Universidade Federal de São Carlos– UFSCar
Centro de Ciências Exatas e de Tecnologia– CCET
Departamento de Computação– DC
Programa de Pós-Graduação em Ciência da Computação– PPGCC

Guilherme Brandão Martins

Optimum-Path Forest in support of Collaborative Filtering

São Carlos
2023

Guilherme Brandão Martins

Optimum-Path Forest in support of Collaborative Filtering

Tese apresentada ao Programa de Pós-Graduação em Ciência da Computação do Centro de Ciências Exatas e de Tecnologia da Universidade Federal de São Carlos, como parte dos requisitos para a obtenção do título de Doutor em Ciência da Computação.

Área de concentração: Metodologias e Técnicas de Computação

Supervisor: Prof. Dr. João Paulo Papa

São Carlos
2023

— *To my mom, a true symbol of persistence, ethics, and love.*

Acknowledgements

First and foremost, I would like to express my deepest gratitude to my *mother* and *father* for dedicating a significant portion of their lives to ensure that I had access to a quality education, which enabled me to reach the current stage of my academic life. The encouragement for reading and studying that I received from childhood, as well as the emotional and financial support, was indispensable. I will always be thankful to you for this demonstration of sacrifice and love.

I am grateful to my brothers, *Gu* and *Fer*, for the friendship and companionship that make life lighter and more joyful. With you, I started to learn the beauty of sharing what I had and knew.

I am grateful to *Helô*, my beloved and dedicated partner, for supporting me wholeheartedly since the beginning of this journey. Her attention to my emotional needs has been crucial in many aspects throughout these challenging years.

I also thank other family members, especially uncles, aunts, and grandparents, for the love and support I have always received from them.

I thank my friends from the youth spiritist movement for the countless experiences we have shared, experiences that have been enriching my humanity every day.

I thank *Rhuan*, a longtime friend who welcomed me when I returned to Bauru to conduct this research. To my colleagues at the Recogna laboratory, I am grateful for the assistance I received. In particular, I want to thank *Luis Afonso, Gustavo Rosa*, and *Marcos Cleison* for their feedback and valuable suggestions to improve this research work.

I thank UNESP and UFSCar for providing the physical infrastructure and necessary human resources for the development of this work. In particular, I am grateful to Professors *Wilson, Nilceu, Alexandre*, and *Vânia* for the significant influence they had on my personal and academic growth.

At last, I want to thank Professor *João Paulo Papa*, who has offered me so many opportunities and provided invaluable guidance throughout this journey. Professor Papa, thank you for your immense kindness.

Resumo

Algoritmos de aprendizado de máquina têm sido aplicados em diversos desafios computacionais, dentre os quais Sistemas Recomendadores (do inglês, *Recommender Systems, RS*) contém um conjunto de técnicas e abordagens para lidar efetivamente com extensos volumes de dados e oferecer conteúdos personalizados e relevantes aos usuários. Tais sistemas devem ser capazes de lidar com problemas relativos aos dados, como esparsidade, escalabilidade e *cold start*, e a Filtragem Colaborativa (do inglês, *Collaborative Filtering, CF*) tradicionalmente tem sido a principal estratégia para lidar com esses desafios. Uma das maneiras de aprimorar os resultados de recomendação é utilizar fontes auxiliares de informação para compensar a falta de dados de CF, como interações usuário-item. Todavia, diferentes interpretações acerca dos problemas mencionados poderiam ser exploradas. O presente trabalho contribui na área de aprendizado de máquina propondo abordagens para lidar com os desafios supracitados. Esta tese é constituída por uma coletânea de trabalhos desenvolvidos pelo autor durante o período de pesquisa, que foram publicados ou submetidos até a atualidade, apresentando: (i) uma revisão sistemática da literatura que analisa e discute abordagens recentes baseadas em aprendizagem profunda para recomendação sob condições de esparsidade, além de identificar desafios e limitações na área de CF; (ii) uma abordagem baseada em Fatoração de Matriz (do inglês, *Matrix Factorization, MF*) que explora esparsidade relativa a CF para fusão de classificadores; (iii) um modelo alternativo do classificador não-supervisionado Floresta de Caminhos Ótimos (do inglês, *Optimum-Path Forest, OPF*) projetado para operar eficientemente em conjuntos de dados de grande escala, utilizando relação de adjacência baseada em grafo de k-vizinhos-aproximados; e (iv) um modelo OPF para agrupamento de dados baseado no conceito de vizinhança compartilhada para aliviar esparsidade e alta dimensionalidade durante a recomendação baseada em CF. Os resultados experimentais alcançados por meio de tais trabalhos corroboram as hipóteses da presente tese.

Palavras-chave: Filtragem Colaborativa. Floresta de Caminhos Ótimos. Esparsidade.

Abstract

Machine learning algorithms are being applied in various computational challenges, among which Recommender Systems (RS) present a range of techniques and approaches to effectively manage large volumes of data and provide personalized and relevant content to users. Such systems must be able to handle data-related issues such as sparsity, scalability, and the cold start problem and Collaborative Filtering (CF) has traditionally been the primary strategy for addressing those challenges. One way to tackled those problems and improve recommendation results is by leveraging auxiliary information sources to compensate the lack of CF data, such as user-item interactions. However, different interpretations of the mentioned problems should be explored. The current work contributes in the field of machine learning by proposing approaches to address the mentioned challenges. This thesis presents a collection of works developed by the author throughout the research period, which have been published or submitted up to the present, encompassing: (i) a systematic literature review which analyzes and discuss recent deep learning approaches employed for CF under sparse-related conditions, while also identifying the challenges and limitations within the field; (ii) a Matrix Factorization (MF)-based approach that leverages CF-related sparsity for the purpose of classifiers fusion; (iii) an alternative unsupervised Optimum-Path Forest (OPF) designed to perform efficiently in large-scale datasets by employing k-approximate-nearest-neighbors graph as its adjacency relation; and (iv) an OPF clustering model built upon the shared-neighborhood concept to alleviate sparsity and high dimensionality issues during CF-based recommendation. The experimental results achieved through such works corroborate the hypotheses of the present thesis.

Keywords: Collaborative Filtering. Optimum-Path Forest. Sparsity.

List of Figures

List of Tables

List of acronyms

ACDA Attentive Contextual Denoising Autoencoder

ACC Accuracy Score

AE Autoencoder

AFM Attention Factorization Machine

ANR Aspect-Based Neural Recommender

ARSR Attentive Recurrent Social Recommendation

BERT Bidirectional Encoder Representations from Transformers

BPR Bayesian Personalized Ranking

BCW Breast Cancer

CAAE Collaborative Adversarial Autoencoders

CB Content-Based

CF Collaborative Filtering

CNN Convolutional Neural Networks

CM Constrain Model

CSRM Collaborative Session-based Recommendation Machine

DAE Denoising Autoencoder

DB Davies-Bouldin

DBSCAN Density-Based Spatial Clustering for Applications with Noise

DCCR Deep Collaborative Conjunctive Recommender for Rating Prediction

DMF Deep Matrix Factorization

DNN Deep Neural Network

DT Decision Templates

EM Expectation-Maximization

FM Factorization Machines

GAN Generative Adversarial Network

GCN Graph Convolutional Networks

GMF Generalized Matrix Factorization

GMM Gaussian Mixture Model

GNN Graph Neural Networks

GRU Gated Recurrent Units

HIN Heterogeneous Information Network

HR Hit Ratio

HF Hybrid Filtering

IFE Implicit Feedback Embedding

IME Inner Memory Encoder

JCR Journal Citation Reports

JRL Joint Representation Learning

KG Knowledge Graphs

KNN k-Nearest Neighbor

KL Kullback–Leibler

KV-MN Key-Value Memory Network

LSTM Long Short-Term Memory

MAE Mean Absolute Error

MF Matrix Factorization

MLP Multi-Layer Perceptron

MSE Mean Squared Error

mSDA marginalized Stacked Denoising Autoencoder

NDCG Normalized Discount Cumulative Gain

NLP Natural Language Processing

NCF Neural Network-based Collaborative Filtering

NMF Non-Negative Matrix Factorization

NN Nearest Neighbors

OME Outer Memory Encoder

OPF Optimum-Path Forest

OPT Optimum-Path Trees

PMF Probabilistic Matrix Factorization

ReLU Rectified Linear Unit

RIM Rating Independent Model

RMSE Root Mean Squared Error

RNN Recurrent Neural Networks

RS Recommender System

SNN Shared Near Neighbor

SDAE Stacked Denoising Autoencoder

SLR Systematic Literature Review

SPE Semi-Parametric Embedding

SVD Singular Value Decomposition

SRCMF Social Review-enhanced Convolutional Matrix Factorization

TF Tensor Factorization

VAE Variational Autoencoder

Contents

Chapter 1

Introduction

Over the past few decades, machine learning techniques have been widely applied to address various computational challenges, and their widespread adoption has reached numerous research fields. These fields include image segmentation, video processing, spam filtering, remote sensing, speech recognition, and content recommendation, among others. Regarding the latter research area, Recommender System (RS) plays a fundamental and strategic role in data management and information filtering, particularly in the context of the Big Data Era. Such systems have been consistently embraced by companies to enhance their online services, including multimedia streaming, e-commerce, and social network visibility, to name a few examples. Furthermore, the recommendation task has demonstrated its importance in helping online users cope with information overload. In this sense, RS has become one of the most popular and powerful tools in the digital world, benefiting both researchers and industry professionals alike (RICCI; ROKACH; SHAPIRA, 2021).

According to Bobadilla et al. (2013), there are fundamental actions and considerations that should be followed when developing a content recommender, with two of them standing out: (i) identifying the type of data available in the database and (ii) the filtering algorithm to be used. The former encompasses data such as rating scores, user profiles, social information, and items inner content; the latter involves choosing the primary strategy, whether it is Content-Based (CB), Collaborative Filtering (CF), or Hybrid Filtering (HF), based on the available data to address the recommendation task (BALABANOVIC; SHOHAM, 1997; ADOMAVICIUS; TUZHILIN, 2005). In summary, CB filtering relies on item features and internal content data, CF focuses solely on user-item interactions, and HF combines both filtering algorithms to provide recommendations.

CF stands as one of the most prominent strategies to effectively managing large

datasets and providing users with personalized content. By following such a strategy, the system recommends to a given user the items other users with similar preferences rated in the past (RICCI; ROKACH; SHAPIRA, 2021). In other words, the historical users profile, i.e., known interactions or *rating scores*, act as the primary data source for either select similar users and to make recommendations. This can be a significant advantage, especially in recommendation scenarios where the internal content details of users or items are either partially or entirely unavailable. CF's ability to make recommendations based on user interactions alleviates the need for extensive content data, making it valuable in situations where CB-based approaches may be limited by data availability or quality.

The user-item interactions could be obtained explicitly (rating score given by the user) or implicitly (e.g., user's most frequent search queries, number of times a video is watched). However, since it is well-known users tend to interact with a limited items set (COVINGTON; ADAMS; SARGIN, 2016; LIANG et al., 2018a; RICCI; ROKACH; SHAPIRA, 2021), typical CF databases tend to be highly sparse, meaning that most of the user-item interactions are missing or unknown. Also, data sparsity can become more pronounced as the user base or item catalog grows large, which can lead to scalability challenges. These challenges are common in CF-based recommenders and must be addressed effectively. Therefore, data sparsity is critical drawback in CF-based models because the lack of sufficient data can adversely impact the quality of recommendations, particularly when machine learning techniques are employed to process large volumes of data.

Among the numerous recommendation approaches, graph-based approaches have shown promising results in addressing data sparsity and scalability in collaborative filtering. Graph-based CF offers several advantages, including the ability to capture complex user-item relationships, handle sparse data, and incorporate various types of interactions. Algorithms like graph embeddings (ZHANG; WANG; LUO, 2020), random walks (GUO et al., 2023), and Graph Neural Networks (GNN) (MOLAEI et al., 2021; QIAN et al., 2022) are often employed to analyze these graphs and make recommendations based on structural information and connectivity patterns. Such an approach is particularly useful in scenarios where traditional CF methods may struggle with scalability, data sparsity or when additional contextual information, such as implicit feedback or social connections, can be incorporated into the recommendation process.

In recent machine learning literature, the Optimum-Path Forest (OPF), a graph-based classifier, has demonstrated promising results across several application domains such as image segmentation (CHEN et al., 2018c; CAPPABIANCO et al., 2012), video summarization (MARTINS et al., 2020), sampling (PASSOS et al., 2022), intrusion detection (COSTA et al., 2015), and Gaussian parameters initialization (ROSA et al., 2014; MARTINS; PAPA, 2022b). Featuring supervised (PAPA; AO, 2008; PAPA; FALCÃO; SUZUKI, 2009), semi-supervised (AMORIM; FALCÃO; CARVALHO, 2014) and unsu-

pervised (ROCHA; CAPPABIANCO; FALCÃO, 2009) versions, OPF does a graph-based interpretation about the dataset, in which graph nodes are samples, and a adjacency relation guides how nodes are connected. In essence, OPF partitions the graph following some predefined methodology to select the most representative nodes (prototypes), from which optimum-paths will be calculated up to the remaining samples through a prototype competition process that employs some path-cost function.

Sparse and high-dimensional feature spaces often characterize the data processed by CF and represent a constant challenge to the development of effective CF-based techniques. Besides, it is well-known those characteristics negatively impact the effectiveness of distance-based classifiers. Although OPF may not be based on distance computation explicitly, the latter is used to calculate the "connectivity strength" between samples. OPF may exhibit certain limitations when applied to CF data, as it was originally designed for handling computer vision tasks where dense data is dominant. This implies that the characteristics and requirements of CF data may differ from those of other scenarios where OPF is effectively employed. For instance, CF data is inherently sparse and large-scale, easily reaches high dimensionality, and often includes noise and outliers. Hence, it is imperative to conduct investigations into OPF taking into account those conditions to develop more robust solutions for this graph-based classifier.

Another data sparsity-related constraint in CF is the inability to make reliable and meaningful recommendation due to the lack of sufficient known interactions in the system. From that, a particular challenge arises when new users join the system or new items are added to it called the cold start problem (AGGARWAL, 2016b; BOBADILLA et al., 2013). Under such circumstances there may not be enough data for accurate recommendations. To address this challenge, a commonly adopted strategy in many studies involves seeking additional sources of information related to users and items, instead of depending solely on scarce rating scores. In essence, researchers combine various types of filtering algorithms in a hybrid recommendation approach to mitigate the cold start effects.

Recently, many works addressed the cold start problem using Deep Neural Network (DNN) techniques (MAGRON; FÉVOTTE, 2022; QIAN et al., 2022; WANG et al., 2020; ZHANG et al., 2022). Their current popularity in RS research is reinforced by the outstanding results those techniques have achieved in different application domains such as classification, feature learning, data augmentation, dimensionality reduction, and Natural Language Processing (NLP) for instance. The architectural flexibility of those kinds of bio-inspired algorithms enables the development of hybrid RS approaches capable of jointly incorporating various types of data[1] and generating predictive values that represent recommendations. Moreover, their chained layer-wise and non-linear optimiza-

[1] The data would come from different sources, which is also referenced by some authors as side information or auxiliary information (SHI; LARSON; HANJALIC, 2014; CHEN et al., 2018b).

tion structure allows the system to hierarchically and efficiently extract various types of information, with the potential to improve recommendation quality, interpretability, and personalization.

While the literature is filled with studies exploring DNN architectures in RS, the challenges of data sparsity and cold start, for instance, still persist in such systems. Additionally, it is widely acknowledged that there is a notable lack of experimental standardization (KOREN; RENDLE; BELL, 2022; DACREMA; CREMONESI; JANNACH, 2019) concerning RS researches which can potentially impact the comprehension of findings and breakthroughs in the field. Therefore, a comprehensive investigation of these multifaceted issues is necessary.

1.1 Hypothesis

This thesis is driven by the investigation of algorithms and machine learning approaches in the context of collaborative filtering recommendation. We propose techniques capable of addressing its arising the main challenges, namely, data sparsity, the cold start problem, and scalability. Thus, the hypotheses this thesis verifies are: *Proposing and investigating DNN architectures provides a solution to challenges in CF, particularly addressing issues related to data sparsity and the cold start problem. Data sparsity inherent from CF could be explored as leverage beyond recommendation applications, such as for the classifiers fusion through decision templates. The combination of k-approximate neighborhood techniques with OPF improves its scalability during clustering, with minimum impact on recognition rates. Introducing the shared neighbors concept as OPF adjacency relation enables it to alleviate sparsity and high dimensionality issues in CF-based recommendation.*

To address that set of hypothesis, four studies were conducted during the development of this thesis:

(i) A rigorous systematic literature review to map recent CF-based recommendation research on the application of deep learning techniques to address sparsity-related issues;

(ii) An approach that explores CF-based matrix factorization in the construction of sparse decision templates for distance-based classifiers;

(iii) The modification of OPF's adjacency relation to efficiently and scalably generate k-approximate neighborhoods during costly clustering tasks;

(iv) A new adjacency relation inspired by the shared neighbors concept, which enables OPF to cluster CF-based sparse and high dimensional data.

The findings and discussions presented in the following chapters provide support for the hypotheses of this work.

1.2 Thesis organization

This thesis is organized as a collection of papers published and submitted by the authors during the research period. Chapter 2 presents a systematic review focusing on state-of-the-art literature on deep learning techniques applied in collaborative filtering recommendations and yet featuring primary studies related to the cold start problem mitigation. Additionally, it considers the diverse non-linear modeling strategies to deal with rating data and side information, the combination of deep learning techniques with traditional CF-based linear methods, and an overview of the most employed public datasets and evaluation metrics concerning CF scenarios.

The paper presented in Chapter 3 introduces a framework that explores collaborative filtering-based latent factors models for fast decision template generation, assuming the latter has a sparse matrix structure. Chapter 4 introduces a Gaussian Mixture Model (GMM) initialization approach based on a fast an scalable OPF clustering that explores k-approximate-nearest-neighbor graphs as the basis for constructing adjacency relations. On the same line of work, Chapter 5 reports a novel OPF adjacency relation inspired by the concept of k-shared neighborhoods to enable OPF for clustering under sparsity and high dimensionality conditions such as CF recommendation. Finally, Chapter 6 provides the final considerations about this work, other contributions, and future research paths we intend to follow.

Chapter 2

Deep learning techniques for recommender systems based on collaborative filtering

This chapter introduces a work based on a survey of deep learning techniques within the context of CF, with a particular focus on those addressing the cold start problem (MARTINS; PAPA; ADELI, 2020). Additionally, the final sections are dedicated to providing an updated discussion concerning research papers published between 2021 and 2023 within the scope of this work. We have also focused on the experimental and evaluative aspects of these papers, which we consider worthy of further examination.

2.1 Introduction

Powered by the continuous and rapid advances in communication technology and data storage, Internet users end up being inevitably overloaded by Web's large volume of data, mostly due to the great diversity of information to which users are regularly exposed. Either through e-commerce, social networks (FAN; MOSTAFAVI, 2019), or streaming services, the Big Data phenomenon[1] has brought significant challenges to sustain effective and efficient information retrieval, providing interesting opportunities for the rise of new solutions for data management and information filtering (LIU; CHEUNG, 2018; RICCI; ROKACH; SHAPIRA, 2021).

[1] It is related to the adoption of a set of techniques like parallel and distributed computing, and artificial intelligence algorithms to face up the problems derived from the management and analysis of huge quantities of data (LIU; CHEUNG, 2018; TORRES et al., 2018).

Considering the current scenario, recommender systems have undoubtedly assisted users to have a more satisfying online experience, by suggesting content or services that well fit users' preferences and particular interests (ADOMAVICIUS; TUZHILIN, 2005; CAMACHO; ALVES-SOUZA, 2018). Thus, recommender systems presented themselves as appropriate techniques for handling information overload, justifying their current popularity, meaning that many online services have adopted them over the past years. As major examples, one can cite social networks like Twitter, Instagram, and Facebook; Spotify, Youtube, Netflix, and Amazon Prime as streaming services, and Amazon, Netshoes, and Google Ads representing e-commerce services.

To recommend potential content to a user, a recommender system relies on key data, for example those extracted from the user's profile and historical preferences, social network connections, and any other sort of interaction that could evidence a user's browsing pattern. In general, the way a recommendation is performed defines a basic classification of major types of recommender systems, which are traditionally categorized as content-based (CB), collaborative filtering (CF), and hybrid approaches (ADOMAVICIUS; TUZHILIN, 2005). Among them, collaborative filtering appears as one of the most popular and successful strategies for recommendation (EKSTRAND; RIEDL; KONSTAN, 2011; HERLOCKER; KONSTAN; RIEDL, 2000; RICCI; ROKACH; SHAPIRA, 2021; SU; KHOSHGOFTAAR, 2009), receiving great attention in the past years from both academy and industry.

A collaborative filtering strategy for recommendation could be defined as the use and manipulation of historical preferences of a set of users aiming to predict unknown preferences for other users (EKSTRAND; RIEDL; KONSTAN, 2011; ELAHI; RICCI; RUBENS, 2016; SU; KHOSHGOFTAAR, 2009). This definition usually is stated based on the assumption that if two users interact with the same items or they present similar interaction behavior, then their future interactions probability will be also alike.

In the aforementioned context, the data is structured as a user $\times$ items matrix (i.e., a 2D array), where the matrix rows usually represent users and its columns express the items, as show in Figure 1. When a user u interacts with an item i, a value or weight usually determines the strength of that interaction, which essentially depends on the domain application (i.e., movies, music, social media, news) and the type of recommendation to be made (i.e., prediction or ranking). The two basic forms of interaction found in the literature can be qualified as explicit (i.e., ratings, reviews or any kind of score a user has explicitly given to an item) or implicit (i.e., number of clicks, time spent watching a video, listening to music or browsing a web site, as calculated from user actions).

In general, it is unfeasible for a user to interact with all items of a database. Therefore, only a small proportion of items is rated or is interacted per user (CHEN et al., 2018b; RICCI; ROKACH; SHAPIRA, 2021). Also, as the data volume grows, CF-based approaches suffer from scalability, data sparsity, and cold start problems, which cause a

Figure 1 – An example of rating matrix.

Users \ Items	i_1	i_2	i_3	i_4	i_5	i_6	i_7	i_8
u_1	4		3	1	5			
u_2		5	3	4		2	3	
u_3		1		5	4		3	5
u_4	5			2		4		
u_5	2					3	1	
u_6		5	4		1			4

Source: Produced by the author.

direct performance decay to recommendations interests (ADOMAVICIUS; TUZHILIN, 2005; EKSTRAND; RIEDL; KONSTAN, 2011; ELAHI; RICCI; RUBENS, 2016). Data sparsity and cold start are correlated issues, since both of them arise from the lack of data. The first issue is characterized by a low number of ratings for available items, making it difficult to find a relationship between users and items (CAMACHO; ALVES-SOUZA, 2018); the latter one occurs when the lack of both user data and item rating history hinders the system in making reliable recommendations (BOBADILLA et al., 2013; CAMACHO; ALVES-SOUZA, 2018).

Since the emergence of collaborative filtering, many algorithms, methods, and recommendation approaches have been proposed. From traditional to the most recent techniques, neighborhood methods (KOREN, 2010; SARWAR et al., 2001) and latent factor models like Matrix Factorization (MF)-based algorithms (KOREN; BELL; VOLINSKY, 2009; KOREN, 2009; SALAKHUTDINOV; MNIH, 2009; ZHANG et al., 2006) are currently the most popular ones. While the former is an easy-to-implement statistical method that delivers reasonable and interpretable recommendations, the latter is a scalable approach that is effective for achieving fast and accurate prediction results even for large databases (BOBADILLA et al., 2013).

Despite their prevalence in collaborative filtering literature, the aforementioned techniques still have their limitations. Even though neighborhood methods are straightforward to implement and deliver reasonable recommendations, they suffer from scalability problems, which are naturally more critical in Big Data scenarios.

In general, latent factor models have proven to do efficient recommendations under high sparsity conditions. Essentially, they require a learning phase in advance to discover optimal model parameters in low-dimensional feature space for both users and items before making a recommendation (CHEN et al., 2018b). Still, these linear methods usually restrict the explainability of the recommended results, since their prediction ability depends on users' and items' low dimensional factors. Moreover, they struggle to learn more

complex and abstract latent representations from available data, and also are computationally expensive.

Mainly due to the aforementioned challenges, deep learning algorithms have received significant attention over the past few years (HUA et al., 2019; LIANG, 2019; YANG et al., 2019; ZHANG; CHENG; REN, 2019). Those methods rely on non-linear optimization techniques to learn model parameters, which make them substantially more robust for data representation, and their high generalization strength also facilitates dealing efficiently with raw or non-structured data. In this way, DNN are able to achieve deeper abstractions of data, effectively demonstrating considerable advances in a variety of tasks like detection (ANSARI et al., 2019; BANG et al., 2019; LI; ZHAO; ZHOU, 2019; MAEDA et al., 2018; MAEDA et al., 2019; ZHANG et al., 2019a), prediction (LUO; PAAL, 2019; NGUYEN et al., 2019), clustering (GAO; KONG; MOSALAM, 2019; REYES; VENTURA, 2019), and classification (LECUN; BENGIO; HINTON, 2015; MAEDA et al., 2018; MANZANERA et al., 2019). Last but not least, we emphasize that many other classification techniques could be considered to cope with the problem of collaborative filtering, such as the Enhanced Probabilistic Neural Networks (AHMADLOU; ADELI, 2010), Neural Dynamic Classification (RAFIEI; ADELI, 2017), and the Finite Element Machine classifier (PEREIRA et al., 2020).

Consequently, the study and development of deep learning techniques have facilitated important improvements in many computer science research areas, from which we quote object detection (ANTONIADES et al., 2018; MOLINA-CABELLO et al., 2018; VERA-OLMOS et al., 2018; WANG; BAI, 2018), speech recognition, computer vision (LUO; PAAL, 2019; SHEN et al., 2019), and natural language processing (LECUN; BENGIO; HINTON, 2015). In this sense, studies involving deep learning in recommender systems have also gained attention, which is easily seen from the increasing number of published articles over the past years (ZHANG et al., 2019).

2.1.1 Leading objectives

In this research, the authors conducted a rigorous systematic review focusing on state-of-the-art literature on deep learning techniques applied in collaborative filtering recommendation, as well as primary studies related to mitigating the cold start problem. The main objectives of this work are summarized as follows:

- ❏ To present a comprehensive overview of the most recent advances in collaborative filtering recommendation focused on deep learning-based approaches.

- ❏ To highlight studies committed to alleviating the cold start problem through deep learning.

- ❏ To inform on the most recent literature in the area as well to be a new source for researchers interested in deep learning-based approaches applied to collaborative filtering.

- ❏ To synthesize the challenges encountered by researchers and to highlight promising new topics to guide future works.

The remainder of this paper is structured as follows. Some of the recent literature reviews and surveys correlated to this work are summarized in Section 2.2. In Section 2.3, the methodology adopted to conduct the Systematic Literature Review (SLR) is properly described. Section 2.4 details the SLR results, pointing out quality standards chosen during the final selection phase and group-based summaries of the primary studies. Section 2.5 describes the primary studies chosen to be synthesized. In Section 2.6, a discussion of the main topics and findings is presented to clarify the selected primary studies. Finally, the main challenges and research opportunities related to the subject of this work are presented in Section 2.7.

2.2 Related work

The popularity of collaborative filtering-based recommender systems has resulted in a large number of published research papers. Accordingly, there are also many contributions presented in the form of surveys and systematic literature reviews that synthesize and discuss the main aspects, definitions, and recommendation approaches related to collaborative filtering.

The study conducted by Nikzad–Khasmakhi, Balafar e Feizi–Derakhshi (2019) presented an overview of expert recommender systems from the Information Retrieval perspective. The authors compiled and summarized different recommendation approaches from the literature, also pointing out their main pros and cons. Besides, the study presented some main concepts, definitions, and evaluation measures often reported in the literature of expert recommender systems.

Chen et al. (2018b) presented the main developments in collaborative filtering research focusing on social network-based hybrid techniques. Presenting both traditional and current state-of-the-art collaborative filtering methods and recommendation approaches, the authors focused on the sparsity and cold start problem resolutions using social network factors.

The systematic review conducted by Camacho e Alves-Souza (2018) specifically investigated the literature dedicated to the use of social network information to mitigate the cold start problem in CF-based recommender systems. This work highlighted the proposed recommender algorithms and techniques, similarity measures, experimental evaluation

procedures, and the different strategies adopted by the revised works to take advantage of social network factors.

An interesting systematic review about the use of machine learning in recommender systems was conducted by Portugal, Alencar e Cowan (2018). Its main purpose was to precisely identify open research questions regarding the adoption of machine learning for the recommendation task. The authors presented a taxonomy of machine learning-based recommender systems, and discussed the existing relations between big data and recommendation, also presenting an analysis of evaluation metrics.

Focusing on deep learning studies, Zhang et al. (2019) presented an overview of the scientific research that investigated the adoption of non-linear algorithms based on deep learning in recommender systems. A taxonomy and synthesis of methods and approaches directly related to the aforementioned topic were proposed and described. Conclusively, the authors point out the rapid increase of deep learning research for recommendation tasks, also highlight some main challenges and limitations, as well as promising paths for future research.

In contrast to other recent related survey articles, which focus mostly on the general overview and analysis of algorithms, techniques, and methods for recommendation scenarios, this work presents a systematic review to investigate the adoption and development of deep learning techniques for collaborative filtering systems, especially the ones centering on mitigating the cold start problem.

2.3 Systematic review procedure

In this section, our systematic literature review process is properly described, focusing on the phases, activities, and necessary resources required (KITCHENHAM, 2004; NAKAGAWA et al., 2017). Through this SLR methodology, works can be identified, evaluated, and interpreted meaningfully.

When conducting a systematic review, research questions have a central role, thus their elaboration is considered an essential task for the success of any SLR (KITCHENHAM, 2004). Besides, they cause a direct influence over the analysis and extraction data summarization activities of the primary studies (NAKAGAWA et al., 2017).

Therefore, to efficiently organize the research questions, we adopted the PICOC criteria (Population, Intervention, Comparison, Outcome, and Context) introduced by Pai et al. (2004). Table 1 presents our formulation based on the prior criteria.

From the PICOC criteria two initial questions were elaborated. They served as the basis to define the research questions, which are presented in Table 2. The searched papers were assembled based on the following initial questions:

- First question (Q_1): to apply deep learning techniques in collaborative filtering.

Table 1 – PICOC criteria chosen to narrow down the SLR scope and to define the research questions.

Criteria	
Population	Collaborative filtering recommender systems
Intervention	Deep learning techniques
Comparison	Does not apply
Outcome	Adopting deep learning causes improvement in the recommendation process, which could include mitigating the cold start problem
Context	Academy and industry

Source: Produced by the author.

- ❑ Second question (Q_2): to adopt deep learning to mitigate the cold start problem in collaborative filtering-based recommender systems.

Table 2 – Research questions elaborated based on PICOC criteria.

Research questions	
RQ$_1$	Was there a significant improvement in the recommendation process while using a deep learning technique?
RQ$_2$	Was the cold start problem mitigated while using a deep learning technique?

Source: Produced by the author.

2.3.1 Keywords, search strings and database sources

The next step was to extract keywords from the research questions to build the search strings required to quest the expected primary studies. The chosen keywords were (i) collaborative filtering, (ii) deep learning, (iii) cold start, and (iv) recommender system.

In total, two search strings (S_1 and S_2) were built, taking into account the redundancy of some terms. They were defined as follows:

- ❑ S_1: “collaborative filtering” AND (“ recommender system” OR “recommendation system”) AND “deep learning”.
- ❑ S_2: “collaborative filtering” AND (“recommender system” OR “recommendation system”) AND “deep learning” AND (“cold start” OR “cold-start”).

The search strings were applied to the following sources to collect the subject-related papers: ACM Digital Library[2], IEEE Xplore Digital Library[3] and Scopus[4]. These search sources were chosen mostly due to their credibility and adequacy to the computing research field (CAMACHO; ALVES-SOUZA, 2018).

[2] http://portal.acm.org
[3] http://ieeexplore.ieee.org
[4] http://www.scopus.com

2.3.2 Inclusion and exclusion selection criteria

To narrow the search for potential articles and to allow a precise selection of relevant candidate primary studies, specific selection criteria were elaborated (NAKAGAWA et al., 2017). The respective inclusion and exclusion criteria are depicted in Table 3, having been elaborated essentially to pre-select papers through reading only their respective title and abstract.

Table 3 – Selection criteria.

Selection criteria	**Description**
Inclusion	Adoption of a deep learning technique in collaborative filtering Adoption of a deep learning to mitigate the cold start problem in collaborative filtering Studies published between January/2017 and January/2020
Exclusion	The search source does not allow full text access The study is not written in English The article is a secondary or tertiary study The article has a Journal Impact Factor less than 2 The study is not a conference or journal article

Source: Produced by the author.

One of the strategies considered to select the most relevant primary works was to concentrate the search efforts on papers published in journals or scientific conferences whose impact factor is greater than two. The one considered was based on Journal Citation Reports (JCR)[5].

It is worth noting selecting only journal papers led the search to degraded results. Given the speed of change in the computer science field nowadays, the most recent state-of-the-art research papers are also published in conferences and proceedings. This fact serves as a plausible justification for including the aforementioned scientific communication vehicles as viable sources of primary studies.

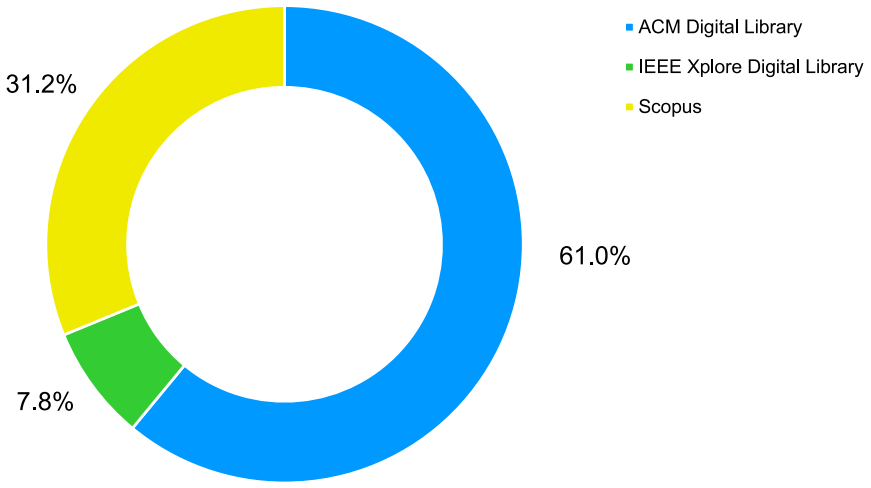
2.4 Systematic review results

In this section, we describe how the primary studies were selected and narrowed to be further synthesized and discussed in subsequent sections. Also, we describe and justify the chosen quality assessments, which served us as an essential restriction strategy to compile the final SLR results.

Throughout the procedures described in Section 2.3, there were found a total of 730 papers distributed among the search sources. After eliminating duplicated entries, the title and abstract of the remaining papers were checked, aiming to identify the ones directly related to the research questions. Thus, 130 research papers were pre-selected to be fully read. The corresponding statistics are depicted in Figures 2 and 3.

[5] http://www.webofknowledge.com/JCR

Figure 2 – Articles found per source.



Source: Produced by the author.

The final selection of articles was chosen via a quality assessment of the primary studies. In general, this strategy was adopted to investigate whether quality differences provide an explanation for variance in study results, and also as means of weighting the importance of individual studies during the ones synthesized (KITCHENHAM, 2004). For this purpose, a checklist composed of quality assessment questions (QA) was elaborated as follows:

QA1: Do the authors propose a deep learning technique to deal with CF-based recommendation?

QA2: Does adopting deep learning consistently improve recommendation in cold start scenarios?

QA3: Does adopting deep learning cause improvement in any stage of the recommendation process?

