs are employed to analyze visual content, such as product images, to improve recommendation accuracy. For instance, Visual Bayesian Personalized Ranking (VBPR) utilizes CNNs to incorporate visual signals into the recommendation process [31].

The field of CNNs continues to evolve with numerous advancements and research directions:

– Architectural Innovations: New architectures like ResNet (Residual Networks) introduce shortcut connections to mitigate the vanishing gradient problem, allowing for the training of very deep networks [30].

– Transfer Learning (TL): Utilizing pre-trained CNN models on large datasets like ImageNet and fine-tuning them for specific tasks has become a standard practice, significantly improving performance in domains with limited labeled data [79].

– Efficiency Improvements: Techniques such as model pruning, quantization, and the development of lightweight architectures like MobileNets aim to make CNNs more efficient and suitable for deployment on resource-constrained devices [52].

- **Recurrent Neural Network (RNN) :**

  They are a category of ANNs able to uncover patterns within sequential data. Unlike traditional feed-forward neural networks, RNNs have connections that form directed cycles, allowing them to maintain a memory of previous inputs [84]. Figure 1.16, illustrates the basic functionning of RNN networks.

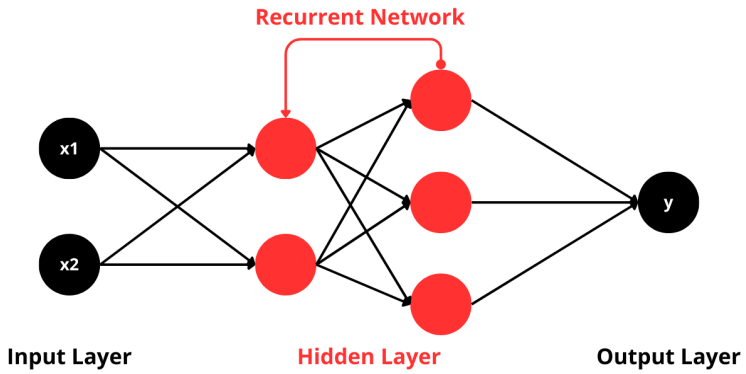

Figure 1.16: Simplified RNN Architecture

This unique architecture makes RNNs particularly well-suited for tasks where context and sequence information are crucial, following these formulas :

Current state is given by:

$$h_t = f(h_{t-1}, x_t) \tag{15}$$

where:

- h_t denotes the current state,
- h_{t-1} denotes the previous hidden state,
- x_t denotes the current input,
- f denotes the function representing the transition from the previous state to the current state.

Activation function:

$$h_t = \tanh(W_{hh}h_{t-1} + W_{hx}x_t) \tag{16}$$

where:

- h_t is the current state,
- W_{hh} is the weight matrix for the previous hidden state,
- h_{t-1} is the previous hidden state,
- W_{hx} is the weight matrix for the current input state,
- x_t is the current input.

Output:

$$y_t = W_{hy}h_t \tag{17}$$

where:

- y_t is the output at time t,
- W_{hy} is the weight matrix for the output layer.

The initial layer processes an encoded item as its input, adapting its size to match the dataset's unique item count. Serving as the neural network's memory, the hidden layer is pivotal in handling sequential data, employing a basic linear transformation followed by a non-linear activation function. Determining the size of this layer, representing the number of hidden units, is a key hyperparameter. the output layer, designed as a linear entity, predicts probabilities for the next item in the sequence. This dynamic facilitates the network's assimilation of insights from the user's sequential item interactions [84].

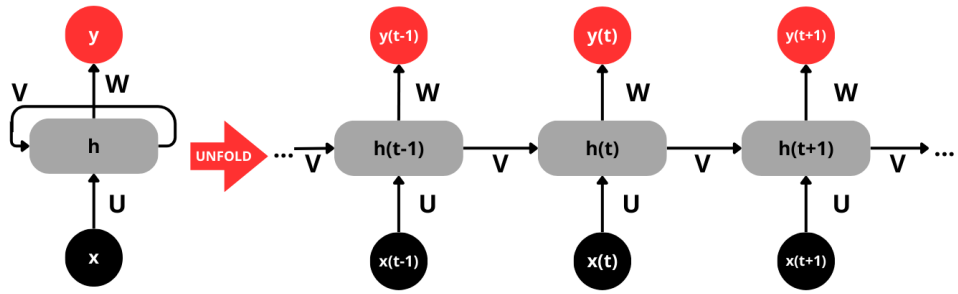

Figure 1.17: Zoom on the RNN Architecture

The output layer, as illustrated in Figure 1.17, leverages its learned weights to generate a prediction for the current time step. Throughout the training process, the network iteratively refines these weights by adjusting them based on the difference between its predictions and the actual data. This process, known as backpropagation, allows the LSTM to progressively improve its predictive accuracy.

- **Long Short-Term Memory (LSTM) :**

  Long Short-Term Memory (LSTM) networks, a specialized type of recurrent neural networks (RNNs), are particularly adept at capturing long-term dependencies in sequential data. Their architecture generally includes an input layer, one or more LSTM layers tailored for handling such dependencies, and an output layer that generates predictions [84], as illustrated in Figure 1.18.

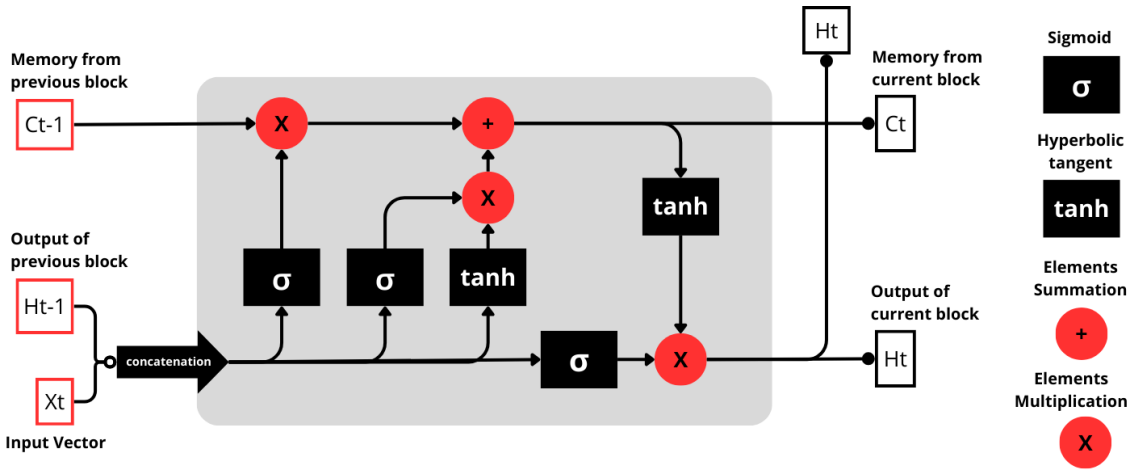

Figure 1.18: LSTM Architecture

- The input layer consists of an embedding layer that transforms each encoded item into a dense vector of

a fixed size, known as an 'embedding'. This approach reduces dimensionality and provides a more expressive representation.

- The simple hidden layer of traditional RNNs is replaced by an LSTM layer. Each LSTM unit includes a cell and three gates (input, forget, and output), which work together to regulate the flow of information across time steps.

- While the final output layer retains a linear activation function to generate the final prediction, the preceding layers within the LSTM model are adept at processing sequential data.

LSTM handles sequences similarly to traditional RNNs but with a more sophisticated mechanism within the LSTM units. This enhancement allows it to remember longer sequences and effectively address the vanishing gradient problem often encountered in RNNs.

Deep learning has facilitated the modeling of high-dimensional and non-linear user-item interactions. Convolutional Neural Networks (CNNs) are used to analyze image data, which is particularly useful in recommending visually-rich content [31]. RNNs and their variants, such as LSTM networks, excel in capturing sequential patterns in user behavior, making them ideal for session-based recommendations [35]. Autoencoders and Generative Adversarial Networks (GANs) have also been explored for collaborative filtering, where they help in capturing complex latent structures and generating plausible user-item interaction data [69]. The integration of machine learning into recommender systems has significantly improved their performance, but it also introduces challenges. Supervised learning algorithms, while accurate, often require large labeled datasets, which may not always be available. Unsupervised and semi-supervised methods help mitigate this issue but may not achieve the same level of precision.

Deep learning approaches, particularly neural networks, offer powerful tools for capturing complex patterns and interactions. However, they demand significant computational resources and may suffer from interpretability issues. For instance, while CNNs and RNNs provide impressive results, their black-box nature makes it difficult to understand how specific recommendations are made.

1.5.1.2.3 Addressing the Sparsity Problem in Recommender Systems: A Focus on Matrix Factorization Techniques

The user-item matrix sparsity problem arises when most of the entries in the matrix are missing or unknown, typically represented as zeros [8]. In recommendation systems, this arises because users engage with a limited subset of the available items, implying a sparse matrix with a high percentage of missing values [45] [32]. Mathematically, the sparsity of a user-item matrix R can be quantified by the sparsity ratio S, defined as the proportion of zero entries to the total number of entries in the matrix:

$$S = \frac{Number \quad of \quad zero \quad entries}{Total \quad number \quad of \quad entries}$$

A high sparsity ratio indicates a greater degree of sparsity in the matrix, which poses challenges for recommendation algorithms as it reduces the amount of available information for making accurate predictions. Table 1.4, represents a sparse User-Item Matrix with sparsity ratio equal to $\frac{11}{16} \approx 0.6875$. The majority of entries are "-", indicating missing interactions and highlighting the sparsity issue.

	Item a	Item b	Item c	Item d
User a	4.6	-	2.9	-
User b	-	3.4	-	-
User c	4.7	-	-	2
User d	-	-	4.1	-

Table 1.4: Example of a Sparse User-Item

1. **Solution : Dimensionality reduction with Matrix Factorization Techniques ?**

Matrix factorization is a commonly used technique in RSs, where a large user-item matrix is decomposed into user feature and item attribute matrices to predict missing ratings.