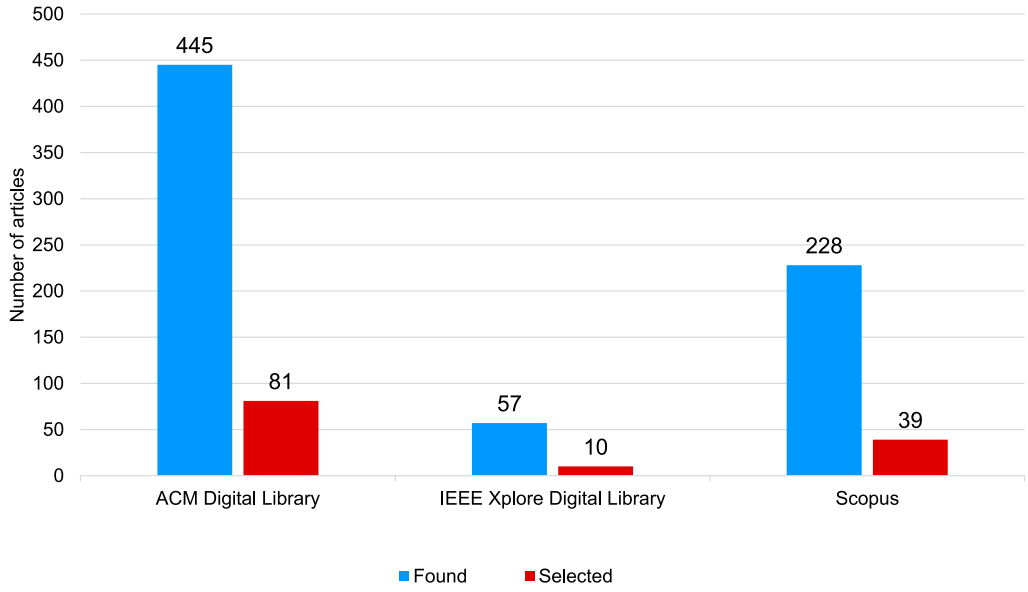
QA4: Does the study clearly describe the methodology designed to improve recommendation?

QA5: Is the proposed approach compared against the state-of-the-art literature?

QA6: Are the experiments conducted using real-world public accessable datasets?

QA7: Are the experimental method and statistical analysis considered to evaluate the research results?

Figure 3 – Found and selected articles grouped by search source.



Source: Produced by the author.

Answering the quality assessment questions enabled us to classify the primary studies according to their relevance level concerning the research questions. To weight the answers, each study received a real-value score that denotes the relevance to be complete, partial, or none. The corresponding details about the quality assessment scores are presented in Table 4.

Table 4 – Quality assessment scores.

Relevance level	**Score**	**Maximum score per article**	**Cutoff score**
Fully relevant	1.0		
Partially relevant	0.5	7.0	5.0
Not relevant	0.0		

Source: Produced by the author.

Despite all QA having the same importance weight, they perform distinct roles during the assignment-score process. QA_1 acted as complementary inclusion criteria, since usually reading and analyzing only the article's title and abstract are insufficient actions to correctly accept or refuse the corresponding study (NAKAGAWA et al., 2017). Hence, studies whose QA_1 score was weighted zero (i.e., not relevant) were automatically classified under the cutoff score.

To weight QA_2, we analyzed the experimental strategies adopted in the studies for evaluating the proposed CF-based methods in a cold start scenario. This kind of assessment is significant to precisely identify studies that investigated the cold start problem and proposed a solution to relieve its influence during the CF recommendation process.

Finally, QA_5, QA_6, and QA_7 allowed us to assess the experimental methods and strategies considered in the primary studies. The ones that conducted experiments using the most adopted public real-world datasets (also avoiding synthetic data) and evaluation metrics scored higher among others. Also, we took into account if the leading and most popular state-of-the-art CF-based algorithms were considered as comparable candidates during experimentation.

Only primary studies having scored at least 5.0 were included in the final selection. This left 50 articles. At last, a summary of the selection process is presented in Table 5.

Table 5 – Summary of the selection process.

Search sources	All articles found	Articles pre-selected (title and abstract reading)	Articles selected (full reading)	Quality assessment	
				Relevant	Synthesis
ACM Digital Library	445	357	81	51	32
IEEE Xplore Digital Library	57	46	10	5	4
Scopus	228	188	39	26	14
Total	730	593	130	82	50

Source: Produced by the author.

2.5 Synthesis of main primary studies

In this section, we describe the main aspects of the 50 most relevant primary studies. We paid special attention to the application domain where the techniques were applied, the deep learning techniques that structured the proposals, datasets, and evaluation metrics adopted during experimentation. Also, we took into account the mitigation of the cold start problem and how the different authors introduced deep learning in their studies and investigations.

Zheng et al. (2018) propose the Spectral Collaborative Filtering (SpectralCF), a deep CF method to deal with the cold start problem through a Graph Convolutional Networks (GCN) architecture (KIPF; WELLING, 2016). To do so, a spectral convolution operation is introduced for modeling in a non-linear fashion, both proximity and connectivity information extracted from a bipartite graph of users and items. Using polynomial approximations, SpectralCF tries to alleviate the high computation burden of both convolutional filters and data manipulation on the spectral domain.

Also exploring GNN structures, Wu et al. (2019a) propose a social recommendation approach called DiffNet, which focuses on recursive influence diffusion of users through a neural attention mechanism to mitigate the cold start problem. The approach contains three essential layers: embedding, fusion and influence diffusion. Users and items are first embedded separately in one-dimensional vector representations, the former based on social network and the latter on content features. The resulting representations are then integrated within a fully connected fusion layer, which feeds an influence diffusion layer

responsible for modeling the dynamics of users' latent preference diffusion in the social network (WU et al., 2019a). Also, non-linear modeling is conducted with a pair-wise loss function resembling the Bayesian Personalized Ranking (BPR) (RENDLE et al., 2012) for implicit feedback.

Wang et al. (2020) integrate a deep Factorization Machines (FM) with a marginalized Stacked Denoising Autoencoder (mSDA) to optimize the network input. Linear modeling of low-level feature interactions is performed by a FM, and high-order latent factors from users and items are learned by the mSDA. The autoencoder embeddings are jointly learned using a Multi-Layer Perceptron (MLP), outputting deep latent factors which are computed together with the linear representation generated by FM to predict ratings.

Nisha e Mohan (2019) propose the AESR, a hybrid recommendation method that models a joint optimization function extending a deep autoencoder with top-k user semantic social information. Two-levels of users' preferences represent the embedding, been composed of the user rating behavior, and the top-k friends of the active user. The semantic information is captured from a trust-relation graph derived from the user-item interactions, and the weighting is computed using cosine similarity. An autoencoder is employed for modeling the predicted ratings instead of a traditional CF method like matrix factorization. So the former low-dimensional representation capabilities could be explored in an efficient unsupervised manner. The encoder network learns user and item embeddings independently from the rating matrix, and these are jointly optimized incorporating top-k trust information into the decoder network to generate the predicted ratings.

Being one of the most popular DNN-based approaches for CF recommendation, He et al. (2017) introduce the Neural Network-based Collaborative Filtering (NCF) to overcome the known limitations of linear MF methods. Adopting an MLP network architecture with a pointwise objective function, the authors aim to obtain a generalized MF form to perform CF-based recommendations. In contrast to other techniques, NCF models user–item interactions through non-linear neural optimization instead of the simple inner product of latent factors. The approach deals with user and item embeddings in independently latent spaces to better generalize the user–item interactions, which can be amplified by adding hidden layers to the neural architecture (HE et al., 2017).

A group recommendation method using an attention mechanism and MLP architecture based on the NCF framework is proposed by Cao et al. (2018). The method is structured in a two-stage process: aggregation strategy learning and interaction learning. The former embeds different deep representations of user groups through neural attention, whereas the latter models the corresponding complex interactions between users, items, and groups using NCF shared hidden layers.

Lee et al. (2018) investigate data imputation to deal with missing ratings. Based on three different claims for data imputation, the authors propose a pre-use preference inference method for data imputation on the original rating matrix. Also, user and item

feature extraction is done by means of a Variational Autoencoder (VAE). The inference of post-use preferences (i.e., the rating prediction task) is done by jointly optimizing the VAE latent factors and the pre-use inferred preferences using an MLP network.

Liu, Wang e Ding (2017) propose a hybrid recommendation approach called PhD, which considers both user–item interactions and side information to generate ratings using a Probabilistic Matrix Factorization (PMF). A Stacked Denoising Autoencoder (SDAE) encodes the user–item interactions, and a Convolutional Neural Networks (CNN) models features extracted from user tags, demographics, movie descriptions, and reviews, just to name a few. At last, the resulting deep representations serve as the basis of a PMF for predicting ratings.

Taking advantage of textual review capabilities, Zheng, Noroozi e Yu (2017) propose the Deep Cooperative Neural Networks (Deep-CoNN) for CF-based recommendation. Essentially, it models users and items jointly through review text information for the rating prediction task. To do so, a two-coupled CNN is used for latent factor generation. One embeds textual information from user reviews and the other one does the same using written reviews for items. The corresponding latent factors are learned in parallel and a joint deep representation is obtained as an output through a shared layer between the networks. All textual reviews are first pre-processed using the word embedding technique word2vec (MIKOLOV et al., 2013) before the non-linear neural optimization procedure.

Zhang et al. (2017) propose the Joint Representation Learning (JRL) framework integrating both user-item interactions and side information using CNNs. The framework is structured under a view perspective, where each view stores different user/item representations obtained by linear combinations of entities (users, items, review text, image, etc). According to the authors, this structure was proposed to allow the adoption of different views, making it extendable to new views in practice, in the sense that the framework is sufficiently flexible to input new forms of data to be modeled (ZHANG et al., 2017). The views produce multiple representation embeddings, which are integrated with a merge function, i.e., a unified pairwise learning to rank framework.

Wu et al. (2017) consider the assumption that users' preferences change over the time and proposed to model user-item interactions using Long Short-Term Memory (LSTM) and Recurrent Neural Networks (RNN) to capture temporal and causal aspects from users and items under a non-parametric and dynamic mapping strategy. The rating prediction came from newly updated states of the RNN model and could be interpreted as naturally causal effects brought by previous states (WU et al., 2017).

Chen et al. (2020a) study efficient strategies for whole-data learning to avoid sampling techniques, which are frequently part of CF-based recommenders. For this purpose, the authors propose an NCF-based approach and three optimization algorithms for whole-data-based learning. In contrast to NCF, the proposed approach considers as the network input an all-interactions user/item vector and the model learning procedure follows a

weighted regression loss introduced by Hu, Volinsky e Koren (2008). The proposed approach was adapted to a mini-batch form in order to associate a confidence weight to each prediction made.

Focused on item recommendation, Chae, Shin e Kim (2019) explore Generative Adversarial Network (GAN) in the context of collaborative filtering. To do so, the authors propose the Collaborative Adversarial Autoencoders (CAAE), a GAN architecture where an autoencoder acts as the generator network, and the discriminator network is based on BPR. CAAE seeks to explore both the data reconstruction capabilities of adversarial networks and the generalization power of deep models. Besides, aiming to increase recommendation quality, a generator network of negative items is considered to extend the deep modeling of items a user may dislike.

Fu et al. (2019) propose a CF-based approach that explores local and global co-occurrence of user-user and item-item relations to capture context information. Throughout this strategy, target user/item and their historical related users/items are independently computed as dictionaries of embeddings, which are generated through global and local co-occurrence techniques like Constrain Model (CM) and Rating Independent Model (RIM), respectively. Finally, ratings are predicted with a deep MLP model, where the contextual embedding representations are jointly learned to reveal the variety and complex interaction forms of the rating data.

Seeking a better integration of different side information, He, Meng e Zhang (2019) propose a VAE CF-based approach for unsupervised embedding generation. Their main contribution is related to the integrated modeling of document content and tag information during inference and generation processes of the VAE model, which allows jointly deep modeling of latent distributions to uncover complex data aspects. Finally, the VAE's latent factors are used to generate a list of recommendations through PMF.

Also taking advantage of the unsupervised capabilities of autoencoders for deep latent representation, Wang et al. (2019b) propose the Deep Collaborative Conjunctive Recommender for Rating Prediction (DCCR). This approach models low-level and high-level user/item latent factors derived from rating data integrating two neural network architectures. The former is embedded by a regularized Denoising Autoencoder (DAE) and the latter unifies user and item representations by learning their respective interactions through an MLP-shared layer, which is optimized considering an L2 regularization term. Also, the authors designed a variant of the Rectified Linear Unit (ReLU) activation function which tries to guarantee the output values are limited to a certain interval, which indicates the minimum and maximum of the input rating data should be reflected in the activation function (WANG et al., 2019b).

Chen et al. (2019a) adopt a DNN joint-learn strategy to model user-item interactions. The proposed approach has two deep MLP networks, the first one designed to model the rating data and the second one to learn the corresponding user/item interactions. The

rating data modeling considers users and item embeddings to be generated in parallel, consequently favoring a more efficient model computation. Also, the authors introduced a hybrid loss function allowing the model to learn latent factors based on pointwise and pairwise losses in a joint fashion, making the proposed approach suitable for both rating prediction and recommendation raking tasks.

The study conducted by Wu et al. (2019b) investigate the over-time user preferences changing problem in the context of collaborative filtering and its effects on latent factor generation. To do so, the authors propose a neural Tensor Factorization (TF) model for dynamic relation prediction that accounts for LSTM layers to capture temporal aspects of the data. Also, the final predictions are generated using an MLP network to avoid the simple inner product operation commonly used by linear MF-based methods for predicting ratings. The latent factors are encoded based on a 3-D tensor input and represent the embeddings of user, item, and time, which are modeled in a similar way to the NCF approach.

Yi et al. (2019) propose the Deep Matrix Factorization (DMF), a CF-based approach that aims to allow the merging of different side information from both users and items. The authors designed two feature transformation functions bound to generate user and item latent factors instead of estimating them directly from the observed data. Positive and negative interactions are extracted from an implicit feedback information graph and are mapped to a low-dimensional real-valued vector retaining key patterns from the data (YI et al., 2019). The aforementioned procedure is performed by Implicit Feedback Embedding (IFE), a feature extraction technique also proposed by the authors. The resulting embeddings are unified with side information in a one-dimensional vector, serving as data input to be modeled in a deep MLP network and to generate the final predictions.

Focusing on research paper recommendation, Zhou et al. (2019) propose a neural network CF-based approach that extracts latent factors from both explicit and implicit feedback data, and also captures textual representations from items using an RNN model. To unify different data representations, an embedding function using latent factor weighting is considered to fulfill this task. To compute recommendations for rating prediction or ranking list, the inner product of users (items) latent factors (rating data) and auxiliary text embeddings (item side information) is calculated.

Zhang et al. (2019b) investigates the adoption of attention mechanisms in CNN architectures to improve feature extraction effectiveness without augmenting the number of layers. The attention mechanisms focus on the low-dimension model space for dense data (i.e., document text information) and the high dimension model space for sparse data (i.e., user rating preferences). Finally, the resulting user and item embeddings are concatenated in one-hot vector representation, feeding a Generalized Matrix Factorization (GMF) model (HE et al., 2017) to produce the predicted ratings.

Feng e Zeng (2019b) interpret the review-based CF as a sequence matching problem as

in the NLP field. User and item sequences are matched through a fine-grained interaction leading to a better result (FENG; ZENG, 2019b). To do so, the authors propose an approach based on the Bidirectional Encoder Representations from Transformers (BERT) technique (DEVLIN et al., 2018) to encode user/item text reviews and a multi-contextual layer attention mechanism to capture complex dynamic relations in multi-level, aspect-level and review-level from the input data. The resulting embeddings serve as input to an Attention Factorization Machine (AFM), which learns user-item interactions from the observed rating data and completing the final prediction.

Wang et al. (2019) propose a fusion-gated mechanism in a hybrid CF framework for session-based recommendation called the Collaborative Session-based Recommendation Machine (CSRM). The authors' main hypothesis stands on the assumption that collaborative information contained in neighborhood sessions may help to improve recommendation performance for the current session (WANG et al., 2019). To do so, an RNN architecture with Gated Recurrent Units (GRU) is considered to capture both active session behaviors and neighborhood session information, representing two different modules, namely Inner Memory Encoder (IME) and Outer Memory Encoder (OME). Finally, the latent factors are unified in a decoding layer, which predicts the possible next item to be clicked by the user in the current session.

Da Costa e Dolog (2019) propose a technique that generates temporal embeddings of users and items while integrating them with user-item interactions during a CNN training phase. The main idea is to better capture contextual information related to overtime user preferences. Using interaction and convolutional layers, user, item, and time embeddings are jointly learned to amplify the quality of items' list predictions (ranking recommendation task).

The approach proposed by Song, Chang e Hu (2019) is based on variational inference probabilistic models. Supposing all user-item interactions could be decomposed into stationary and dynamic terms, the authors intended to model dynamic embeddings through a VAE network, capturing the complex non-linear user preferences patterns and item popularity overtime. In this sense, the inference network follows an RNN-based architecture employing GRU layers to infer users/items latent factors' posterior distributions for each interaction time interval. For rating prediction purposes, user and item temporal embeddings are later unified with stationary factors through an expectation process.

Liu et al. (2019) investigate local and mutual attention of CNN to jointly learn latent factors from rating data and review texts to increase the recommendation results' interpretability. The proposed approach (DAML - Dual Attention Mutual Learning) considers a local attention strategy to model review information and a mutual one to capture user-item relevance pairs. Further, neural factorization machines are used to perform high-order nonlinear interaction of latent features (LIU et al., 2019) and to generate predicted ratings.

The study conducted by Xue et al. (2019) focuses on effective ways to capture high-order item relations in item-based collaborative filtering (ICF) models. With this intent, the authors propose an NCF-based approach specially designed to model high-order item relations. A multi-hot encoding is considered to represent each item in the input neural network layer, and an attention layer focuses on extracting the most representative user/item interactions, which are then projected using a pooling operation. At last, an MLP non-linear optimization models the latent attention vectors generating a list of predictions for top-N recommendation purposes.

The method proposed by Fan et al. (2019) explores neighborhood information from users in the context of social recommendation. The authors assume that it is not necessarily the user's direct neighbors which are capable of offering the most representative influence to produce recommendations. To do so, a proposed multi-layer structure using LSTM integrates users' random walks paths, latent factors from users, items and ratings intending to better capture from input data the neighborhood's structure, interaction, and sequential information.

Hu et al. (2019) propose the Semi-Parametric Embedding (SPE), a hybrid item-item CF approach which incorporates content-based information to overcome the cold start problem and models the latent factor simultaneously using an SDAE. An item embedding is composed of two parts, i.e., the parametric part from content information and the non-parametric part designed to encode behavior information (HU et al., 2019). In this sense, through a simultaneous learning process, the approach can alleviate the lack of data concerning cold start items, thus improving recommendations.

Otunba, Rufai e Lin (2019) propose an ensemble framework for CF-based recommendation. The method integrates GMF and MLP networks to take advantage of the generalization power of deep neural architectures in representing user-item interactions and the efficiency of shallow networks for recommendation ranking tasks. The authors adopt an unweighted stacking strategy and a Hadamard product layer to design the ensemble. Since the prediction accuracy is maximized when there is a balance between the classifiers concerning the number of classes, the aforementioned strategy is valid in 2-class problems like CF scenarios considering implicit feedback only.

Using Conditional GAN as the deep model basis, Wang et al. (2019a) explore the generation of augmented interaction data from side information in a CF setup. The authors propose a discriminator loss function and a Gumbel-Softmax approximation to overcome the gradient block problem, aiming to validate the assumption that learning from interaction data produced by generative methods reduces the data sparsity, and consequently allowing pure CF-based methods to augment the quality of recommendations.

The approach proposed by Wang e Caverlee (2019) integrates an RNN short-term sequence and a GRU for local coherence and dynamic latent factor generation. The authors intend to capture user-item interaction sequences from neighboring information

of implicit feedback data. This procedure generates latent factors capable of representing structural information from users and items' neighborhood, and also to encode the diverse interaction sequences from rating data. Subsequently, a GRU-based RNN jointly learns the latent factors using an element-wise loss function to produce predicted ratings.

In a session-based recommendation scenario, the study conducted by Sun et al. (2019) aims to identify which user's sessions are useful to model the current session favoring the prediction of next-item recommendation. Thus, an RNN architecture structured with LSTM and GRU integrates both long-term and short-term user preferences to facilitate the joint modeling of users' current session and historical sessions' sequential information.

Wang et al. (2019) explore the capabilities of ConvMF (KIM et al., 2016) and propose the Social Review-enhanced Convolutional Matrix Factorization (SRCMF), a CNN-based approach which integrates context-aware information from user's and item's reviews and social influence to perform recommendation for rating prediction. The main aspect of SRCMF is the deep modeling of user and item reviews through a CNN model, which represents the encoding of contextual information. Essentially, a shared layer integrates both CNN and MF allowing social network information and reviews to be jointly learned and to bridge the user's social interests and user's general preferences in the same latent space (WANG et al., 2019).

Song et al. (2018) investigate the so-called negative sampling problem under the assumption that observed items are preferred to those the users not observed. In addition, the authors propose a classification strategy based on the NCF framework and pairwise loss for item ranking. The network's input data is an interaction triple formed by the user, an unobserved item, and an observed item. During training, the model sets higher weights for observed items and lower weights for unobserved ones to favor rated items under the aforementioned assumption. Like NCF, the proposed approach learns complex interactions from data using shared layers to predict whether an item is observed or not.

Chen et al. (2019b) studied a recommendation problem reformulation, considering external information modeled by a deep architecture, also retaining the model efficiency at test time. In this sense, the authors propose a generalized distillation framework that integrates, employing an adaptive layer, a "teacher module" to generate CNN-based review embeddings, and a "student module" focused on a user-item interaction latent encoding. Following an adversarial learning paradigm, the teacher module transports its distillation learning to the student module, which incorporates those in an FM to efficiently predict new ratings. It is worth mentioning the aforementioned approach is quite similar to the review recommendation state-of-the-art TransNet when it comes to considering a proper network for modeling and encoding review information.

Chin et al. (2018) explore the adoption of a neural attention mechanism for an aspect-based recommendation. The authors propose the Aspect-Based Neural Recommender (ANR), a deep neural architecture that extracts essential information from text reviews

and also applies a co-attention mechanism to jointly encode user-item interactions to capture aspect-level information from data. ANR computes its predictions based on aspect importance levels separately for users and items. This is a different strategy comparing to the interaction-shared layer commonly used by a considerable number of DNN-based recommenders.

Jhamb, Ebesu e Fang (2018) propose a DAE architecture to encode contextual information via an attention mechanism. The Attentive Contextual Denoising Autoencoder (ACDA) adopts context weights for users and items for further incorporating them to user rating history to learn a unified latent representation suitable for recommendation ranking tasks.

The study conducted by Sun, Wu e Wang (2018) focuses on dynamic latent modeling of social influence and how it could be integrated with static features like user historical preferences. To do so, the authors propose the Attentive Recurrent Social Recommendation (ARSR), an RNN architecture that captures social influence employing LSTM and integrates the resulting embedding with static latent representations from user preferences.

Chen et al. (2018a) explore Heterogeneous Information Network (HIN) capabilities to represent heterogeneous data types, which is accomplished by the so-called meta paths describing the complex semantics among HIN entities. Thus, the proposed approach is structured in the form of an MLP network, including attentive layers to map low-level features from users, items, and their respective interactions. This network architecture summarizes the meta path feature extraction, which allows relevant semantic information to be automatically selected, thus alleviating the data sparsity problem during an FM recommendation step.

The study conducted by Huang et al. (2018) focuses on sequential recommendation using knowledge base (KB) information. To do so, the proposed model integrates an RNN-based network with Key-Value Memory Network (KV-MN). The former considers GRU to capture sequential information, while the latter is responsible for learning attribute-level user preferences. It is worth mentioning KB information is incorporated into the model to enhance semantic representation, therefore providing higher interpretability power for the prediction results.

Tay, Anh Tuan e Hui (2018) focus on the study of metric learning applied to user-item interactions in the CF context. The authors propose a method for learning adaptive translations for each user-item interaction, and its respective latent factors are generated using the word2vec (MIKOLOV et al., 2013). The network model is build based on neural attention and augmented memory modules, and it also considers a translational principle minimization process integrating hinge loss and negative sampling. With these strategies, the model is capable of scaling to large datasets while sustaining a competitive ranking performance.

The study conducted by Li et al. (2018) aims at the next-item recommendation problem. The authors propose a neural item embedding algorithm called w-item2vec, which models item similarities based on a huge number of items' sequential behaviors, aiming to get a unified item representation of them. Essentially, the proposal extends the item2vec algorithm (BARKAN; KOENIGSTEIN, 2016) taking into account the item frequency as a factor to weigh user interactions, which is important in encoding the sequential relationships of items. Thus, the resulting sequential behavior embeddings act as inputs for two LSTM-based RNN, which are responsible to jointly model session information and historical user preferences. Also, the optimization process considers a Mean Squared Error (MSE) loss to compute the predicted ratings and to generate the list of recommendations.

Hyun et al. (2018) present the SentiRec, a review-aware recommendation approach that incorporates sentiments of reviews during user and item modeling. The authors' goal is to leverage ratings that accompany the reviews (HYUN et al., 2018). The review rating acts as a label, and the text review itself serves as data input to a CNN, which learns the non-linear patterns in a supervised fashion. The last hidden layer outputs a fixed-size review vector that encodes the user's sentiment over items, which is finally used as input in a DeepCoNN-based framework to generate predictions. The strategy of not directly using raw words as input, but a low-dimensional sentiment vector for users and items instead, favors efficiency in terms of training time and memory usage.

(LIANG et al., 2018b) propose a CF-based technique to model rating data using a variational autoencoder. The authors introduce a generative model based on conditional multinomial likelihood and Bayesian inference to estimate the parameters of implicit feedback, leveraging the known limitations of linear models for latent factor generation. Also, the authors investigate the influence of regularized weights in the Kullback–Leibler (KL) divergence term during VAE's optimization process and its influence over the quality of top-N recommendations.

Bai et al. (2017) explore the extraction of local information from user/item neighborhood and its contributions, in terms of representation power in a deep learning architecture for the top-N recommendations. As such, the authors propose to encode user/item neighborhood information using convolutional filters and to integrate its resulting latent factors with user-item interactions during the joint learning procedure of the NCF algorithm.

Manotumruksa, Macdonald e Ounis (2017) studied neural sequential models for venue recommendation. To do so, the authors propose a recommendation approach based on the NCF framework that extends the latter adopting an RNN layer to model sequential information from historical check-in localization using a negative sampling strategy. The resulting latent factors are then integrated with static rating data to be jointly learned through a pairwise loss, consequently uncovering complex interactions from users, items, and check-in information.

In the study conducted by Lee, Song e Moon (2017), different VAE architectures to deal with side information in the context of CF are explored. As such, conditional and joint distributions, adversarial learning, and latter structures are introduced, as part of VAE architectures. Also, the authors propose a novel negative sampling strategy based on the Bernoulli distribution that acts in the generative network specifically designed to deal with sparse data and implicit feedback.

Bellini et al. (2017) conducted a study that explores knowledge graphs to extract semantic information using autoencoder networks to mitigate data sparsity and cold start users. Since autoencoders are optimal in data reconstruction tasks, the authors aimed to reconstruct the rating data (usually in the form of user $\times$ item matrix) based on user $\times$ latent representation setup. Then, the list of recommendations is obtained through a memory-based CF strategy, in which the cosine similarity of the user weights are generated from the resulting semantic latent representations.

Ruocco, Skrede e Langseth (2017) focus on session-based recommendation and propose to independently model intra-session and inter-session user behaviors employing two RNN. The recent user behavior is captured through inter-session learning, in which the sequential modeled output becomes the initial hidden state during RNN learning employing GRU.

2.6 Discussion

In this section, we present a summary of the primary works synthesized, later discussing its results. Table 6 presents a classification of the different deep learning techniques used or studied in the reviewed articles. As one can observe, multilayer perceptron, autoencoder, and sequential models are the most explored deep neural architectures in the context of collaborative filtering recommendation. The preference for MLP-based networks presumably lies in their capability to model rating data, proving latent factors that better represent the complex user-item interactions. Further, as demonstrated by He et al. (2017), a multi-layer perceptron is well suited for rating prediction tasks, since its architecture generalizes the traditional matrix factorization linear methods.

The popularity of deep autoencoder networks is due to their representation learning capability, from which we highlight their power to reconstruct data from low-dimensional latent features. That makes autoencoders well suited for CF-based recommendation problems since there is a need to deal with high sparse data, which includes cold-start users (items). It is noteworthy that, from articles using some autoencoder-based network, almost half of them explored the DAE architecture (HU et al., 2019; JHAMB; EBESU; FANG, 2018; LIU; WANG; DING, 2017; WANG et al., 2019b; WANG et al., 2020).

Another finding is that about 50% of the primary studies adopted a neural sequential model to deal with CF recommendation. In this SLR, sequential models are essentially represented by recurrent neural networks and attention mechanism methods. While the

Table 6 – Studies classified by deep learning techniques.

Category	Number of studies	Primary studies
Autoencoder	13	Wang et al. (2020), Nisha e Mohan (2019), Lee et al. (2018), Liu, Wang e Ding (2017), Chae, Shin e Kim (2019), He, Meng e Zhang (2019), Wang et al. (2019b), Song, Chang e Hu (2019), Hu et al. (2019), Jhamb, Ebesu e Fang (2018), Liang et al. (2018b), Lee, Song e Moon (2017), Bellini et al. (2017).
CNN	9	Liu, Wang e Ding (2017), Zheng, Noroozi e Yu (2017), Zhang et al. (2017), Zhang et al. (2019b), Da Costa e Dolog (2019), Liu et al. (2019), Wang et al. (2019), Chen et al. (2019b), Hyun et al. (2018).
GAN	3	Chae, Shin e Kim (2019), Wang et al. (2019a), Lee, Song e Moon (2017).
GNN	2	Zheng et al. (2018), Wu et al. (2019a).
MLP	20	Wang et al. (2020), Cao et al. (2018), Lee et al. (2018), Chen et al. (2020a), Fu et al. (2019), Wang et al. (2019b), Chen et al. (2019a), Wu et al. (2019b), Yi et al. (2019), Zhou et al. (2019), Zhang et al. (2019b), Liu et al. (2019), Xue et al. (2019), Fan et al. (2019), Otunba, Rufai e Lin (2019), Song et al. (2018), Chen et al. (2018a), He et al. (2017), Bai et al. (2017), Manotumruksa, Macdonald e Ounis (2017).
Neural attention	13	Cao et al. (2018), Zhang et al. (2019b), Feng e Zeng (2019a), Wang et al. (2019), Liu et al. (2019), Xue et al. (2019), Fan et al. (2019), Chin et al. (2018), Jhamb, Ebesu e Fang (2018), Sun, Wu e Wang (2018), Chen et al. (2018a), Huang et al. (2018), Tay, Anh Tuan e Hui (2018).
RNN	12	Wu et al. (2017), Wu et al. (2019b), Zhou et al. (2019), Wang et al. (2019), Song, Chang e Hu (2019), Wang e Caverlee (2019), Sun et al. (2019), Sun, Wu e Wang (2018), Huang et al. (2018), Li et al. (2018), Manotumruksa, Macdonald e Ounis (2017), Ruocco, Skrede e Langseth (2017).

Source: Produced by the author.

former is optimal to learn different states from the data, the latter processes distinct parts from the input data at each output generation step. In CF scenarios, user preferences may change over time, i.e., they are not necessarily static since user needs vary. Thus, exploring a sequential architecture comes as a natural and reasonable choice to learn data dynamics, especially when data representations tend to be sparse.

Table 7 groups primary studies as to whether cold start problem mitigation is explicitly investigated or not. Here, we considered as explicitly investigated only the studies that included an experimental protocol precisely designed to analyze a CF-based approach concerning its robustness to the cold start problem and its mitigation through deep modeling. Thereby, nearly a quarter of the primary studies satisfied the aforementioned experimental requirements, the majority at which is related to the adoption of side information as an effective strategy to reduce cold start effects in CF recommendations, which is a well-established finding in the recommender systems literature (CAMACHO; ALVES-SOUZA, 2018; CHEN et al., 2018b; SHI; LARSON; HANJALIC, 2014).

In spite of the predominance of Autoencoder (AE), MLP, and Sequential models among primary studies considered in this paper, only two articles employed a GNN to alleviate the cold start problem. One (ZHENG et al., 2018) modeled user-item interaction data directly in the spectral domain, while the other (WU et al., 2019a) explored the non-linear neural optimization to learn side information embeddings. In both cases, the proposed approaches were effectively able to mitigate cold-start users/items, which could indicate a promising exploration field for future research.

Table 7 – Focus on cold start problem.

Focus	Number of studies	Primary studies
Cold start	13	Ruocco, Skrede e Langseth (2017), Nisha e Mohan (2019), Hu et al. (2019), Wang et al. (2020), Lee et al. (2018), Cao et al. (2018), Wu et al. (2019a), Liu et al. (2019), Wang e Caverlee (2019), Li et al. (2018), Manotumruksa, Macdonald e Ounis (2017), Bellini et al. (2017), Zheng et al. (2018)
Non-cold start	37	Liu, Wang e Ding (2017), Zheng, Noroozi e Yu (2017), Zhang et al. (2017), Wu et al. (2017), Chen et al. (2020a), Chae, Shin e Kim (2019), Fu et al. (2019), He, Meng e Zhang (2019), Wang et al. (2019b), Chen et al. (2019a), Wu et al. (2019b), Yi et al. (2019), Zhou et al. (2019), Zhang et al. (2019b), Feng e Zeng (2019a), Wang et al. (2019), Da Costa e Dolog (2019), Song, Chang e Hu (2019), Xue et al. (2019), Fan et al. (2019), Otunba, Rufai e Lin (2019), Wang et al. (2019a), Sun et al. (2019), Wang et al. (2019), Song et al. (2018), Chen et al. (2019b), Chin et al. (2018), Jhamb, Ebesu e Fang (2018), Sun, Wu e Wang (2018), Chen et al. (2018a), Huang et al. (2018), Tay, Anh Tuan e Hui (2018), Hyun et al. (2018), Liang et al. (2018b), He et al. (2017), Bai et al. (2017), Lee, Song e Moon (2017)

Source: Produced by the author.

Table 8 shows how the primary studies adopted a deep architecture to model data, i.e., learning only from rating data, side information, or considering both strategies. Unsurprisingly, more than half of the primary studies explored a deep latent factor learning from side information, which indicates hybrid CF methods could be reliable strategies to augment the quality of recommendations, especially when the available rating data is scarce.

Table 8 – Studies classified by the type of CF deep modeling.