In tackling the issue of high sparsity in Recommender System databases, dimensionality reduction techniques play a crucial role. They are valuable tools for reducing the computational complexity of RS algorithms and improving their scalability. Additionally, employing multiple dimensionality reduction methods such as Singular Value Decomposition and Principle Components Analysis can be extremly beneficial to further enhance the performance of RS models [45].

- **Singular Value Decomposition (SVD)**

  Singular Value Decomposition (SVD) is a pillar technique within linear algebra used to decompose a matrix into three distinct matrices. it is used to enable the system to discern hidden patterns and preferences. Matrix Factorization techniques like SVD pave the way for scalable and efficient recommendation systems [3].

  At its core, SVD operates on the principle of decomposing a matrix into three constituent matrices, each representing distinct aspects of the data:

  - U Matrix (User Matrix) : It acts as a user portrait. This user portrait reveals underlying preferences and affinities towards items within the system, enabling the RS to tailor suggestions to each individual's tastes.
  - S Matrix (Singular Values) : These values set the importance of each latent feature, guiding the recommendation engine towards optimal decision-making.
  - V^T Matrix (Item Matrix) : also known as the transposed item matrix, plays a crucial role in uncovering the latent attributes that resonate with user preferences.

  Singular Value Decomposition (SVD) emerges as a powerful solution to the challenges of data sparsity. By capturing the intricate relationships between users and items in concise and meaningful representations, SVD significantly enhances recommendation accuracy. By extracting latent features from sparse datasets, SVD paves the way for highly personalized recommendations that align closely with users' preferences [45].

- **Principal Component Analysis (PCA) :**

  Principal Component Analysis (PCA) stands as a cornerstone in the realm of dimensionality reduction, offering a mathematical framework to uncover the underlying structure of high-dimensional datasets while preserving important characteristics and minimizing information loss. At its essence, PCA aims to transform the original feature space into a new orthogonal basis where the majority of the variance is captured by a smaller number of dimensions, known as principal components [8]. Principal Component Analysis (PCA) is often employed also as a solution to sparsity issues in the User-Item Matrix, in the context of recommender systems [3].

  PCA Procedure :

– Standardization :

PCA typically begins with the standardization of the dataset to ensure that all features have a mean of zero and a standard deviation of one. This step ensures that each feature contributes equally to the analysis, preventing features with larger scales from dominating the principal components.

– Covariance Matrix Calculation :

Following standardization, PCA computes the covariance matrix Σ of the standardized dataset.By encapsulating the co-variation between features, the covariance matrix offers a window into the nature of their linear associations. This window reveals both the direction (positive or negative) and the intensity of these relationships, it's calculated using this formula :

$$\Sigma = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^T(x_i - \bar{x}) \tag{18}$$

Where:

* Σ represents the co-variation matrix.
* n represents number of data points used to compute the covariance matrix.
* x_i represents the i-th data point.
* $\bar{x}$ is the mean deviation of all data point

– Eigendecomposition :

Next, we perform eigendecomposition on the covariance matrix, denoted by Σ. This process yields the eigenvalues (λ) and corresponding eigenvectors ($\mathbf{v}$) of Σ. The eigenvectors represent the principal axes of the data distribution, while the eigenvalues quantify the proportion of variance explained by each principal component. This relationship is mathematically expressed by the eigendecomposition equation:

$\Sigma\mathbf{v} = \lambda\mathbf{v}$

– Principal Component Selection :

PCA sorts the eigenvalues in descending order to prioritize the principal components that capture the most variance within the input data. These components are then selectively retained to form a lower-dimensional representation, often determined by the cumulative explained variance ratio.

– Projection :

Finally, PCA projects the original dataset onto the selected principal components to obtain the reduced-dimensional representation :

$$Projected\,Data = Standardized\,Data \times Selected\,Eigenvectors \tag{19}$$

2. **How do PCA and SVD improve the recommendation and alleviate the Sparsity Problem?**

Matrix factorization techniques reduce the dimensionality of the feature space by capturing the most significant features in the data. By employing those techniques,they have the ability to enable effective dimensionality reduction, compressing the information while preserving the important features of the original input. This technique finds particular application within RSs, this reduction in dimensionality helps alleviate sparsity by focusing on the most relevant features and reducing noise [8]. Simultaneously, they identify the latent factors or underlying patterns in the dataset, which may correspond to user preferences or item characteristics in recommendation systems. By extracting these features, They enable recommendation algorithms to operate in a lower-dimensional space defined by the principal components. This feature extraction process helps uncover meaningful relationships between users and items, even in sparse datasets where direct interactions are limited [45].

The reduced-dimensional representation obtained through Matrix factorization techniques provides a more compact and generalized view of the data. Instead of considering each individual feature or interaction,

recommendation systems can leverage the principal components to capture broader trends and similarities among users and items. This enhanced generalization allows recommendation models to make more accurate predictions and recommendations, even for users or items with sparse data [39], those techniques have also the inherent ability to filter out noise and irrelevant information by prioritizing the principal components that capture the most variance in the data. In sparse recommendation datasets, where noise and irrelevant signals may be prevalent, Matrix factorization techniques help distinguish meaningful patterns from random fluctuations. This noise reduction contributes to more robust and reliable recommendation models, leading to improved performance, even in sparse environments.

1.5.1.2.4 Clustering Strategies for Enhanced Recommender Systems

Recommender systems can be enriched by employing methods that cluster users or items. Clustering might be really efficient in matters that involve scalability, sparsity as well as cold-start issues. Data clustering ensures more efficient recommendation algorithms that lower computational costs besides enabling accurate predictions which don't require much input. In the following sections, we shall be looking at different clustering techniques and their utility in the context of RSs enhancement.

1. **Cold users problem :**

Within recommender systems (RS), the cold start problem manifests as the inherent challenge of generating accurate recommendations for new users who possess a limited or entirely absent interaction history with the system (zero interaction). This problem arises because traditional recommendation algorithms rely on historical user-item interactions to generate personalized recommendations [77].

	Item a	**Item b**	**Item c**	**Item d**
User 1	3.6	2.7	-	3.9
User 2	4.5	5	1.7	-
User 3	-	-	-	-
⋮	⋮	⋮	⋮	⋮
User n	-	-	-	-

Table 1.5: Illustration of Cold Users in RSs

In Table 1.5, For new users (e.g., From User 3 to User n) Most entries in their row are "-", indicating that they have not interacted with many items in the system. As a result, because of the lack of historical data, standard recommendation algorithms may find problems when it comes to providing accurate recommendations for these users.

2. **Clustering Models :**

The inherent sparsity of RSs databases can reduce the effectiveness of those systems, particularly in cold-start scenarios. To address this challenge and enhance prediction quality, RS can employ clustering techniques. A common approach involves clustering users based on their rating behavior to identify groups with similar preferences. Alternatively, bi-clustering, which clusters both users and items, can be employed. Furthermore, for RS that incorporate social network information, leveraging techniques such as tagging, explicit social links, and trust information within the clustering process has demonstrated efficacy in improving recommendation accuracy [7].

Clustering, as explained prior in the anterior sections, is an unsupervised machine learning technique that groups similar data points together. In the context of RSs, clustering can be used for several purposes like:

- Find set of users with similar preferences: This is employed to make the suggestions to users based on the preferences of other users in their cluster.
- Discover new item categories or genres: This can be used to improve the organization of items in a recommender system and to make more relevant recommendations.
- Identify outliers or anomalies: This can be used to detect and filter out items that are not relevant to the majority of users.

And in this section, we're gonna dive into the most known clustering models:

- **K-Means**

  K-Means clustering is a centroid-based algorithm widely used in recommender systems due to its simplicity, efficiency, and effectiveness in handling sparse data. Here's how it works [29] [51]:

  - **Centroids and Cluster Formation:**
    - K-Means first requires pre-defining the desired number of clusters (denoted by "K").
    - The algorithm then selects K data points (randomly) as the first step centroids, which represent the centers of each oen of these clusters.
    - In the context of RS, these data points could be users or items, depending on the chosen clustering approach (user clustering or item clustering).
  - **Iterative Refinement:**
    1. **Assignment Step:** For each user or item (data point), K-Means calculates its distance (usually Euclidean distance) to all K centroids, using the formula :

$$\text{distance}(x, y) = \sqrt{\sum_{i=1}^{d} (x_i - y_i)^2} \tag{20}$$

    2. **Recentering Step:** Once all data points are assigned to the closest clusters, K-Means repositions the centroid to be more representative of the current cluster members :

$$C_k = \frac{1}{|S_k|} \sum_{x \in S_k} x \tag{21}$$

    3. These two steps (assignment and recentering) are repeated iteratively until a convergence criterion is met. This criterion typically involves reaching a point where the centroids no longer significantly change between iterations, indicating that the clusters have stabilized.

  Eventual Benefits for RS:

  - Reduced Sparsity Impact: K-Means groups similar users or items together. This reduces the sparsity problem by allowing recommendations to be drawn from a smaller set of more relevant neighbors within a cluster, even for users with limited rating history.
  - Cold Start Mitigation: New users or items with few ratings can be assigned to a cluster based on their limited features or initial ratings. This allows the system to make preliminary recommendations based on the cluster's preferences, mitigating the cold start problem.
  - Scalability: K-Means is a relatively efficient algorithm, making it suitable for handling large datasets often encountered in recommender systems.

  Eventual Limitations and Considerations:

  - Predefined Clusters: The number of clusters (K) needs to be determined beforehand, which can be challenging and might impact performance.

- Sensitivity to Initialization: K-Means can get trapped in local optima depending on the initial placement of centroids. Different initializations might lead to different cluster configurations.
- Distance Metric: The choice of distance metric (e.g., Euclidean distance) can influence the clustering results.

- **Gaussian Mixture Models (GMM)**

  While K-Means clustering offers a powerful approach for handling sparsity in recommender systems, another technique gaining traction is Gaussian Mixture Models (GMMs).

  GMMs: A Probabilistic Approach

  Unlike K-Means, which works with hard assignments of data points to single clusters, GMMs take a probabilistic approach. They assume that the data (users or items in RS) are generated from a mixture of several Gaussian distributions (bell-shaped curves). Each Gaussian distribution refers to a cluster then we calculate the points belonging to each one of the clusters, given by:

$$p(x) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x|\mu_k, \Sigma_k) \tag{22}$$

  - π_k represents the mixing coefficient, μ_k denotes the mean vector, and Σ_k stands for the covariance matrix of the k-th Gaussian distribution.

  The GMM Learning Process:

  Learning a GMM involves estimating the model parameters (mixing coefficients, means, and covariances) that best fit the observed data [76]. **Eventual Benefits for RS:**

  - Soft Clustering: GMMs provide a more nuanced representation of user and item preferences compared to K-Means. Users or items can have varying degrees of membership in multiple clusters, reflecting a more realistic scenario where user preferences aren't always strictly confined to a single category.
  - Modeling Heterogeneity: GMMs can effectively capture the heterogeneity present in user and item characteristics. Different user segments might have diverse rating patterns, which GMMs can model by adjusting the parameters of the underlying Gaussian distributions.
  - Cold Start Handling: Similar to K-Means, GMMs can assign new users or items to clusters based on their limited features or initial ratings, enabling recommendations even with sparse data.

  Eventual Disadvantages for RS:

  - Computational Complexity: GMMs require more computational resources compared to simpler clustering methods like K-Means, especially with large datasets. The Expectation-Maximization (EM) algorithm used to fit GMMs can be computationally intensive.
  - Overfitting Risk: With a large number of components or complex models, overfitting might happen, which results in poor generalization to new data.
  - Assumption of Gaussian Distribution: GMMs assume that the data within each cluster follows a Gaussian distribution. If the real data distribution significantly deviates from this assumption, it can be a problem.

- **Agglomerative Clustering**

  K-Means and GMMs represent popular clustering techniques for recommender systems. However, another approach worth considering is agglomerative clustering, which offers a hierarchical perspective on data organization [74].

  Building a Hierarchy:

  Unlike K-Means and GMMs, which directly partition data points into predefined clusters, agglomerative clustering takes a bottom-up approach. It starts with each data point as an individual cluster. Then, it

iteratively merges the most similar clusters based on a chosen distance metric (e.g., Euclidean distance) to form a hierarchy of clusters.

For example, the similarity between two clusters C_1 and C_2 can be defined as:

$$\text{sim}(C_1, C_2) = \frac{\sum_{x \in C_1} \sum_{y \in C_2} \text{similarity}(x, y)}{|C_1| \times |C_2|} \tag{23}$$

Types of Agglomerative Clustering:

There are two main types of agglomerative clustering [53]:

- **Ward's Method:** This method merges clusters that minimize the increase in within-cluster variance after merging. It aims to create clusters with high internal homogeneity (similarity of data points within a cluster).
- **Average Linkage:** This approach combines clusters based on the average distance between all data points in one cluster and all data points in another cluster. Its goal is to create clusters that exhibit similar average pairwise distances between their data points.

Eventual Benefits for RS:

- Discovery of Hierarchical Relationships: Agglomerative clustering allows you to explore the data and uncover natural groupings at different levels of granularity. This can be particularly useful in RS for identifying diverse user segments or item categories with varying degrees of similarity.
- Flexibility in Choosing Cluster Number: Unlike K-Means and GMMs where the number of clusters needs to be predefined, agglomerative clustering doesn't require this upfront decision. You can analyze the resulting hierarchy and determine the most appropriate level of granularity for your RS application.
- Effective Handling of Outliers: Agglomerative clustering can be less sensitive to outliers compared to centroid-based methods like K-Means. This can be beneficial for RS datasets that might contain some noisy or atypical user/item profiles.

Eventual Disadvantages for RS:

- Computational Complexity: Agglomerative clustering can be computationally expensive for large datasets, especially with complete linkage methods.
- Hierarchical Representation: While the hierarchy provides valuable insights, interpreting and choosing the most relevant level of clustering for RS applications can require additional analysis.
- Distance Metric Selection: The choice of distance metric significantly impacts the clustering results. Selecting an appropriate metric that reflects the underlying relationships in your RS data is crucial.

3. **Deep Learning Based Clustering Models :**

Deep clustering techniques combine the representational power of deep neural networks with traditional clustering methods to enhance performance on complex datasets. However, these methods face challenges such as accumulated errors during alternating optimization and the necessity to process the entire dataset simultaneously [50] [24].

- **Self Organizing Map (SOM) Clustering**

  Self-Organizing Maps (SOMs) are a type of artificial neural network that excels in data visualization, dimensionality reduction, and clustering. In the realm of recommender systems (RS), SOMs

have emerged as a promising technique for addressing sparsity and cold start issues, particularly in collaborative filtering approaches [57].

Understanding SOMs:

SOMs are inspired by the human brain's ability to organize and interpret sensory information. They consist of a two-dimensional grid of neurons, each represented by a vector of weights. The goal of SOM training is to adjust these weights such that the resulting map captures the underlying patterns and relationships within the data [23].

Several variants of SOMs have been proposed for RS applications, addressing specific challenges and enhancing performance [23]:

- **Weighted SOMs:** Incorporate user or item weights to reflect the importance or relevance of certain data points in the training process.
- **Hybrid SOMs:** Combine SOMs with other techniques, such as collaborative filtering or content-based filtering, to leverage the strengths of each approach.
- **SOMs with Dynamic Neighborhoods:** Adapt the neighborhood size during training to better capture the varying density and relationships in the data.

Eventual Benefits for RS:

- Visualizing User and Item Relationships: SOMs provide a visual representation of user and item relationships, allowing for insights into user preferences and item similarities. This visualization can aid in understanding user behavior and identifying potential recommendations.
- Dimensionality Reduction: SOMs can effectively reduce the dimensionality of high-dimensional data, such as user-item rating matrices in RS. This can be particularly beneficial for large datasets where computational efficiency is crucial.
- Clustering and Recommendation: SOMs can be used to cluster users or items based on their preferences or characteristics. These clusters can then be leveraged for recommendation purposes, suggesting items similar to those enjoyed by users within the same cluster.

Eventual Disadvantages for RS:

- Data Preprocessing: SOMs perform best on normalized and scaled data. Proper data preprocessing is essential to ensure meaningful and accurate representations.
- Interpretation of SOM Maps: While SOMs provide a visual representation, interpreting the map and extracting meaningful insights can require domain knowledge and additional analysis.

- **Deep Contrastive Clustering :**

Contrastive clustering is a technique that aims to learn representations by jointly optimizing clustering assignments and a contrastive loss function. It leverages the idea of embedding data points into a latent space where similar points are grouped together while dissimilar points are pushed apart.

Peng et al.'s (2021) [46] method introduces a novel approach by conducting contrastive learning at both instance and cluster levels. This dual-level learning incorporates "label as representation" to facilitate clustering, making it distinct from general-purpose representation learning methods. By treating labels as special representations, the method enables effective instance- and cluster-level representation learning in row and column spaces, respectively.

Learning Objective:

Given a dataset $X = \{x_1, x_2, ..., x_n\}$, the objective of contrastive clustering is to learn a set of cluster assignments $C = \{c_1, c_2, ..., c_n\}$ and a set of embeddings $Z = \{z_1, z_2, ..., z_n\}$ such that similar points are assigned to the same cluster and dissimilar points are assigned to different clusters.

The method consists of three key components: a pair construction backbone (PCB), an instance-level contrastive head (ICH), and a cluster-level contrastive head (CCH) [46].

– Uses data augmentations to create data pairs.
– Extracts features from augmented samples.

Instance-Level Contrastive Head (ICH)

– Applies contrastive learning on the row space of the feature matrix.
– Uses cosine similarity and a two-layer nonlinear MLP to measure pair-wise similarities and compute instance-level contrastive loss.

Cluster-Level Contrastive Head (CCH)

– Projects data samples into a cluster space, where each dimension represents the probability of belonging to a specific cluster.
– Uses cosine similarity and a two-layer nonlinear MLP to measure similarities between cluster pairs and compute cluster-level contrastive loss.

Objective Function The overall objective function combines the instance-level contrastive loss L_{ins} and the cluster-level contrastive loss L_{clu}, formulated as:

$$L = L_{\text{ins}} + L_{\text{clu}} \tag{25}$$

The combined loss is optimized in a one-stage, end-to-end process, ensuring both instance-level and cluster-level learning. Despite the potential to use dynamic weights for balancing, simple addition of the two losses has been found effective [46].

Contrastive Loss Function:

The contrastive loss function encourages similar points to be close together in the embedding space while pushing dissimilar points apart. It is typically defined as:

$$\mathcal{L}_{\text{contrastive}} = -\frac{1}{n}\sum_{i=1}^{n} \log \frac{e^{s\cdot \text{sim}(z_i, z_{c_i})}}{\sum_{j=1}^{n} e^{s\cdot \text{sim}(z_i, z_j)}} \tag{26}$$

where:

– n is the number of data points.
– z_i is the embedding of data point x_i.
– c_i is the cluster assignment of data point x_i.
– $\text{sim}(z_i, z_j)$ is a similarity function between embeddings z_i and z_j.
– s is a scaling parameter that controls the sharpness of the contrastive function.