CF category	Type of deep modeling	Number of studies	Primary studies
Pure CF (model-based)	User-item interaction	23	Zheng et al. (2018), Wang et al. (2020), Cao et al. (2018), Lee et al. (2018), Wu et al. (2017), Chen et al. (2020a), Chae, Shin e Kim (2019), Fu et al. (2019), Wang et al. (2019b), Chen et al. (2019a), Wu et al. (2019b), Wang et al. (2019), Xue et al. (2019), Otunba, Rufai e Lin (2019), Wang e Caverlee (2019), Sun et al. (2019), Song et al. (2018), Chen et al. (2018a), Tay, Anh Tuan e Hui (2018), Liang et al. (2018b), He et al. (2017), Bai et al. (2017), Ruocco, Skrede e Langseth (2017)
Hybrid[6]	Side information	14	Liu, Wang e Ding (2017), Zheng, Noroozi e Yu (2017), He, Meng e Zhang (2019), Hu et al. (2019), Feng e Zeng (2019a), Wu et al. (2019a), Liu et al. (2019), Wang et al. (2019a), Chen et al. (2019b), Jhamb, Ebesu e Fang (2018), Sun, Wu e Wang (2018), Hyun et al. (2018), Lee, Song e Moon (2017), Bellini et al. (2017)
	Both	13	Nisha e Mohan (2019), Zhang et al. (2017), Yi et al. (2019), Zhou et al. (2019), Zhang et al. (2019b), Da Costa e Dolog (2019), Song, Chang e Hu (2019), Fan et al. (2019), Wang et al. (2019), Chin et al. (2018) Huang et al. (2018), Li et al. (2018), Manotumruksa, Macdonald e Ounis (2017)

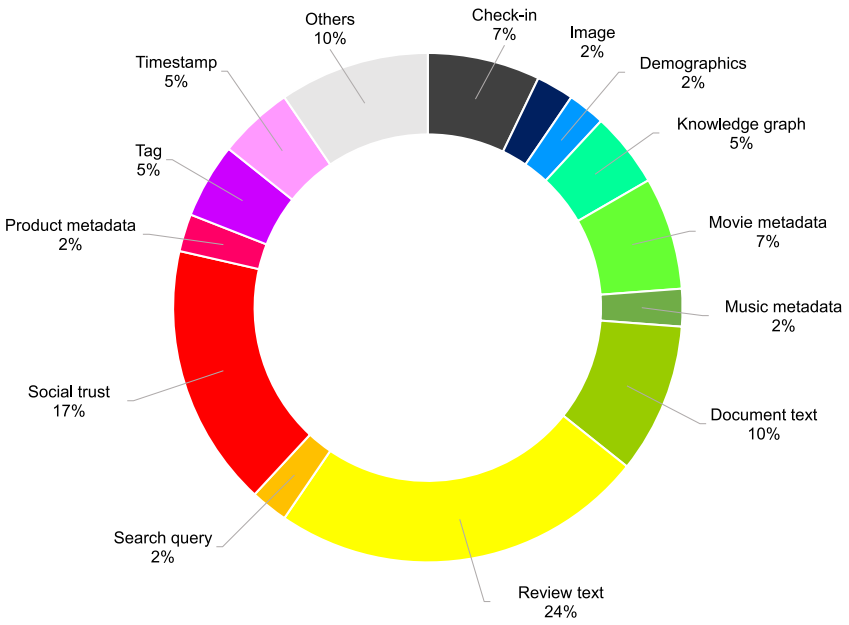
Source: Produced by the author.

Many studies use rating data as a single source for latent factor generation. Since pre-

cise user and item embedding learning is the key for building a successful recommender system (WU et al., 2019a), this may be due to a need in reaching in-deep representations for user-item interactions through deep architectures and to surpass the known limitations of traditional linear CF-based techniques like Matrix Factorization and its derivatives (CHEN et al., 2018b).

Figure 4 shows the various side information considered by the primary studies, showing textual reviews and social trust to be most preferred. Users' and items' reviews are considered a valuable source of information in the sense of augmenting the quality of recommendations (CAMACHO; ALVES-SOUZA, 2018; SHI; LARSON; HANJALIC, 2014). Also, taking into account recent breakthroughs in Natural Language Processing field, the extraction of more representative latent factors from users and items reviews, using NLP techniques, reasonably justifies researchers' high interest in textual deep modeling.

Figure 4 – Different types of side information.



Source: Produced by the author.

The high percentage of studies exploring social network information to improve recommendation quality essentially indicates researchers' interest not only in enhancing recommendation quality, but also to understand how a user's friends in a social network influence her/his behaviors and interests, and which relation degree exists between the different social influence factors and the generation of CF-based recommendations.

In Table 9, primary studies are organized by whether they make use of deep learning or not for collaborative filtering recommendation. In this sense, one-third of studies consider using a state-of-the-art non-deep learning CF technique specifically as a prediction module in their respective proposed recommendation architectures. Among CF-based techniques, Matrix Factorization and Factorization Machines were the most preferred. This may be

due to their high prediction capacity and flexibility to be extended or integrated into other methods. Besides, the main drawback of the aforementioned techniques is that they follow a linear optimization strategy to model the input data and generate recommendations.

Table 9 – Studies which integrate DNN models with another CF-based technique.

Integration	Number of studies	CF technique	Primary studies
DNN only	34	-	Zheng et al. (2018), Nisha e Mohan (2019), Cao et al. (2018), Lee et al. (2018), Zhang et al. (2017), Wu et al. (2017), Chen et al. (2020a), Fu et al. (2019), Wang et al. (2019b), Hu et al. (2019), Chen et al. (2019a), Zhang et al. (2019b), Wang et al. (2019), Wu et al. (2019a), Xue et al. (2019), Fan et al. (2019), Otunba, Rufai e Lin (2019), Wang et al. (2019a), Wang e Caverlee (2019), Sun et al. (2019), Song et al. (2018), Chin et al. (2018), Jhamb, Ebesu e Fang (2018), Sun, Wu e Wang (2018), Huang et al. (2018), Tay, Anh Tuan e Hui (2018), Li et al. (2018), Hyun et al. (2018), Liang et al. (2018b), He et al. (2017), Manotumruksa, Macdonald e Ounis (2017), Lee, Song e Moon (2017), Bellini et al. (2017), Ruocco, Skrede e Langseth (2017)
Non-DNN	16	MF-based	Liu, Wang e Ding (2017), He, Meng e Zhang (2019), Yi et al. (2019), Zhou et al. (2019), Da Costa e Dolog (2019), Song, Chang e Hu (2019), Wang et al. (2019)
		FM-based	Wang et al. (2020), Zheng, Noroozi e Yu (2017), Wu et al. (2019b), Feng e Zeng (2019a), Liu et al. (2019), Chen et al. (2019b), Chen et al. (2018a)
		BPR-based	Chae, Shin e Kim (2019)
		Neighborhood-based	Bai et al. (2017)

Source: Produced by the author.

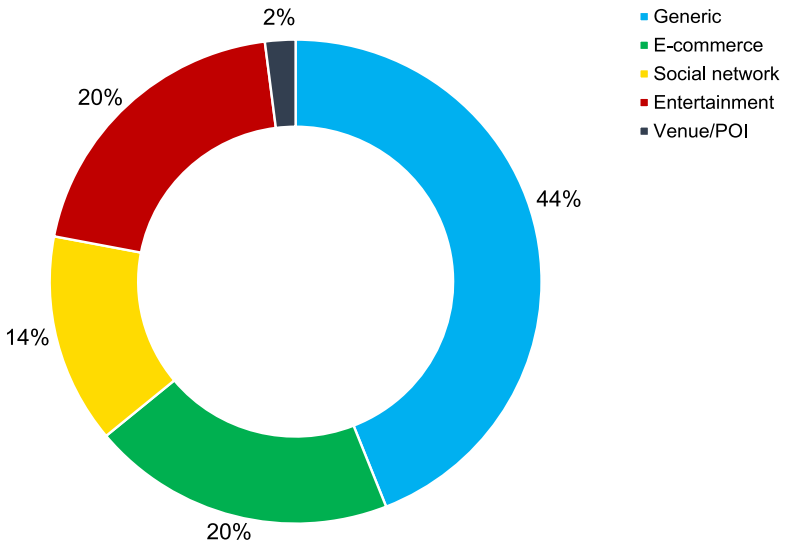
In these studies, the adoption of a linear method to perform CF prediction tasks may be related to the necessity of adding some interpretability to the recommendation results, since neural networks frequently act like a black box producing results difficult to be understood, thus to be analyzed.

Figure 5 shows the application domains where the approaches were proposed to perform recommendations. We considered as domain-specific only the ones which have their recommendation structure directly dependent on the information extracted from users or items. Thus, the approaches that did not have those content dependency levels were treated as generic, i.e., domain-independent. In this sense, more than 55% of the studies presented domain-specific CF-based approaches. The most popular ones are related to e-commerce, entertainment[7] and social networks.

As expected, the majority of CF-based approaches in some way use or adapt a deep learning technique to model the latent factor from domain-specific side information. That indicates the preference for deep hybrid architectures to learn features from users or items. In other words, the authors chose to extract latent representations not only from sparse data (ratings from users and items, for example) but also from dense ones.

[7] It includes video and audio recommendations in stream services, where movies, TV-series, and music are the most prevalent entertainment content.

Figure 5 – Distribution of the application domains.



Source: Produced by the author.

Figure 6 shows the percentage of each dataset used in the primary studies. The most frequently ones are "MovieLens", "Amazon Datasets" and "Yelp", respectively. The main indications of their popularity are related to (i) the necessity for CF-based consistent and reliable ground truth in rating dataset; and (ii) the availability of users and items textual reviews.

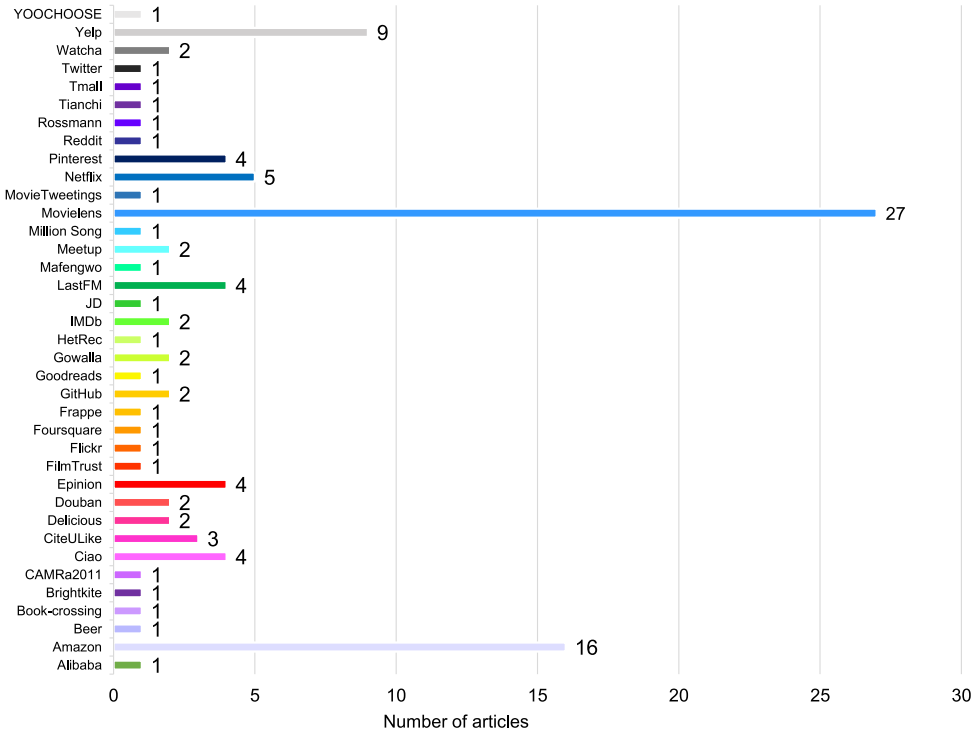
"MovieLens" could be associated to the first indication, since the aforementioned dataset is one of the first successful initiatives to provide publicly rating data, allowing novel approaches to be compared against the known performance of existing systems in a consistent environment (EKSTRAND; RIEDL; KONSTAN, 2011). As a result, "MovieLens" is a well-established dataset among researchers.

The second indication is associated with "Amazon Datasets" and "Yelp". These datasets provide not only rating data, but also content information in the form of textual reviews. This occurrence directly reflects the growing interest in review data modeling using deep architectures, as we already discussed previously in this section.

Figure 7 shows the evaluation metrics used in the primary studies. For the sake of clarity, we organized the number of articles per metric, considering the two most common recommendation tasks adopted: rating prediction (Figure 7(a)) and ranking (Figure 7(b)).

In Figure 7(a), Mean Absolute Error (MAE) and RMSE are the most popular evaluation metrics. That is not a surprise, given that these are still the most used metrics in the literature for evaluating rating prediction. In Figure 7(b), Normalized Discount Cumulative Gain (NDCG), Hit Ratio (HR) and Recall are shown to be the most used ranking evaluation metrics. We have noticed the aforementioned ranking metrics are used at least in pairs, where NDCG-HR and NDCG-Recall are the most frequent combinations. In

Figure 6 – Datasets used by the studies.



Source: Produced by the author.

general, while HR and Recall assess the recommender's power to retrieve for the current user as many relevant items as possible, NDCG weights higher better-rated items, i.e., the ones at the top of the predicted list.

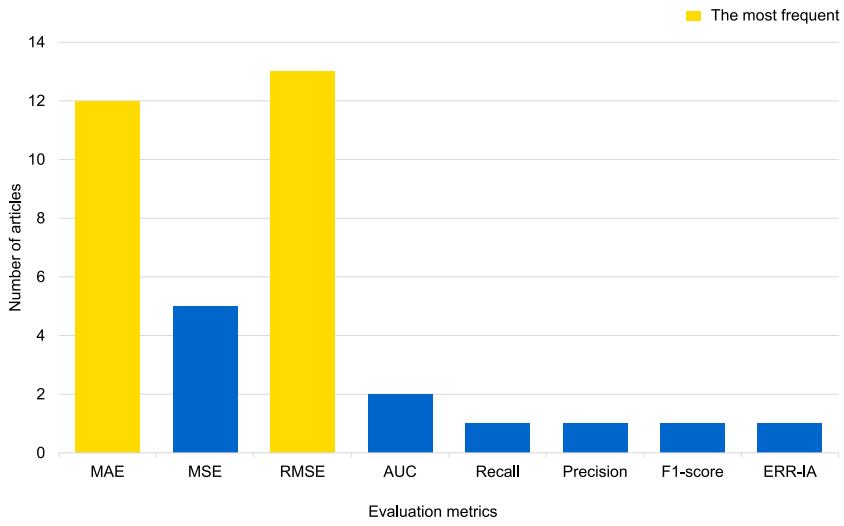
To summarize, HR and Recall prioritize the number of relevant items, and NDCG focuses on the list's ranking quality. As such, they may be seen as complementary to each other, capturing different aspects of the predicted ranking lists, which is a good strategy to enhance experimental results evaluation in terms of objectivity and effectiveness.

2.7 Conclusions

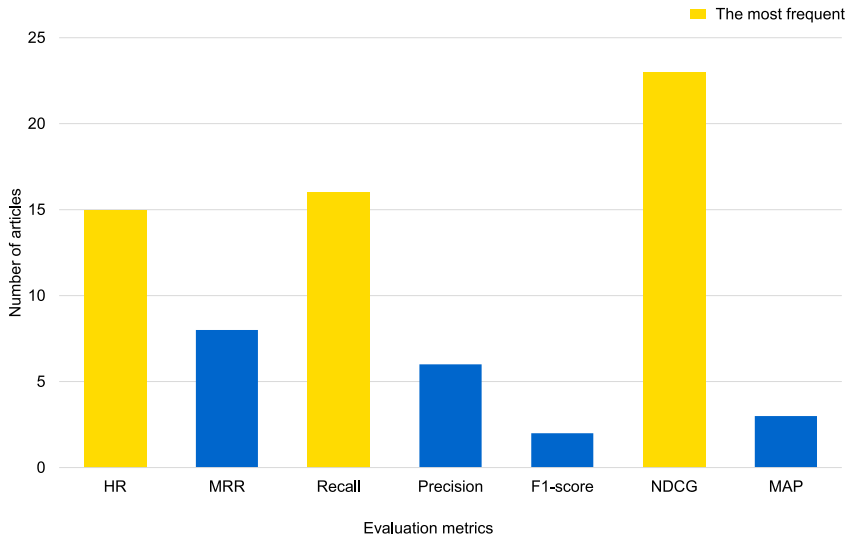
In this work, we have identified and selected research articles associated with collaborative filtering recommendation exploring deep learning techniques which also could be used to mitigate the cold start problem in those scenarios. To do so, we followed rigorous procedures in a Systematic Literature Review fashion aiming to effectively compile, analyze, and synthesize the most relevant primary studies according to the objectives of this work.

From the primary studies synthesis we were able to identify the most explored Deep Neural Network-based techniques in recent years considering CF recommendation scenarios, from which we highlight promising methods and approaches to handle data scarceness.

Figure 7 – Evaluation metrics used in the primary studies.



((a)) Rating prediction



((b)) Ranking

Source: Produced by the author.

We discuss diverse strategies for modeling sparse or dense data through deep neural architectures, the adoption of experimental protocol focused on cold start scenarios, the most explored side information sources, the integration of traditional state-of-the-art CF-based techniques with DNN-based approaches, among others.

Based on the primary studies analyzed, the adoption of non-linear neural optimization to model user-item interactions or to learn latent factor from content-based features, i.e.,

review text, check-in localization, time information, or metadata, clearly brings benefits to augment recommendation quality in terms of rating prediction and ranking tasks according to the majority of experimental results.

We have also pointed out promising future directions concerning the area. Since attention mechanisms have been demonstrated as an effective way to deal with sparse data, Gated Recurrent Units appear as a promising alternative to traditional Long Short-Term Memory, thus deserving further investigations. Analyzing and evaluating Natural Language Processing-based techniques to model textual side information and its behavior in collaborative filtering scenarios still could lead to interesting findings with respect to in-deep users/items latent representations. Regarding unsupervised feature learning, denoising autoencoders and variational autoencoders architectures presented effectiveness for user/item embedding learning based on auxiliary information, and also for the task of rating prediction.

Lastly, Graph Neural Networks exhibited promising recommendation results when dealing with cold-start users (items), also having flexible structures capable to efficiently combine robust and time-consuming modeling strategies, like convolutional operations.

2.8 Systematic Literature Review update

In this section, we provide an update of the SLR presented throughout this chapter. Considering the main conclusions outlined in Section 2.7, the objective was to summarize the most pertinent papers within the scope of this study, published between 2021 and 2023. From an extensive list of works gathered through the methodology described in Section 2.3, and considering the findings synthesized in Section 2.7, the final selection comprises twelve research papers that best align with the established criteria. Such papers have been reviewed, analyzed, and are the subject of discussion in the present section.

Table 10 summarizes the main characteristics found in those papers. As one can see, the great majority of works propose a hybrid CF-based approach, which mainly explores a DNN architecture for embedding data such as item attributes, user profile information and timestamp features. In general, the authors claim to model side information – from the same domain of rating data or from external sources – to alleviate the effects of data sparsity (ZHAO et al., 2022; AHMED et al., 2022; NAHTA et al., 2021; WANG et al., 2021), whose majority focused on addressing the cold start problem (QIAN et al., 2022; MAGRON; FÉVOTTE, 2022; TANUMA; MATSUI, 2022; ZHANG et al., 2022; HUANG et al., 2021; MOLAEI et al., 2021; MAROTO; VIGNAC; FROSSARD, 2021).

We have identified several important aspects related to the exploration of DNN architectures. In general, the authors proposed recommendation approaches introducing a DNN module based on a well established neural architecture. All of the works have performed data modeling solely using DNN, meaning that other CF-based techniques or

Table 10 – A summary of the main characteristics regarding the most relevant papers.

Related work	DNN architecture	DNN models user-item interactions	DNN models side information	Mitigating sparsity	Alleviating cold start
Qian et al. (2022)	GNN, VAE	✓	✓	✓	✓
Magron e Févotte (2022)	MLP	–	✓	–	✓
Zhao et al. (2022)	GNN	–	✓	✓	–
Ahmed et al. (2022)	DAE	–	✓	✓	–
Tanuma e Matsui (2022)	VAE	✓	✓	–	✓
Zhang et al. (2022)	DAE	–	✓	✓	✓
Sidana et al. (2021)	MLP	✓	–	–	–
Huang et al. (2021)	RNN	–	✓	–	✓
Molaei et al. (2021)	GNN	✓	–	✓	✓
Nahta et al. (2021)	VAE	✓	✓	✓	–
Wang et al. (2021)	GCN	–	✓	✓	–
Maroto, Vignac e Frossard (2021)	MLP	✓	✓	–	✓

approaches have not been utilized. The proposed recommenders embed user-item interactions, content information from users and/or items (side information), or have combined those type of data during the neural learning process.

Another finding is that almost half of the papers explored some variation of autoencoder architectures to do feature learning of side information, and one third of the works have chosen to learn embeddings from CF data through a GNN-based architecture. The recommendation approach proposed by Qian et al. (2022) faced the problem of sparsity in user-item interaction graphs. Seeking to alleviate such a problem, the authors assumed that every user or item had available attributes to be embedded and structured as an attribute graph using a Gated GNN. The rating prediction is done by means of a VAE modified to considered during training a reconstruction factor to minimize the error between user-item interaction and its corresponding user attributes. Hence, the authors attempted to further mitigate sparsity in complete cold start scenarios as well.

Ahmed et al. (2022) explored a hybrid recommender based on DAE to deal with the problem of user overlap in cross-domain contexts. The proposed approach builds a confidence matrix from the dense user data that comes from different source domains, such that the sparsity on the target domain could be reduced. The author focused on embedding user reviews and temporal data using DAE, and also explored clustering of reviews based on sentiment analysis seeking to increase recommendation accuracy. One critical limitation is that the approach showed to be vulnerable when the computed embeddings come from falsified reviews, which could produce unexpected and undesired recommendation results.

Another work employing DAE for deep feature learning is introduced by Zhang et al. (2022), and it is focused on the cold start problem. Content item features are first extracted using a DAE architecture, been further learned together with user-item interactions through the proposed Deep Pairwise Hashing. This hashing-based approach reduces computational burden of recommendation task by mapping both user and item vectors to the Hamming space, where the similarity computation can be conducted more efficiently.

The cold start problem is also the main challenge for Tanuma e Matsui (2022). The authors proposed a hybrid recommender based on a generative deep model seeking to augment interaction data from embedded item content information. To do so, a VAE architecture is introduced, which has its reconstruction loss defined by assuming the data follows a Gamma distribution instead of a Gaussian. Then, both item content and user preferences – i.e., user-item interactions – are jointly modeled during the learning process.

Nahta et al. (2021) proposes a hybrid recommender based on a VAE architecture. User attributes and item content are explored as input of two parallel VAE, such that users and item embeddings are first learned independently and then are unified for generating predictions. According to the authors, the main idea is to assume auxiliary information to act as the prior probability and the user-item interactions (ratings) to be the posterior probability during the model learning. In other words, user-item interactions are interpreted as non-deterministic data, which is a central assumption to further compute high quality recommendations while reducing sparsity.

Zhao et al. (2022) also explore the stochastic nature of user-item interactions to propose a GNN-based hybrid recommender. It relates users, items and textual reviews as a tripartite graph, having their corresponding node embeddings computed following a Bayesian learning process which integrates an attention mechanism to enhance node representations. The rating prediction task is perform by estimating neighbor relations in the graph, and it incorporates the prior knowledge of user preferences to regularize the posterior inference of attention weights (ZHAO et al., 2022). That means the recommendation quality of the approach is related to the ability of identifying the most representative graph nodes during the GNN model training.

Following a similar motivation, the work conducted by Wang et al. (2021) employs textual information to modeled together with rating data in a heterogeneous graph based on GCN for reducing the negative effects of sparsity. Also, the authors introduced a neural matching function to perform rating prediction instead of the conventional inner product of embedding vectors. Both GCN model and prediction network are optimized through the BPR loss function, i.e., the proposed recommender is specialized for ranking tasks.

Up to this point, some of the previous works have incorporated attention mechanisms into the recommendation approaches aiming to better extract high quality embedding representations during the model learning. Huang et al. (2021) adopted a similar strategy by introducing a bidirectional LSTM layer into a RNN with the goal of embed paths and meta-paths from Knowledge Graphs (KG) to either reduce the effects of cold start and increase the explainability of the hybrid DNN-based recommender the authors proposed. Basically, while the RNN learns features from KG paths employing the bidirectional path extraction algorithm, an entropy-based weighting module is proposed to embed the meta-paths information. Rating predictions are generate by integrating the RNN and the

entropy encoder through a weighted polling of their corresponding outputs.

Regarding learning embeddings exclusively from user-item interactions, only two research papers proposed to follow that path (MOLAEI et al., 2021; SIDANA et al., 2021). The CF-based approach proposed by Molaei et al. (2021) conducts a two-stage graph-based strategy for recommendation. First, a user-item bipartite graph is turned into a homogeneous graph of embedding representations through the node2vec algorithm (GROVER; LESKOVEC, 2016). Then, such node embeddings become input samples in a Cascade Tree Forest designed for rating prediction, which basically is a hierarchical multi-level organization of Decision Trees ensembles.

Sidana et al. (2021) explored theoretical properties of ranking functions and incorporated them as the objective function of an MLP-based recommender. The proposed approach could be considered a "pure" CF-based recommender and is inspired on the BPR framework (RENDLE et al., 2012). It tries to deep model user-item interactions exploring such a framework, having that as the main strategy to enrich embeddings representation and consequently enhancing the quality of recommendations.

There are other two works which introduced an MLP-based architecture for recommendation, however the main problem to be solved (or alleviated) is the cold start for both proposed approaches. The work conducted by Maroto, Vignac e Frossard (2021) assumes the distribution of user-item interactions suffers from concept drift, i.e., that rating data change over time. To deal with such a challenge, the authors introduced an adaptive weighting procedure to the learning process. The proposed approach is a hybrid recommender based on an general MLP architecture for computing embeddings from ratings, time features, user attributes and item content information. It also integrates the AutoRec (SEDHAIN et al., 2015) as recommendation module, but adapted with dropout layers to reduce overfitting.

In the work developed by Magron e Févotte (2022), the cold start problem is addressed in the music recommendation context. To do so, the acoustic features are embedded via MLP, and then are jointly learned with implicit user-item interactions by means of a deep generative model analogous to a Weighted MF. The objective is to predict music playcounts, which is performed using a conventional dot product between the embeddings of both users and items.

2.8.1 Considerations about experimental choices

In this section, a brief discussion is conducted regarding certain experimental aspects identified in the selected papers. Table 11 summarizes the main findings with respect to the experimental choices done by the researchers. Four major aspects have been considered of great relevance, which are:

(i) The optimization of model hyperparameters before running experiments;

(ii) The employment of statistical tests during results assessment;

(iii) The specific design of experiments to evaluate the cold start problem;

(iv) The source code availability concerning the experiments.

Table 11 – Summary of the main experimental aspects regarding the most relevant works.

Related work	Hyperparameter optimization		Assessing results through statistics	Cold start specific evaluation	Source code availability
	Proposed approach	Baselines			
Qian et al. (2022)	✓	–	✓	–	✓
Magron e Févotte (2022)	✓	–	–	✓	✓
Zhao et al. (2022)	✓	✓	✓	None	–
Ahmed et al. (2022)	–	–	–	✓	–
Tanuma e Matsui (2022)	✓	✓	–	✓	–
Zhang et al. (2022)	✓	✓	–	✓	–
Sidana et al. (2021)	✓	–	✓	None	✓
Huang et al. (2021)	✓	–	–	✓	–
Molaei et al. (2021)	✓	–	–	✓	–
Nahta et al. (2021)	✓	✓	✓	None	–
Wang et al. (2021)	✓	–	–	None	✓
Maroto, Vignac e Frossard (2021)	–	–	–	✓	✓

As depicted in Table 11, the majority of papers (10/12) optimized the hyperparameters of their corresponding DNN-based proposed approaches as well as described how such procedure has been done. Besides the quality of data and the amount of it considered, it is worth noting deep learning algorithms usually have their prediction performances constrained to a combination of values (hyperparameters) needed for initializing such algorithms. Thus, following that optimization procedure is paramount to effectively employ DNN and allow its model to fit the data properly. This also applies to other machine learning-based approaches.

Nonetheless, among those 10 works only 4 of them presented and described hyperparameter optimization procedures regarding recommender approaches taken as baselines during the experiments. Such a finding is at least unsettling from the perspective of the scientific correctness demanded to conduct fair experimental comparisons between different recommendation approaches. Also, those research papers claimed to outperform the baselines in every scenario they have been compared against. That been said, naturally it comes to mind the following question: have the approaches proposed in these papers really demonstrated superiority in terms of recommendation quality?

Another important finding relates to the robustness and reliability of the results. Only 4 papers performed statistical tests to assess experimental results, from which half described a full hyperparameters optimization procedure applied to the proposed recommender and baselines (ZHAO et al., 2022; NAHTA et al., 2021). We understand such experimental choices would give a more solid support to the achieved conclusions on those papers.

Employing statistical tests and other related procedures to evaluate experimental results is a well established practice in research that is being develop in other application

fields of machine learning like video processing, image segmentation, anomaly detection, and speech recognition, for instance. If well managed, statistical tests can help researchers better analyze the experiments, and reach reliability when describing their results and stating research breakthroughs. In that sense, such strategy could also be employed for the establishment of best practices and standardization of robust experimental protocols in the recommendation research field.

Regarding the adoption of a specific setting for evaluating the cold start problem, out of 8 works aimed at alleviating this issue, only one did not introduce any design to assess this type of sparsity challenge. In general, the remaining works simulated cold start scenarios by producing different subsets of data with very restrict number of user-item interaction (partial cold start) or none of them for a few users (complete cold start) and then evaluating the results employing some recommendation-specific metrics. From an experimental design perspective, we understand that strategy is essential for better analyzing cold start users/items and further be able to assess more objectively how to mitigate its effects over the quality of recommendations.

The last major experimental aspect we considered was if the authors have made available online all the source codes regarding the proposed recommender and the experiments. Less than half of the papers (5/12) provided source codes, which directly impacts both the publicity and reproducibility of the results.

2.8.2 Final considerations

In this section, we provide some remarks that we consider pertinent, thereby complementing the previously discussed topics. It seems well established in the literature that the most intuitive and effective way to deal with recommendations challenges like sparsity and cold start users/items is through manipulating auxiliary data – content information coming from users and/or items. Since this type of data is widely available on today's web, collecting and processing it has become easier when compared to the internet's historical context of the 90s and early 2000s.

Broadly speaking, what can be inferred from the analyzed papers is that its effectiveness is probably more closely related with modeling large volumes of dense data, such as textual reviews, interaction timestamps, user profiles, and other inner content from items, rather than focusing on embedding sparse rating data. Therefore, the importance of selecting extensive datasets with available side information should be emphasized, as it can assist researchers in avoiding issues related to DNN model training, such as underfitting, for instance.

Concerning experiments, the central issue remains the lack of a standardized framework for developing robust experimental protocols in recommendation research. As previously discussed in Section 2.8.1, the adoption of well-established practices such as (i) hyperparameters optimization for all tested approaches/methods, (ii) a well-founded and

justified data splitting method, (iii) multiple randomized experimental runs, (iv) the use of a appropriate statistical test for the data, and (v) making experiments' source code available online would significantly enhance the quality of produced research in terms of robustness, reliability, and reproducibility.

Chapter 3

Collaborative filtering matches Decision Templates: A practical approach to estimate predictions

This chapter presents an adaptation of the work published in the 35$^{\text{th}}$ Conference on Graphics, Patterns and Images (SIBGRAPI). The scope of the work is to introduce a classical CF-based technique, such as MF, for building an ensemble of OPF classifiers based on sparse decision templates (MARTINS; PAPA, 2022a). As part of the development of this thesis, such a work enabled the investigation of CF-like sparsity as an advantage for modeling OPF-based classifiers.

3.1 Introduction

Machine learning has shaped the world we live in. The number of applications has grown exponentially in the past years, mainly regarding computer-aided diagnosis and recommendation systems. The latter started their popularity from streaming services, in which recommendations play a crucial role. Based on the idea that similar users tend to watch related content, most modern recommender systems learn patterns encoded in latent layers using numerous techniques that range from matrix factorization (MF) to encoder/decoders (MARTINS; PAPA; ADELI, 2020).

Usually, reliable recommender systems are the ones that consistently deal with large data-related problems like high sparsity and scalability, for instance. In this sense, Collaborative Filtering (CF) stands as one of the most popular recommendation strategies since it relies on user-item interactions, i.e., the ratings users give to items. Thus, CF estimates

unobserved user-item interactions directly from the observed ones, which is possible because observed interactions are highly correlated across users and items (AGGARWAL, 2016a).

However, this type of setting does not only occur in CF-related tasks. Data generated from classifiers fusion tend to show a similar correlation between classifiers and their outcomes, i.e., labeled samples. A representative example that arises in this sense are Decision Templates (DT), which are combiners that capture the most typical patterns between classifiers and their respective outputs to classify the new ones by comparing them using some distance measure (KUNCHEVA, 2004). Such an ensemble technique has its structure closely related to that adopted in collaborative filtering and thus could enable the construction of DT from a novel perspective. Another circumstance that favors such an interpretation is that DT rely on having a complete set of outputs to be built. Hence, scenarios in which there is a lack of classifiers' outcomes tend to hinder DT robustness, consequently reducing their effectiveness during ensemble tasks.

From the above, consider the following problem: we have M samples to be labeled by N classifiers. When M and N are suitable numbers, i.e., the time required to classify all samples is within the problem needs, we can perform standard classification and take the predictions to whatever we need, e.g., to build decision templates for classifier fusion. However, when M and N grow large, the computational burden may be prohibitive. In this paper, this problem is addressed by assuming the decision template matrix is sparse; i.e., it has been partially filled only. Based on such information, one can thus estimate a given classifier's prediction based on the outcomes of other models that are somehow similar to it. In this work, employing the framework provided by CF techniques to learn such a similarity in a latent space that models both learners and samples to be classified is proposed.

The evaluation of the proposed approach has been conducted through experiments on five datasets and assessed costly-test phase algorithms like Optimum-Path Forest (OPF) and k-Nearest Neighbor (KNN). Its feasibility for fast decision template generation has been attested under specific situations like low data sparsity levels and datasets labeled with fewer classes.

3.2 Related works

Ye et al. (2019) adopted Non-Negative Matrix Factorization (NMF) to model low-rank factors of clustering ensemble information. The authors' main goal is to explore useful information – such as parameters, covariance, and probability data – that is commonly hidden or unused in big data or complex models (YE et al., 2019) (i.e., dark knowledge), which could be used as information basis during the construction of cluster ensembles through NMF.

Suh et al. (2018) introduced an ensemble method based on non-negative matrix factorization to extract useful and non-redundant information about documents retrieved from large-scale datasets. The method explores a residual matrix obtained by a gradient boosting model and the application of a local weighting scheme followed by a divide-and-conquer strategy for extracting local topics.

For tackling the problem of NMF-based stability in topic modeling, Qiang et al. (2018) proposed an NMF ensemble where each trained model starts its respective optimization with a larger learning rate than the subsequent trained model, thus avoiding the current local optima. This strategy was adopted to reduce any additional training computation costs and to enhance the ensemble's stability.

3.3 Theoretical Background

3.3.1 Collaborative Filtering

Liang et al. (2018b) define the problem of CF-based recommendation as the task of predicting or estimating what items a user would prefer by discovering and exploiting the similarity patterns across users and items. User-item interactions represent the ratings given explicitly or implicitly to items by users. Those interactions compose the so-called rating matrix, i.e., a two-dimensional structure in N rows (users) by M columns (items).

Generally, CF-based rating matrices have few interactions, which characterizes one of their main drawbacks, i.e., a high sparsity rate. Roughly speaking, it happens when the majority of user-item interactions are unknown or missing. To deal with such a challenge, many techniques have been developed over the years, among which the ones based on computational models stand out. They are represented by machine learning models that are trained to composite patterns based on training data and then make intelligent predictions for the unseen/missing entries (test data) of the rating matrix.

3.3.2 Matrix Factorization

Matrix Factorization is the leading technique in model-based CF, and it has been of great interest in the past years from both academy and industry (MARTINS; PAPA; ADELI, 2020). MF maps both users and items onto a joint latent factor space of dimensionality k, such that user-item interactions are modeled as inner products in that low dimensional space. Also, MF techniques scale very well as data grows large, presenting competitive accuracy recognition rates, as well as being capable of estimating recommendations very quickly (KOREN; BELL; VOLINSKY, 2009).

Now, a formal definition for the matrix factorization problem from a collaborative filtering perspective is presented. Let $\mathbf{R} \in \mathbb{R}^{N \times M}$ be an interaction matrix between N users and M items. Still, consider both users and items ratings could be factorized

into low-rank matrices $\mathbf{P} \in \mathbb{R}^{N \times k}$ and $\mathbf{Q} \in \mathbb{R}^{M \times k}$ in a k-dimensional latent space. The fundamental assumption is that $\mathbf{R}$ could be estimated from the product between low-rank matrices $\mathbf{P}$ and $\mathbf{Q}$, producing an approximation matrix $\hat{\mathbf{R}} \in \mathbb{R}^{N \times M}$, as described bellow:

$$\mathbf{R} \approx \hat{\mathbf{R}} = \mathbf{P}\mathbf{Q}^{\top}. \tag{1}$$

From $\mathbf{P}$ e $\mathbf{Q}$, each user i and item j has its latent factors represented by row vectors $\mathbf{p}_i \in \mathbb{R}^k$ and $\mathbf{q}_j \in \mathbb{R}^k$, respectively. They express the interaction weight between user/item and its k latent factors. Therefore, the prediction $\hat{r}_{ij}$ of association weights (i.e., ratings) a user i would give to an item j is defined by the dot product of their respective latent vectors, and it stands as the position (i, j) in the approximation matrix $\hat{\mathbf{R}}$.