Eventual Benefits for RS:

– Enhanced Representation Learning: It enhances the quality of learned embeddings by maximizing the similarities between augmented views of the same pair and minimizing those between different pairs. This leads to more accurate and meaningful embeddings, which can significantly enhance the performance of recommender systems.
– Better Handling of Sparse Data: it's particularly effective in dealing with sparse datasets. By leveraging augmented data, it can extract robust features even when user-item interactions are limited, leading to improved recommendations in scenarios with sparse data.
– Improved Clustering Performance: The dual contrastive learning approach, involving both instance-level and cluster-level contrastive loss, helps in decoupling the learning processes for individual instances and clusters. This decoupling enhances the ability to form well-defined and distinct clusters, which in turn leads to more precise and personalized recommendations by accurately identifying user segments and item categories.

Eventual Disadvantages for RS:

- Choice of Embedding Space: It is important to carefully design CC components to capture meaningful relationships in the data.
- Scalability: Contrastive clustering can be computationally expensive, especially for large datasets and high-dimensional embeddings. Efficient algorithms and optimization techniques are needed to scale the method to real-world applications.
- Interpretability: Interpreting the learned clusters and embeddings can be challenging, particularly in complex datasets. Domain knowledge and visualization techniques may be required to gain insights from the clustering results.

4. **Comparison of Clustering Techniques for Recommender Systems :**

In this section, we present a tabular that will contrast various clustering techniques in recommender systems :

Feature	**K-Means**	**Gaussian Mixture Models (GMMs)**	**Agglomerative Clustering**	**Self-Organizing Maps (SOMs)**	**Contrastive Clustering**
Clustering Approach	Partitional	Probabilistic (soft assignments)	Hierarchical	Neural network-based	Similarity-based representations
Cluster Number	Predefined (K)	Predefined (K)	Flexible	Not predefined (grid size)	Emerges from data similarity
Data Representation	Assumes spherical clusters [29]	Models data with Gaussian distributions [76]	Considers pairwise similarities [53]	Projects high-dimensional data to lower dimensions [23]	Learns representations that maximize similarity within clusters and minimize similarity between clusters [46]
Preference Modeling	Limited ability to capture nuances [51]	Captures heterogeneity in user/item characteristics [76]	No explicit preference modeling [74]	Can visualize user/item relationships [50]	Captures preferences by learning contrastive representations [76]
Cold Start Handling	Assigns new data to existing clusters	Assigns new data to existing clusters	Iteratively merges with similar clusters	May require additional techniques	Assigns new data based on its learned representation
Computational Cost	Low	Moderate	High (especially for large datasets)	Moderate	Can be computationally expensive
Outlier Sensitivity	More sensitive	Less sensitive	Less sensitive	Can handle outliers to some extent	Outliers can influence the learned representations
Distance Metric	Impacts cluster formation	Impacts cluster formation	Impacts merging decisions	Impacts map topology	Impacts similarity calculations

Feature	K-Means	Gaussian Mixture Models (GMMs)	Agglomerative Clustering	Self-Organizing Maps (SOMs)	Contrastive Clustering
General Data Analysis	Widely used for efficient clustering [51]	Well-suited for heterogeneous data [76]	Useful for exploring hierarchical structures [74]	Effective for dimensionality reduction and visualization [57]	Can be used for unsupervised representation learning [46]
RS Applications	Effective for handling sparsity and cold start	Captures user/item heterogeneity for personalized recommendations	Explores user/item relationships at different granularities	Visualizes user/item relationships and supports collaborative filtering	Can be used for recommendation tasks such as item retrieval and personalized ranking

Table 1.6: Comparison of Clustering Techniques in Recommender Systems

1.5.1.3 Memory-Based vs. Model-Based

We will, in this section, look at how recommendations differ with different types of systems: those using memory vs those relying on models. This will involve us tabulating their differences across various dimensions such as how they can be relevant in diverse recommendation situations.

Table 1.7: Comparison of Memory-Based and Model-Based Recommender Systems

Feature	Memory-Based	Model-Based
Approach	Analyzes past user-item interactions to find similar users/items	Uses statistical or machine learning models to capture underlying patterns
Underlying Assumption	Users with similar historical preferences will continue to have similar preferences	Hidden relationships and patterns exist in the data
Recommendation Calculation	Based on ratings of similar users/items	Based on model predictions for unseen items
Advantages	• Simple and easy to implement • Transparent recommendations • Can offer serendipitous discoveries	• Scalable for large datasets • Handles cold start (with limitations) • Can incorporate additional data

Feature	Memory-Based	Model-Based
Disadvantages	• Scalability issues with large datasets • Struggles with cold start (new users/items) • Limited representation of complex patterns	• More complex to develop and tune • Black box nature - recommendations less interpretable • Prone to overfitting .

1.5.2 Content-Based Filtering

In RSs, Content-Based Filtering (CBF) serves to suggests items to the users they are most likely to like depending on what they like and the characteristics of the items themselves, it's a well-known technique for the enhancement of RSs, and we will diving into the its core principle in the following sections.

1.5.2.1 Data Types Spectrum

In constructing CBF models and data-driven systems, the availability and variety of data are crucial. Real-world data can be categorized into several forms: structured, semi-structured, and unstructured. Additionally, metadata, which provides information about other data, plays an essential role. Below, we briefly explore that buy comparing the three data types [64].

Table 1.8 put into highlight each one's characteristics :

Aspect	Structured Data	Semi-Structured Data	Unstructured Data
Definition	Data that is organized in a predefined format, often in rows and columns.	Data that does not conform to a rigid structure but contains tags or markers to separate semantic elements.	Data that lacks a specific format or structure, not organized in a predefined manner.
Examples	Databases, spreadsheets, CSV files, SQL tables.	JSON, XML files, HTML documents, emails with metadata.	Text documents, emails, images, videos, audio files, social media posts.
Storage	Relational databases (e.g., MySQL).	NoSQL databases (e.g., MongoDB)	NoSQL databases (e.g., MongoDB), file systems.
Type of Data	Numbers, Dates.	Tagged data, key-value pairs.	Text, multimedia content, logs.
Ease of Analysis	Easily analyzed with standard querying and analysis tools (SQL).	Requires parsing and may need specialized tools for querying and analysis.	Requires advanced techniques like natural language processing (NLP), image processing, machine learning.
Schema	Fixed schema with predefined fields and data types.	Flexible schema, can evolve over time.	No fixed schema, flexible and adaptable to various data types.

Aspect	Structured Data	Semi-Structured Data	Unstructured Data
Searchability	Highly searchable and indexable due to structured format.	Searchable with appropriate indexing and parsing methods.	Less searchable without metadata or indexing, requires sophisticated search algorithms.
Example Use Cases	Financial reporting, inventory management, customer databases.	Web data exchange, API responses, configuration files.	Sentiment analysis, image recognition, video analysis, content recommendation.
Data Integration	Easier to integrate due to consistency and structured format.	Moderate integration efforts, may require transformation.	More challenging to integrate, requires data preprocessing and transformation.
Scalability	Can be scalable but often limited by rigid schema constraints.	Generally scalable with flexible schema designs.	Highly scalable with appropriate tools and techniques for handling large volumes.

Table 1.8: Comparison of Structured, Semi-Structured, and Unstructured Data

In addition to these categories, data can also be classified based on its content, such as textual and visual data:

- Textual Data: It's an unstructured text form. Textual data is generated from sources such as emails, social media posts, documents, and web pages. It is crucial for applications like sentiment analysis, information retrieval, and machine translation. Advances in NLP have significantly improved the ability to process and understand textual data, enabling applications like chatbots, automated summarization, and recommendation systems.
- Visual Data: This includes data in the form of images and videos. Visual data is prevalent in applications like facial recognition, object detection, medical imaging, and video surveillance. The rise of deep learning, particularly Convolutional Neural Networks (CNNs), has revolutionized the processing and analysis of visual data. These models can automatically extract features from images and videos, facilitating advanced applications in fields such as autonomous driving, augmented reality, and visual search.

1.5.2.2 Visual Data Insights : Feature Extraction and Applications in Modern Recommender Systems

Visual data, encompassing images and videos, represents one of the most complex and information-rich forms available today. The rapid advancement in imaging technologies and the proliferation of visual content on platforms like social media, surveillance systems, and medical imaging devices have led to an explosion in the volume of visual data generated daily. This type of data is highly unstructured, requiring sophisticated techniques for effective processing and analysis [40].

The fundament of visual data analysis in recent years has been the development and application of Convolutional Neural Networks (CNNs). These deep learning models have achieved state-of-the-art performance in a variety of visual tasks. Beyond CNNs, more recent advancements include Generative Adversarial Networks (GANs) and Vision Transformers (ViTs) [19].

Production of visual data is seeing an exponential growth, according to approximate statistics, as illustrated in Figure 1.19. Applications of visual data are vast and varied. In the medical field, visual data from

MRI, CT scans, and X-rays are analyzed using deep learning models to assist in diagnosing diseases, detecting tumors, and planning surgeries. In the automotive industry, visual data is critical for the development of autonomous vehicles, enabling tasks such as object detection, lane tracking, and pedestrian recognition. In the realm of security, facial recognition systems and surveillance cameras rely on visual data to enhance public safety and security measures. Moreover, visual data plays a pivotal role in the entertainment industry, fashion, and e-commerce [20]. In addition, visual data is essential for recommender systems, which enhance user experience by suggesting personalized content, products, and services based on individual preferences and behaviors [40].

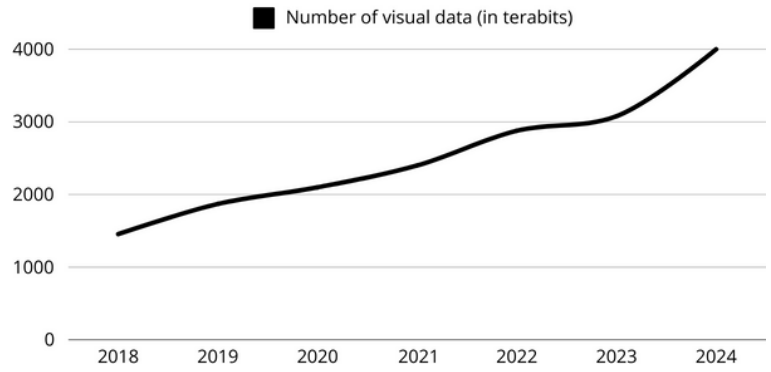

Figure 1.19: Approximative Visual Data Growth over the Last 6 Years

The challenges associated with visual data include the need for large labeled datasets, significant computational resources, and the development of robust algorithms that can handle variations in lighting, occlusion, and perspective. Additionally, ethical concerns such as privacy, security, and bias in visual data systems need to be addressed.