Since Equation 1 tells us the product of $\mathbf{P}$ and $\mathbf{Q}$ approximates the interaction matrix $\mathbf{R}$, we could face the problem of finding the best coefficient estimation for the low-rank matrices from an optimization perspective. First, $\mathbf{P}$ and $\mathbf{Q}$ coefficients are initialized and then we calculate the difference between those matrices and $\mathbf{R}$ aiming to minimize some error measure.

To learn the latent factor vectors $\mathbf{p}_i$ and $\mathbf{q}_j$, the system tries to minimize the regularized squared error on the set of known interaction weights (KOREN; BELL; VOLINSKY, 2009), as follows:

$$\min_{\mathbf{p}^*, \mathbf{q}^*} \sum_{(i,j) \in \mathcal{A}} (r_{ij} - \mathbf{p}_i^{\top} \mathbf{q}_j) + \lambda(\|\mathbf{p}_i\|^2 + \|\mathbf{q}_j\|^2), \tag{2}$$

where $\mathcal{A}$ is the set of known interaction weights for each user-item pair (i, j) (i.e., the training set), and λ acts as a regularization factor to avoid overfitting during training.

The optimization problem as presented in Equation 2 allows us to rearrange the training data (i.e., the known interaction weights given by $\mathcal{A}$) as tuples $\langle i, j, r_{ij} \rangle$, such that all the missing entries of the original matrix $\mathbf{R}$ are avoided. This strategy is commonly adopted in CF and allows an MF-based algorithm to be optimized more efficiently over the known interaction weights without the need to store the missing ratings as zero-value entries.

3.3.3 Decision templates

In the ensemble classifier's research field, a well-established strategy for combining classifiers when they have continuous outputs stand as Decision Templates. The fundamental idea of DT is to identify the most representative decision profile $\mathbf{D}$ for each class j and then compare it with the current decision profile for a given data $\mathbf{x}$ adopting some similarity (or dissimilarity) measure.

A decision profile is a two-dimensional matrix represented in an intermediate feature space that contains the outputs of l classifiers in an ensemble. Each entry in the matrix

is the support a classifier i offers to the hypothesis that the pattern $\mathbf{x}$ comes from class j. A decision profile $\mathbf{D} \in \mathbb{R}^{l \times c}$ is depicted as follows:

$$\mathbf{D}(\mathbf{x}) = \begin{bmatrix} d_{1,1}(\mathbf{x}) & \dots & d_{1,j}(\mathbf{x}) & \dots & d_{1,c}(\mathbf{x}) \\ \vdots & & \vdots & & \vdots \\ d_{i,1}(\mathbf{x}) & \dots & d_{i,j}(\mathbf{x}) & \dots & d_{i,c}(\mathbf{x}) \\ \vdots & & \vdots & \ddots & \vdots \\ d_{l,1}(\mathbf{x}) & \dots & d_{l,j}(\mathbf{x}) & \dots & d_{l,c}(\mathbf{x}) \end{bmatrix}, \tag{3}$$

where $d_{i,j}(\mathbf{x})$ stands for the output of classifier i concerning class j, c denotes the number of classes, and l corresponds to the number of classifiers.

A decision template combiner could be viewed as a two-step process, named training and operation (KUNCHEVA, 2004). The former is responsible for generating one decision template $\mathbf{T}^j$ per class in the labeled training data set, while the latter outputs j support values for a given data $\mathbf{x}$ based on some similarity measure computation.

Let $\mathcal{V} = \{\mathbf{v}_1, \dots, \mathbf{v}_m\}$ be a labeled training data set such that $\mathbf{v}_i \in \mathbb{R}^n$. The decision template $\mathbf{T}^j(\mathcal{V}) \in \mathbb{R}^{l \times c}$ concerning class j is a matrix whose (s, u)-th entry, i.e., $T^j_{s,u}(\mathcal{V})$, is computed as follows:

$$T^j_{s,u}(\mathcal{V}) = \frac{\sum_{i=1}^{m} \mathbb{1}(\mathbf{v}_i, j)\ d_{s,u}(\mathbf{v}_i)}{\sum_{i=1}^{m} \mathbb{1}(\mathbf{v}_i, j)}, \tag{4}$$

such that $\forall s \in \{1, \dots, l\}$ and $\forall u \in \{1, \dots, c\}$. Recall that $\mathbb{1}(\mathbf{v}_i, j)$ denotes an indicator function that assigns 1 if $\mathbf{v}_i$ is labeled with class j, and 0, otherwise. Also, for the sake of simplicity, $\mathbf{T}^j(\mathcal{V})$ will be denoted by $\mathbf{T}^j$. As such, the decision template $\mathbf{T}^j$ for class j is the average of the decision profiles of the elements of the training set $\mathcal{V}$ that belong to class j.

The goal of having all $\mathbf{T}^j$ calculated is to find a support μ_j for each class j, which is measured through a similarity function $\mathcal{S}$ between a decision profile $\mathbf{D}(\mathbf{x})$ and all decision templates $\mathbf{T}^j$:

$$\mu_j(\mathbf{x}) = \mathcal{S}(\mathbf{D}(\mathbf{x}), \mathbf{T}^j), \forall j \in \{1, \dots, c\}. \tag{5}$$

Although any similarity measure can be used, usually the normalized squared euclidean is considered during the operation step.

Finally, the class ω assigned to $\mathbf{x}$ will be the one with the maximum support μ_j.

3.4 Proposed approach

The approach proposed in this paper can be understood as a pre-processing stage for building decision templates for large datasets, particularly when computational efficiency is preferred. The design involves assuming the adoption of Collaborative Filtering-based

techniques, like Matrix Factorization, to fill a sparse matrix partially populated by predicted labels in an ensemble-based learning setup. The goal is to enable decision templates to be built quickly in contexts where the number of classifiers and samples grow large, making label prediction (classification) an unfeasible task in terms of processing time.

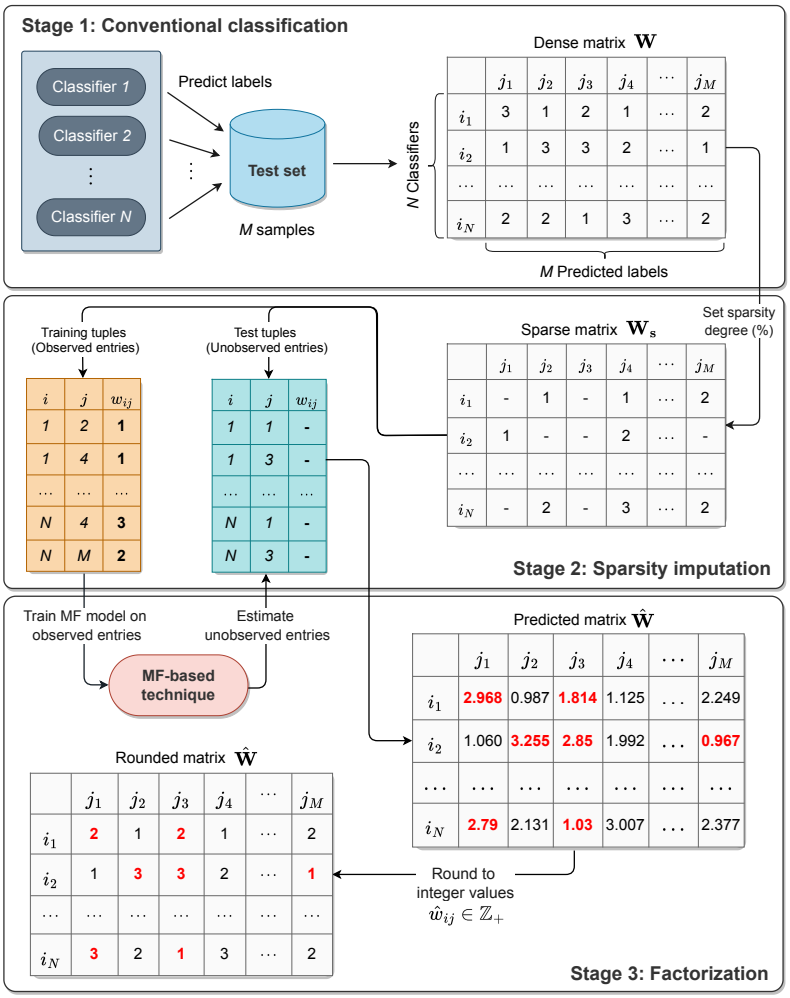
Consider the following scenario: for a given dataset having M test samples, one may want to combine N classifiers for getting more accurate classification results. Also, take into account the predicted labels for the test samples are organized in N rows and M columns, forming the *dense matrix* $\mathbf{W}$, whose entries are denoted by w_{ij}. If N and M grow large, the computational burden may be prohibitive for generating $\mathbf{W}$ in terms of data storage and processing time, mostly for some classifiers that have a costly testing phase, in general.

In situations like the one described, a more efficient approach is needed. If $\mathbf{W}$ is assumed to be sparse, i.e., it has a limited number of observed labels and the remaining entries w_{ij} are unknown, and the classifier models tend to present similar prediction outcomes, one could adopt a Collaborative Filtering-based strategy for matrix completion, i.e., to predict the outcomes of the ensemble for a given test sample. In this sense, the intent is to estimate the unobserved entries w_{ij} using a machine learning model trained on observed labels from the sparse matrix extracted from $\mathbf{W}$.

Depicted in Figure 8, the proposed approach is organized in three main stages, which are: (i) conventional classification, (ii) sparsity imputation, and (iii) factorization. In the first stage, N classifiers predict labels for M test samples, and the results are organized as the bi-dimensional matrix $\mathbf{W} \in \mathbb{R}^{N \times M}$. Each entry of $\mathbf{W}$ represents the label w_{ij} a classifier i predicted for a given test sample j. Usually, $\mathbf{W}$ is a full-populated matrix. However, in real-world situations, this may not happen, and is very likely that $\mathbf{W}$ shows a high sparsity level. That is the goal of stage two, i.e., a percentage s of predicted labels w_{ij} will be omitted from $\mathbf{W}$ to simulate real-world scenarios, resulting in the sparse matrix $\mathbf{W_s} \in \mathbb{R}^{N \times M}$. The sparsity imputation also has a experimental motivation concerning the need for having a ground-truth (i.e., the omitted entries w_{ij}) to allow the proposed approach results to be measured and validated. Otherwise, it would be impossible to analyze the experimental results in terms of recognition rate.

During the third stage, a Matrix Factorization technique is employed for decomposing $\mathbf{W_s}$ into two low-rank matrices $\mathbf{P}$ and $\mathbf{Q}$ such that their product is the approximation matrix $\hat{\mathbf{W}}$. Basically, the MF-based model is trained considering the observed labels from $\mathbf{W_s}$ (training tuples), seeking to estimate the unobserved labels (test tuples), i.e., the ones that were omitted from $\mathbf{W}$ during stage (ii). It is relevant to remind $\hat{\mathbf{W}}$ contains *approximated* real values for the predicted labels, i.e., $\hat{w}_{ij} \in \mathbb{R}$. Nevertheless, the final outcome $\hat{\mathbf{W}}$ must have categorical integer entries such that $\hat{w}_{ij} \in \mathbb{Z}_+$.

Figure 8 – Pipeline of the proposed approach.



Source: Produced by the author.

3.5 Materials and methods

3.5.1 Datasets

The experiments have been carried on five public-access datasets. Table 12 describes each dataset principal aspects like the number of samples, features, and class labels.

Table 12 – Description of the datasets adopted in this work.

Dataset	# samples	# features	# classes
Blood Transfusion (Blood) (DUA; GRAFF, 2019)	748	4	2
Breast Cancer (Cancer) (DUA; GRAFF, 2019)	569	30	2
Contraceptive Method Choice (CMC) (DUA; GRAFF, 2019)	1,473	9	3
Handwritten Digits (Digits) (DUA; GRAFF, 2019)	1,797	64	10
Iris Dataset (Iris) (DUA; GRAFF, 2019)	150	4	3

Source: Produced by the author.

3.5.2 Baselines

The proposed approach includes two types of machine learning algorithms: (i) a conventional learner to perform classification task (stage 1) and (ii) an MF-based classifier for label prediction. Since our goal is on the classification stage for fast label prediction, the baselines chosen were those acknowledged to be comparably inefficient at test time. In this sense, the conventional classifiers we considered as baselines rely, at some level, on distance computation and are described as follows:

- **OPF**: The Optimum-Path Forest (PAPA; FALCÃO; SUZUKI, 2009) is a graph-based supervised classifier that partitions the dataset (i.e., the graph nodes) during the training process into multiple optimum-path trees, each one rooted by a prototype node. The learning process is led by these nodes following a competition fashion and adopts some cost function to measure the strength of connectivity among nodes. In this work, the OPF version using a complete graph adjacency relation was considered, and a python implementation (de Rosa; PAPA, 2021) of the algorithm was adopted.

- **KNN**: It stands for the traditional KNN for supervised classification. We considered our own implementation of KNN algorithm, whose source code is available at the project repository[1].

Regarding MF-based algorithms, the most-adopted ones in collaborative filtering research were selected, i.e., Non-negative Matrix Factorization (NMF) (ZHANG et al., 2006), Probabilistic Matrix Factorization (PMF) (SALAKHUTDINOV; MNIH, 2009), and Singular Value Decomposition (SVD) (KOREN; BELL; VOLINSKY, 2009).

Regarding MF implementation, the Surprise recommendation library (HUG, 2020) was employed. Finally, all source codes and supplementary files regarding the conducted experiments are available at the project repository.

3.5.3 Experimental setup

The experiments considered pre-processing the data such that all features were normalized to have zero mean and unitary standard deviation, and the Accuracy Score (ACC)[2] was employed as the measure to assess the label prediction performance on the test set. Also, the datasets were randomly splitted using 5-fold cross-validation repeated over 20 times for further statistical analysis. To ensure the results reliability, a Wilcoxon signed-rank test was conducted considering the following related groups:

- G_1: Test samples are classified by a conventional machine learning model (a baseline classifier).

[1] <https://github.com/guibmartins/mf_based_dt_ensemble>
[2] We adopted the implementation from scikit-learn (PEDREGOSA et al., 2011).

- ❑ G2: Test samples are classified by a MF-based technique during the third stage of the proposed approach.

We want to evaluate under what circumstances the proposed approach (G2) present statistically similar recognition rates when compared to conventional classification (G1). Additionally, a second experimental test was conducted to assess the computation burden, i.e., whether G2 can be faster than G1.

The processing time means the time (seconds) spent by groups G1 and G2 to label test samples. With respect to group G1, it represents the computational cost to classify the omitted entries from sparse matrix $\mathbf{W_s}$. For group G2, the computation cost regards the total time to perform all tasks related to the factorization stage of the proposed approach.

In summary, our goal is to find out in which sparsity conditions the predicted matrix $\hat{\mathbf{W}}$ (Figure 8) is able to keep similar recognition rates while computation time decreases.

Hyperparameter tuning

We also used one fold from holdout (validation set) to tune the baseline hyperparameters. OPF has no hyperparameters, but KNN needs to initialize the number of neighbors. We accomplished that task by selecting the one that maximized the accuracy in the validation set. Concerning NMF, PMF, and SVD, we set the number of latent factors to be $k = 2$, while the learning rate and the regularization terms were tuned through a randomized search procedure[3].

Sparsity imputation

During the second stage, different sparsity degrees were considered, in which a 20-step is taken within the interval $[10, 70]$ to simulate distinct data sparsity scenarios. In practice, the sparsity imputation acts as a split operation on $\mathbf{W_s}$ such that the applied sparsity degree s represents the test set size, while the training set size is defined by $1 - s$.

Machine configuration

All experiments were performed using a Kubuntu 20.04 Linux 64 bits machine with 8Gb of RAM running on 8 × Intel® Core™ i5-9300H CPU @ 2.40GHz processors.

[3] The interval of values to be searched was defined empirically.

3.6 Experimental results

3.6.1 Prediction accuracy evaluation

Table 13 presents the average accuracy and standard deviation of the proposed approach and OPF as the baseline classifier. Similarly, Table 14 shows the comparison between the KNN classifier and the proposed approaches. In both tables, the statistically similar best accuracy results according to the Wilcoxon signed-rank with 95% confidence have been highlighted in bold.

Table 13 – Comparison of average accuracy among OPF and the proposed approaches across different datasets.

Dataset	Sparsity (%)	Baseline	Proposed approaches		
		OPF	NMF	PMF	SVD
Blood	10	0.6289 ± 0.0115	$\mathbf{0.6366 \pm 0.0114}$	$\mathbf{0.6356 \pm 0.0143}$	$\mathbf{0.631 \pm 0.0123}$
	30	0.6289 ± 0.0115	$\mathbf{0.6615 \pm 0.0125}$	$\mathbf{0.6602 \pm 0.0134}$	0.6387 ± 0.0151
	50	0.6289 ± 0.0115	$\mathbf{0.6923 \pm 0.0097}$	$\mathbf{0.6954 \pm 0.0151}$	0.6438 ± 0.0162
	70	0.6289 ± 0.0115	0.7108 ± 0.008	$\mathbf{0.736 \pm 0.0074}$	0.6575 ± 0.0164
Cancer	10	$\mathbf{0.9499 \pm 0.004}$	$\mathbf{0.9497 \pm 0.0046}$	$\mathbf{0.9481 \pm 0.0041}$	$\mathbf{0.9497 \pm 0.0039}$
	30	$\mathbf{0.9499 \pm 0.004}$	$\mathbf{0.949 \pm 0.0054}$	0.9457 ± 0.0059	$\mathbf{0.9486 \pm 0.0054}$
	50	$\mathbf{0.9499 \pm 0.004}$	0.9386 ± 0.0073	0.9331 ± 0.0059	0.938 ± 0.0066
	70	$\mathbf{0.9499 \pm 0.004}$	0.8888 ± 0.0102	0.8322 ± 0.021	0.8902 ± 0.008
CMC	10	$\mathbf{0.4294 \pm 0.0064}$	$\mathbf{0.4283 \pm 0.0064}$	$\mathbf{0.4265 \pm 0.0062}$	0.4255 ± 0.0071
	30	$\mathbf{0.4294 \pm 0.0064}$	0.4124 ± 0.0087	0.4143 ± 0.0071	0.4113 ± 0.0069
	50	$\mathbf{0.4294 \pm 0.0064}$	0.3873 ± 0.0075	0.3875 ± 0.0065	0.3914 ± 0.0085
	70	$\mathbf{0.4294 \pm 0.0064}$	0.3522 ± 0.0093	0.3492 ± 0.01	0.3571 ± 0.0092
Digits	10	$\mathbf{0.9678 \pm 0.0016}$	0.9547 ± 0.0037	0.9582 ± 0.0024	0.9633 ± 0.0028
	30	$\mathbf{0.9678 \pm 0.0016}$	0.8755 ± 0.0146	0.904 ± 0.0129	0.9222 ± 0.0115
	50	$\mathbf{0.9678 \pm 0.0016}$	0.7168 ± 0.0309	0.7868 ± 0.019	0.8215 ± 0.0241
	70	$\mathbf{0.9678 \pm 0.0016}$	0.4914 ± 0.0241	0.5615 ± 0.0289	0.59 ± 0.03
Iris	10	$\mathbf{0.9302 \pm 0.0079}$	$\mathbf{0.9301 \pm 0.0106}$	$\mathbf{0.9311 \pm 0.0074}$	$\mathbf{0.9303 \pm 0.0081}$
	30	$\mathbf{0.9302 \pm 0.0079}$	$\mathbf{0.9299 \pm 0.011}$	$\mathbf{0.9289 \pm 0.0104}$	$\mathbf{0.9315 \pm 0.0095}$
	50	$\mathbf{0.9302 \pm 0.0079}$	0.9059 ± 0.014	0.8953 ± 0.0144	0.9065 ± 0.0176
	70	$\mathbf{0.9302 \pm 0.0079}$	0.7891 ± 0.0251	0.6091 ± 0.048	0.8092 ± 0.0241

Source: Produced by the author.

According to Table 13, the best results for at least one of the proposed approaches in 4 of 5 datasets have been obtained. The majority of those best accuracy predictions attained for sparsity levels varying from 10% to 30% in general. Cancer and Iris datasets achieved the best overall performance, both having balanced data, a relatively small number of classes, and been the two smallest datasets among the ones considered for experimentation.

Regarding the CMC dataset, NMF and PMF were capable of matching the baseline's prediction accuracy, but only when the data was 10% sparse. Concerning the Digits dataset, the proposed approaches demonstrated to be less effective for any sparsity level when compared to traditional OPF prediction. It is noteworthy Digits dataset has its

data distributed among ten balanced classes, while the other ones vary from 2 to 3 classes, which possibly made it harder for the model to train and classify the data effectively.

Table 14 – Comparison of average accuracy among KNN classifier and the proposed approaches across different datasets.

Dataset	**Sparsity (%)**	**Baseline**	**Proposed methods**		
		KNN	NMF	PMF	SVD
Blood	10	$\mathbf{0.7827 \pm 0.0065}$	0.778 ± 0.018	$\mathbf{0.7825 \pm 0.0188}$	$\mathbf{0.7879 \pm 0.0188}$
	30	$\mathbf{0.7827 \pm 0.0065}$	$\mathbf{0.7814 \pm 0.0145}$	$\mathbf{0.781 \pm 0.0128}$	$\mathbf{0.7825 \pm 0.0128}$
	50	$\mathbf{0.7827 \pm 0.0065}$	$\mathbf{0.7795 \pm 0.0105}$	$\mathbf{0.7769 \pm 0.0112}$	$\mathbf{0.7831 \pm 0.0072}$
	70	$\mathbf{0.7827 \pm 0.0065}$	0.7754 ± 0.0086	0.7662 ± 0.0086	0.777 ± 0.009
Cancer	10	0.9592 ± 0.0053	$\mathbf{0.9625 \pm 0.0115}$	$\mathbf{0.9619 \pm 0.0119}$	$\mathbf{0.9658 \pm 0.0098}$
	30	$\mathbf{0.9592 \pm 0.0053}$	$\mathbf{0.9632 \pm 0.0081}$	$\mathbf{0.9605 \pm 0.0069}$	$\mathbf{0.9595 \pm 0.0084}$
	50	$\mathbf{0.9592 \pm 0.0053}$	0.9421 ± 0.0085	0.9415 ± 0.0083	0.9391 ± 0.0094
	70	$\mathbf{0.9592 \pm 0.0053}$	0.8796 ± 0.0127	0.7413 ± 0.0615	0.8803 ± 0.012
CMC	10	$\mathbf{0.5013 \pm 0.0066}$	0.4507 ± 0.0212	0.4431 ± 0.0223	0.4478 ± 0.0125
	30	$\mathbf{0.5013 \pm 0.0066}$	0.4456 ± 0.0104	0.4291 ± 0.0126	0.4261 ± 0.0083
	50	$\mathbf{0.5013 \pm 0.0066}$	0.4264 ± 0.0126	0.3957 ± 0.0105	0.4051 ± 0.0142
	70	$\mathbf{0.5013 \pm 0.0066}$	0.3937 ± 0.0125	0.3502 ± 0.0075	0.3648 ± 0.0123
Digits	10	$\mathbf{0.9705 \pm 0.0022}$	0.665 ± 0.0505	0.9043 ± 0.0297	0.9197 ± 0.0149
	30	$\mathbf{0.9705 \pm 0.0022}$	0.5958 ± 0.045	0.8356 ± 0.024	0.8483 ± 0.022
	50	$\mathbf{0.9705 \pm 0.0022}$	0.4731 ± 0.0416	0.6412 ± 0.0427	0.7144 ± 0.0342
	70	$\mathbf{0.9705 \pm 0.0022}$	0.3542 ± 0.0248	0.3033 ± 0.0539	0.4326 ± 0.0479
Iris	10	$\mathbf{0.9527 \pm 0.0268}$	$\mathbf{0.9447 \pm 0.02}$	$\mathbf{0.9433 \pm 0.0293}$	$\mathbf{0.9547 \pm 0.0218}$
	30	$\mathbf{0.9527 \pm 0.0268}$	0.9396 ± 0.0189	$\mathbf{0.9358 \pm 0.0221}$	$\mathbf{0.9411 \pm 0.0174}$
	50	$\mathbf{0.9527 \pm 0.0268}$	0.9023 ± 0.0316	0.8523 ± 0.0384	0.9157 ± 0.0248
	70	$\mathbf{0.9527 \pm 0.0268}$	0.772 ± 0.029	0.4959 ± 0.042	0.7856 ± 0.0246

Source: Produced by the author.

Following the results presented in Table 14, one could observe the proposed approaches achieved an almost similar prediction performance in comparison to the results in Table 13 using OPF. The same behavior occurs: the proposed approaches demonstrated statistically similar accuracy effectiveness in Blood, Cancer and Iris datasets mainly for sparsity levels equal to 30% or less, in general. Besides, KNN's accuracy scores have not been surpassed or matched by any proposed approach in CMC and Digits datasets.

3.6.2 Processing time evaluation

Also, experiments to verify the computational efficiency of the proposed approaches have been conducted. Their performances have been assessed against a baseline algorithm in terms of processing time (seconds). The baseline's processing time is the one spent by the algorithm to predict labels for all test samples that will compose a decision template, while for the proposed approaches it includes both training (i.e., matrix factorization) and prediction steps duration.

The results concerning OPF and KNN are depicted in Tables 15 and 16, respectively. As one can observe in Table 15, at least one of the proposed approaches outperformed the OPF classifier in all datasets considered. PMF and SVD achieved the lowest processing times, being statistically similar to one another in most cases. Concerning the results presented in Table 16, the proposed approaches showed to be more efficient than KNN classifier in five datasets for all sparsity levels considered. Statistically, NMF achieved the most efficient performance in terms of processing time when compared to the other MF-based techniques.

Table 15 – Comparison of average processing time between OPF classifier and the proposed approaches across different datasets.

Dataset	**Sparsity (%)**	**Baseline**	**Proposed approaches**		
		OPF	NMF	PMF	SVD
Blood	10	0.4138 ± 0.1158	0.041 ± 0.0035	$\mathbf{0.03344 \pm 0.00373}$	$\mathbf{0.0341 \pm 0.0043}$
	30	1.1507 ± 0.2852	0.0348 ± 0.0033	$\mathbf{0.027 \pm 0.0026}$	$\mathbf{0.02693 \pm 0.00325}$
	50	1.9126 ± 0.4656	0.0273 ± 0.0023	$\mathbf{0.02039 \pm 0.00213}$	$\mathbf{0.0205 \pm 0.0021}$
	70	2.6744 ± 0.6534	0.0209 ± 0.0032	$\mathbf{0.01399 \pm 0.0021}$	$\mathbf{0.0143 \pm 0.0016}$
Cancer	10	0.1646 ± 0.0073	0.0304 ± 0.004	$\mathbf{0.02252 \pm 0.00209}$	$\mathbf{0.0243 \pm 0.0034}$
	30	0.4509 ± 0.0171	0.025 ± 0.0028	$\mathbf{0.01848 \pm 0.00233}$	$\mathbf{0.0192 \pm 0.0023}$
	50	0.7395 ± 0.0241	0.0201 ± 0.0023	$\mathbf{0.01389 \pm 0.0019}$	$\mathbf{0.0146 \pm 0.0015}$
	70	1.0364 ± 0.0275	0.0143 ± 0.0017	$\mathbf{0.00944 \pm 0.00114}$	$\mathbf{0.0111 \pm 0.0027}$
CMC	10	1.565 ± 0.0629	0.0765 ± 0.0037	$\mathbf{0.06207 \pm 0.00331}$	0.0646 ± 0.0042
	30	4.6949 ± 0.1886	0.0643 ± 0.0047	$\mathbf{0.05035 \pm 0.00292}$	0.052 ± 0.0028
	50	7.8248 ± 0.3144	0.0517 ± 0.0032	$\mathbf{0.03899 \pm 0.00369}$	0.0404 ± 0.003
	70	10.9547 ± 0.4402	0.0377 ± 0.0034	$\mathbf{0.02698 \pm 0.00283}$	$\mathbf{0.0288 \pm 0.0032}$
Digits	10	1.3836 ± 0.1081	0.0962 ± 0.0094	$\mathbf{0.08001 \pm 0.00795}$	$\mathbf{0.0803 \pm 0.0058}$
	30	4.1508 ± 0.3242	0.0809 ± 0.0069	$\mathbf{0.0642 \pm 0.0046}$	$\mathbf{0.06396 \pm 0.00406}$
	50	6.9181 ± 0.5403	0.0642 ± 0.0051	$\mathbf{0.0486 \pm 0.0035}$	$\mathbf{0.04854 \pm 0.00253}$
	70	9.6853 ± 0.7564	0.0472 ± 0.0042	$\mathbf{0.03417 \pm 0.00396}$	$\mathbf{0.0357 \pm 0.0042}$
Iris	10	0.01909 ± 0.00103	0.0101 ± 0.0016	$\mathbf{0.0071 \pm 0.001}$	$\mathbf{0.00701 \pm 0.00064}$
	30	0.0384 ± 0.0014	0.0085 ± 0.0013	$\mathbf{0.00561 \pm 0.00055}$	0.0062 ± 0.0011
	50	0.0586 ± 0.0021	0.0066 ± 0.0007	$\mathbf{0.00427 \pm 0.00031}$	$\mathbf{0.0045 \pm 0.0006}$
	70	0.0787 ± 0.0025	0.0051 ± 0.0005	$\mathbf{0.00298 \pm 0.00028}$	$\mathbf{0.0033 \pm 0.0005}$

Source: Produced by the author.

In general, the datasets size has a power of influence over the computational efficiency of a classifier. As the number of samples to train and classify grows, the computational load tends to increase. Based on the experimental results, this expected behavior occurs for datasets with distinct sizes. In this sense, conventional classification performed by OPF showed itself to be more sensitive to the dataset's size than the proposed approaches, especially when the number of samples grows large (i.e., beyond $1,000$).

For that situation, one possible explanation is related to the classification of a new test sample by OPF. Once OPF has a model trained, it requires that a new to-be-classified node (test sample) be connected to all other graph nodes before the competition-based classification procedure starts. Concerning KNN, it is well known the prediction phase

relies on multiple distance calculations between the test sample and all other data points, which in practice is characterized as an exhaustive procedure directly influenced by the number of neighbors to be searched and the dataset size, for instance.

Table 16 – Comparison of average processing time among KNN classifier and the proposed approaches across different datasets.

Dataset	**Sparsity (%)**	**Baseline**	**Proposed approaches**		
		KNN	NMF	PMF	SVD
Blood	10	0.3291 ± 0.025	$\mathbf{0.07664 \pm 0.00794}$	0.1447 ± 0.0103	0.1427 ± 0.0079
	30	0.9436 ± 0.0444	$\mathbf{0.0617 \pm 0.00338}$	0.1123 ± 0.0087	0.1114 ± 0.0049
	50	1.5752 ± 0.0855	$\mathbf{0.04987 \pm 0.00337}$	0.0824 ± 0.0073	0.0815 ± 0.0038
	70	2.1967 ± 0.1269	$\mathbf{0.03618 \pm 0.00241}$	0.0516 ± 0.0046	0.0495 ± 0.0017
Cancer	10	0.1963 ± 0.0135	$\mathbf{0.05426 \pm 0.0027}$	0.1074 ± 0.0062	0.1081 ± 0.0079
	30	0.5737 ± 0.0408	$\mathbf{0.0472 \pm 0.00474}$	0.0845 ± 0.0058	0.0853 ± 0.0075
	50	0.9425 ± 0.0645	$\mathbf{0.03674 \pm 0.00211}$	0.0632 ± 0.0075	0.0605 ± 0.002
	70	1.298 ± 0.0788	$\mathbf{0.02653 \pm 0.00132}$	0.0375 ± 0.0017	0.0382 ± 0.0023
CMC	10	1.2364 ± 0.1486	$\mathbf{0.14674 \pm 0.01839}$	0.2942 ± 0.0322	0.2925 ± 0.0296
	30	3.7328 ± 0.4747	$\mathbf{0.12036 \pm 0.01243}$	0.2289 ± 0.0213	0.2293 ± 0.0223
	50	6.2303 ± 0.7518	$\mathbf{0.11115 \pm 0.05115}$	0.1791 ± 0.0506	0.1778 ± 0.0518
	70	8.6708 ± 0.9819	$\mathbf{0.0696 \pm 0.00663}$	0.1056 ± 0.012	0.1022 ± 0.0081
Digits	10	2.1029 ± 0.2788	$\mathbf{0.20734 \pm 0.02831}$	0.4183 ± 0.0797	0.3994 ± 0.0499
	30	6.284 ± 0.642	$\mathbf{0.17044 \pm 0.02172}$	0.3365 ± 0.0602	0.3123 ± 0.0424
	50	10.6274 ± 1.1597	$\mathbf{0.13726 \pm 0.0173}$	0.2412 ± 0.0485	0.2258 ± 0.0322
	70	14.9125 ± 1.8734	$\mathbf{0.09736 \pm 0.01661}$	0.147 ± 0.026	0.1391 ± 0.02
Iris	10	0.0162 ± 0.0008	$\mathbf{0.01385 \pm 0.00071}$	0.0342 ± 0.0087	0.0374 ± 0.0087
	30	0.0395 ± 0.0019	$\mathbf{0.01143 \pm 0.00049}$	0.0264 ± 0.007	0.0283 ± 0.0064
	50	0.0634 ± 0.002	$\mathbf{0.0093 \pm 0.00027}$	0.0202 ± 0.0061	0.0213 ± 0.0064
	70	0.0875 ± 0.0027	$\mathbf{0.00694 \pm 0.00029}$	0.0117 ± 0.0027	0.0125 ± 0.0028

Source: Produced by the author.

On the other hand, label prediction by MF-based algorithms is much straightforward since it solely relies on the inner product of the two latent matrices learned. That allows the MF-based proposed approaches to label all test samples in a single matrix operation, being more efficient than the incremental strategy performed by conventional OPF and distance-based classifiers like KNN.

3.7 Conclusions

In this paper, an approach for building decision templates based on latent factor learning is introduced, especially considering scenarios where computational efficiency is preferred. The experiments have shown that the proposed approach is robust for fast generation of decision models when the data is distributed over a limited number of classes and assuming that it is at most 30% sparse. Moreover, the feasibility of the proposed

approach is identified in situations where learning multiple classifier models from training datasets presents a high computational burden, especially for distance-based classifiers.

Chapter 4

How to proper initialize Gaussian Mixture Models with Optimum-Path Forest

This chapter presents an adaption of the work published in the 35$^{\text{th}}$ Conference on Graphics, Patterns and Images, which introduces a fast and scalable unsupervised Optimum-Path Forest for improving the initialization of Gaussian mixture models. (MARTINS; PAPA, 2022b). The intuition behind this work is to mitigate scalability issues related to OPF clustering, thus enabling it to be applied for particular tasks involving large sets of data that demand efficiency, such as CF-based recommendation.

4.1 Introduction

GMM stands out as a robust and flexible probabilistic model to cope with soft clustering and density estimation. Clustering algorithms usually are sensitive to their hyperparameters, mainly models based on Finite Mixture like the traditional GMM, which relies on the number of mixture components. Also, such likelihood estimation-based algorithms can suffer from local optimal solutions, making the optimized estimated parameters highly dependent on the initial values to start the Expectation-Maximization (EM) algorithm (SHIREMAN; STEINLEY; BRUSCO, 2017).

In this sense, several works addressed the mentioned problem employing clustering algorithms. Qiu, Fang e Yuan (2019) introduced an Improved Density Peaks Clustering-based EM algorithm that employs an adaptive searching strategy of the probability density peaks to set initial values for GMM parameters. Blömer e Bujna (2016) proposed a seeding

method that explores the k-means++ and Gonzalez algorithm (GONZALEZ, 1985) for estimating spherical covariance matrix adaptively for each mixture component.

Although the literature has many studies that cope with finding Gaussian parameters, two traditional strategies remain popular to perform such a task: (i) random initialization and (ii) initial estimation based on k-means clustering. These strategies are simple to implement, provide reasonable initial guesses for the EM algorithm, and generally have a low computational cost when compared to other initialization techniques. However, for both strategies – as well as for other clustering-based techniques – usually the number of clusters must be set a priori.

Some years ago, the Optimum-Path Forest (OPF) framework (PAPA; FALCÃO; SUZUKI, 2009; ROCHA; CAPPABIANCO; FALCÃO, 2009) was proposed to overcome some limitations of other pattern classifiers with supervised, semi-supervised, and unsupervised variants. The latter interprets the clustering task as a graph partition problem, in which data samples are interpreted as graph nodes connected by a pre-defined symmetric adjacency relation. The learning process is led by a few key nodes (prototypes) following a competition fashion and adopts a path-cost function for calculating the connectivity strength among nodes. Each key node becomes the root of an optimum-path tree (cluster) and is located approximately in the center of each cluster, i.e., a high-density region (ROSA et al., 2014).

Although OPF clustering has shown promising results concerning several tasks, it presents some drawbacks: (i) the adjacency relation construction relies on the definition of a k-nn graph and is conducted through a linear search over all data points; (ii) the graph partitioning strategy is done by an exhaustive search seeking incrementally for the k (number of neighbors) that minimizes a graph-cut measure. Thus, when large neighborhoods and extensive datasets are considered, such procedures may be impractical due to their high inherent computation cost. In that sense, an alternative strategy to deal with k-nn scalability issues is to assume a small number of errors are allowed during the search for nearest neighbors.

Therefore, this work proposes a graph-based clustering approach to address the initialization of the EM algorithm employed for optimizing Gaussian Mixture Models. It adapts OPF to run with a k-approximate nearest neighbor (k-ann) graph as adjacency relation, speeding up the k-neighborhood construction during multiple passes OPF does toward clusters computation. The main idea is to explore some of OPF clustering properties like its non-parametric unsupervised nature, the estimation of the number of clusters on-the-fly, its capability to output clusters with multiple shapes, and the prototypes tendency to be located in the center of the highest dense regions, thus allowing the Gaussian parameters to be encode directly from the clustering structure computed by OPF.

This work figures two main contributions: (i) to present an alternative unsupervised OPF suitable to perform in large-scale datasets by adopting an adjacency relation that

employs data structures based on k-ann; and (ii) to experimentally analyze the proposed approach for the task of initialization of GMM parameters, comparing it against baselines such as the conventional unsupervised OPF and k-means[1] on several public datasets.

4.2 Theoretical background

4.2.1 Unsupervised Optimum-Path Forest

Let $\mathcal{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_n\}$ be an unlabeled dataset such that $\boldsymbol{x}_i \in \mathbb{R}^m$ denotes the i-th sample represented by an m-dimensional feature vector. Also, consider a graph $\mathcal{G} = (X, A)$ is defined such that the arcs $(\boldsymbol{x}_i, \boldsymbol{x}_j) \in \mathcal{A}$ connect k-nearest neighbors in the feature space, $i \neq j$.

Initially, graph $\mathcal{G}$ has its nodes structured by a well-defined adjacency relation based on a k-neighborhood for the unsupervised OPF version. Therefore, an undirected k-nn graph is computed, whose arcs are weighted by $d(\boldsymbol{x}_i, \boldsymbol{x}_j)$, i.e., the distance between those nodes relies on some symmetric dissimilarity measure. In practice, the adjacency relation sets initial connections among graph nodes.

The second stage starts the construction of Optimum-Path Trees (OPT) by weighting the nodes and then selecting the most representative ones, which are called prototypes and are usually located at the center of the most dense regions on their respective k-neighborhood (ROSA et al., 2014). To fulfill that purpose, all graph nodes are weighted by a density value $\rho(\boldsymbol{x}_i)$ computed through a Gaussian probability density function (pdf), given by:

$$\rho(\boldsymbol{x}_i) = \frac{1}{\sqrt{2\pi\sigma^2}|\mathcal{A}(\boldsymbol{x}_i)|} \sum_{\forall \boldsymbol{x}_j \in \mathcal{A}(\boldsymbol{x}_i)} \exp\left\{\frac{-d^2(\boldsymbol{x}_i, \boldsymbol{x}_j)}{2\sigma^2}\right\}, \tag{6}$$

where $|\mathcal{A}(\boldsymbol{x}_i)| = k$, $\sigma = \frac{d_f}{3}$, and d_f is the maximum arc weight in (X, A). This parameter choice considers all nodes for density computation, since a Gaussian function covers most samples within $d(\boldsymbol{x}_i, \boldsymbol{x}_j) \in [0, 3\sigma]$.

By taking into account the k-nearest neighbors for each graph node, unsupervised OPF handles different concentrations and reduces the scale problem to the one of finding the optimum value of k within $[1, k_{\max}]$, for $1 \leq k_{\max} \leq |\mathcal{X}|$. The solution provided by Rocha, Cappabianco e Falcão (2009) considers the minimum graph cut provided by the clustering results for $k \in [1, k_{\max}]$ according to a measure on normalized graph.

The third stage refers to the competition process lead by prototypes to conquer the remaining nodes, resulting in the partition of $\mathcal{G}$ into multiple OPTs, which are rooted by a single prototype and will constitute the final clusters composed of their most strongly

[1] Notice parameter k has a distinct meaning concerning OPF (neighborhood size) and k-means (number of clusters). For the sake of simplicity, we decided to keep the same notation.

connected nodes (samples). The OPF clustering objective is to compute OPTs by maximizing a cost function, which is established in terms of paths on $\mathcal{G}$ and is defined as an acyclic sequence of adjacent nodes in $\mathcal{A}$.

Let a path $\pi_{\boldsymbol{x}_j}$ be a sequence of adjacent samples starting from a root set $\mathcal{R}$, i.e., the set containing only prototype nodes, and ending at a sample $\boldsymbol{x}_j$. Also, let $\pi_{\boldsymbol{x}_j} = \langle \boldsymbol{x}_j \rangle$ be a trivial path and $\pi_{\boldsymbol{x}_i} \cdot \langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle$ the concatenation of $\pi_{\boldsymbol{x}_i}$ and the arc $(\boldsymbol{x}_i, \boldsymbol{x}_j)$. Given some connectivity function $f(\cdot)$, a path $\pi_{\boldsymbol{x}_j}$ is optimal only if $f(\pi_{\boldsymbol{x}_j}) > f(\tau_{\boldsymbol{x}_j})$, where $\tau_{\boldsymbol{x}_j}$ represents any other possible path ending at sample $\boldsymbol{x}_j$. Among all possible paths $\pi_{\boldsymbol{x}_j}$ with roots on the pdf maxima, unsupervised OPF assigns to $\boldsymbol{x}_j$ a path whose minimum density value along it is maximum. Each pdf maximum should then define an influence zone (i.e., a cluster) by selecting the samples that are the most strongly connected to it than to any other maximum. Formally, it finds a connectivity map $\mathcal{V}(\boldsymbol{x}_j) = \max_{\forall \pi_{\boldsymbol{x}_j} \in (\mathcal{X}, \mathcal{A})} \{f(\pi_{\boldsymbol{x}_j})\}$ and the objective is to maximize $f(\pi_{\boldsymbol{x}_j})$ for all $\boldsymbol{x}_j \in \mathcal{X}$, where

$$\begin{aligned} f(\langle \boldsymbol{x}_j \rangle) &= \begin{cases} \rho(\boldsymbol{x}_j) & \text{if } \boldsymbol{x}_j \in \mathcal{R} \\ \rho(\boldsymbol{x}_j) - \delta & \text{otherwise} \end{cases} \\ f(\pi_{\boldsymbol{x}_i} \cdot \langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle) &= \min\{f(\pi_{\boldsymbol{x}_i}), \rho(\boldsymbol{x}_j)\}, \end{aligned} \tag{7}$$

for $\delta = \min_{\forall (\boldsymbol{x}_i, \boldsymbol{x}_j) \in \mathcal{A} | \rho(\boldsymbol{x}_j) \neq \rho(\boldsymbol{x}_i)} |\rho(\boldsymbol{x}_j) - \rho(\boldsymbol{x}_i)|$ and $\mathcal{R}$ being a root set containing one element for each pdf maximum and being discovered on-the-fly. It is worth mentioning that δ acts as a handicap and filtering parameter for the pdf, where higher values reduce either the number of maxima and the number of clusters.

Therefore, the unsupervised OPF algorithm maximizes the connectivity map $\mathcal{V}(\boldsymbol{x}_j)$ such that the optimum paths form an optimum-path forest, i.e., an acyclic predecessor map $\mathcal{P}$ that assigns to each sample $\boldsymbol{x}_j \notin \mathcal{R}$ its predecessor $\mathcal{P}(\boldsymbol{x}_j)$ in the optimum path from $\mathcal{R}$ or a marker *nil* when $\boldsymbol{x}_j \in \mathcal{R}$. Each prototype node will be the root of an OPT (cluster), and the collection of all OPTs creates the optimum-path forest.

4.2.2 Gaussian Mixture Model

Gaussian Mixture Model is a classical statistical clustering technique founded on the finite mixture theory and stands as one of the most popular algorithms for dealing with clustering problems. In short, a mixture is defined as a set of C probability distributions representing C clusters, and they are based on some set of parameters that lead the feature values for samples of that cluster (WITTEN; FRANK; Mark A. Hall, 2011).

Again, let $\mathcal{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_n\}$ be a unlabeled dataset such that $\boldsymbol{x}_i \in \mathbb{R}^m$. Consider all the observed samples are independent and identically distributed random variables (i.i.d.) from a density $p(\boldsymbol{x})$ parameterized by both mean $\boldsymbol{\mu} \in \mathbb{R}^m$ and covariance matrix $\boldsymbol{\Sigma} \in \mathbb{R}^{m \times m}$, i.e, a conditional probability that represents a finite mixture of c components,

given by:

$$p(\boldsymbol{x};\boldsymbol{\theta}) = \sum_{c=1}^{C} w_c\, p(\boldsymbol{x} \mid \boldsymbol{\theta}_c), \tag{8}$$

where w_c is the contribution weight regarding component c, $\sum_{c=1}^{C} w_c = 1$, and $\boldsymbol{\theta} = (w_1, ..., w_C, \boldsymbol{\mu}_1, ..., \boldsymbol{\mu}_C, \boldsymbol{\Sigma}_1, ..., \boldsymbol{\Sigma}_C)$. Normally, $p(\boldsymbol{x};\boldsymbol{\theta})$ is defined as a multivariate Gaussian density probability, as follows:

$$p(\boldsymbol{x} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{m/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\left\{ -\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^{\top}\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right\}. \tag{9}$$

The fundamental goal is to estimate all parameters $\boldsymbol{\theta}$, such that the conditional probabilities for each $\boldsymbol{x} \in \mathcal{X}$ are maximized with respect to the component c of the finite mixture. Nevertheless, since $\mathcal{X}$ is an unlabeled dataset, and distribution-based mixture models are unable to provide an analytical solution to the optimization problem, the Maximum Likelihood Estimation is not an appropriate method given those circumstances. Alternatively, given that the parameter estimation problem involves an incomplete statistical model, the Expectation-Maximization algorithm should be considered. Thus, the model is reformulated and an unobserved latent variable $z \in \mathcal{Z}$ is assumed, which specifies the component to which each sample $\boldsymbol{x}$ belongs. During Expectation (E-step), membership probabilities must be calculated for all samples $\boldsymbol{x}$ with respect to each component c from the finite mixture. To be more specific, given sample $\boldsymbol{x}$ and estimated parameters $\boldsymbol{\theta}$, the conditional probability of z is determined using Bayes Theorem as posterior probability $\boldsymbol{b}_c$, given by:

$$\begin{aligned} \boldsymbol{b}_c &= p(z=c|\boldsymbol{x},\boldsymbol{\theta}) \\ &= \frac{p(\boldsymbol{x}|\boldsymbol{\mu}_c,\boldsymbol{\Sigma}_c)p(z=c)}{\sum_{c=1}^{C} p(\boldsymbol{x}|\boldsymbol{\mu}_c,\boldsymbol{\Sigma}_c)p(z=c)} \\ &= \frac{p(\boldsymbol{x}|\boldsymbol{\mu}_c,\boldsymbol{\Sigma}_c)w_c}{\sum_{c=1}^{C} p(\boldsymbol{x}|\boldsymbol{\mu}_c,\boldsymbol{\Sigma}_c)w_c}, \end{aligned} \tag{10}$$

where $\boldsymbol{b}_c \in \mathbb{R}^n$ and $p(\boldsymbol{x}|\boldsymbol{\mu}_c,\boldsymbol{\Sigma}_c)$ is a multivariate Gaussian density probability calculated through Equation 9.

At the end of E-step, a likelihood function must be calculated to obtain the expected value with respect to the conditional distribution of $\mathcal{Z}$ given $\mathcal{X}$ using the current estimation $\boldsymbol{\theta}^{(t)}$. Thus, the likelihood is defined as $L(\boldsymbol{\theta};\mathcal{X},\mathcal{Z}) = p(\mathcal{X},\mathcal{Z}|\boldsymbol{\theta})$, and the expected value of its log-likelihood is calculate by means of an equivalent function Q, as follows:

$$\begin{aligned} Q(\boldsymbol{\theta}|\boldsymbol{\theta}^{(t)}) &= E_{\mathcal{Z}|\mathcal{X},\boldsymbol{\theta}^{(t)}}[\log L(\boldsymbol{\theta};\mathcal{X},\mathcal{Z})] \\ &= \sum_{i=1}^{n}\sum_{c=1}^{C} p(z_i=c|\boldsymbol{x}_i,\boldsymbol{\theta}^{(t)}) \log L(\boldsymbol{\theta}_c;\boldsymbol{x}_i,z_i) \\ &= \sum_{i=1}^{n}\sum_{c=1}^{C} {b_{ic}}^{(t)} \log L(\boldsymbol{\theta}_c;\boldsymbol{x}_i,z_i). \end{aligned} \tag{11}$$

The next stage of EM algorithm is the M-step, and it consists of maximizing Q with respect to $\boldsymbol{\theta}$, as follows:

$$\boldsymbol{\theta}^{(t+1)} = \arg\max_{\boldsymbol{\theta}} \Big\{ Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{(t)}) \}. \tag{12}$$

In practice, $\boldsymbol{\theta}$ is updated for each term, i.e., $\boldsymbol{w}$, $\boldsymbol{\mu}$, and $\boldsymbol{\Sigma}$ are re-estimated independently for each mixture component c:

$$w_c{}^{(t+1)} = \frac{1}{n} \sum_{i=1}^{n} \boldsymbol{b}_{ic}{}^{(t)}, \tag{13}$$

$$\boldsymbol{\mu}_c{}^{(t+1)} = \frac{\sum_{i=1}^{n} \boldsymbol{b}_{ic}{}^{(t)} \boldsymbol{x}_i}{\sum_{i=1}^{n} \boldsymbol{b}_{ic}{}^{(t)}}, \tag{14}$$

$$\boldsymbol{\Sigma}_c{}^{(t+1)} = \frac{\sum_{i=1}^{n} \boldsymbol{b}_{ic}{}^{(t)} (\boldsymbol{x}_i - \boldsymbol{\mu}_c{}^{(t+1)})(\boldsymbol{x}_i - \boldsymbol{\mu}_c{}^{(t+1)})^\top}{\sum_{i=1}^{n} \boldsymbol{b}_{ic}{}^{(t)}}. \tag{15}$$

The EM algorithm alternates between the estimation of $\boldsymbol{b}_c{}^{(t)}$ through E-step and the calculation of $w_c{}^{(t+1)}, \boldsymbol{\mu}_c{}^{(t+1)}$, and $\boldsymbol{\Sigma}_c{}^{(t+1)}$ given the values of $\boldsymbol{b}_c{}^{(t)}$ during the M-step (WEBB; COPSEY, 2011). This process is repeated until the log-likelihood is maximized, making the EM algorithm converge to a local or global solution.

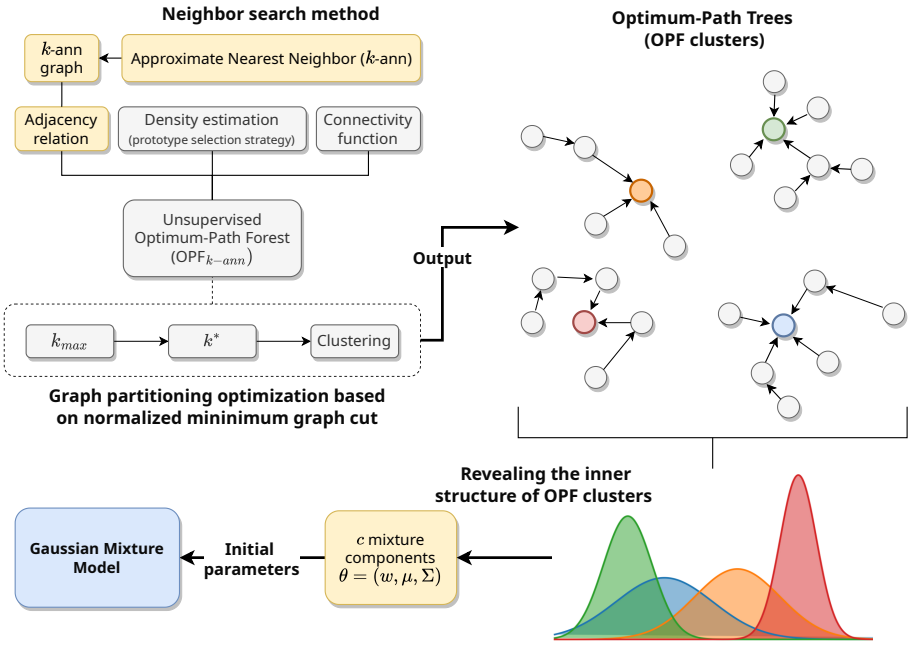
4.3 Proposed Approach

The maximum likelihood estimation calculated with the EM algorithm has its overall performance associated with starting values of the hyperparameter c (number of mixture components) and the model parameters $\boldsymbol{\theta} = (\boldsymbol{w}, \boldsymbol{\mu}, \boldsymbol{\Sigma})$ described in Section 4.2.2. Choosing c depends on many factors, including clusters' shape, separation, relative sizes, sample size, and data dimension (WEBB; COPSEY, 2011). Besides, such characteristics also impose a degree of influence over the clustering process considering different initialization values for $\boldsymbol{\theta}$. Therefore, the parameter estimation of Gaussian mixtures does not represent a trivial task. That said, this section describes an approach to initialize GMM parameters by exploiting different attributes of an alternative version of the unsupervised OPF algorithm. Figure 9 depicts the proposal, with its main contributions highlighted as yellow boxes.

One can observe that two main processes are conducted: (i) clustering samples with the unsupervised OPF using a k-ann graph to output the computed OPTs, and (ii) the unraveling of clusters' inner structure into distribution parameters that will be further assumed as initial estimations to initialize the GMM algorithm.

The main idea is to first cluster the data using OPF and then take the distributions from computed clusters to estimate w, $\boldsymbol{\mu}$, and $\boldsymbol{\Sigma}$ concerning each component (cluster) from the Gaussian mixture. Let $X_c \in \mathbb{R}^{m \times n_c}$ be the matrix of n_c concatenated samples

Figure 9 – A diagram synthesizing the proposed approach.



Source: Produced by the author.

from cluster c. The mean of a given cluster c is the average feature vector calculated for all samples within the c-th cluster, as follows:

$$\boldsymbol{\mu}_c = \frac{1}{n_c} \sum_{i=1}^{n_c} \boldsymbol{x}_i, \tag{16}$$

where $\boldsymbol{\mu}_c \in \mathbb{R}^m$ and $\boldsymbol{x}_i$ belongs to cluster c. Another way to represent $\boldsymbol{\mu}_c$ is to take the feature vector located at the center of its corresponding cluster. Usually, that is achieved by prototypes because these key nodes are approximately centered at high-density regions due to the pdf computation. This latter strategy was chosen in this work to set $\boldsymbol{\mu}_c$. Furthermore, the covariance matrix from cluster c is $\boldsymbol{\Sigma}_c \in \mathbb{R}^{m \times m}$, and is given by:

$$\boldsymbol{\Sigma}_c = \frac{1}{n_c - 1}(X_c - \boldsymbol{\mu}_c)(X_c - \boldsymbol{\mu}_c)^\top. \tag{17}$$

The initial contribution weight (w_c) is defined as the proportion between the number of samples from c-th cluster over the total number of samples from the dataset, as follows:

$$w_c = \frac{n_c}{n}. \tag{18}$$

Clustering algorithms like k-means have their clustering results directly related to the choice of the number of clusters and the initial centroids' position in the feature space. In contrast, unsupervised OPF only relies on the definition of a single hyperparameter: the number of neighbors (k) to construct the k-nn graph (adjacency relation). That allows the clustering model to estimate the number of clusters on-the-fly and is performed via a brute-force search for finding the best k based on a graph-cut optimization procedure, as already described in Section 4.2.1.

Still, a severe drawback associated with k-nn related techniques is the complexity in the search for nearest neighbors among n available training samples, which becomes critical in high-dimensional feature spaces and with large-scale datasets.That also constitutes a handicap for unsupervised OPF, since it relies on the choice of the adjacency parameter (k) by means of a graph cut optimization. Seeking to deal with high-dimensional data and large-scale datasets, Rocha, Cappabianco e Falcão (2009) suggested two strategies: (i) imposing a spatial constraint to the k-nn graph construction and (ii) optimizing k on a randomly generate small subset $\mathcal{X}' \subset \mathcal{X}$. Essentially, both alternatives try to narrow the k-nn search space favoring computational efficiency, but adding the possibility of degrading the quality of clustering results since not all samples from the original dataset are taken into account during the clusters computation. Yet, a brute-force search is employed in the two cases.

We propose an OPF architecture that employs a data structure for indexing and fast querying of data points, where any k-approximate nearest neighbors search technique could be adopted to perform such a task. The primary idea is to boost the k-nn graph construction, consequently accelerating the graph-cut optimization and the competition process to compute OPTs, especially when extensive datasets and large k-neighborhoods are considered. In that sense, employing a k-ann technique enhances OPF scalability.

Thus, the proposed approach offers a specific clustering structure from which the number of components (clusters) and the remaining GMM parameters ($\boldsymbol{\theta}$) could be calculated for boosting the initialization of the EM algorithm. Finally, for the sake of simplicity, the proposed approach and the conventional unsupervised OPF will be denoted as OPF$_{k\text{-ann}}$ and OPF, respectively.

4.4 Material and methods

4.4.1 Datasets and evaluation measure

The experiments have been conducted in eight datasets, from which six are general-purpose ones, and the two remaining are associated with collaborative filtering-based recommendation tasks. Table 17 presents the description of each dataset, focusing on the number of samples, features, and classes (when those are available).

To assess clustering performance regarding recognition quality, an intrinsic measure has been employed since some of the datasets have target labels unavailable and, considering the remaining labeled ones, not always the number of classes matches the best-computed clusters structure. In that sense, the Davies-Bouldin (DB) (DAVIES; BOULDIN, 1979) have been employed for intrinsic measurement of the results in this work. Roughly speaking, DB aims to capture both the separation and the compactness of the clusters by calculating the ratio of within-cluster scatter and between-cluster separation. Thus, a

Table 17 – Description of the datasets adopted in this work.

Dataset	# samples	# features	# classes
Breast Cancer (BCW) (DUA; GRAFF, 2019)	569	30	2
Blood Transfusion (Blood) (DUA; GRAFF, 2019)	748	4	2
Cervical Cancer (CCRF) (DUA; GRAFF, 2019)	858	32	2
Mammographic Mass (MM) (DUA; GRAFF, 2019)	961	5	2
Diabetic Retinopathy (Diabetic) (DUA; GRAFF, 2019)	1,151	19	2
Google Reviews (GR) (DUA; GRAFF, 2019)	5,456	24	–
Movielens-1M (ML1M) (HARPER; KONSTAN, 2015)	6,040	18	–
Anuran Calls (Frogs) (DUA; GRAFF, 2019)	7,195	22	–

Source: Produced by the author.

smaller DB index indicates a better clustering solution.

4.4.2 Experimental setup

The datasets have been pre-processed, such that all missing feature values have been replaced by their corresponding average ones. Additionally, the data has been normalized to have zero mean and unitary standard deviation. Then, the data have been split at random into 80/20 proportion training and validation sets, respectively. Notice that the validation set has the purpose of finding a suitable value for the hyperparameter $k_{\max}$, since it directly influences over the estimated number of clusters computed by OPF. As a result, OPF$_{k\text{-ann}}$ clusters the training data using the fine-tuned $k_{\max}$. The experimental routine has been replicated ten times for further statistical validation done through Wilcoxon signed-rank test. Its choice is justified since the data did not necessarily come from a Gaussian distribution and because it is suitable for paired tests of dependent samples.

4.5 Experimental Results

The experiments have been conducted in two-fold: (i) OPF$_{k\text{-ann}}$ is compared against OPF[2], focusing on computational cost and clustering quality (DB) measurements (Section 4.5.1); (ii) the proposed approach is evaluated against k-means regarding the initialization task of GMM parameters (Section 4.5.2). To evaluate OPF$_{k\text{-ann}}$ against other techniques in both scenarios, we employed k-ann search algorithms based on their relevance in the literature and technical robustness:

- ANNOY[3]: it is a tree-based algorithm that explores random projections to perform multiple partitioning of the feature space, thus producing a forest of search trees.
- KD-Tree: it is a space-partitioning binary search tree where data in each node is represented by a K-dimensional point in space.

[2] The OPFython library (ROSA; PAPA, 2021) was employed.
[3] Available at: <https://github.com/spotify/annoy>

- ❏ HNSW (MALKOV; YASHUNIN, 2018): it explores small-world graphs to partition the feature space into a hierarchy of several layers, such that a fast and reliable search for neighbors is allowed.

Notice we employed k-means, GMM and KD-Tree implementations from scikit-learn (PEDREGOSA et al., 2011). Finally, OPF$_{k\text{-ann}}$ has been implemented using Python 3.8, and all source codes and supplementary files regarding the conducted experiments are available at the project's GitHub repository[4].

4.5.1 OPF$_{k\text{-ann}}$ evaluation against OPF

As aforementioned, this section presents the OPF$_{k\text{-ann}}$ evaluation, comparing it against OPF concerning the average execution time. Such an information is depict in Figure 10, in which both OPF and OPF$_{k\text{-ann}}$ are represented by the Nearest Neighbors (NN) search algorithm employed to build their k-neighborhood. Overall, one can observe OPF$_{k\text{-ann}}$ versions (KD-Tree, ANNOY, and HNSW) are faster than OPF (k-nn) considering all datasets, which has been attested according to the Wilcoxon signed-rank test with 95% of confidence. Besides, the results evidenced that OPF$_{k\text{-ann}}$ is statistically similar regarding the execution time.

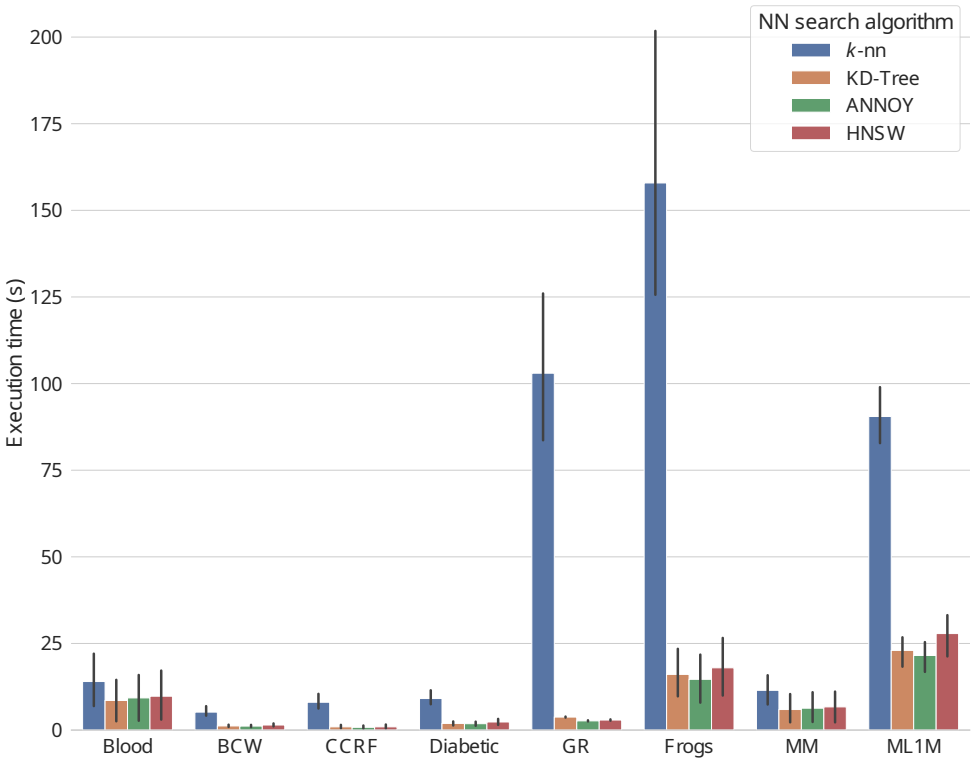
On average, for datasets with more than $1,500$ samples, OPF$_{k\text{-ann}}$ proved to be 15.5 times faster than OPF, while for smaller datasets ($|\mathcal{X}| < 1,500$), the former proved to be 4.2 times faster than the latter. Therefore, this evidence suggests that the computational efficiency gains of OPF$_{k\text{-ann}}$ over larger datasets tend to be higher, as seen in the results concerning GR and Frogs datasets. In other words, as data grows, an exhaustive search for neighbors becomes overly expensive to conduct during the graph-cut optimization of OPF.

Moreover, we can draw two important observations: (i) the high standard deviation values obtained in some datasets regardless of their size (Blood, Frogs, GR, and ML1M), and (ii) the relatively higher computational cost of OPF$_{k\text{-ann}}$ versions obtained on those mentioned datasets in contrast to their respective performances achieved on the remaining datasets, i.e., the ones having a small average standard deviation. From such observations, we presume that the most influential factor on the execution time of OPF$_{k\text{-ann}}$ is more related to the size of the optimized k-neighborhood than the dataset dimension, since for those datasets the average computed best $k \approx 28$.

Additionally, the clustering quality has been also evaluated. Since k-ann techniques assume a restricted number of errors is allowed during the NN search, it is intuitive to think that the above will impact the quality of the computed clusters. In that sense, a pairwise comparison between OPF and each OPF$_{k\text{-ann}}$ version was conducted to assess the average DB index considering all datasets. The corresponding results of such an evaluation

[4] Available at <https://github.com/guibmartins/opf_kann_gmm_initialization>

Figure 10 – Average execution time for OPF and OPF$_{k\text{-ann}}$ considering different nearest neighbors search algorithms.



Source: Produced by the author.

are presented in Table 18. For most results, the OPF$_{k\text{-ann}}$ versions demonstrated to be statistically equivalent to OPF, i.e., results highlighted in bold. However, when HNSW is employed, OPF$_{k\text{-ann}}$ performed better than OPF in CCRF and GR datasets.

Table 18 – Average DB index for OPF and OPF$_{k\text{-ann}}$ using different k-ann search algorithms. The best results (statistically similar) are denoted in bold according to the Wilcoxon signed-rank test with 95% of confidence.

Dataset	**OPF**	**OPF$_{k\text{-ann}}$**		
		KD-Tree	**ANNOY**	**HNSW**
Blood	1.3411 ± 0.34	$\mathbf{1.346 \pm 0.34}$	$\mathbf{1.351 \pm 0.33}$	$\mathbf{1.3311 \pm 0.2}$
BCW	0.7283 ± 0.17	$\mathbf{0.7283 \pm 0.17}$	$\mathbf{0.7208 \pm 0.18}$	$\mathbf{0.7101 \pm 0.16}$
CCRF	1.2132 ± 0.16	$\mathbf{1.2253 \pm 0.15}$	$\mathbf{1.2729 \pm 0.19}$	1.1294 ± 0.14
Diabetic	1.5394 ± 0.22	$\mathbf{1.5394 \pm 0.22}$	$\mathbf{1.4777 \pm 0.18}$	$\mathbf{1.4271 \pm 0.16}$
GR	0.8374 ± 0.01	$\mathbf{0.8374 \pm 0.01}$	$\mathbf{0.8313 \pm 0.01}$	0.8245 ± 0.01
Frogs	1.2498 ± 0.11	$\mathbf{1.2498 \pm 0.11}$	$\mathbf{1.2697 \pm 0.11}$	$\mathbf{1.2164 \pm 0.11}$
ML1M	1.2627 ± 0.15	$\mathbf{1.2627 \pm 0.15}$	$\mathbf{1.3204 \pm 0.16}$	$\mathbf{1.2424 \pm 0.13}$
MM	1.8428 ± 0.88	$\mathbf{2.2085 \pm 1.58}$	$\mathbf{1.9699 \pm 0.97}$	$\mathbf{1.5335 \pm 0.25}$

Source: Produced by the author.

4.5.2 Assessing OPF$_{k\text{-ann}}$ for the GMM initialization task

In these experiments, we evaluate the quality of the clusters computed with the GMM algorithm when it is initialized with the proposed approach, statistically comparing its results against those obtained using the k-means algorithm. Its worth noting that we set the number of clusters of k-means to be the one computed with OPF$_{k\text{-ann}}$, thus allowing a fairer assessment of the GMM clustering results. Table 19 presents such results with respect to the average DB index considering all datasets employed in this work. Note that now the goal is to verify whether there is a significant difference between the paired samples, such that the proposed approach (OPF$_{k\text{-ann}}$) indicates superior performance against the baseline (k-means). Hence, the best-achieved results are denoted in bold.

Table 19 – Average DB index achieved by GMM algorithm employing different initialization methods: k-means and OPF$_{k\text{-ann}}$. The best results are denoted in bold according to the Wilcoxon signed-rank test with 95% of confidence.

Dataset	**k-ann search algorithm**	**k-means**	**OPF$_{k\text{-ann}}$**
Blood	KD-Tree	1.5073 ± 0.36	$\mathbf{1.346 \pm 0.34}$
	ANNOY	1.5027 ± 0.35	$\mathbf{1.351 \pm 0.33}$
	HNSW	1.3616 ± 0.18	$\mathbf{1.3311 \pm 0.2}$
BCW	KD-Tree	1.6961 ± 0.07	$\mathbf{0.7283 \pm 0.17}$
	ANNOY	1.7063 ± 0.07	$\mathbf{0.7208 \pm 0.18}$
	HNSW	1.6828 ± 0.06	$\mathbf{0.7101 \pm 0.16}$
CCRF	KD-Tree	1.3933 ± 0.22	$\mathbf{1.2253 \pm 0.15}$
	ANNOY	1.4002 ± 0.26	$\mathbf{1.2729 \pm 0.19}$
	HNSW	1.5365 ± 0.33	$\mathbf{1.1294 \pm 0.14}$
Diabetic	KD-Tree	1.851 ± 0.42	$\mathbf{1.5394 \pm 0.22}$
	ANNOY	1.9404 ± 0.52	$\mathbf{1.4777 \pm 0.18}$
	HNSW	1.9752 ± 0.43	$\mathbf{1.4271 \pm 0.16}$
GR	KD-Tree	0.953 ± 0.01	$\mathbf{0.8374 \pm 0.01}$
	ANNOY	0.9466 ± 0.01	$\mathbf{0.8313 \pm 0.01}$
	HNSW	0.9556 ± 0.01	$\mathbf{0.8245 \pm 0.01}$
Frogs	KD-Tree	1.6863 ± 0.24	$\mathbf{1.2498 \pm 0.11}$
	ANNOY	1.6829 ± 0.22	$\mathbf{1.2697 \pm 0.11}$
	HNSW	1.6904 ± 0.22	$\mathbf{1.2164 \pm 0.11}$
MM	KD-Tree	$\mathbf{1.7694 \pm 0.71}$	2.2085 ± 1.58
	ANNOY	$\mathbf{1.8576 \pm 0.81}$	1.9699 ± 0.97
	HNSW	2.8646 ± 2.28	$\mathbf{1.5335 \pm 0.25}$
ML1M	KD-Tree	2.0468 ± 0.38	$\mathbf{1.2627 \pm 0.15}$
	ANNOY	2.1637 ± 0.4	$\mathbf{1.3204 \pm 0.16}$
	HNSW	2.0203 ± 0.37	$\mathbf{1.2424 \pm 0.13}$

Source: Produced by the author.