1.5.2.2.1 Feature Extraction Models

Feature extraction models are pivotal in the domain of computer vision, serving as the backbone for processing and analyzing visual data. These models are designed to automatically identify and extract meaningful patterns, textures, and structures from raw images, converting them into a structured format that machine learning algorithms can readily utilize. The importance of feature extraction lies in its ability to reduce the dimensionality of visual data while preserving essential information, enabling more efficient and accurate analysis [67].

However, the rise of Deep Learning, particularly CNNs, has significantly advanced the field. CNNs, with their multi-layered architecture, excel in capturing complex hierarchical features directly from raw pixel data, making them exceptionally powerful for various visual tasks. These models transform raw visual data into high-level feature representations that encapsulate both spatial and semantic information, facilitating their integration into machine learning pipelines for applications like object detection, image classification, and visual-based recommendation systems. As a result, feature extraction models are indispensable in harnessing the rich information embedded in visual data, driving innovations and enhancing the capabilities of modern machine learning systems [73].

1. **Types of Feature Extraction Models :**

Feature extraction models can be broadly categorized into *traditional methods* and *deep learning-based approaches*:

SIFT (Scale-Invariant Feature Transform) and HOG (Histogram of Oriented Gradients) were pillar techniques that significantly advanced early image analysis. SIFT identifies distinctive features in an image and generates robust vector, leading it to be highly effective for object recognition and matching tasks. HOG, on the other hand, focuses on the distribution of gradient orientations in localized portions of an image, excelling in tasks like pedestrian detection by capturing edge and texture information [67].

CNNs have demonstrated unparalleled effectiveness in automatically learning and extracting hierarchical features directly from raw pixel data. Unlike traditional methods, which rely on handcrafted features, CNNs leverage multiple layers of convolutional filters to capture increasingly abstract and complex features, from edges and textures in the initial layers to high-level semantic features in deeper layers. Architectures such as AlexNet, VGGNet, ResNet, and Inception have set new benchmarks in various visual tasks, including image classification, object detection, and segmentation. These deep learning models not only enhance the accuracy and robustness of feature extraction but also facilitate the end-to-end learning process, significantly reducing the need for manual feature engineering [20].

2. **Popular CNN Architectures:**

- AlexNet: One of the early CNN architectures that demonstrated the potential of deep learning in visual tasks.
- VGGNet: Known for its simplicity and depth, contributing to high performance in image recognition tasks.
- ResNet: Introduced residual learning, allowing for the training of much deeper networks without the vanishing gradient problem.
- Inception: With its unique architecture combining multiple convolutional layers with different filter sizes. Transfer Learning.

Extracted features from visual data play a crucial role in enhancing the performance of recommendation algorithms. By converting raw visual inputs into high-level feature representations, these models enable more accurate and personalized recommendations. These integrations allow recommendation algorithms to move beyond simple collaborative filtering or text-based approaches, incorporating rich visual information to deliver more nuanced and contextually appropriate suggestions, ultimately improving user satisfaction.

1.5.3 Hybrid Models

Hybrid RSs introduce two or more recommendation approaches to improve performance while reducing the limitations of individual methods. Typically, collaborative filtering is combined with another technique, often content-based filtering (CBF), as illustrated in Figure 1.20, in an attempt to avoid the ramp-up problem [14].

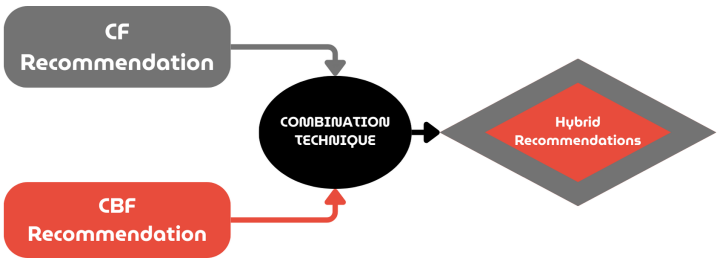

Figure 1.20: The Hybridization Process in RSs

Hybrid recommender systems merge diverse recommendation algorithms or data representations to boost performance. They recognize that different approaches may excel in various aspects of recommendation tasks; for instance, collaborative filtering captures user-item interactions well, while content-based filtering is adept at recommending items based on attributes. By combining these strengths, hybrid systems achieve better recommendation performance and robustness than individual methods. As a result, hybrid recommender systems stand out as potent solutions, promising improved accuracy, adaptability, and resilience to diverse user preferences and item traits [13].

1. **Type of Hybrid Models :**

- **Weighted :** A weighted hybrid recommender system integrates scores from different recommendation techniques. For example, the P-Tango system combines collaborative and content-based recommendations through a linear formula. This method leverages the strengths of each system and allows for straightforward adjustments, but it presumes equal effectiveness across techniques, which is not always accurate. For instance, collaborative recommenders may struggle with items that have few ratings [14] [13].
- **Feature combination :** In a CF-CBF hybrid, collaborative data enhances content-based recommendations, mitigating data sparsity issues. For instance, a CF-CBF book recommender uses collaborative filtering to create new user features, then applies fuzzy c-means clustering and type-2 fuzzy logic to categorize books for each user type. Finally, content-based filtering suggests relevant books to each user [14] [13].
- **Cascade :** Cascade hybridization differs from other hybrid methods by operating in stages. Initially, one technique produces a basic ranking, which a second technique then refines. This approach optimizes efficiency by applying the secondary technique only to items that need further differentiation, thus avoiding unnecessary processing. Unlike weighted hybrids that apply all techniques to all items, cascade hybrids focus on where additional processing is required. This method also tolerates noise in low-priority techniques since the primary recommender's ratings are only used for refinement [14] [13].
- **Switching :** Switching hybrids alternate between recommendation techniques based on predefined criteria. For instance, a CF-CBF system may switch to content-based recommendations when collaborative filtering cannot provide sufficient suggestions. Variations of the same strategy, such as CBF1-CBF2 or CF1-CF2, can also employ a switching method [13] [14].
- **Meta-level :** Meta-level hybrid recommender systems use an order-sensitive approach, utilizing the entire model generated by one technique as input for another. Typically, content-based recommenders build item representation models, which collaborative recommenders then use to match items with the user preferences.[14] [13].
- **Mixed :** When multiple recommendations are needed simultaneously, a 'mixed' hybrid approach combines suggestions from various techniques. For example, the PTV system recommends TV schedules by merging content-based analysis of show descriptions with collaborative user preferences. This method addresses the 'new item' issue by recommending unrated shows through content analysis but does not fully resolve the 'new user' problem. However, when integrated into digital TV, it can track viewing habits to refine profiles and introduce new items that a content-focused approach might miss [14] [13].

Hybrid recommender systems improve recommendation accuracy by combining diverse information sources and algorithms, capturing various user preferences and item characteristics [14], The following will elucidate the benefits and highlight the challenges associated with this approach :

2. **Benefits and Advantages of Hybrid Recommender Systems:**

- Enhanced Accuracy: Combining multiple recommendation algorithms and data representations leads to more accurate and personalized recommendations.
- Greater Robustness: Hybrid systems filter out noise and sparse data, providing reliable and stable recommendations.
- Improved Adaptability: They can address various recommendation scenarios effectively, from collaborative filtering to context-aware recommendation.
- Interpretability: Some hybrid systems offer transparency, enhancing understanding of user preferences and item characteristics.

3. **Challenges and Considerations:**

- Data Integration: Ensuring compatibility and consistency among different data sources is challenging.
- Algorithm Selection: Choosing the right combination of algorithms and tuning parameters requires careful experimentation.
- Computational Complexity: Managing computational resources efficiently, especially for large datasets, is essential.
- Overfitting: Balancing model complexity with generalization performance is crucial to avoid overfitting.
- User Experience: Ensuring relevant and timely recommendations is essential for user acceptance and satisfaction.

1.6 Recent Recommendation Algorithms

As we move beyond classical recommendation algorithms, it is essential to explore recent advancements that leverage new technologies and methodologies. These advancements offer enhanced RSs.

- **Technologies of IoT in RSs**

  IoT devices, such as smart home appliances, wearable technology, and connected vehicles, continuously collect real-time data about user preferences and behaviors. This data enables recommendation systems to provide context-aware suggestions, enhancing user experience in various smart environments. For instance, a smart refrigerator can recommend recipes based on the available ingredients, while a wearable fitness tracker can suggest workout routines tailored to the user's activity levels and health goals [9]. Service recommendations can be achieved using Social IoT (SIoT), which leverages data from various IoT devices. In 2012 the concept of social objects was introduced [4], where objects mimic social behaviors to create relationships [55].

  To improve intelligent accessibility [10], SIoT gives devices a flexible social structure, allowing them to independently form social connections. SIoT and IoT contribute to the rapid growth of data on the Internet, challenging conventional research methods.