According to the results, one can verify that the GMM initialized with OPF$_{k\text{-ann}}$ approach obtained the lowest DB index in seven out of eight datasets. For the remaining

ones, OPF$_{k\text{-ann}}$ presented statistical equivalence to k-means when performing GMM initialization task, according to the Wilcoxon signed-rank test with 95% confidence. Such results indicate OPF$_{k\text{-ann}}$ can consistently identify Gaussian centers, generating reliable initial estimations for the EM algorithm, thus optimizing both intra-cluster scatter and between-cluster separation regarding clusters computed through GMM.

4.6 Conclusions

This work proposed a fast and scalable unsupervised OPF based on k-ann graph construction to cope with the initialization task of GMM parameters. It estimates the number of clusters on-the-fly and encodes Gaussian parameters more naturally and intuitively than other clustering algorithms like k-means, presenting itself as a viable alternative for the clustering task.

The experiments indicated that OPF$_{k\text{-ann}}$ is more scalable than OPF, demonstrating viability for handling extensive datasets. It can drastically reduce the computation cost regarding the graph construction and the partitioning optimization when the k-neighborhood goes large.

Chapter 5

OPF_{SNN}: A Novel Optimum-Path Forest Clustering Approach based on Shared Near Neighbors for Collaborative Filtering Recommendation

This chapter presents an adaptation of the work submitted to the Neurocomputing journal which is under review[1]. The scope of this work is to introduce a novel adjacency relation based on the concept of shared neighborhoods to the unsupervised OPF algorithm, seeking to alleviate sparsity and high dimensionality data issues particularly related to collaborative filtering recommendation.

5.1 Introduction

The recent and constant advances in computational power and the ability to efficiently process large volumes of data have demonstrated the great potential of machine learning to shape the reality we live in terms of how we deal with data and exploit it to improve people's lives. In this context, RS has been an important tool to reduce users effort in the search for content, products, and services on the web, remaining as one of the most popular and necessary machine learning technologies to deal with currently computational

[1] A preprint of this research is available at SSRN: <https://dx.doi.org/10.2139/ssrn.4531711>.

needs. RS have the goal of collecting information on the preferences of its users for a given set of items like movies, songs, books, textual reviews and comments, websites and other sources of interaction (BOBADILLA et al., 2013).

There are different ways of thinking strategies to build recommender systems. The majority of researches lies within a traditional taxonomy based on the kind of data a RS may process, e.g., inner content from images, videos and text, interaction data like user ratings on items, or a combination of them. One type of recommender system that explores interaction data is CF and it prevails as the most important and influential technique in the field – since it is the most successful and adopted one in both industry and academy (MARTINS; PAPA; ADELI, 2020) over the last decades. The main premise of such a technique is that if two users, A and B, interact similarly with the same set of items, it is more likely user A will present a similar interaction pattern regarding another item that user B has already interacted with, rather than a randomly chosen user C. Hence, through collaboration among users, one may be able to estimate the interest level a individual user may have on a item or a set of items. As such, CF could be defined as the technique of recommending content to a target user exploring only explicit response data (user ratings on items) or implicit interactions (clicks, time spent on web pages, etc) from other users with similar interaction patterns.

Although a "pure" CF-based approach can deliver robust and personalized recommendations, it is highly dependent on interaction data which is very scarce due to the fact users rate only a very restricted set of items (LIANG et al., 2018b). Consequently, CF datasets tend to be highly sparse, i.e, there are very few user-item interactions in relation to the total number of users and items available in the data corpus. In other words, the high data sparsity is a common problem that affects the quality of CF-based recommenders.

There is a large number of researches in the literature to mitigate the effects of data sparsity in CF-based systems. The most relevant strategies to cope with this challenge involve, for instance, exploring external data sources or side information (DUAN; JIANG; JAIN, 2022; CHEN et al., 2021), modeling the user-item rating matrix by means of dimensionality reduction techniques (LAVANYA; BHARATHI, 2021; ZHANG et al., 2021), exploring deep learning architectures (ALHARBE; RAKROUKI; ALJOHANI, 2023; DO; NGUYEN, 2022; LIANG et al., 2018b), and feature space partitioning through unsupervised learning (POUDEL; BIKDASH, 2022; LI; WEN; CHEN, 2021; CHEN et al., 2020b).

With respect to the last strategy, the key benefit of clustering-based CF techniques lies in the increase of computational efficiency at recommendation stage, since the data points gathered during computation – which is performed in an offline stage – will be only those within the cluster the target user (item) has been assigned for. In other words, clustering-based techniques traditionally perform recommendations based on the k-closest

samples to the target item within the same cluster (AGGARWAL, 2016b).

Still, there are also some drawbacks related to its adoption. First, the efficiency improvement usually comes with some loss in recognition rates. Second, the incomplete nature of CF data reflects over the feature vector sparseness, whose dimensions represent the available and/or rated items in the dataset; that is the most common way of representing samples in clustering-based CF according to the literature, and it causes a sensitive influence over the computed clusters quality. At last, the majority of clustering-based CF techniques require setting the number of clusters *a priori*, condition which may not be suitable in recommendation scenarios since the process of clustering CF data is not considered a trivial task.

The second issue could be more severe in situations where the number of users and items rapidly grows, causing the samples to be high dimensional, especially when the Euclidean distance is used to measure the dissimilarity between samples. To face this problem, Jarvis e Patrick (1973) explored the use of similarity measures in shared neighborhoods for clustering algorithms, demonstrating their effectiveness in minimizing the negative effects of both high dimensionality and sparsity in the feature space.

Regarding the first issue, different clustering algorithms are found in the literature that compute the number of clusters indirectly. Among them, the Optimum-Path Forest (OPF) (PAPA; FALCÃO; SUZUKI, 2009; PAPA et al., 2012; ROCHA; CAPPABIANCO; FALCÃO, 2009) has received attention in recent years, and involves a growing number of researches in unsupervised learning with applications in several areas such as image segmentation (CHEN et al., 2018c; CAPPABIANCO et al., 2012), intrusion detection (COSTA et al., 2015), video summarization (MARTINS et al., 2020), sampling (PASSOS et al., 2022), and Gaussian parameters initialization (ROSA et al., 2014; MARTINS; PAPA, 2022b), to name a few.

In summary, OPF is a graph-based machine learning algorithm, which interprets the data as a graph structure and can be manipulated to perform supervised, semi-supervised and unsupervised tasks. The data points or samples are represented by graph nodes and are organized according to a predefined adjacency relation, from which some nodes are selected as the representatives ones (i.e., the prototypes) and further compete among themselves to conquer the other nodes based on some path-cost function. The unsupervised OPF version – focused on clustering tasks – sets a k-Nearest Neighbors (KNN) graph as adjacency relation and defines the prototype nodes as those approximately located at the center of regions of higher concentration according to a probability density function. Furthermore, during the learning stage, the clusters are computed iteratively and their number is estimated/defined at runtime. Such characteristics seem to fit the data structure in collaborative filtering scenarios, where it is not possible to trivially decide the "correct" number of clusters. Under such circumstances, computing the number of clusters on-the-fly based on the k-neighborhood size as the starting point seems to be

a more "natural" solution when performing such a task.

Although these key strength characteristics, OPF may show some limitations when dealing with CF-based data. Such classifier has been initially designed for addressing computer vision tasks like image segmentation. That means the particularities of such application and their data may not be the same for other scenarios where OPF clustering could be employed. CF-based data, for instance, usually can achieve high dimensionality with easy, contains noise and outliers, and is naturally sparse. Also, in terms of data distribution, those kind of data contain frequent low and medium density representing uniform regions which could be overlooked by OPF during clustering.

In that sense, employing a KNN graph as OPF's adjacency relation could not suits well for CF-related tasks and thus a Shared Near Neighbor (SNN) strategy could be explored instead. SNN is able to mitigate – to some extent – the effects of outliers, varying densities, high dimensionality, and sparsity without making OPF lose the capability of generating clusters with different sizes and shapes. Also, it handles low similarities (or distance) between samples than KNN, making it a valuable alternative in the context of collaborative filtering where the sparseness of feature vectors can often result such a challenge.

That being said, the goal of this work is to introduce a SNN graph as the adjacency relation of the unsupervised OPF algorithm and fit it to perform clustering-based CF tasks. The idea is that the combination of clustering by OPF and SNN could be an alternative for dealing effectively with both sparsity and high dimensionality challenges, thus reducing their degrading effects over the quality of CF-based recommendations.

The remainder of this paper is organized as follows. In Section 5.2, the main background concepts and definitions regarding collaborative filtering, optimum-path forest and shared nearest neighbors are described. In Section 5.3, the proposed Optimum-Path Forest clustering algorithm based on Shared Near Neighbor graph is presented. The conducted experiments concerning clustering and recommendation tasks are detailed and discussed in Sections 5.4 and 5.5, respectively, while Section 5.6 presents the final considerations about the work and directions for future research.

5.2 Background

5.2.1 Collaborative Filtering

Since we have already defined collaborative filtering in Section 5.1, here we will stick to briefly describing its overall categories, as well as the main strategies of each modality, with a focus on the clustering-based ones.

In summary, CF-based recommenders are classified into two groups: model- and memory-based (BREESE; HECKERMAN; KADIE, 2013). Roughly speaking, the for-

mer employs machine learning by mapping the historical preferences of users into a low-dimensional feature space from which recommendations are made. The latter exploits the history of observed user-item interactions by calculating the similarity among users who evaluated common items. Analogously, a set of items evaluated by common users could be taken for similarity calculation. Recommendations are generated through aggregation functions and aim to estimate unobserved interactions directly from the set of users (items) with greater proximity to a given target user (item).

To calculate similarity, functions such as cosine, Pearson correlation, mean squared difference and Jaccard index are usually employed. To estimate unobserved interactions, simple average, weighted average, and mean-centered weighted average are common examples of aggregation functions, the latter being the most commonly adopted. Let $\mathcal{U}$ and $\mathcal{I}$ be the sets of users and items, respectively, and one may want to estimate an unobserved interaction p_{ui} between user $u \in \mathcal{U}$ and item $i \in \mathcal{I}$. In collaborative filtering, the mean-centered weighted average is defined as follows:

$$p_{ui} = \mu_u + \frac{\sum_{v \in \mathcal{V}(i)} \mathbb{W}(u,v) * (r_{vi} - \mu_v)}{\sum_{v \in \mathcal{V}(i)} \mid \mathbb{W}(u,v) \mid}, \tag{19}$$

where μ_u is the average of known ratings given by user u, $\mathcal{V}(i) \subset \mathcal{U}$ is the subset of users v that have previously rated item i, r_{vi} is the rating a user v gave to item i, and the similarity between users u and v is calculated through function $\mathbb{W}(u,v)$.

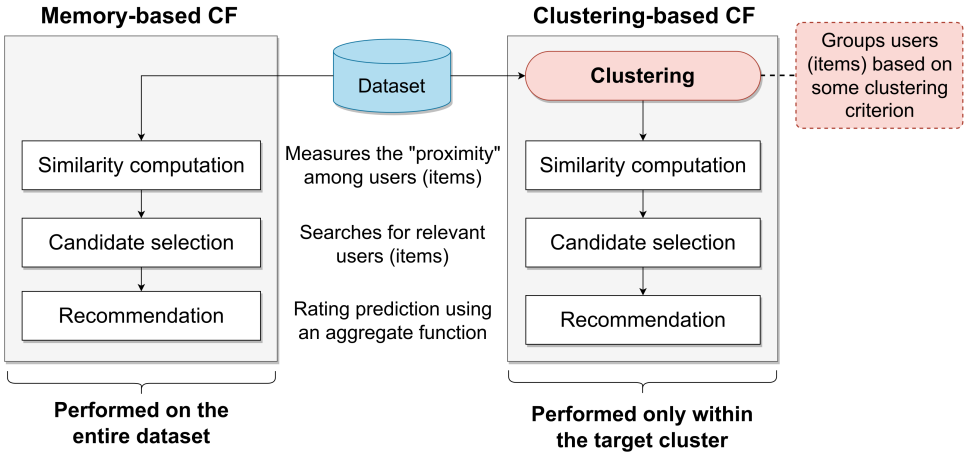
Memory-based CF have its most influential approaches the ones that exploit nearest neighbors (NN) algorithms, such as traditional user- and item-based approaches. The former sets the k-neighborhood of a target user from the rows (users) of the rating matrix of size $\mid \mathcal{U} \mid \times \mid \mathcal{I} \mid$, while the latter is based on the columns (items) of it. Roughly speaking, from the k-neighborhood of a user the recommendations for unseen items are estimated.

On the other hand, model-based CF consists of approaches that generate recommendations from a computed model of user-item interactions – i.e., a rating matrix. In fact, the principal approaches are those based on Matrix Factorization (MF) (YEHUDA; ROBERT; CHRIS, 2009; SALAKHUTDINOV; MNIH, 2009; HIEN et al., 2015), which aim at reducing the dimensionality of the rating matrix by mapping users and items to a latent space, i.e., through some optimization algorithm. From the modeled latent space of users and items – a compact and dense representation of the original feature space, the unobserved interactions will be predicted. In general, MF-based approaches present better scalability and calculate recommendations efficiently when compared to memory-based approaches.

Regarding boosting efficiency and scalability, clustering is also employed in collaborative filtering. It can be explored with both memory- and model-based approaches, and generally are interpreted as a pre-processing stage in CF process. To illustrate such a situ-

ation, two process are depicted in Figure 11: a memory-based CF and one complemented with clustering.

Figure 11 – The process of recommending through memory-based on the left and one employing clustering as a pre-processing stage on the right.



Source: Produced by the author.

As one can see in Figure 11, recommending through memory-based CF can be organized in three basic components: (i) similarity calculation to define proximity among users (items), (ii) candidate selection of the k-most similar users (items), and (iii) employment of an aggregation function to further predict the top-N recommendations. In contrast, when clustering is adopted the set of k common users (items) is not defined directly on similarity computation, but is first set on the cluster (or clusters) the target user (item) has been assigned for. With such a strategy, the CF process may reduce the computational cost during candidate selection and influences rating prediction to be conducted in a reduced search space, i.e., only users (items) grouped together at the same cluster will be explored to generate recommendations.

5.2.2 Optimum-Path Forest Clustering

The fundamental problem in data clustering is to identify natural groups (if they exist) in some dataset. Let $\mathcal{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_n\}$ be an unlabeled dataset such that $\boldsymbol{x}_i \in \mathbb{R}^m$ denotes the i-th sample given by an m-dimensional feature vector. Also, consider a graph $\mathcal{G} = (X, A)$ is defined such that $\mathcal{A}$ is a given adjacency relation and the arcs $(\boldsymbol{x}_i, \boldsymbol{x}_j) \in \mathcal{A}$ connect k-nearest neighbors in the feature space, $i \neq j$.

The Optimum-Path Forest framework constitutes a three-stage process for partitioning $\mathcal{G}$ into subgraphs called Optimum-Path Trees (OPT), each representing the structure of one different cluster. OPF follows a competition-fashion strategy lead by a subset of nodes marked as prototypes, which will compete among themselves to conquer the remaining nodes in a connectivity-maximization scheme. In that sense, OPF relies on (i)

an adjacency relation for initially connecting graph nodes, (ii) the definition of a prototype selection strategy and (iii) a connectivity function to rule the competition-based learning process.

Initially, graph $\mathcal{G}$ has its nodes connected by a well-defined adjacency relation, which we defined as $\mathcal{A}_1$ and is based on a k-neighborhood for the unsupervised OPF version. Therefore, an undirected k-NN graph is computed, whose arcs are weighted by $d(\boldsymbol{x}_i, \boldsymbol{x}_j)$, i.e., the distance between those nodes, and relies on some distance function. In practice, the adjacency relation sets initial connections among the nodes of $\mathcal{G}$.

In a second stage, the construction of OPTs is initiated by weighting the nodes and further selecting the most representative ones. These are called prototypes and are usually located at the center of the densest regions on their respective k-neighborhood (ROSA et al., 2014). To fulfill that purpose, all nodes are weighted by a density value $\rho(\boldsymbol{x}_i)$ computed through a Gaussian probability density function (pdf), given by:

$$\rho(\boldsymbol{x}_i) = \frac{1}{\sqrt{2\pi\sigma^2} \mid \mathcal{A}(\boldsymbol{x}_i) \mid} \sum_{\forall \boldsymbol{x}_j \in \mathcal{A}(\boldsymbol{x}_i)} \exp\left\{\frac{-d^2(\boldsymbol{x}_i, \boldsymbol{x}_j)}{2\sigma^2}\right\}, \tag{20}$$

where $\mid \mathcal{A}(\boldsymbol{x}_i) \mid = k$, $\sigma = \frac{d_f}{3}$, and d_f is the maximum arc weight in (X, A). This parameter choice considers all nodes for density computation, since a Gaussian function covers most samples within $d(\boldsymbol{x}_i, \boldsymbol{x}_j) \in [0, 3\sigma]$.

By taking the k-nearest neighbors of each node, OPF handles different concentrations and reduces the scale problem to the one of finding the optimum value of k within $[1, k_{\max}]$, for $1 \leq k_{\max} \leq \mid \mathcal{X} \mid$. The solution provided by Rocha, Cappabianco e Falcão (2009) considers the minimum graph cut provided by the clustering results for $k \in [1, k_{\max}]$ according to a measure on normalized graph cut suggested by Shi e Malik (2000).

The third stage refers to the competition process lead by prototypes to conquer the remaining nodes. The result is the partition of $\mathcal{G}$ into multiple OPTs, which are rooted by a single prototype and will constitute the final clusters consisting of their most strongly connected nodes (samples). The OPF clustering objective is to compute OPTs by maximizing a cost function, which is established in terms of paths on $\mathcal{G}$ and defined as an acyclic sequence of adjacent nodes in $\mathcal{A}$.

Let a path $\pi_{\boldsymbol{x}_j}$ be a sequence of adjacent samples starting from a root set $\mathcal{R}$, i.e., the set containing only prototype nodes, and ending at a sample $\boldsymbol{x}_j$. Also, let $\pi_{\boldsymbol{x}_j} = \langle \boldsymbol{x}_j \rangle$ be a trivial path and $\pi_{\boldsymbol{x}_i} \cdot \langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle$ the concatenation of $\pi_{\boldsymbol{x}_i}$ and the arc $(\boldsymbol{x}_i, \boldsymbol{x}_j)$. Given some connectivity function $f(\cdot)$, a path $\pi_{\boldsymbol{x}_j}$ is optimal only if $f(\pi_{\boldsymbol{x}_j}) > f(\tau_{\boldsymbol{x}_j})$, where $\tau_{\boldsymbol{x}_j}$ represents any other possible path ending at sample $\boldsymbol{x}_j$. Among all possible paths $\pi_{\boldsymbol{x}_j}$ with roots on the pdf maxima, unsupervised OPF assigns to $\boldsymbol{x}_j$ a path whose minimum density value along it is maximum. Each pdf maximum should then define an influence zone – i.e., a cluster – by selecting the samples that are the most strongly connected to

it than to any other maximum. Formally, it finds a connectivity map $\mathcal{V}(\boldsymbol{x}_j)$, such that

$$\mathcal{V}(\boldsymbol{x}_j) = \max_{\forall \pi_{\boldsymbol{x}_j} \in (\mathcal{X}, \mathcal{A})} \{f(\pi_{\boldsymbol{x}_j})\}. \tag{21}$$

The objective is to maximize $f(\pi_{\boldsymbol{x}_j})$ for all $\boldsymbol{x}_j \in \mathcal{X}$, where

$$\begin{aligned} f(\langle \boldsymbol{x}_j \rangle) &= \begin{cases} \rho(\boldsymbol{x}_j) & \text{if } \boldsymbol{x}_j \in \mathcal{R} \\ \rho(\boldsymbol{x}_j) - \delta & \text{otherwise} \end{cases} \\ f(\pi_{\boldsymbol{x}_i} \cdot \langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle) &= \min\{f(\pi_{\boldsymbol{x}_i}), \rho(\boldsymbol{x}_j)\}, \end{aligned} \tag{22}$$

for $\delta = \min_{\forall (\boldsymbol{x}_i, \boldsymbol{x}_j) \in \mathcal{A} | \rho(\boldsymbol{x}_j) \neq \rho(\boldsymbol{x}_i)} \mid \rho(\boldsymbol{x}_j) - \rho(\boldsymbol{x}_i) \mid$ and $\mathcal{R}$ being a root set containing one node for each pdf maximum and being discovered on-the-fly. It is worth mentioning that δ acts as a handicap and filtering parameter for the pdf, where higher values reduce either the number of maxima and the number of clusters. Generally, it is set to $\delta = 1.0$ and $\rho(\boldsymbol{x}_j)$ is scaled within an interval of real numbers, i.e., $\rho(\boldsymbol{x}_j) \in [1, 1000]$. These settings have been adopted in this work.

Therefore, the unsupervised OPF algorithm maximizes the connectivity map $\mathcal{V}(\boldsymbol{x}_j)$ such that the optimum paths form an optimum-path forest, i.e., an acyclic predecessor map $\mathcal{P}$ that assigns to each sample $\boldsymbol{x}_j \notin \mathcal{R}$ its predecessor $\mathcal{P}(\boldsymbol{x}_j)$ in the optimum path from $\mathcal{R}$ or a marker *nil* when $\boldsymbol{x}_j \in \mathcal{R}$. Each prototype will be the root of an OPT, having its own root label $\mathcal{L}(\boldsymbol{x}_j)$ propagated to all other samples $\boldsymbol{x}_j$ within the same optimum-path tree. Ultimately, the collection of all OPTs creates the optimum-path forest. Algorithm 1 described the OPF clustering working mechanism.

In Lines $1-4$, maps $\mathcal{P}$ and $\mathcal{V}$ are initialized and all samples are inserted in a priority queue $\mathcal{Q}$. At each iteration of the main loop (Lines $5-17$), an optimum path $\mathcal{P}^*(\boldsymbol{x}_i)$ with value $\mathcal{V}(\boldsymbol{x}_i)$ is obtained in $\mathcal{P}$ when its last sample $\boldsymbol{x}_i$ is removed from $\mathcal{Q}$ (Line 6). Possible ties are broken in $\mathcal{Q}$ using first-in-first-out (FIFO) policy, i.e., when two optimum paths reach an ambiguous sample $\boldsymbol{x}_i$ with the same maximum value, $\boldsymbol{x}_i$ will be assigned to the first path that reached it. In Line 7, the conditional test $\mathcal{P}(\boldsymbol{x}_i) = nil$ identifies $\mathcal{P}^*(\boldsymbol{x}_i)$ as a trivial path $\langle \boldsymbol{x}_i \rangle$. Given that the optimum paths are found in a non-increasing order of values, trivial paths thus indicate samples in the maxima. As indicated in Line 8, by changing $\mathcal{V}(\boldsymbol{x}_i)$ to $\rho(\boldsymbol{x}_i)$ (Equation 22), the first sample in each maximum is forced to conquer the remaining samples in that maximum. Thus, $\boldsymbol{x}_i \in \mathcal{R}$ becomes root of the forest and a distinct label l is assigned to it (Line 8). Lines $10^{\smile}14$ evaluate if the path that reaches through $\boldsymbol{x}_i$ an adjacent sample $\boldsymbol{x}_j$ is better than the current one ending at node $\boldsymbol{x}_j$. Then, $\mathcal{L}$, $\mathcal{Q}$, $\mathcal{P}$, and $\mathcal{V}$ are updated accordingly. Note that in Line 10, the condition $\mathcal{V}(\boldsymbol{x}_j) < \mathcal{V}(\boldsymbol{x}_i)$ makes the algorithm avoid evaluating adjacent nodes already removed from $\mathcal{Q}$. Finally, the clustering ends when all samples $\boldsymbol{x}_i$ have been evaluated and removed from $\mathcal{Q}$.

Algorithm 1 Clustering by Optimum-Path Forest

Input: Graph $(\mathcal{X}, \mathcal{A})$ and function ρ.
Output: Path-value map $\mathcal{V}$, predecessor map $\mathcal{P}$, and label map $\mathcal{L}$.
Auxiliary: Priority queue $\mathcal{Q}$, variables tmp, and $l \leftarrow 1$.

for each $\boldsymbol{x}_j \in \mathcal{X}$ **do**
  $\mathcal{P}(\boldsymbol{x}_j) \leftarrow nil$, $\mathcal{V}(\boldsymbol{x}_j) \leftarrow \rho(\boldsymbol{x}_j) - \delta$.
  Insert $\boldsymbol{x}_j$ in $\mathcal{Q}$.
end for
while $\mathcal{Q}$ is not empty **do**
  Remove from $\mathcal{Q}$ a sample $\boldsymbol{x}_i$ such that $\mathcal{V}(\boldsymbol{x}_i)$ is maximum.
  if $\mathcal{P}(\boldsymbol{x}_i) = nil$ **then**
    $\mathcal{L}(\boldsymbol{x}_i) \leftarrow l$, $l \leftarrow l + 1$, $\mathcal{V}(\boldsymbol{x}_i) \leftarrow \rho(\boldsymbol{x}_i)$.
  end if
  for each $\boldsymbol{x}_j \in \mathcal{A}(\boldsymbol{x}_i)$ such that $\mathcal{V}(\boldsymbol{x}_j) < \mathcal{V}(\boldsymbol{x}_i)$ **do**
    $tmp \leftarrow \min\{\mathcal{V}(\boldsymbol{x}_i), \rho(\boldsymbol{x}_j)\}$.
    if $tmp > \mathcal{V}(\boldsymbol{x}_j)$ **then**
      $\mathcal{L}(\boldsymbol{x}_j) \leftarrow \mathcal{L}(\boldsymbol{x}_i)$, $\mathcal{P}(\boldsymbol{x}_j) \leftarrow \boldsymbol{x}_i$, $\mathcal{V}(\boldsymbol{x}_j) \leftarrow tmp$.
      Update position of $\boldsymbol{x}_j$ in $\mathcal{Q}$.
    end if
  end for
end while

5.3 Optimum-Path Forest Clustering based on Shared Near Neighbor Graph

In this approach, data samples are modeled as graph nodes, whose arcs connect the k-neighbors in the feature space. We then derive from this structure a new connection between the arcs, which we will be defined here as the one consisting of the k-shared nearest neighbors. As in traditional OPF clustering, the graph nodes are weighted by probability density values based on a Gaussian kernel (ROCHA; CAPPABIANCO; FALCÃO, 2009).

5.3.1 Theoretical foundation

In clustering tasks, the central problem is to identify groups of samples in $\mathcal{X}$, where samples in the same group should exhibit some degree of similarity according to a given semantic meaning. We say a sample $\boldsymbol{x}_j$ is adjacent to another sample $\boldsymbol{x}_i$, i.e., $(\boldsymbol{x}_i, \boldsymbol{x}_j) \in \mathcal{A}$, when an adjacency relation is satisfied. For instance,

$$(\boldsymbol{x}_i, \boldsymbol{x}_j) \in \mathcal{A}_1 \text{ if}$$
$$\boldsymbol{x}_j \text{ is } k\text{-nearest-neighbor of } \boldsymbol{x}_i, \tag{23}$$

where $k \geq 1$ is an integer parameter. Thus, the pair $(\mathcal{X}, \mathcal{A})$ defines a k-NN graph where $\mathcal{A}$ is an $\mathcal{A}_1$-type adjacency relation.

Still, let $\mathcal{A}_2$ be an adjacency relation derived from $\mathcal{A}_1$ and being based on the following assumptions (JARVIS; PATRICK, 1973):

i. When $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are similar such samples tend to unity.

ii. Samples are similar to the extent that they share the same nearest neighbors and are assumed to be part of their respective k-neighborhoods.

Hence, we say that a sample $\boldsymbol{x}_j$ belongs to the k-shared neighborhood of $\boldsymbol{x}_i$ when

$$\begin{gathered} \text{if } \boldsymbol{x}_j \in \mathcal{A}_1(\boldsymbol{x}_i) \text{ and} \\ \boldsymbol{x}_i \in \mathcal{A}_1(\boldsymbol{x}_j) \text{ , then} \\ (\boldsymbol{x}_i, \boldsymbol{x}_j) \in \mathcal{A}_2. \end{gathered} \tag{24}$$

The pair $(\mathcal{X}, \mathcal{A}_2)$ defines a k-shared-near-neighbors (k-SNN) graph, where $\mathcal{A}_2$ is the adjacency relation. However, $\mathcal{A}_2$ is asymmetric and one may achieve such a graph symmetric on the density plateaus employing $\mathcal{A}_3$, as proposed by Rocha, Cappabianco e Falcão (2009):

$$\begin{gathered} \text{if } \boldsymbol{x}_j \in \mathcal{A}_2(\boldsymbol{x}_i), \\ \boldsymbol{x}_i \notin \mathcal{A}_2(\boldsymbol{x}_j) \text{ and} \\ \rho(\boldsymbol{x}_i) = \rho(\boldsymbol{x}_j), \text{ then} \\ \mathcal{A}_3(\boldsymbol{x}_j) \longleftarrow \mathcal{A}_2(\boldsymbol{x}_j) \cup \{\boldsymbol{x}_i\}. \end{gathered} \tag{25}$$

Thus, using $\mathcal{A}_3$ ensures only one representative per maximum is required.

By introducing $\mathcal{A}_3$, we wish to take advantage of some features enabled by shared neighborhoods, such as the ability to prevent a group of samples with high density from being combined with relatively isolated samples, and to favor the formation of non-spherical clusters. Regarding the latter, OPF already present such an attribute, therefore by adopting $\mathcal{A}_3$ will not make OPFSNN fail to compute clusters with distinct shapes and concentrations, which is suitable for data in a recommendation context. Moreover, with $\mathcal{A}_3$ scale changes are handled automatically in the feature space.

5.3.1.1 SNN-based distance measure

As first introduced by Jarvis e Patrick (1973), a similarity measure can be devised from common neighbors. Given that OPF has weighted arcs connecting its nodes computed by any distance measure, one can employ a SNN-based similarity measure to do such a task, for instance:

$$d(\boldsymbol{x}_i, \boldsymbol{x}_j) = 1 - \frac{\text{SNN}_k(\boldsymbol{x}_i, \boldsymbol{x}_j)}{k}, \tag{26}$$

where $\text{SNN}_k(\boldsymbol{x}_i, \boldsymbol{x}_j)$ stands as the number of neighbors shared by both $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, and k (number of neighbors) is used to scale the similarity between $[0, 1]$.

In this work, a SNN distance based on Jaccard index has been chosen, as proposed by Ertöz, Steinbach e Kumar (2003), and is defined as follows:

$$\text{SNN}_k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \frac{\mid \Gamma(\boldsymbol{x}_i) \cap \Gamma(\boldsymbol{x}_j) \mid + c}{\mid \Gamma(\boldsymbol{x}_i) \cup \Gamma(\boldsymbol{x}_j) \mid + c}, \tag{27}$$

where $\Gamma(\cdot)$ is the set of nearest neighbors of a data sample and c is a constant we added to control the inclusion of a sample within its own k-neighborhood[2]. Equation 27 has been employed because it performs better than the Euclidean distance in high-dimensional spaces, at least when the data is sparse, such as for textual documents and collaborative filtering.

Finally, once the adjacency relation and the distance function are defined, the remaining steps of OPFSNN are conducted as the conventional unsupervised OPF.

5.3.2 OPFSNN for CF-based recommendation

OPFSNN can now be employed in collaborative filtering and Figure 12 depicts the proposed approach, which follows a clustering-based CF setup for recommendation as described in Section 5.2.1. First, users, items, and their interactions are organized in a rating matrix, which is further splitted row-wise into training and test sets. Note that the rating matrix rows are sparse feature vector representations of users, whose dimensions are rated items. During the clustering phase, training users are clustered by OPFSNN, which further outputs an OPF-based model as result.

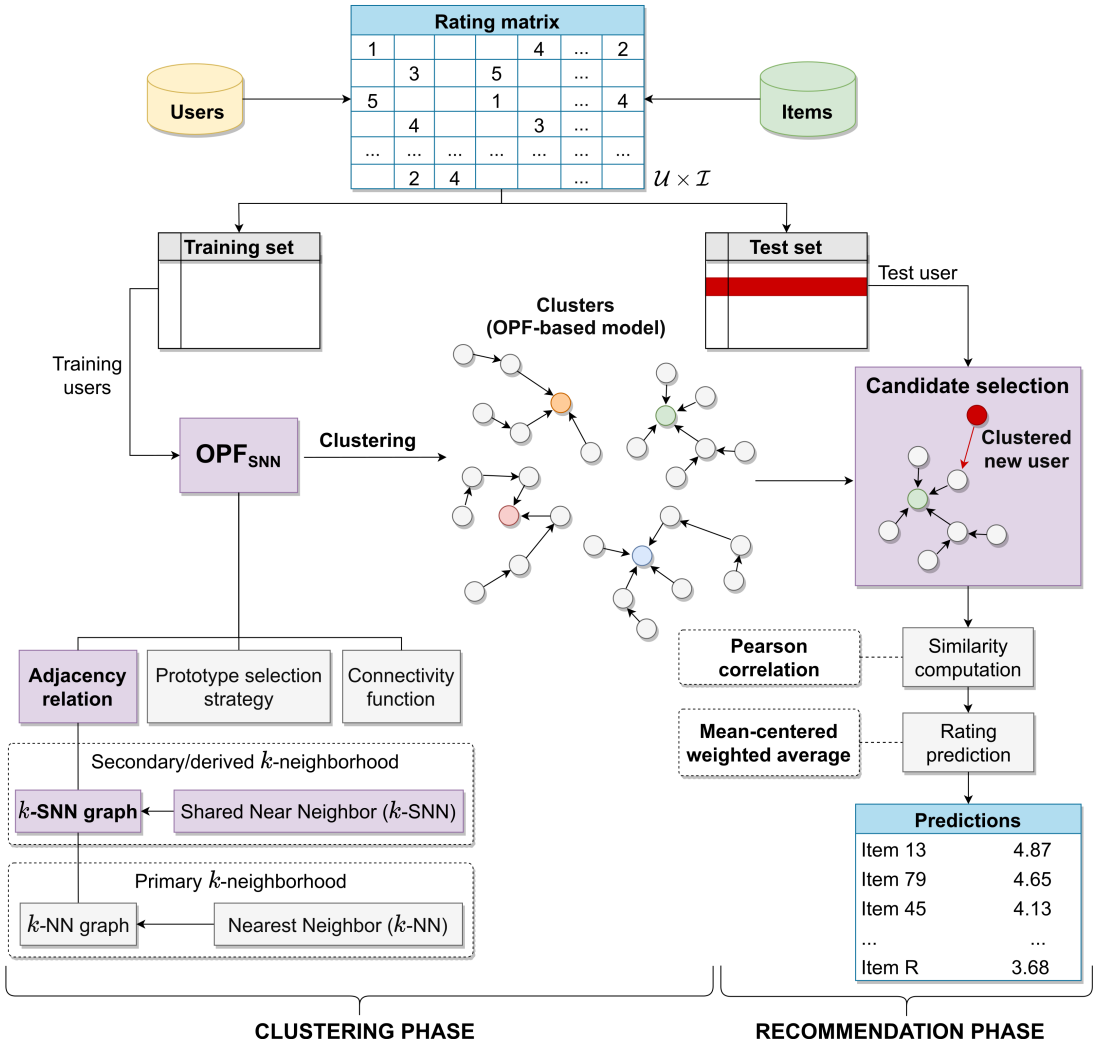
Afterwards, such a model is used during the recommendation phase to group test users and then recommending unobserved items following a CF strategy. This involves identifying the most similar user within the same cluster (similarity computation) and performing rating predictions using an aggregate function such as mean-centered weighted average (Equation 19), for instance.