  However, IoT features such as heterogeneity, real-time communication, scalability [10], and mobility present complexities and challenges to IoT recommendation systems, potentially limiting their performance [9]

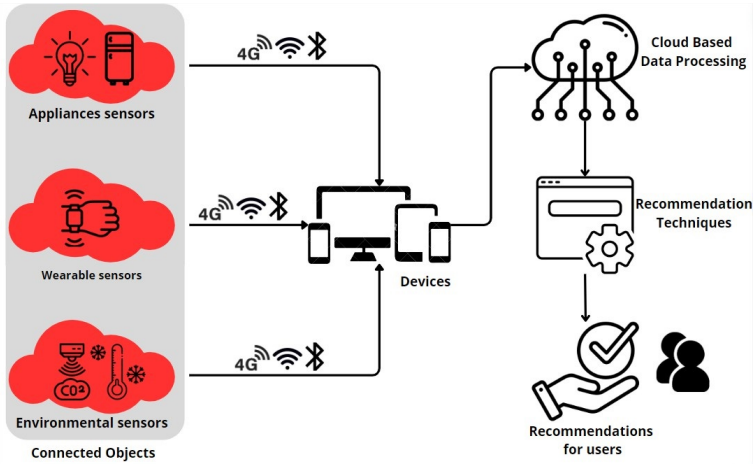

Figure 1.21: Integrated Framework for Data Collection, Transmission, Processing, and User Recommendations from Connected Objects

The figure 1.21 illustrates a framework for integrating and processing data from various connected objects to generate user recommendations:

- Connected Objects: This section includes:
  - Appliances Sensors: Monitor and control household or industrial appliances.
  - Wearable Sensors: Track physiological and activity data.
  - Environmental Sensors: Measure environmental parameters like temperature and humidity.
- WiFi: Wireless networking for short to medium ranges.
- 4G: High-speed mobile data transmission.
- Bluetooth: Short-range wireless data exchange.
- Devices: Act as intermediaries, aggregating and securely transmitting data to the cloud.
- Cloud-Based Data Processing: In the cloud, advanced processing and analysis are performed to handle large data volumes and apply complex algorithms.
- Recommendation Techniques: Insights from data processing generate technical recommendations through analytical and machine learning techniques.
- Recommendations for Users: Technical recommendations are translated into user-friendly advice, offering actionable guidance to enhance user experience and device performance.

- **Web 3.0 and Recommender Systems**

  Web 3.0, characterized by its decentralized architecture, graph-based data structures, and extensive data diversity [61], presents unique opportunities for recommendation systems. Hybrid recommendation algorithms can harness the power of Web 3.0 to analyze complex relationships within vast datasets. This leads to more accurate and diverse recommendations, catering to a broader spectrum of user preferences. The integration of blockchain technology ensures data privacy and security, which is crucial in decentralized networks [42]. The rise of Web3 technology has brought about a new paradigm in recommendation systems. With Web3, users have more control over their data, and blockchain technology ensures the security and privacy of that data. This new paradigm enables recommendation systems to deliver more personalized and accurate recommendations to users, it's widely known that this technology facilitates decentralized systems, giving users control over their data and preventing misuse or manipulation. This allows users to share more accurate information about their preferences and behaviors, leading to better recommendations, its use of Blockchain technology further ensures data security and privacy. Its immutable nature and encryption techniques keep data tamper-proof

and accessible only to authorized users. This boosts user confidence in sharing their data, enhancing recommendation accuracy [83].

- **Generative Models in RSs**

  These models can generate synthetic user data to address cold-start problems [68], where little to no historical data exists for new users or items. Additionally, generative models can create novel recommendations by learning underlying data distributions, thus enhancing the diversity and creativity of suggestions. The ability to generate new content tailored to user preferences opens up exciting possibilities for personalized experiences. Generative AI can be used in recommendation systems to overcome limitations of retrieval-based paradigms. It offers the potential to generate personalized items that meet users' specific information needs. Additionally, the use of natural language instructions, such as those facilitated by ChatGPT, allows users to express their information needs more precisely. This combination of generative AI and user instructions points towards a next-generation recommender paradigm. One proposed approach is the GeneRec paradigm, which uses an AI generator to personalize content generation and leverages user instructions to acquire information needs [80]. The AI generator can repurpose existing items and create new items based on the guidance provided by the user instructions. Various fidelity checks are emphasized to ensure the trustworthiness of the generated items.

- **Deep Learning and Feature Extraction**

  Deep learning models can analyze complex data types, such as images, text, and time-series data, to uncover intricate patterns and user preferences. Compared to traditional statistical methods, deep learning approaches offer superior performance in capturing the nuances of user behavior. Kim and Lim (2023) proposed a deep neural collaborative filtering model for e-book service recommendations that utilizes minimal data inputs like user numbers, book numbers, and ratings. Their model includes an input layer for embedding data, a feature extraction layer for analyzing correlations, a multilayer perceptron, and an output layer. This approach addresses data sparsity and improves recommendation accuracy through Bayesian optimization to fine-tune hyperparameters [41].

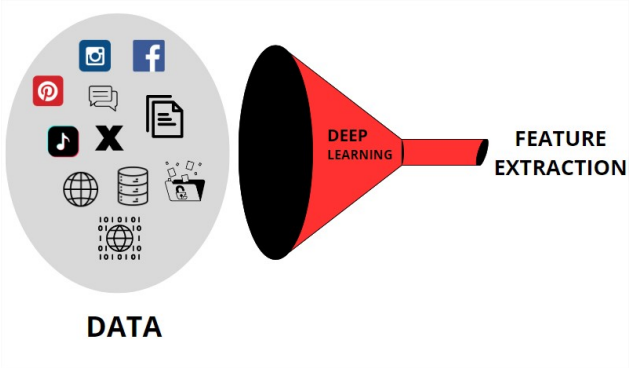

Figure 1.22: Data Processing Pipeline for Feature Extraction in Deep Learning

This Figure 1.22 illustrates the process of feature extraction from various data sources using deep learning. On the left, data from social media, textual sources, and the internet are collected. These data types are rich but often unstructured. The funnel in the center represents deep learning approaches like the ones details earlier, these features capture essential characteristics and are ready for use in tasks like classification, clustering, and recommendation.

- **Temporal Historical Data and Successful Modelization**

  Modeling temporal historical data is crucial for understanding and predicting user behavior over time. Long Short-Term Memory (LSTM) networks, a type of RNN, are particularly effective in this regard. LSTMs can capture long-term dependencies and patterns in sequential data, making them ideal for

recommendations based on historical usage trends. By incorporating temporal dynamics, recommendation systems can deliver timely and relevant suggestions, adapting to changes in user preferences and item popularity.

These advancements in recommendation algorithms demonstrate the ongoing evolution of the field, driven by technological innovations and the growing complexity of user data. Embracing these modern techniques can lead to more personalized, accurate, and engaging recommendations, ultimately enhancing the user experience across various applications.

1.7 Evaluation of Recommender Systems

Assessing a recommendation system involves measuring its performance against predefined goals, which dictates the selection of appropriate metrics. The dataset is typically split into training and test sets. Predictions are made for each user's rating on an item using ratings from others in the training set and their own ratings for other items [16].

In recommender systems, various evaluation metrics are employed to assess the performance of the system. Some common metrics include:

- **Accuracy**: Accuracy measures the proportion of correct predictions made by the recommender system.

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}} \tag{27}$$

  Where:

  - Number of correct predictions: Number of recommendations that match the user's actual preferences.
  - Total number of predictions: Total number of recommendations made by the system.

- **Recall**: Recall measures the proportion of relevant items that are recommended by the system.

$$\text{Recall} = \frac{\text{Number of relevant items recommended}}{\text{Total number of relevant items}} \tag{28}$$

  Where:

  - Number of relevant items recommended: Number of items recommended by the system that are actually relevant to the user.
  - Total number of relevant items: Total number of items that are relevant to the user.

- **Precision**: Precision measures the proportion of recommended items that are relevant to the user.

$$\text{Precision} = \frac{\text{Number of relevant items recommended}}{\text{Total number of items recommended}} \tag{29}$$

  Where:

  - Number of relevant items recommended: Number of items recommended by the system that are actually relevant to the user.
  - Total number of items recommended: Total number of items recommended by the system.

- **MAE (Mean Absolute Error)**: MAE measures the average absolute difference between the predicted ratings and the actual ratings given by users.

$$\text{MAE} = \frac{1}{n} \sum_{i=1}^{n} |r_{ui} - \hat{r}_{ui}| \tag{30}$$

  Where:

  - n: Total number of ratings.
  - r_{ui}: Actual rating given by user u for item i.
  - $\hat{r}_{ui}$: Predicted rating for user u on item i.

- **RMSE (Root Mean Squared Error)**: RMSE measures the square root of the average of the squared differences between the predicted ratings and the actual ratings given by users.

$$\text{RMSE} = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (r_{ui} - \hat{r}_{ui})^2} \tag{31}$$

  Where:

  - n: Total number of ratings.
  - r_{ui}: Actual rating given by user u for item i.
  - $\hat{r}_{ui}$: Predicted rating for user u on item i.

- **Diversity**: Diversity measures the range of different items recommended by the system. It ensures that the recommendations are not too similar to each other, thus providing a broader spectrum of options to the user [15], we can ditinct two types of it :

  - Intra-list Diversity: This is the average dissimilarity between all pairs of items in the recommendation list for a user. Higher intra-list diversity indicates a more varied set of recommendations. The dissimilarity between items i and j can be measured using a distance metric $d(i, j)$, such as cosine similarity or Jaccard distance.

$$\text{Intra-list Diversity} = \frac{2}{n(n-1)} \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} d(i, j) \tag{32}$$

    Where:

    * n is the number of items in the recommendation list.
    * $d(i, j)$ is the distance between items i and j.

  - Aggregate Diversity : This measures the coverage of different items across all users, promoting a wide usage of the item catalog. It can be calculated as the number of unique items recommended to all users divided by the total number of items in the catalog.

$$\text{Aggregate Diversity} = \frac{\text{Number of unique items recommended}}{\text{Total number of items in catalog}} \tag{33}$$

- **Novelty**: measures the extent to which the recommended items are new or unexpected to the user, enhancing the discovery experience [15]. we can distinct two type of it too :

  - Self-Novelty: This metric evaluates the novelty of recommendations based on the user's history. A higher self-novelty score suggests that the recommendations include items with which the user hasn't previously rated. Self-novelty for a user u can be defined as :

$$\text{Self-Novelty}(u) = \frac{1}{|R_u|} \sum_{i \in R_u} (1 - p_u(i)) \tag{34}$$

    Where:

* R_u is the ensemble of recommended items for user u.
* $p_u(i)$ represents the probability that u has interacted with i in the past.

– Catalog Novelty: This evaluates how new the recommendations are relative to the entire user base. It can be measured by the average inverse popularity of recommended items.

$$\text{Catalog Novelty} = \frac{1}{|R|} \sum_{i \in R} \frac{1}{\log(1 + \text{popularity}(i))} \tag{35}$$

Where:

* R is the ensemble of recommended items.
* The term popularity(i) typically is the number of users who have interacted with item i.

1.8 Concrete Objectives of a Recommendation System

Previously, we thoroughly examined the underlying technologies used in recommendation systems, including their operation, specific characteristics, and technical specifications. Having thus consolidated a solid foundation of technical understanding, it is now imperative to explore the very purpose of these systems. Our goal is to shed light on their intrinsic value and fundamental objectives in the context of user experience. This involves illustrating the multiple facets of interactions between users and recommendation systems, as well as the benefits for both users and service providers.

Service providers have several compelling reasons for adopting recommendation engines, as these systems play a pivotal role in the success of their services :

- Maximizing revenue and optimizing operational strategies: The primary goal is to increase sales by recommending products or services tailored to users' tastes and needs.
- Alter recommendations: it's an essential feature aims to encourage users to discover lesser-known options. For example, in the context of a movie recommendation system, the service provider aims to offer a wide range of movies rather than limiting to the most popular titles.
- Understanding users and enhancing their engagement: An important aspect is the ability to understand users' preferences, whether explicitly or predicted. These data can be leveraged by the service provider to optimize its business strategy.

1.9 Recommendation Engines in Modern Culture

In contemporary society, recommendation engines have become ubiquitous, profoundly impacting various aspects of our daily lives. These intelligent systems analyze vast amounts of data to provide personalized suggestions, thereby shaping our consumption patterns, entertainment choices, and even decision-making processes.

E-commerce Platforms: Pinterest's Visual Discovery - Pinterest employs advanced visual recommendation algorithms to suggest pins based on users' interactions, preferences, and browsing history. By analyzing the visual content of pins and user engagement metrics, Pinterest provides personalized recommendations that inspire users to discover new ideas and products.

Social Media Platforms: TikTok's For You Page - TikTok's For You page utilizes visual data analysis to curate a personalized feed of short videos tailored to each user's interests and preferences. By analyzing

video content, user interactions, and trends, TikTok delivers a captivating and engaging browsing experience that encourages users to explore a diverse range of content.

Art and Design Platforms: Adobe Stock's Visual Search - Adobe Stock leverages visual search technology to enable users to discover images based on their visual attributes. By analyzing image content and similarity metrics, Adobe Stock provides users with relevant and visually appealing image recommendations that streamline the creative process.

Augmented Reality Applications: Snapchat's AR Lenses - Snapchat's AR lenses utilize visual data processing techniques to overlay interactive and immersive effects onto users' photos and videos. By analyzing facial features and environmental cues, Snapchat delivers personalized AR experiences that captivate users and drive engagement.

1.10 Conclusion :

In conclusion, this chapter has provided a detailed exploration of recommender systems, covering their definition, historical development, classification into collaborative filtering, content-based filtering, and hybrid models, as well as an overview of classical and modern algorithms. It also discussed evaluation techniques and the societal impact of recommendation engines. This chapter sets the stage for deeper dives into specific methodologies.

General Conclusion

This dissertation provided an deep exploration of the state-of-the-art in RSs, exploring the multiple techniques and approaches shaping the IT field. Delving into collaborative filtering, content-based and hybrid approaches, diverse approaches has been examined, each offering unique strengths and challenges. Moreover, the integration of visual data emerges as a pivotal frontier in recommendation technology, promising to enhance recommendation accuracy and user satisfaction.

By synthesizing the findings presented in this chapter, it becomes evident that recommender systems continue to evolve, driven by innovations in data analysis, machine learning, and user modeling. As we move forward, leveraging the synergies between traditional data sources and visual information holds immense potential for unlocking new opportunities in personalized recommendation experiences.

However, alongside the promise of visual data integration, it is imperative to acknowledge the challenges and limitations inherent in its adoption. These include issues related to data privacy and security, algorithmic bias, scalability, and interpretability. Addressing these challenges requires a holistic approach that encompasses ethical considerations, regulatory frameworks, and technological innovations.

Looking ahead, several promising avenues emerge for advancing the integration of visual data in recommender systems. One direction involves exploring novel deep learning architectures tailored for visual recommendation tasks, capable of extracting meaningful representations from complex visual data. Additionally, research efforts should focus on developing hybrid approaches that seamlessly use visual and non-visual information to enhance recommendation quality and diversity. Furthermore, there is a growing need to investigate the role of user context and situational factors in shaping visual preferences, paving the way for context-aware recommendation systems.

To conclude this state-of-the-art, the synthesis of findings from this study underscores the dynamic nature of recommender systems and the transformative potential of integrating visual data. As recommender systems continue to evolve, driven by advancements in data analysis and machine learning, leveraging the synergies between traditional data sources and visual information holds promise for revolutionizing personalized recommendation experiences. By addressing the associated challenges we can unlock new frontiers in recommendation technology and enrich the lives of users in the digital era.

Bibliography

[1] Gediminas Adomavicius and YoungOk Kwon. Improving aggregate recommendation diversity using ranking-based techniques. *IEEE Transactions on Knowledge and Data Engineering*, 24(5):896–911, 2011.

[2] Gediminas Adomavicius and Alexander Tuzhilin. Personalization technologies: a process-oriented perspective. *Communications of the ACM*, 48(10):83–90, 2005.

[3] Farzana Anowar, Samira Sadaoui, and Bassant Selim. Conceptual and empirical comparison of dimensionality reduction algorithms (pca, kpca, lda, mds, svd, lle, isomap, le, ica, t-sne). *Computer Science Review*, 40:100378, 2021.

[4] Luigi Atzori, Antonio Iera, Giacomo Morabito, and Michele Nitti. The social internet of things (siot)–when social networks meet the internet of things: Concept, architecture and network characterization. *Computer networks*, 56(16):3594–3608, 2012.

[5] Amokrane Belloui. *L'usage des concepts du web sémantique dans le filtrage d'information collaboratif.* PhD thesis, ESI, 2008.

[6] Rianne van den Berg, Thomas N Kipf, and Max Welling. Graph convolutional matrix completion. *arXiv preprint arXiv:1706.02263*, 2017.

[7] J. Bobadilla, F. Ortega, A. Hernando, and A. Gutiérrez. Recommender systems survey. *Knowledge-Based Systems*, 46:109–132, 2013.

[8] Jesus Bobadilla and Francisco Serradilla. The effect of sparsity on collaborative filtering metrics. volume 92, pages 9–17, 01 2009.

[9] Halima Bouazza, Bachir Said, and Fatima Zohra Laallam. A hybrid iot services recommender system using social iot. *Journal of King Saud University - Computer and Information Sciences*, 34(8, Part B):5633–5645, 2022.

[10] Halima Bouazza, Laallam Fatima Zohra, and Bachir Said. Integration of internet of things and social network: Social iot general review. In *Advances in Data Science, Cyber Security and IT Applications: First International Conference on Computing, ICC 2019, Riyadh, Saudi Arabia, December 10–12, 2019, Proceedings, Part II 1*, pages 312–324. Springer, 2019.

[11] John Breese, David Heckerman, and Carl Kadie. Empirical analysis of predictive algorithm for collaborative filtering. *UAI*, 01 2013.

[12] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

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

[14] E. Cano and M. Morisio. Hybrid recommender systems: a systematic literature review. *Intelligent Data Analysis*, 21:1487–1524, 2017.

[15] Pablo Castells, Saul Vargas, and Jun Wang. Novelty and diversity metrics for recommender systems: Choice, discovery and relevance. *Proceedings of International Workshop on Diversity in Document Retrieval (DDR)*, 01 2011.

[16] Mingang Chen and Pan Liu. Performance evaluation of recommender systems. *International Journal of Performability Engineering*, 13(8):1246, 2017.

[17] Leah E. Steinberg Amrit S. Tuladhar Daniel Lew, Ben Sowell. Memory-based recommender systems, 2007. Consulté le: 24 mai 2024.

[18] Minh-Phung Do, Dung Nguyen, and Loc Nguyen. Model-based approach for collaborative filtering. 08 2010.

[19] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[20] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021.

[21] Michael D Ekstrand, John T Riedl, Joseph A Konstan, et al. Collaborative filtering recommender systems. *Foundations and Trends® in Human–Computer Interaction*, 4(2):81–173, 2011.

[22] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Knowledge Discovery and Data Mining*, 1996.

[23] Arthur Flexer. On the use of self-organizing maps for clustering and visualization. *Intelligent Data Analysis*, 5(5):373–384, 2001.

[24] Mudasir A Ganaie, Minghui Hu, Ashwani Kumar Malik, Muhammad Tanveer, and Ponnuthurai N Suganthan. Ensemble deep learning: A review. *Engineering Applications of Artificial Intelligence*, 115:105151, 2022.

[25] Sarik Ghazarian and Mohammad Ali Nematbakhsh. Enhancing memory-based collaborative filtering for group recommender systems. *Expert systems with applications*, 42(7):3801–3812, 2015.

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

[27] Carlos A Gomez-Uribe and Neil Hunt. The netflix recommender system: Algorithms, business value, and innovation. *ACM Transactions on Management Information Systems (TMIS)*, 6(4):1–19, 2015.

[28] Davide Feltoni Gurini, Fabio Gasparetti, Alessandro Micarelli, and Giuseppe Sansonetti. A sentiment-based approach to twitter user recommendation. *RSWeb@ RecSys*, 1066:1–4, 2013.