5.3.2.1 Candidate selection: classifying test users

Similarity computation and candidate selection are conducted only within the target cluster. What we need to describe is how to choose the "correct" cluster. Essentially, the OPF solution for clustering a new sample is employed, but adjusted to consider a k-shared neighborhood adjacency relation based on $\mathcal{A}_3$.

[2] In this work, $c = 2$.

Figure 12 – CF-based recommendation process through OPFSNN.



Source: Produced by the author.

First, let $\mathcal{X}_1$ and $\mathcal{X}_2$ be partitions of a given dataset $\mathcal{X}$, such that $\mathcal{X} = \mathcal{X}_1 \cup \mathcal{X}_2$. Also, consider $(\mathcal{X}_1, \mathcal{A}_3)$ the OPFSNN-based model computed by Algorithm 1 that clustered all samples $\boldsymbol{x}_i \in \mathcal{X}_1$. A sample $\boldsymbol{x}_j \in \mathcal{X}_2$ is classified (grouped) in one of the clusters by identifying which prototype would offer to it an optimum path. Note that there is no need to guarantee the graph symmetry on the density plateaus for test samples, and thus $\mathcal{A}_2$ is employed. By considering the adjacent nodes $\boldsymbol{x}_i \in \mathcal{A}_2(\boldsymbol{x}_j) \in \mathcal{X}_1$, through Equation 20 $\rho(\boldsymbol{x}_j)$ is computed and the optimum paths $\pi_{\boldsymbol{x}_i} \cdot \langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle$ are evaluated by Equation 22. The one path may be chosen as indicated in (ROCHA; CAPPABIANCO; FALCÃO, 2009), such that it satisfies

$$\mathcal{V}(\boldsymbol{x}_j) = \max_{\forall (\boldsymbol{x}_i, \boldsymbol{x}_j) \in \mathcal{A}_2} \{\min\{\mathcal{V}(\boldsymbol{x}_i), \rho(\boldsymbol{x}_j)\}\}. \tag{28}$$

Consider $\boldsymbol{x}_i^* \in \mathcal{X}_1$ to be the node that satisfies Equation 28. Finally, the classification

simply assigns $\mathcal{L}(\boldsymbol{x}_j) \leftarrow \mathcal{L}(\boldsymbol{x}_i^*)$, i.e., $\boldsymbol{x}_j$ belongs to the cluster of $\boldsymbol{x}_i^*$.

Note that by adopting Equations 24 and 25, the process of building the k-shared neighborhood of $\boldsymbol{x}_j \in \mathcal{X}_2$ with OPFSNN slightly differs from OPF regarding the clustering of test samples. After finding $\boldsymbol{x}_i \in \mathcal{A}_1(\boldsymbol{x}_j)$, $\boldsymbol{x}_j$ must be temporally added to every $\boldsymbol{x}_i$ neighborhood, such that Equation 24 is satisfied. That enables the adoption of $\mathcal{A}_2$ during the evaluation of optimum paths. These steps are synthesized as a pseudo code described by Algorithm 2.

Algorithm 2 Prediction by OPFSNN

Input: Optimum-path forest $(\mathcal{X}_1, \mathcal{A}_3)$, path-value map $\mathcal{V}$, and label map $\mathcal{L}$.
Output: Updated label map $\mathcal{L}$.
Auxiliary: Test graph $(\mathcal{X}_2, \mathcal{A}_3)$, variable tmp.

for each test sample $\boldsymbol{x}_j \in \mathcal{X}_2$ **do**
  Set $\boldsymbol{x}_j$'s k-neighborhood with respect to all $\boldsymbol{x}_i \in \mathcal{X}_1$. ▷ Equation 23.
  for each $\boldsymbol{x}_i \in \mathcal{A}_1(\boldsymbol{x}_j)$ **do**
    Insert $\boldsymbol{x}_j$ to $\boldsymbol{x}_i$'s k-neighborhood. ▷ Sorted by arc weight.
    if $\boldsymbol{x}_j \in \mathcal{A}_1(\boldsymbol{x}_i)$ **and** $\boldsymbol{x}_i \in \mathcal{A}_1(\boldsymbol{x}_j)$ **then**
      Make $(\boldsymbol{x}_j, \boldsymbol{x}_i) \in \mathcal{A}_2$. ▷ Equation 24.
    end if
    Remove $\boldsymbol{x}_j$ from $\boldsymbol{x}_i$'s k-neighborhood.
  end for
  Compute $\rho(\boldsymbol{x}_j)$. ▷ Equation 20.
  $\mathcal{V}(\boldsymbol{x}_j) \leftarrow -1$.
  for each $\boldsymbol{x}_i \in \mathcal{A}_2(\boldsymbol{x}_j)$ **do** ▷ Equation 28.
    $tmp \leftarrow \min\{\mathcal{V}(\boldsymbol{x}_i), \rho(\boldsymbol{x}_j)\}$.
    if $tmp > \mathcal{V}(\boldsymbol{x}_j)$ **then**
      $\mathcal{L}(\boldsymbol{x}_j) \leftarrow \mathcal{L}(\boldsymbol{x}_i)$, $\mathcal{V}(\boldsymbol{x}_j) \leftarrow tmp$.
    end if
  end for
end for

5.3.2.2 Similarity computation and rating prediction

After computing the cluster of a target user $\boldsymbol{x}_j$ through the strategy described in the previous section, the recommendation phase can be conducted following a traditional CF setup, i.e., finding the most similar users with respect to the target user and estimating ratings for unobserved items, respectively. To do so, the Pearson correlation has been employed in this work for similarity computation and is defined as follows:

$$Pearson(\boldsymbol{x}_i, \boldsymbol{x}_j) = \frac{\sum_{t \in \mathcal{T}} (\boldsymbol{x}_{it} - \mu_i) * (\boldsymbol{x}_{jt} - \mu_j)}{\sqrt{\sum_{t \in \mathcal{T}} (\boldsymbol{x}_{it} - \mu_i)^2} * \sqrt{\sum_{t \in \mathcal{T}} (\boldsymbol{x}_{jt} - \mu_j)^2}}, \tag{29}$$

where $\mathcal{T} = \mathcal{I}(\boldsymbol{x}_i) \cap \mathcal{I}(\boldsymbol{x}_j)$ is the set of items rated by users i and j, $\boldsymbol{x}_{it}$ is the dimension from user $\boldsymbol{x}_i$'s rating vector referring to the rated item $t \in \mathcal{T}$, and μ is the average rating

of a given user (rating vector). Finally, the rating prediction is conducted by Equation 19, whose term $\mathbb{W}(\cdot)$ is a function calculated through Equation 29.

5.4 Experiments: clustering

In this section, OPFSNN has been assessed for the clustering task. The goal was to compare the proposed approach against the traditional unsupervised OPF. Then, the quality of the computed clusters using dense data are evaluated.

5.4.1 Experimental setup

For evaluating OPFSNN in this phase, the experiments have been performed with dense datasets, each presenting different number of samples, data dimensionality and the presence (or absence) of samples' true labels. All datasets employed in this stage of the work have public accessibility and are available at UCI Machine Learning Repository (DUA; GRAFF, 2019) (BCW, CCRF, Digits, Frogs, and Google Reviews datasets) and Kaggle[3] (Olivetti Faces). Table 20 describes the mentioned datasets.

Table 20 – Description of dense datasets employed for clustering experiments.

Dataset	# samples	# features	True labels
Breast Cancer (BCW)	569	30	2
Cervical Cancer (CCRF)	858	32	2
Digits	1,797	64	10
Olivetti Faces	400	4,096	40
Google Reviews	5,456	24	–
Anuran Calls (Frogs)	7,195	22	–

Source: Produced by the author.

Clustering results have been assessed through a set of cluster validity measures, namely Davies-Bouldin (DB) Index (DAVIES; BOULDIN, 1979), Completeness, Homogeneity, and V-Measure (ROSENBERG; HIRSCHBERG, 2007). DB Index is an internal measure, while the remaining three depend on the samples' true labels, i.e., external measures. It is worth mentioning V-Measure is independent of the number of class labels, the number of clusters, the size of the data, and the clustering algorithm employed. That makes it suitable for assessing both OPFSNN and OPF on the sort of data we have chosen for these experiments.

Regarding experimental procedures, data have been first normalized and then randomly split into 80% for training and the remaining 20% for testing. Also, the ex-

[3] <https://www.kaggle.com/datasets/sahilyagnik/olivetti-faces>

periments have been repeated 20 times and further validated statistically through the Wilcoxon signed-rank test[4] (WILCOXON, 1945).

Several experiments have been conducted to compare OPFSNN and the traditional OPF clustering – i.e., the baseline. These approaches have been initialized with different values for k_{max} parameter within the interval $[5, 50]$, thus allowing a more appropriate comparison between the proposed approach and the baseline. Finally, all experiments have been performed on a Ubuntu 22.04 Linux 64 bits machine with 8Gb of RAM running 8 × Intel® Core™ i5-9300H CPU @ 2.40GHz processors, and the corresponding source code of all performed experiments are hosted in the project repository on GitHub[5].

5.4.2 Results

Tables 21 and 22 present the results obtained on labeled datasets. The former depicts results regarding $k_{max} \in [5, 25]$, while the latter shows results assuming $k_{max} \in [30, 50]$. Average values of Homogeneity, Completeness, and V-Measure are depicted assuming different k_{max} setups for both OPFSNN and OPF, as well as their respective standard deviations.

The majority of results indicate that the general performance of OPFSNN is superior to OPF in terms of V-Measure in three out of four of the labeled datasets – i.e., Breast Cancer (BCW), Digits, and Olivetti Faces. Also, OPFSNN achieved a better consistency than OPF with respect to the measured homogeneity of the computed clusters as k_{max} grows. That means, on average, OPFSNN presented more similar samples within each computed cluster regardless of the maximum size of the shared neighborhood.

Another finding is that when analyzing the results for the Digits dataset and considering different k_{max} parameter settings, OPFSNN demonstrated to be less sensitive to such parameter changing than OPF in terms of V-Measure performance, in general. Concerning Completeness evaluation, i.e., how much similar samples are grouped, the results indicated a more balanced performance between OPFSNN and OPF performances in contrast to the general achieved Homogeneity and V-Measure results, for instance.

The results that attested a greater difference between OPFSNN and OPF performances (in favor of the former) considering all external measures have been achieved with samples from the Olivetti Faces dataset. Among the other employed data, such a dataset presents the highest number of labels (40 classes) and data dimensionality (4,096 features per sample). Also, the proposed approach achieved superior results against OPF in Digits, the dataset showing the second highest number of labels with data distributed among ten different classes.

Regarding the k_{max} setting, the results showed that best recognition rates for both approaches have been achieved by exploring small k_{max} values, particularly for $5 \leq k_{max} \leq$

[4] Such statistical test has been chosen since the data does not necessarily follows a Gaussian distribution.
[5] <https://github.com/guibmartins/opf_snn_collaborative_filtering>

Table 21 – Average results considering labeled datasets and $k_{max} \in [5, 25]$. The best statistical results are denoted in bold according to the Wilcoxon signed-rank test with 95% of confidence.

$\mathbf{k_{max}}$	Dataset	Approach	Homogeneity	Completeness	V-Measure
5	BCW	OPF	0.5370 ± 0.0656	0.5822 ± 0.0703	0.5584 ± 0.0664
		OPFSNN	$\mathbf{0.6022 \pm 0.0996}$	0.6141 ± 0.0966	$\mathbf{0.6078 \pm 0.0976}$
	CCRF	OPF	0.0141 ± 0.0192	0.1448 ± 0.3084	0.0186 ± 0.0276
		OPFSNN	0.0184 ± 0.0223	0.0249 ± 0.0346	0.0196 ± 0.0234
	Digits	OPF	0.9615 ± 0.0142	0.9621 ± 0.0137	0.9618 ± 0.0139
		OPFSNN	0.9657 ± 0.0115	0.9661 ± 0.0115	0.9659 ± 0.0115
	Olivetti Faces	OPF	0.7564 ± 0.0421	0.8570 ± 0.0251	0.8031 ± 0.0313
		OPFSNN	$\mathbf{0.8480 \pm 0.0251}$	$\mathbf{0.8880 \pm 0.0204}$	$\mathbf{0.8674 \pm 0.0211}$
10	BCW	OPF	0.4386 ± 0.0935	0.5257 ± 0.0779	0.4769 ± 0.0869
		OPFSNN	$\mathbf{0.5594 \pm 0.0923}$	$\mathbf{0.5696 \pm 0.0922}$	$\mathbf{0.5643 \pm 0.0918}$
	CCRF	OPF	0.0080 ± 0.0139	0.2141 ± 0.4045	0.0085 ± 0.0171
		OPFSNN	0.0129 ± 0.0118	0.0146 ± 0.0184	0.0128 ± 0.0128
	Digits	OPF	0.9435 ± 0.0189	0.9448 ± 0.0181	0.9441 ± 0.0185
		OPFSNN	$\mathbf{0.9618 \pm 0.0113}$	$\mathbf{0.9623 \pm 0.0114}$	$\mathbf{0.9621 \pm 0.0114}$
	Olivetti Faces	OPF	0.5820 ± 0.0493	0.8133 ± 0.0262	0.6775 ± 0.0383
		OPFSNN	$\mathbf{0.8139 \pm 0.0318}$	$\mathbf{0.8824 \pm 0.0211}$	$\mathbf{0.8466 \pm 0.0238}$
15	BCW	OPF	0.4321 ± 0.0923	0.5268 ± 0.0757	0.4738 ± 0.0861
		OPFSNN	$\mathbf{0.5002 \pm 0.0906}$	0.5190 ± 0.0867	0.5091 ± 0.0879
	CCRF	OPF	0.0080 ± 0.0127	0.2144 ± 0.4043	0.0085 ± 0.0165
		OPFSNN	0.0128 ± 0.0133	0.0194 ± 0.0363	0.0142 ± 0.0190
	Digits	OPF	0.9107 ± 0.0244	0.9159 ± 0.0214	0.9133 ± 0.0228
		OPFSNN	$\mathbf{0.9588 \pm 0.0131}$	$\mathbf{0.9592 \pm 0.0131}$	$\mathbf{0.9590 \pm 0.0131}$
	Olivetti Faces	OPF	0.4100 ± 0.0554	0.8144 ± 0.0378	0.5430 ± 0.0519
		OPFSNN	$\mathbf{0.7915 \pm 0.0311}$	$\mathbf{0.8760 \pm 0.0186}$	$\mathbf{0.8313 \pm 0.0215}$
20	BCW	OPF	0.3933 ± 0.1023	0.4972 ± 0.0835	0.4377 ± 0.0967
		OPFSNN	$\mathbf{0.5224 \pm 0.1000}$	0.5427 ± 0.1066	$\mathbf{0.5321 \pm 0.1026}$
	CCRF	OPF	0.0039 ± 0.0073	0.2559 ± 0.4408	0.0038 ± 0.0067
		OPFSNN	$\mathbf{0.0107 \pm 0.0103}$	0.0120 ± 0.0158	$\mathbf{0.0108 \pm 0.0120}$
	Digits	OPF	0.8951 ± 0.0234	0.9040 ± 0.0218	0.8995 ± 0.0224
		OPFSNN	$\mathbf{0.9464 \pm 0.0186}$	$\mathbf{0.9471 \pm 0.0184}$	$\mathbf{0.9467 \pm 0.0185}$
	Olivetti Faces	OPF	0.2626 ± 0.0783	0.8325 ± 0.0429	0.3934 ± 0.0855
		OPFSNN	$\mathbf{0.7655 \pm 0.0400}$	$\mathbf{0.8800 \pm 0.0218}$	$\mathbf{0.8182 \pm 0.0265}$
25	BCW	OPF	0.3928 ± 0.0929	0.4965 ± 0.0806	0.4376 ± 0.0892
		OPFSNN	$\mathbf{0.5459 \pm 0.1496}$	$\mathbf{0.5695 \pm 0.1490}$	$\mathbf{0.5572 \pm 0.1491}$
	CCRF	OPF	0.0070 ± 0.0123	$\mathbf{0.3059 \pm 0.4663}$	0.0055 ± 0.0095
		OPFSNN	0.0113 ± 0.0191	0.0192 ± 0.0384	0.0133 ± 0.0234
	Digits	OPF	0.8813 ± 0.0286	0.8955 ± 0.0239	0.8883 ± 0.0258
		OPFSNN	$\mathbf{0.9321 \pm 0.0243}$	$\mathbf{0.9329 \pm 0.0244}$	$\mathbf{0.9325 \pm 0.0244}$
	Olivetti Faces	OPF	0.1298 ± 0.0807	0.8234 ± 0.1091	0.2144 ± 0.1166
		OPFSNN	$\mathbf{0.7593 \pm 0.0479}$	$\mathbf{0.8786 \pm 0.0190}$	$\mathbf{0.8138 \pm 0.0310}$

Source: Produced by the author.

Table 22 – Average results considering labeled datasets and $k_{max} \in [30, 50]$. The best statistical results are denoted in bold according to the Wilcoxon signed-rank test with 95% of confidence.

$\mathbf{k_{max}}$	Dataset	Approach	Homogeneity	Completeness	V-Measure
30	BCW	OPF	0.3718 ± 0.1244	0.5211 ± 0.1367	0.4148 ± 0.1294
		OPFSNN	$\mathbf{0.5297 \pm 0.1259}$	0.5600 ± 0.1215	$\mathbf{0.5441 \pm 0.1233}$
	CCRF	OPF	0.0070 ± 0.0130	0.4061 ± 0.4976	0.0056 ± 0.0100
		OPFSNN	0.0157 ± 0.0155	0.0673 ± 0.2202	$\mathbf{0.0160 \pm 0.0158}$
	Digits	OPF	0.8694 ± 0.0264	0.8895 ± 0.0203	0.8793 ± 0.0229
		OPFSNN	$\mathbf{0.9354 \pm 0.0176}$	$\mathbf{0.9363 \pm 0.0177}$	$\mathbf{0.9359 \pm 0.0177}$
	Olivetti Faces	OPF	0.0831 ± 0.0666	0.8525 ± 0.1187	0.1432 ± 0.1049
		OPFSNN	$\mathbf{0.7241 \pm 0.0840}$	0.8732 ± 0.0268	$\mathbf{0.7896 \pm 0.0575}$
35	BCW	OPF	0.2527 ± 0.1482	0.5191 ± 0.2224	0.2979 ± 0.1611
		OPFSNN	$\mathbf{0.5105 \pm 0.1524}$	0.5332 ± 0.1642	$\mathbf{0.5213 \pm 0.1577}$
	CCRF	OPF	0.0049 ± 0.0106	$\mathbf{0.5548 \pm 0.5051}$	0.0041 ± 0.0083
		OPFSNN	$\mathbf{0.0197 \pm 0.0205}$	0.0786 ± 0.2205	$\mathbf{0.0215 \pm 0.0248}$
	Digits	OPF	0.8568 ± 0.0260	0.8806 ± 0.0228	0.8685 ± 0.0234
		OPFSNN	$\mathbf{0.9238 \pm 0.0162}$	$\mathbf{0.9250 \pm 0.0165}$	$\mathbf{0.9244 \pm 0.0163}$
	Olivetti Faces	OPF	0.0269 ± 0.0315	$\mathbf{0.9352 \pm 0.1176}$	0.0502 ± 0.0578
		OPFSNN	$\mathbf{0.7264 \pm 0.0694}$	0.8733 ± 0.0233	$\mathbf{0.7918 \pm 0.0492}$
40	BCW	OPF	0.1858 ± 0.1555	0.5893 ± 0.2848	0.2239 ± 0.1747
		OPFSNN	$\mathbf{0.5210 \pm 0.1312}$	0.5567 ± 0.1310	$\mathbf{0.5375 \pm 0.1304}$
	CCRF	OPF	0.0052 ± 0.0116	$\mathbf{0.5547 \pm 0.5051}$	0.0041 ± 0.0087
		OPFSNN	$\mathbf{0.0198 \pm 0.0222}$	0.0243 ± 0.0320	$\mathbf{0.0205 \pm 0.0252}$
	Digits	OPF	0.8402 ± 0.0285	0.8732 ± 0.0219	0.8563 ± 0.0241
		OPFSNN	$\mathbf{0.9175 \pm 0.0240}$	$\mathbf{0.9195 \pm 0.0226}$	$\mathbf{0.9185 \pm 0.0233}$
	Olivetti Faces	OPF	0.0159 ± 0.0266	$\mathbf{0.9615 \pm 0.0814}$	0.0299 ± 0.0494
		OPFSNN	$\mathbf{0.7020 \pm 0.0668}$	0.8688 ± 0.0223	$\mathbf{0.7750 \pm 0.0460}$
45	BCW	OPF	0.0610 ± 0.0984	$\mathbf{0.8116 \pm 0.2964}$	0.0785 ± 0.1257
		OPFSNN	$\mathbf{0.4839 \pm 0.1502}$	0.5075 ± 0.1567	$\mathbf{0.4946 \pm 0.1524}$
	CCRF	OPF	0.0050 ± 0.0116	$\mathbf{0.5554 \pm 0.5043}$	0.0043 ± 0.0087
		OPFSNN	$\mathbf{0.0152 \pm 0.0157}$	0.0689 ± 0.2202	$\mathbf{0.0160 \pm 0.0174}$
	Digits	OPF	0.8295 ± 0.0221	0.8721 ± 0.0251	0.8502 ± 0.0224
		OPFSNN	$\mathbf{0.9121 \pm 0.0220}$	$\mathbf{0.9146 \pm 0.0204}$	$\mathbf{0.9134 \pm 0.0211}$
	Olivetti Faces	OPF	0.0069 ± 0.0171	$\mathbf{0.9705 \pm 0.0798}$	0.0132 ± 0.0324
		OPFSNN	$\mathbf{0.6798 \pm 0.0656}$	0.8689 ± 0.0212	$\mathbf{0.7614 \pm 0.0473}$
50	BCW	OPF	0.0260 ± 0.0653	$\mathbf{0.8992 \pm 0.2463}$	0.0337 ± 0.0835
		OPFSNN	$\mathbf{0.5297 \pm 0.1479}$	0.5680 ± 0.1469	$\mathbf{0.5477 \pm 0.1466}$
	CCRF	OPF	0.0028 ± 0.0080	$\mathbf{0.6597 \pm 0.4765}$	0.0041 ± 0.0126
		OPFSNN	$\mathbf{0.0190 \pm 0.0205}$	0.0696 ± 0.2204	$\mathbf{0.0186 \pm 0.0219}$
	Digits	OPF	0.8255 ± 0.0235	0.8702 ± 0.0250	0.8472 ± 0.0236
		OPFSNN	$\mathbf{0.8949 \pm 0.0214}$	$\mathbf{0.8995 \pm 0.0201}$	$\mathbf{0.8972 \pm 0.0207}$
	Olivetti Faces	OPF	0.0023 ± 0.0103	$\mathbf{1.0000 \pm 0.0000}$	0.0044 ± 0.0197
		OPFSNN	$\mathbf{0.6575 \pm 0.0961}$	0.8595 ± 0.0185	$\mathbf{0.7416 \pm 0.0689}$

Source: Produced by the author.

Table 23 – Average results considering unlabeled data and $k_{max} \in [5, 50]$. The best statistical results are denoted in bold according to the Wilcoxon signed-rank test with 95% of confidence.

Dataset	Approach	DB-index				
		$k_{max} = 5$	$k_{max} = 10$	$k_{max} = 15$	$k_{max} = 20$	$k_{max} = 25$
Frogs	OPF	1.4506 ± 0.0567	$\mathbf{1.4969 \pm 0.0591}$	$\mathbf{1.4787 \pm 0.0643}$	$\mathbf{1.4585 \pm 0.0576}$	$\mathbf{1.5014 \pm 0.0766}$
	OPFSNN	$\mathbf{1.3655 \pm 0.0369}$	$\mathbf{1.4747 \pm 0.0480}$	1.5443 ± 0.0409	1.6202 ± 0.0404	1.6450 ± 0.0352
Google Reviews	OPF	0.8955 ± 0.0250	1.1195 ± 0.0299	1.2689 ± 0.0352	1.4177 ± 0.0432	1.5088 ± 0.0448
	OPFSNN	$\mathbf{0.8203 \pm 0.0219}$	$\mathbf{0.9595 \pm 0.0231}$	$\mathbf{1.0380 \pm 0.0284}$	$\mathbf{1.1040 \pm 0.0330}$	$\mathbf{1.1693 \pm 0.0353}$
		$k_{max} = 30$	$k_{max} = 35$	$k_{max} = 40$	$k_{max} = 45$	$k_{max} = 50$
Frogs	OPF	$\mathbf{1.5339 \pm 0.0749}$	$\mathbf{1.5677 \pm 0.0910}$	$\mathbf{1.5591 \pm 0.1115}$	$\mathbf{1.5918 \pm 0.1021}$	$\mathbf{1.6212 \pm 0.1226}$
	OPFSNN	1.6682 ± 0.0445	1.6850 ± 0.0460	1.7100 ± 0.0538	1.7373 ± 0.0567	1.7422 ± 0.0684
Google Reviews	OPF	1.5929 ± 0.0387	1.6851 ± 0.0430	1.7303 ± 0.0458	1.7521 ± 0.0621	1.7397 ± 0.0696
	OPFSNN	$\mathbf{1.1986 \pm 0.0431}$	$\mathbf{1.2381 \pm 0.0397}$	$\mathbf{1.2599 \pm 0.0427}$	$\mathbf{1.2938 \pm 0.0592}$	$\mathbf{1.3328 \pm 0.0645}$

Source: Produced by the author.

15. Yet, as the number of neighbors increases, the results indicated a significant drop in OPF's recognition rates in comparison to OPFSNN.

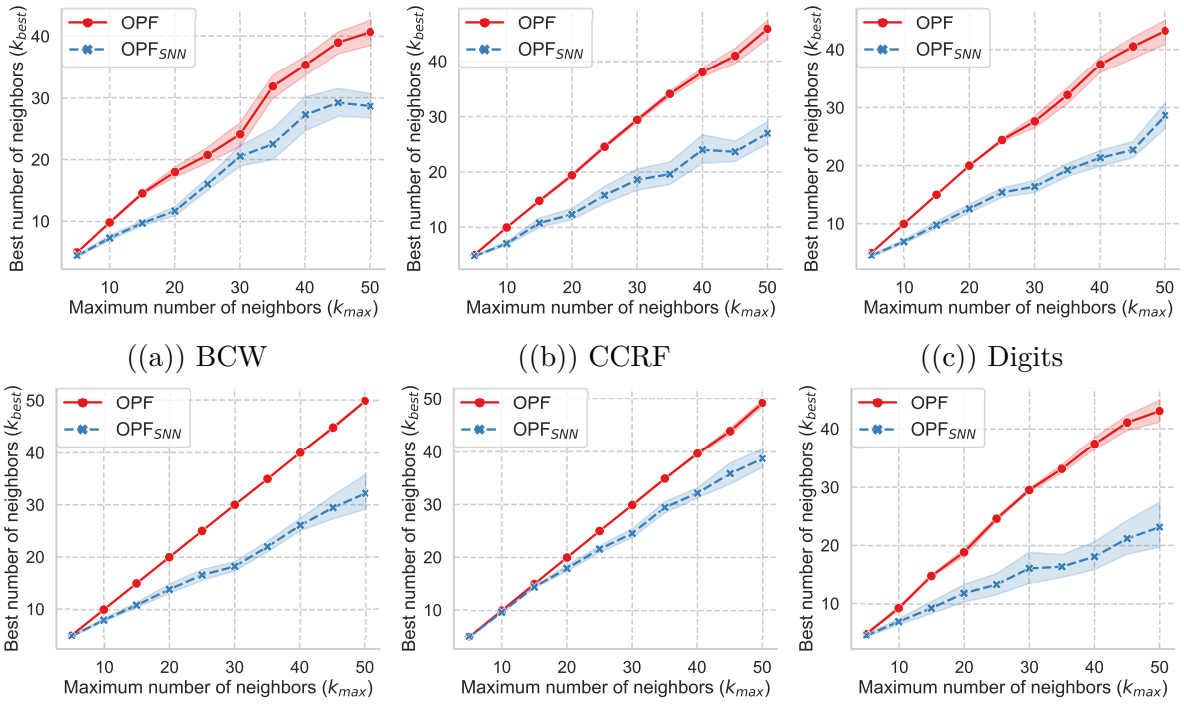
Some experiments have been run with unlabeled datasets. The results are shown in Table 23 and describe the average DB Index and standard deviations assuming different k_{max} settings for OPFSNN and OPF. The results indicated statistical DB Index superiority in the Frogs dataset towards OPF, while in Google Reviews data OPFSNN demonstrated the best overall performance. One may note that superior recognition rates have been achieved by both approaches when they are initialized with a small number of neighbors, i.e., $\{k_{max} \in \mathbb{Z} : 5 \leq k_{max} \leq 20\}$. As previously reported, such behavior also has been observed with labeled data experiments.

Whereas DB Index essentially measures the compactness of clusters and how well they are separated from each other, when $k_{max} \in [5, 10]$, one may notice the proposed approach performs better than OPF on both unlabeled datasets. Such a result could express OPFSNN's ability to generate more compact and well-separated clusters than OPF in situations where small shared-neighborhoods are explored, which corresponds with some SNN properties.

There are some key aspects to be discussed about the connection between the maximum k-neighborhood size and the computed best number of neighbors (k_{best}). Figure 13 illustrates such relation for OPFSNN and OPF considering all datasets employed during experiments.

The main finding one may notice is that, for all datasets, the k_{best} parameter computed by OPFSNN is relatively lower than the one optimized by OPF. Such behavior was expected because, during the SNN graph computation, the shared-neighbors search is prone to limit the number of neighbors. Another aspect is that, for a given data, multiple shared-neighborhoods may be computed keeping at most a radius of size k_{max}, i.e., each data point may present its own particular number of shared neighbors. That contrasts

Figure 13 – Relation between k_{max} and k_{best} parameters considering all datasets.



((a)) BCW ((b)) CCRF ((c)) Digits

((d)) Frogs ((e)) Google Reviews ((f)) Olivetti Faces

Source: Produced by the author.

with the k-neighborhoods generated by OPF, which are also limited to a k_{max} size radius, but hold the same k to every data point. Hence, OPFSNN's shared-neighborhoods building process is prone to be adaptive, while computing a conventional k-nearest neighbor graph like the one conducted by OPF is not. We understand such attribute favors the data partitioning into small and compact clusters.

Besides, there is a situation to be pointed out concerning the number of clusters computed by OPFSNN. Figure 14 presents the relation between k_{best} and the number of clusters computed by OPFSNN and OPF in all employed datasets.

One may notice that, in all scenarios (datasets), OPFSNN computes more clusters than OPF. It may indicate OPFSNN's tendency to generate smaller neighborhoods, thus resulting in small and numerous Optimum-Path Trees (clusters). Another aspect concerning the relation between k_{best} and the number of clusters is that it may be described by exponential functions for OPFSNN and OPF. In other words, it seems the proposed approach is able to preserve the correlation between those mentioned parameters when the SNN-based adjacency relation is employed.

We can connect those findings with the results presented in Tables 21 to 23 since the discovered OPFSNN attributes may reflect on the measured clustering performance. OPFSNN shared-neighborhoods are prone to be formed by very close data points, consequently impacting the path-cost learning process. As demonstrated throughout the experiments, OPFSNN computes numerous but small clusters compared to OPF. That

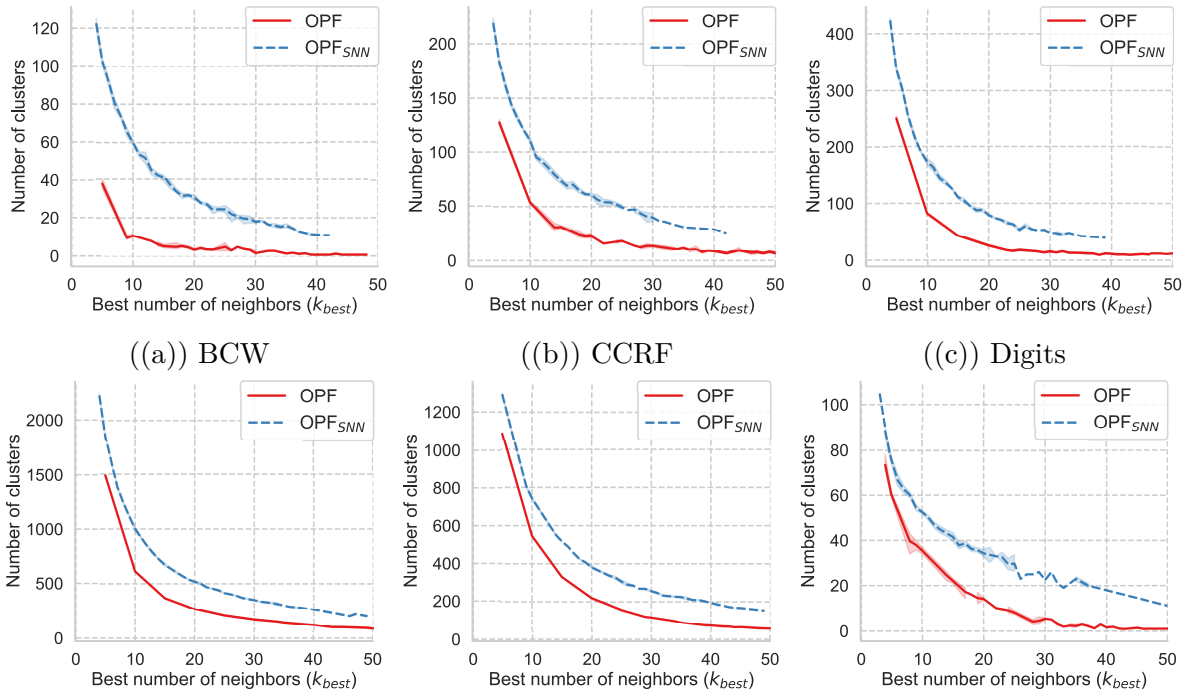
may improve the correct classification of test samples when data labels are available, as one can observe from the results evaluated through the set of external cluster validity measures chosen in this work.

Regarding assessing unlabeled data experiments through DB Index, OPFSNN presented a lower general performance than OPF, as already discussed. Such results may be associated with some OPFSNN aspects, like (i) the high number of computed clusters and (ii) their different size and sample concentrations. That opposes what DB Index measures as a better clustering. i.e., the premise that clusters have spherical shapes with similar sizes and densities. Finally, note that such attributes are known DB Index drawbacks – added to its sensitivity to outliers and noise, thus creating the possibility of leading to false indications of poor clustering results.

5.5 Experiments: recommendation

In this section, OPFSNN has been evaluated for the recommendation task. It describes the experimental setup, protocol, and results, where OPFSNN has been compared to other well-known CF-based recommenders while dealing only with sparse data.

Figure 14 – Relation between number of computed clusters and k_{best} parameters considering all datasets.



((a)) BCW ((b)) CCRF ((c)) Digits

((d)) Frogs ((e)) Google Reviews ((f)) Olivetti Faces

Source: Produced by the author.

5.5.1 Datasets

Three sparse datasets have been employed in the recommendation experiments. Movielens 100K (ML-100K) and Movielens Latest Small (ML-LS) contain sparse data collected and made available by GroupLens Research Group[6]. The third dataset is the Amazon Magazine Subscriptions (AMZ-MS)[7] containing only rating data and is part of the Amazon review data (NI; LI; MCAULEY, 2019). Those are publicly available dataset and have been chosen due to their reliability and popularity among researchers in terms of evaluating CF-based recommenders. All datasets contain rating scores users gave to movies or products and are highly sparse. Also, the rating scores are within 1 to 5 (discrete interval) and each user has rated at least 20 movies or products. Table 24 presents a summary about the main characteristics of the mentioned datasets.