[29] Jiawei Han, M. Kamber, and J. Pei. Data mining: Concepts and techniques. pages 585–631, 01 2006.

[30] Ruining He, Chen Fang, Zhaowen Wang, and Julian McAuley. Vista: A visually, socially, and temporally-aware model for artistic recommendation. In *Proceedings of the 10th ACM conference on recommender systems*, pages 309–316, 2016.

[31] Ruining He and Julian McAuley. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *proceedings of the 25th international conference on world wide web*, pages 507–517, 2016.

[32] Jon Herlocker, Joseph Konstan, and John Riedl. An empirical analysis of design choices in neighborhood-based collaborative filtering algorithms. *Information Retrieval*, 5:287–310, 01 2002.

[33] Jonathan L. Herlocker, Joseph A. Konstan, Al Borchers, and John Riedl. An algorithmic framework for performing collaborative filtering. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '99, page 230–237, New York, NY, USA, 1999. Association for Computing Machinery.

[34] Jonathan L Herlocker, Joseph A Konstan, Al Borchers, and John Riedl. An algorithmic framework for performing collaborative filtering. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 230–237, 1999.

[35] Balázs Hidasi, Massimo Quadrana, Alexandros Karatzoglou, and Domonkos Tikk. Parallel recurrent neural network architectures for feature-rich session-based recommendations. In *Proceedings of the 10th ACM conference on recommender systems*, pages 241–248, 2016.

[36] Bahrudin Hrnjica, Denis Music, and Selver Softic. Model-based recommender systems. *Trends in Cloud-based IoT*, pages 125–146, 2020.

[37] Folasade Isinkaye, Yetunde Folajimi, and Bolanle Ojokoh. Recommendation systems: Principles, methods and evaluation. *Egyptian Informatics Journal*, 16, 08 2015.

[38] Gawesh Jawaheer, Peter Weller, and Patty Kostkova. Modeling user preferences in recommender systems: A classification framework for explicit and implicit user feedback. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, 4(2):1–26, 2014.

[39] Alexandros Karatzoglou and Balázs Hidasi. Deep learning for recommender systems. pages 396–397, 08 2017.

[40] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4401–4410, 2019.

[41] Ji-Yoon Kim and Chae-Kwan Lim. Feature extracted deep neural collaborative filtering for e-book service recommendations. *Applied Sciences*, 13(11):6833, 2023.

[42] Sung Rim Kim and Joon Hee Kwon. A study on recommendation method based on web 3.0. *Journal of Korea Society of Digital Industry and Information Management*, 8(4):43–51, 2012.

[43] Durk P Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. Semi-supervised learning with deep generative models. *Advances in neural information processing systems*, 27, 2014.

[44] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

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

[46] Yunfan Li, Peng Hu, Zitao Liu, Dezhong Peng, Joey Tianyi Zhou, and Xi Peng. Contrastive clustering. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 8547–8555, 2021.

[47] Greg Linden, B. Smith, and J. York. Linden g, smith b and york j: 'amazon.com recommendations: item-to-item collaborative filtering', internet comput. ieee, , 7. *Internet Computing, IEEE*, 7:76 – 80, 02 2003.

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

[49] M Malone, ND Black, M Lydon, and M Cinnamond. Acoustic analysis of infantile stridor: a review. *Medical and Biological Engineering and Computing*, 31:85–96, 1993.

[50] Ioannis Maniadis Metaxas, Georgios Tzimiropoulos, and Ioannis Patras. Divclust: Controlling diversity in deep clustering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3418–3428, 2023.

[51] Mohssen Mohammed, Muhammad Khan, and Eihab Bashier. *Machine Learning: Algorithms and Applications*. 07 2016.

[52] Kriz Moses, Ankur Miglani, Pavan Kumar Kankar, et al. Deep cnn-based damage classification of milled rice grains using a high-magnification image dataset. *Computers and Electronics in Agriculture*, 195:106811, 2022.

[53] Daniel Müllner. Modern hierarchical, agglomerative clustering algorithms. *arXiv preprint arXiv:1109.2378*, 2011.

[54] Luong Vuong Nguyen, Quoc-Trinh Vo, and Tri-Hai Nguyen. Adaptive knn-based extended collaborative filtering recommendation services. *Big Data and Cognitive Computing*, 7(2):106, 2023. Published: 31 May 2023.

[55] Michele Nitti, Luigi Atzori, and Irena Pletikosa Cvijikj. Network navigability in the social internet of things. In *2014 IEEE world forum on internet of things (WF-IoT)*, pages 405–410. IEEE, 2014.

[56] Bryan Perozzi, Leman Akoglu, Patricia Iglesias Sánchez, and Emmanuel Müller. Focused clustering and outlier detection in large attributed graphs. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1346–1355, 2014.

[57] Latchoumi Thamarai Pugazhendhi, Raja Kothandaraman, and Balamurugan Karnan. Implementation of visual clustering strategy in self-organizing map for wear studies samples printed using fdm. *Traitement du Signal*, 39(2):531, 2022.

[58] Paul Resnick, Neophytos Iacovou, Mitesh 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.

[59] Francesco Ricci, Lior Rokach, and Bracha Shapira. *Recommender Systems Handbook*, volume 1-35, pages 1–35. 10 2010.

[60] Daniel A Roberts, Sho Yaida, and Boris Hanin. *The principles of deep learning theory*, volume 46. Cambridge University Press Cambridge, MA, USA, 2022.

[61] Riaan Rudman and Rikus Bruwer. Defining web 3.0: opportunities and challenges. *The electronic library*, 34(1):132–154, 2016.

[62] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. *Advances in neural information processing systems*, 29, 2016.

[63] Imtiaz Hossain Sarker, ASM Kayes, S Badsha, H Alqahtani, Paul Watters, and Andrew Ng. Cybersecurity data science: an overview from machine learning perspective. *Journal of Big Data*, 7(1):1–29, 2020.

[64] Iqbal Sarker. Machine learning: Algorithms, real-world applications and research directions. *SN Computer Science*, 2, 03 2021.

[65] Iqbal Sarker, Alan Colman, Jun Han, · Asif, Asif Khan, · Yoosef, B Abushark, and Khaled Salah. Behavdt: A behavioral decision tree learning to build user-centric context-aware predictive model. *Mobile Networks and Applications*, 25, 06 2020.

[66] 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*, pages 285–295, 2001.

[67] Nellutla Sasikala, V Swathipriya, M Ashwini, V Preethi, A Pranavi, and M Ranjith. Feature extraction of real-time image using sift algorithm. *European Journal of Electrical Engineering and Computer Science*, 4(3), 2020.

[68] Andrew I Schein, Alexandrin Popescul, Lyle H Ungar, and David M Pennock. Generative models for cold-start recommendations. In *Proceedings of the 2001 SIGIR workshop on recommender systems*, volume 6. Citeseer, 2001.

[69] Suvash Sedhain, Aditya Krishna Menon, Scott Sanner, and Lexing Xie. Autorec: Autoencoders meet collaborative filtering. In *Proceedings of the 24th international conference on World Wide Web*, pages 111–112, 2015.

[70] S. Shengli and CX. Ling. Hybrid cost-sensitive decision tree, knowledge discovery in databases. In *PKDD 2005, Proceedings of 9th European Conference on Principles and Practice of Knowledge Discovery in Databases*, volume 3721, 2005.

[71] Xiaoyuan Su and Taghi M Khoshgoftaar. A survey of collaborative filtering techniques. *Advances in artificial intelligence*, 2009, 2009.

[72] V Subramaniyaswamy and R Logesh. Adaptive knn based recommender system through mining of user preferences. *Wireless Personal Communications*, 97:2229–2247, 2017.

[73] Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *Proceedings of the 36th International Conference on Machine Learning*, pages 6105–6114, 2019.

[74] Eric K Tokuda, Cesar H Comin, and Luciano da F Costa. Revisiting agglomerative clustering. *Physica A: Statistical mechanics and its applications*, 585:126433, 2022.

[75] Petar Veličković, William Fedus, William L Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. Deep graph infomax. *arXiv preprint arXiv:1809.10341*, 2018.

[76] Jinghua Wang and Jianmin Jiang. Unsupervised deep clustering via adaptive gmm modeling and optimization. *Neurocomputing*, 433:199–211, 2021.

[77] Chuhan Wu, Fangzhao Wu, Tao Qi, and Yongfeng Huang. Are big recommendation models fair to cold users? *arXiv preprint arXiv:2202.13607*, 2022.

[78] Lirong Wu, Haitao Lin, Yufei Huang, Tianyu Fan, and Stan Z Li. Extracting low-/high-frequency knowledge from graph neural networks and injecting it into mlps: an effective gnn-to-mlp distillation framework. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 10351–10360, 2023.

[79] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? *Advances in neural information processing systems*, 27, 2014.

[80] Fajie Yuan, Qianli Ma, Haitao Qin, Xiaoling Han, Zhongkai Hu, Maoying Chen, Wei Zhang, and Xiangnan Liu. Generative recommendation: Towards next-generation recommender paradigm. *arXiv preprint arXiv:2304.03516*, 2023.

[81] Xiaojin Zhu, Zoubin Ghahramani, and John D Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *Proceedings of the 20th International conference on Machine learning (ICML-03)*, pages 912–919, 2003.

[82] Xiaojin Jerry Zhu. Semi-supervised learning literature survey. 2005.

[83] Solidity Ácademy. Recommendation systems with web3 and machine learning: A game-changing approach to user experience. `https://medium.com/coinmonks/recommendation-systems-with-web3-and-machine-learning-a-game-changing-approach-to-user-experience-`
April 2023. Published in Coinmonks. Followed by Solidity Academy.

[84] Vijay Çagadre. Recurrent neural networks: A beginner's guide. `https://vijaygadre.medium.com/recurrent-neural-networks-a-beginners-guide-16333bd2eeb1`, 2023. Accessed: 2023-10-03.