Table 24 – Main information about the sparse datasets employed for recommendation experiments. Density is given by the ratio between the number observed ratings and the total number of possible ratings (# users × # items).

Dataset	**# users (samples)**	**# items (features)**	**# ratings**	**Density (%)**
Amazon Magazine Subscriptions (AMZ-MS)	645	64,380	84,864	0.20
Movielens 100K (ML-100K)	943	1,682	100,000	6.304
Movielens Latest Small (ML-LS)	610	9724	100,836	1.69

Source: Produced by the author.

5.5.2 Evaluation measures

To measure the performance, we have chosen to employ a broad evaluation strategy considering three types of recommendation measures according to Bobadilla et al. (2013): (i) prediction accuracy, (ii) ranking, and (iii) decision support. Our goal is to capture different aspects about the experimental results such that a deeper analysis and interpretation could lead us to robust conclusions.

The prediction accuracy was evaluated using the most adopted measures in the recommendation literature, i.e, Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE). MAE aims to reduce the absolute difference between the real rating (r_{ui}) and the predicted one (p_{ui}) a user u might give to a movie i for n samples in the test set[8]. Its calculation is given as follows:

$$MAE = \frac{1}{n} \sum_{u,i} \mid p_{ui} - r_{ui} \mid . \tag{30}$$

[6] <https://grouplens.org/datasets/movielens/>
[7] <https://nijianmo.github.io/amazon/index.html>
[8] It consists of all rating that must be predicted considering all test items and test users.

RMSE adopts the same principle as MAE, however it reduces the squared difference between r_{ui} and p_{ui} for n test samples to put greater emphasis on large errors. Its equation is given as follows:

$$RMSE = \sqrt{\frac{1}{n}\sum_{u,i}(p_{ui} - r_{ui})^2}, \tag{31}$$

where the final calculation considers the square root of the mean squared error.

In terms of evaluating the ranking quality of recommendations, Normalized Discount Cumulative Gain (NDCG) was adopted since it is a widely used ranking recommendation measure by researchers. It measures the order of the predicted ranked list and normalizes the result using a baseline which represents the correct ordered list of recommendations. To do so, NDCG penalizes highly relevant items that appear lower in the ranked list by reducing the graded relevance value in a logarithm proportion to the position of the resulting measurement, which is calculated as follows:

$$NDCG@K = \frac{1}{IDCG@K}\sum_{i=1}^{K}\frac{2^{rel_i} - 1}{\log_2(i+1)}, \tag{32}$$

where NDCG@K is the result at the top-K ranked items, IDCG@K is the top-K perfect ranked list, and rel_i represents the predicted relevance for an item at position i.

Additionally, to evaluate the capacity of finding all relevant items while avoiding the irrelevant ones in a given top-K recommendation list, Precision and Recall have been adopted as decision support measures in this work. To define those measures in terms of recommendation, let $\mathcal{U}$ be set of all users, $\mathcal{X}_u$ be the set of recommendations to user u, $\mathcal{Z}_u$ be the set of K recommendations to user u, and $\mathcal{Z}'_u$ define the complement of $\mathcal{Z}_u$. Also, considering all users accept K recommendations, Precision and Recall have been defined as presented in (BOBADILLA et al., 2013):

$$P@K = \frac{1}{\mid \mathcal{U} \mid}\sum_{u\in\mathcal{U}}\frac{\{i \in \mathcal{Z}_u \mid r_{u,i} \geq \theta\}}{K} \tag{33}$$

and

$$R@K = \frac{1}{\mid \mathcal{U} \mid}\sum_{u\in\mathcal{U}}\frac{\{i \in \mathcal{Z}_u \mid r_{u,i} \geq \theta\}}{\{i \in \mathcal{Z}_u \cup \mathcal{Z}'_u \mid r_{u,i} \geq \theta\}}, \tag{34}$$

where $P@K$ and $R@K$ are Precision and Recall at top-K recommendations, respectively, and θ is a rating score cutoff that is set as a relevance threshold.

5.5.3 Experimental protocol

A series of procedures have been adopted to conduct and validate experiments. First, data have been reorganized into a user × item rating matrix, whose rows (users) are sam-

ples/instances and the columns (items) are user features. Thus, each user is represented by a sparse feature vector of rated items the former have interacted with.

From the resulting rating matrix described earlier, the data must be splitted carefully such that the training and test subsets remain mutually disjoint during the experiments. To attend such a condition, a strong generalization strategy (MARLIN, 2004) have been adopted. The data has been splitted row-wise following a 80/20 proportion, such that 80% of users compose the training set and the 20% remaining constitute the test set. In addition, only 20% of the items rated by each user are considered during the test phase, i.e., their respective rating scores are taken as unobserved interactions during recommendation to further allow an objective assessment of the results.

The aforementioned procedure have been repeated over 20 times for further statistical analysis through the Wilcoxon signed-rank test with 5% of significance. The choice of validating the results using some statistical test was done based on the fact that it is a good practice for experimental evaluation in other machine learning-related areas like speech recognition and computer vision, although it is rarely explored in recommender systems research field as far as we know. In that sense, the goal is to add an additional degree of reliability and robustness during the analysis of experimental results.

Baseline approaches

The proposed approach has been compared against some baseline approaches employed for the recommendation task. Such baselines are clustering- and neighborhood-based approaches, attested great relevance to the field, and also fits the strong generalization protocol evaluation adopted in this work, thus allowing a fairer comparison to OPFSNN. Among them, the only exception is OPF, which has never been employed for CF-based recommendation, as far as we known. They are briefly described as follows:

- **User-KNN**: The conventional k-neighborhood-based algorithm for collaborative filtering recommendation, which is inspired on KNN. In this work, the user-oriented version of KNN have been employed.
- **k-Means**: The classic and most popular clustering algorithm, but customized to work with incomplete data as described in (AGGARWAL, 2016b). In that sense, only the intersection of not null dimensions – rated or observed items – are considered during distance computations between data points.
- **OPF**: The conventional unsupervised version of the Optimum-Path Forest algorithm designed for clustering. OPF has been adapted in the same way as k-Means, enabling it to cluster sparse incomplete data in CF-based contexts.
- **DBSCAN**: The Density-Based Spatial Clustering for Applications with Noise (DBSCAN) proposed by Ester et al. (1996). Different from k-Means, it is able to identify clusters

with arbitrary shape and does not need the number of clusters to be set a priori. As the other clustering-based baselines employed during this experimental stage, DBSCAN has been adapted to cluster sparse and incomplete data.

5.5.4 Results

The experiments have been conducted in two steps. First, OPFSNN has been evaluated individually through RMSE, assuming different values of maximum neighborhood size (k_{max}), distance functions, and the number of computed clusters. Further, the best-reported result in each data from combining those parameters is selected, and the optimal OPFSNN is compared against the baseline approaches.

Choosing OPFSNN parameters

Figure 15 depicts average RMSE results achieved by OPFSNN using different distance functions while the maximum neighborhood size parameter varies such that $k_{max} \in [10, 60]$. Regardless of the distance function assumed, lower RMSE measurements have been achieved as k_{max} increases, which means samples have been distributed among a restricted number of computed clusters. Such an aspect impacts the rating estimation through CF-based heuristic functions like the mean-centered weighted average employed in this work. Also, OPFSNN has not demonstrated uncommon performance regarding changing the distance function. From that, we presume the selection of that parameter may be more directly correlated to the nature of the sample data than to the structure of the learning algorithm.

The optimal combination of parameters obtained on each sparse dataset is selected to initialize the OPFSNN model during the second experimental step. The summary of those findings is presented in Table 25, including the average values of computed parameters like the optimized number of shared neighbors (k_{best}) and the estimated number of clusters.

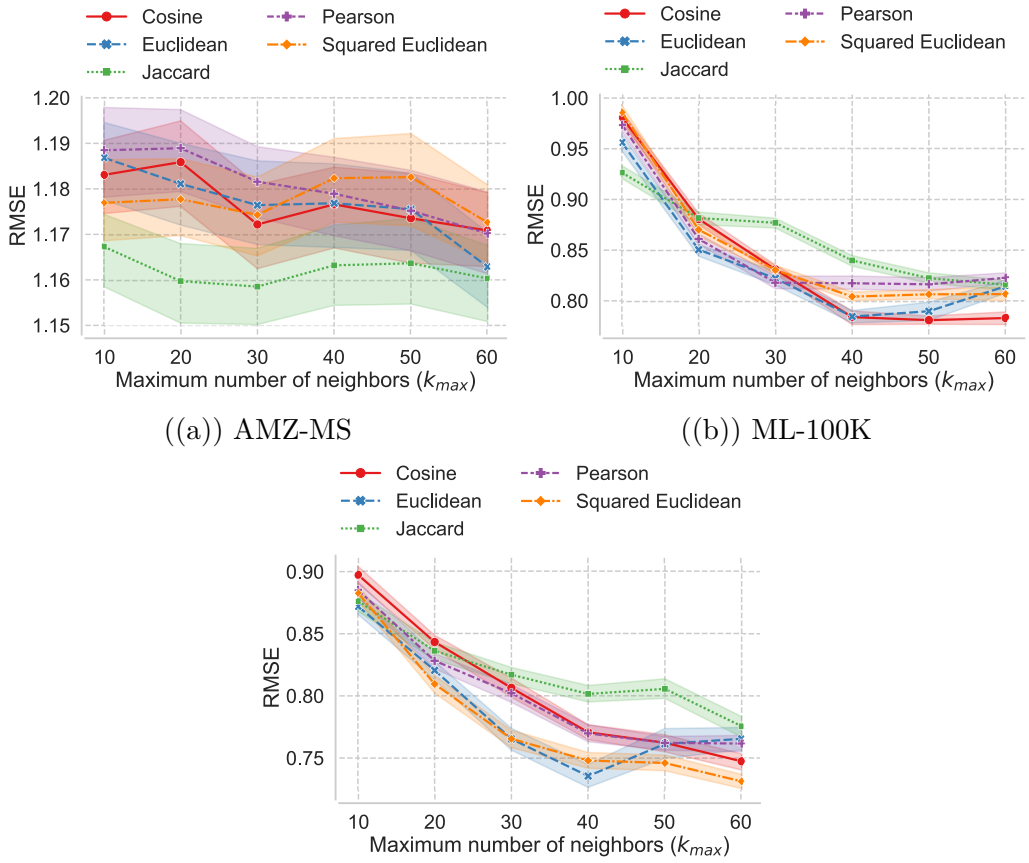
Table 25 – Best combination of OPFSNN parameters based on average RMSE results achieved in sparse datasets.

Dataset	k_{max}	Distance function	k_{best}	# of clusters
AMZ-MS	30	Jaccard	28.15	56.8 (57)
ML-100K	50	Cosine	47.15	94.1 (94)
ML-LS	60	Squared Euclidean	39.25	20.7 (21)

Source: Produced by the author.

To create similar initialization conditions as much as possible, the same k_{max} has been employed by OPF and User-KNN, while to cluster data using k-Means the average number of clusters obtained on-the-fly by OPFSNN has been rounded to the closest integer parameter value. Still concerning such a parameter, it was necessary to empirically

Figure 15 – Average RMSE achieved by OPFSNN using different distance functions while varying the maximum number of neighbors (k_{max}).



((a)) AMZ-MS

((b)) ML-100K

((c)) ML-LS

Source: Produced by the author.

optimize DBSCAN's parameters[9] until the number of clusters computed by DBSCAN and OPFSNN was close enough. Regarding the distance function, all clustering-based approaches required it to be set similarly to OPFSNN, while User-KNN employed the Pearson Correlation similarity, following its conventional implementation. In terms of predicting ratings during the recommendation stage, the mean-centered weighted average function and Pearson correlation have been adopted, respectively, as aggregation function and similarity function by all approaches.

Comparing OPFSNN against baselines

The experiments have been organized by the recommendation aspect we wanted to measure, which includes prediction accuracy, ranking, and decision support assessment.

[9] The following DBSCAN parameters have been optimized in this work: (i) the maximum distance between two samples for one to be considered the other's neighbor; and (ii) the number of samples in a neighborhood for a point to be considered as a core point.

Tables 26, 27, and 28 describe the achieved average results in all sparse datasets concerning those recommendation aspects, respectively.

Table 26 details the results achieved through MAE and RMSE in sparse data. Looking individually at the experiments conducted on each dataset, one may note that the results achieved in AMZ-MS dataset indicate OPFSNN having the best performance among all approaches concerning MAE and RMSE. In ML-100K experiments, OPFSNN achieved the second best prediction accuracy results, and with the ML-LS dataset, it presented the third-best MAE and RMSE evaluations. Looking at those results from an overall perspective, OPF-based approaches demonstrated lower average rating prediction than User-KNN and k-Means, mostly with MovieLens sparse data.

It is noteworthy that the approaches employed in this work are not based on gradient descent algorithms to optimize MAE and RMSE during learning. Given that they are memory- and clustering-based approaches, the recommendation is generated by means of an heuristic function like the mean-centered weighted average. In that sense, it has been expected that the results would lead to a balanced prediction accuracy performance among all approaches. The application of a statistical test to the results only confirmed the expectations we had at that point.

One may notice DBSCAN performances in ML-100K and ML-LS datasets are considerably superior than the other approaches, which is not demonstrated by the results in the AMZ-MS dataset. Looking closely at the clustering results, we have found DBSCAN esti-

Table 26 – Average results considering prediction accuracy assessment measures over sparse data. The best results are indicated in bold according to the Wilcoxon Signed-Rank Test with 95% of confidence.

Dataset	Approach	MAE	RMSE
AMZ-MS	User-KNN	1.0200 ± 0.0649	1.1673 ± 0.0715
	k-Means	1.0069 ± 0.0543	1.1604 ± 0.0624
	OPF	1.0047 ± 0.0542	1.1618 ± 0.0608
	OPFSNN	$\mathbf{1.0012 \pm 0.0501}$	$\mathbf{1.1587 \pm 0.0567}$
	DBSCAN	1.0149 ± 0.0516	1.1748 ± 0.0610
ML-100K	User-KNN	0.8459 ± 0.0366	1.0053 ± 0.0392
	k-Means	0.7185 ± 0.0315	0.8605 ± 0.0383
	OPF	0.8469 ± 0.0293	1.0047 ± 0.0342
	OPFSNN	0.6509 ± 0.0224	0.7818 ± 0.0274
	DBSCAN	$\mathbf{0.5827 \pm 0.0251}$	$\mathbf{0.6960 \pm 0.0296}$
ML-LS	User-KNN	0.7498 ± 0.0367	0.9075 ± 0.0446
	k-Means	0.6403 ± 0.0453	0.7793 ± 0.0481
	OPF	0.5482 ± 0.0340	0.6675 ± 0.0419
	OPFSNN	0.6016 ± 0.0354	0.7313 ± 0.0388
	DBSCAN	$\mathbf{0.5409 \pm 0.0325}$	$\mathbf{0.6582 \pm 0.0393}$

Source: Produced by the author.

mated only one cluster in all experimental cases considering the two MovieLens datasets. In such situations, the approach will estimate ratings exploring all samples from the dataset rather than selecting a few when more clusters are computed. Thus, it is expected that the recommendation performance employing an aggregation function will be the best achievable, since more data will be available to enhance the rating estimation ability.

In practice, the aforementioned behavior may indicate some points regarding DBSCAN: (i) it struggles to deal with sparse feature vectors and too many low density regions in data by biasing the group of all samples into one big cluster; and (ii) it mislead demonstrated the lowest MAE an RMSE while not been able to properly cluster sparse data for further recommendation. In that sense, we understand that the outcomes achieved by DBSCAN through ranking and decision-support measures will also be deceptive at some point. Therefore, the second-best results in the upcoming tables have been underlined.

Regarding the ranking quality evaluation (Table 27), NDCG@K has been employed for measuring $K = 1$, $K = 5$, and $K = 10$ item recommendations. According to the results, DBSCAN achieved the best overall ranking performance among all considered approaches, followed by OPFSNN and OPF. Considering the disclaimer about DBSCAN, the ranking results point out OPFSNN is able to recommend more relevant items while clustering than the baselines to the limit of $K = 10$.

Such positive results achieved by both OPFSNN and OPF in terms of ranking quality may indicate that the competition-based process, which computes optimum-paths during

Table 27 – Average results considering ranking assessment measures over sparse data. The best results are indicated in bold according to the Wilcoxon Signed-Rank Test with 95% of confidence.

Dataset	Approach	NDCG@1	NDCG@5	NDCG@10
AMZ-MS	User-KNN	0.3016 ± 0.0479	0.3506 ± 0.0542	0.3088 ± 0.0693
	k-Means	0.3922 ± 0.0464	0.4291 ± 0.0435	0.4015 ± 0.0698
	OPF	$\mathbf{0.4641 \pm 0.0520}$	$\underline{0.4945 \pm 0.0481}$	$\underline{0.4607 \pm 0.0727}$
	OPFSNN	0.4430 ± 0.0583	$\mathbf{0.5019 \pm 0.0575}$	$\mathbf{0.4935 \pm 0.0694}$
	DBSCAN	$\underline{0.4570 \pm 0.0722}$	0.4861 ± 0.0581	0.4392 ± 0.0452
ML-100K	User-KNN	0.3277 ± 0.0567	0.4531 ± 0.0490	0.5054 ± 0.0537
	k-Means	0.4319 ± 0.0652	0.5473 ± 0.0397	0.5762 ± 0.0537
	OPF	0.2707 ± 0.0568	0.3455 ± 0.0407	0.3626 ± 0.0445
	OPFSNN	$\underline{0.4739 \pm 0.0649}$	$\underline{0.5821 \pm 0.0479}$	$\underline{0.6188 \pm 0.0537}$
	DBSCAN	$\mathbf{0.5766 \pm 0.0623}$	$\mathbf{0.6833 \pm 0.0473}$	$\mathbf{0.7190 \pm 0.0596}$
ML-LS	User-KNN	0.3426 ± 0.0959	0.4198 ± 0.0497	0.4439 ± 0.0671
	k-Means	0.4770 ± 0.0685	0.5389 ± 0.0641	0.5558 ± 0.0613
	OPF	$\underline{0.5516 \pm 0.0595}$	$\underline{0.6144 \pm 0.0545}$	$\underline{0.6284 \pm 0.0560}$
	OPFSNN	0.4861 ± 0.0912	0.5422 ± 0.0657	0.5484 ± 0.0677
	DBSCAN	$\mathbf{0.5566 \pm 0.0644}$	$\mathbf{0.6215 \pm 0.0508}$	$\mathbf{0.6368 \pm 0.0540}$

Source: Produced by the author.

Table 28 – Average results considering decision support assessment measures over sparse data. The best results are indicated in bold according to the Wilcoxon Signed-Rank Test with 95% of confidence.

Dataset	Approach	Precision@1	Precision@5	Precision@10	Recall@1	Recall@5	Recall@10
AMZ-MS	User-KNN	**0.4898 ± 0.0537**	**0.6014 ± 0.0600**	**0.6030 ± 0.0782**	0.2273 ± 0.0493	0.3288 ± 0.0689	0.2702 ± 0.0847
	k-Means	0.4586 ± 0.0487	0.5742 ± 0.0805	**0.5937 ± 0.1018**	0.2242 ± 0.0305	0.3395 ± 0.0663	0.2997 ± 0.0874
	OPF	0.4453 ± 0.0624	0.5616 ± 0.0730	**0.5827 ± 0.0903**	0.2359 ± 0.0456	0.3517 ± 0.0658	0.3151 ± 0.0864
	OPFSNN	0.4367 ± 0.0440	0.5518 ± 0.0735	**0.5686 ± 0.0958**	**0.2375 ± 0.0353**	**0.3543 ± 0.0703**	**0.3237 ± 0.0816**
	DBSCAN	0.4367 ± 0.0636	0.5631 ± 0.0661	**0.5790 ± 0.0730**	0.2336 ± 0.0460	0.3419 ± 0.0661	0.2927 ± 0.0599
ML-100K	User-KNN	0.4723 ± 0.0593	0.5989 ± 0.0520	0.6797 ± 0.0457	0.2223 ± 0.0455	0.3155 ± 0.0466	0.3227 ± 0.0531
	k-Means	0.5266 ± 0.0464	0.6654 ± 0.0420	0.7274 ± 0.0442	0.2995 ± 0.0380	0.3740 ± 0.0386	**0.3687 ± 0.0436**
	OPF	0.5149 ± 0.0596	0.6244 ± 0.0621	0.6920 ± 0.0659	0.2170 ± 0.0481	0.2523 ± 0.0366	0.2590 ± 0.0487
	OPFSNN	0.5606 ± 0.0520	0.7052 ± 0.0473	0.7677 ± 0.0375	0.3176 ± 0.0409	0.3732 ± 0.0435	0.3526 ± 0.0441
	DBSCAN	**0.6426 ± 0.0556**	**0.8093 ± 0.0371**	**0.8772 ± 0.0319**	**0.3420 ± 0.0610**	**0.3869 ± 0.0407**	0.3589 ± 0.0546
ML-LS	User-KNN	0.4385 ± 0.0534	0.5450 ± 0.0569	0.5578 ± 0.0737	0.2451 ± 0.0533	0.3041 ± 0.0425	0.3064 ± 0.0671
	k-Means	0.5148 ± 0.0533	0.6484 ± 0.0730	0.6487 ± 0.0814	0.3139 ± 0.0395	0.3687 ± 0.0570	0.3592 ± 0.0567
	OPF	0.5705 ± 0.0591	0.7278 ± 0.0594	0.7393 ± 0.0740	**0.3410 ± 0.0559**	**0.3920 ± 0.0517**	0.3779 ± 0.0608
	OPFSNN	0.5164 ± 0.0531	0.6399 ± 0.0620	0.6564 ± 0.0670	0.3197 ± 0.0591	0.3619 ± 0.0665	0.3529 ± 0.0706
	DBSCAN	**0.5754 ± 0.0408**	**0.7304 ± 0.0487**	**0.7456 ± 0.0629**	0.3377 ± 0.0566	0.3907 ± 0.0502	**0.3796 ± 0.0596**

Source: Produced by the author.

both learning and classification, holds at least an influence over the generation of better-ranked samples to be recommended. That may favor the NDCG@K evaluation towards the OPF-based approaches compared to the other baselines, whose algorithms follow distinct learning and/or classification strategies.

Table 28 presents the decision support-based recommendation results. Precision@K and Recall@K have been employed assuming top-1, top-5, and top-10 recommendations. The experiments emphasize similar outcomes regarding prediction accuracy and raking quality, i.e., DBSCAN achieved the best performance on MovieLens datasets followed by both OPFSNN and OPF. Also, the OPF-based approaches led the most consistence overall performances in almost every tested scenario. Still, one may note that User-KNN and k-Means achieved the best and second-best Precision when top-1 and top-5 items are recommended on the AMZ-MS dataset, respectively. Concerning Precision at $K = 10$, one may notice there was not attested statistical dissimilarity among all approaches results. Also, k-Means performed consistently on ML-100K dataset concerning Recall at $K = 5$ and $K = 10$.

According to the results, the proposed approach presented the most consistent overall average Recall@K among all tested approaches while preserving its clustering ability. That finding was particularly evident in the AMZ-MS dataset, which feature the sparsest data and the highest dimensionality of feature vectors. In other words, the proportion of relevant items found in the top-K recommendations outputted by OPFSNN is higher than the ones generated by the baselines.

5.6 Conclusions

In this work, a novel collaborative filtering approach based on the Optimum-Path Forest algorithm, called OPFSNN, has been proposed. The unsupervised version of OPF has been explored by introducing an adjacency relation based on a shared near-neighbor graph. Also, an SNN function based on Jaccard Index has been employed for pairwise distance calculation. The goal was to alleviate data sparsity and high-dimensionality problems through clustering samples before generating recommendations. Given the particularities of OPFSNN, we also introduced a prediction algorithm that takes advantage of shared-neighborhood to cluster new samples.

Extensive experiments have been conducted to test and analyze OPFSNN in the context of clustering and recommendation by exploring dense and sparse datasets, respectively. OPFSNN has been compared against four other approaches through a diverse set of assessment measures, with results statistically indicating the superiority of our approach for both clustering and CF-based recommendation tasks.

The main findings are associated with exploring the SNN graph as OPFSNN adjacency relation and the competition-based learning strategy focused on the path costs optimization. We have also identified that the choice of OPFSNN's distance function seems more related to the data structure itself than to the technical specificities of the former. Regarding clustering, OPFSNN tends to produce more numerous and compact clusters than OPF, also presenting superior recognition rates in terms of V-Measure, in general. Concerning the recommendation task, we emphasize the high-ranking quality attested by the results towards OPFSNN. Besides, the proposed approach presented promising relevant items recommendation in terms of Recall at K. After those considerations, we could state OPFSNN demonstrated a more robust performance, showing a better trade-off between clustering and recommendation than DBSCAN.

As future works, OPFSNN characteristics may enable it to deal with other machine learning tasks such as feature selection through clustering. Also, the incomplete graph-based OPF may benefit from SNN adjacency relation to perform supervised classification since such a version is theoretically similar to the unsupervised counterpart. At last, considering the positive results achieved with dense data, clustering auxiliary information by OPFSNN to mitigate the cold start problem in CF may present a promising direction for future research.

Chapter 6

Conclusions

The current thesis has been focused on examining algorithms and machine learning approaches within the context of collaborative filtering-based recommendation. To achieve this goal, a collection of studies have been proposed during the research period, which included a survey covering most-recent advances in DNN-based recommendation with focus on sparsity-related problems (Chapter 2), a classifier fusion-based approach that explores CF-related sparsity (Chapter 3), as well as studies that investigated, discussed and evaluated unsupervised OPF models designed for handling scalability and data sparsity issues (Chapters 4 and 5). The results contributed to better comprehend the objects of this research and prompted a number of questions, some of which were addressed by the works developed and presented in this thesis.

How are DNN architectures being explored to address CF challenges associated with data sparsity and the cold start problem? The study presented in Chapter 2 aimed at answering that question, and the core idea was to understand how the literature had been investigating the application of neural architectures in recommendation systems. Through a rigorous systematic review, we filtered a select set of primary studies that addressed research questions related to this thesis, particularly data sparsity and the cold start problem. Additionally, we extended this work to encompass more up-to-date research and reassess our conclusions. Regarding DNN architectures, attention mechanisms such as GRU and LSTM are extensively utilized in modeling auxiliary data – especially textual data, which evidently enhances the quality of generated recommendations. Graph-based neural architectures demonstrated to be effective in alleviating the cold start problem, while autoencoders like Denoising AE and Variational AE are primarily designed as independent feature learning modules that jointly encodes side information and user-item interactions to produce richer data representations. Concerning experiments, the primary

issue remains the absence of a universally standardized set of procedures to establish robust experimental protocols. Thus, employing best practices from other machine learning application domains can assist the standardization process of experiments in recommendation research. In contrast, there has been progress in recent years when it comes to designing specialized experimental strategies for assessing the cold start problem.

Despite primary studies claiming the superiority of their approaches in virtually all analyzed scenarios, the experimental fragility of most of these works raises doubts about the validity of such assertions. Therefore, given that the most common tactic for improving recommendation quality is to incorporate auxiliary data together with user-item interactions, we understand that the effectiveness of those DNN-based models is probably more related to modeling great volumes of dense data like textual reviews, interaction time stamps, user profiles, and other internal item content, rather than their capability to embed sparse user-item interactions.

How can we leverage the data sparsity inherent in CF for applications beyond recommendation? We addressed this question by introducing a classification fusion approach that explores sparse DT via a CF-based optimization of latent factor models (Chapter 3). The main idea was to employ matrix factorization techniques under a CF setting to predict labels when we have sparse DT generated over the training data. The experimental results demonstrate the proposed approach was able to do fast prediction estimation of classifiers, making it feasible for decision template generation in scenarios with low data sparsity and datasets featuring fewer classes. In such conditions, the proposed framework demonstrated statistically competitive recognition rates while notably improving computational efficiency, particularly in situations distance-based classifiers like OPF and KNN are employed for ensemble learning.

Can the combination of k-approximate nearest neighbor techniques with OPF improve its scalability without a substantial loss in recognition rates? In Chapter 4, we introduced a GMM parameter initialization approach based on a alternative version of OPF clustering that explores the k-Approximate Nearest Neighbors graph for building its adjacency relation, namely OPF$_{k\text{-ann}}$. The results indicated the proposed approach was able to encode Gaussian parameters more naturally and intuitively compared to other clustering algorithms such as k-means. Also, OPF$_{k\text{-ann}}$ demonstrated greater scalability compared to traditional OPF clustering, significantly reducing computation cost in both graph construction and partitioning optimization as the k-neighborhood size increases.

Would exploring the concept of shared neighborhood enable OPF to mitigate the effects of sparsity and high dimensionality in collaborative filtering? The study presented in Chapter 5 introduced the OPFSNN, a novel unsupervised Optimum-Path Forest approach for clustering-based recommendation that employs a new adjacency relation based on shared near neighbors as the key strategy for handling data sparsity and high dimensionality issues of CF data. By assessing different clustering and recommendation aspects

and comparing OPFSNN against a set of clustering algorithms and recommendation approaches, the results statistically demonstrated superior performance towards the proposed approach for clustering dense data and a consistent CF-based recommendation ability when dealing with sparse data.

6.1 Publications and other works

Table 29 provides a compilation of studies conducted during the research period.

Table 29 – A list of works developed by the author.

Paper description	Type	Qualis	Year	Status
Deep learning techniques for recommender systems based on collaborative filtering (MARTINS; PAPA; ADELI, 2020)	Journal	A2	2020	Published
Collaborative Filtering Matches Decision Templates: A Practical Approach to Estimate Predictions (MARTINS; PAPA, 2022a)	Conference	A3	2022	Published
How to proper initialize Gaussian Mixture Models with Optimum-Path Forest (MARTINS; PAPA, 2022b)	Conference	A3	2022	Published
Temporal Dengue Outbreak Prediction from Climatic Variables using Finite Element Machines for Regression (LYDIA et al., 2023)	Conference	A3	2023	Published
OPFSNN: A Novel Optimum-Path Forest Clustering Approach based on Shared Near Neighbors for Collaborative Filtering Recommendation	Journal	A1	2023	Submitted

6.2 Future works

The works developed in this thesis pave the way for future research. Despite the relatively promising results obtained with the CF-based approach for generating decision templates, we acknowledge the necessity of an evaluation employing extensive datasets, i.e., with millions of samples at least. Additionally, we intend to explore DNN architectures to understand the impact of deep autoencoders during the nonlinear process of latent factors learning instead of only employing linear MF-based techniques.

Regarding the research conducted with unsupervised OPF, we recognize the potential for progress concerning changes in the adjacency relation of that classifier. Thus, we aim to integrate SNN with k-approximate neighborhood techniques to enhance computational efficiency during the OPFSNN graph construction. Another interesting possibility is to apply the SNN as the adjacency relation of supervised OPF-KNN. Besides its theoretical similarity to OPF clustering, we understand the OPFSNN capability to output more numerous and compact clusters may benefit OPF-KNN recognition rates during supervised classification tasks. Such behavior could also be explored for other machine learning-related tasks like clustering-based feature selection.

At last, regarding the recommendation task, we intend to address the cold start problem through generative deep architectures such as conditional VAE, aiming to reconstruct

cold users from dense data during the model learning process. We intend to verify both user-item interactions and auxiliary information as dense data sources during experiments.

Bibliography

ADOMAVICIUS, G.; TUZHILIN, A. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. **IEEE Trans. on Knowl. and Data Eng.**, IEEE Educational Activities Department, Piscataway, NJ, USA, v. 17, n. 6, p. 734–749, 2005. ISSN 1041-4347.

AGGARWAL, C. C. An introduction to recommender systems. In: **Recommender Systems**. Switzerland: Springer International Publishing, 2016. p. 1–28. ISBN 978-3-319-29657-9.

_____. **Recommender Systems**. 1. ed. Switzerland: Springer International Publishing, 2016. v. 40. 1–518 p. ISSN 00010782. ISBN 978-3-319-29657-9.

AHMADLOU, M.; ADELI, H. Enhanced probabilistic neural network with local decision circles: A robust classifier. In: **Integrated Computer-Aided Engineering**. NLD: IOS Press, 2010. v. 17, n. 3, p. 197–210. ISSN 1069-2509.

AHMED, A. et al. Trust-aware denoising autoencoder with spatial-temporal activity for cross-domain personalized recommendations. **Neurocomputing**, Elsevier B.V., v. 511, p. 477–494, 2022. ISSN 18728286.

ALHARBE, N.; RAKROUKI, M. A.; ALJOHANI, A. A collaborative filtering recommendation algorithm based on embedding representation. **Expert Systems with Applications**, v. 215, p. 119380, 2023. ISSN 0957-4174.

AMORIM, W. P.; FALCÃO, A. X.; CARVALHO, M. Semi-supervised pattern classification using optimum-path forest. In: **Brazilian Symposium of Computer Graphic and Image Processing**. Rio de Janeiro, Brazil: IEEE, 2014. p. 111–118.

ANSARI, A. H. et al. Neonatal Seizure Detection Using Deep Convolutional Neural Networks. **International Journal of Neural Systems**, 2019. ISSN 17936462.

ANTONIADES, A. et al. Deep Neural Architectures for Mapping Scalp to Intracranial EEG. **International Journal of Neural Systems**, 2018. ISSN 17936462.

BAI, T. et al. A neural collaborative filtering model with interaction-based neighborhood. **International Conference on Information and Knowledge Management, Proceedings**, Part F1318, p. 1979–1982, 2017.

BALABANOVIC, M.; SHOHAM, Y. Fab: Content-based, collaborative recommendation. **Commun. ACM**, Association for Computing Machinery, New York, NY, USA, v. 40, n. 3, p. 66–72, mar 1997. ISSN 0001-0782.

BANG, S. et al. Encoder–decoder network for pixel-level road crack detection in black-box images. **Computer-Aided Civil and Infrastructure Engineering**, 2019. ISSN 14678667.

BARKAN, O.; KOENIGSTEIN, N. ITEM2VEC: Neural item embedding for collaborative filtering. In: **IEEE International Workshop on Machine Learning for Signal Processing, MLSP**. Vietri sul Mare, Italy: IEEE, 2016. p. 1–6. ISBN 9781509007462. ISSN 21610371.

BELLINI, V. et al. Auto-encoding user ratings via knowledge graphs in recommendation scenarios. **ACM International Conference Proceeding Series**, Part F1301, p. 60–66, 2017.

BLÖMER, J.; BUJNA, K. Adaptive seeding for gaussian mixture models. In: **Advances in Knowledge Discovery and Data Mining**. Cham: Springer International Publishing, 2016. p. 296–308. ISBN 978-3-319-31750-2.

BOBADILLA, J. et al. Recommender systems survey. **Knowledge-Based Systems**, v. 46, p. 109 – 132, 2013. ISSN 0950-7051.

BREESE, J. S.; HECKERMAN, D.; KADIE, C. Empirical analysis of predictive algorithms for collaborative filtering. **arXiv preprint arXiv:1301.7363**, 2013.

CAMACHO, L.; ALVES-SOUZA, S. Social network data to alleviate cold-start in recommender system: A systematic review. **Information Processing & Management**, v. 54, n. 4, p. 529 – 544, 2018. ISSN 0306-4573.

CAO, D. et al. Attentive group recommendation. **41st International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2018**, p. 645–654, 2018.

CAPPABIANCO, F. A. et al. Brain tissue MR-image segmentation via optimum-path forest clustering. **Computer Vision and Image Understanding**, v. 116, n. 10, p. 1047–1059, 2012. ISSN 10773142.

CHAE, D. K.; SHIN, J. A.; KIM, S. W. Collaborative Adversarial Autoencoders: An Effective Collaborative Filtering Model under the GAN Framework. **IEEE Access**, IEEE, v. 7, p. 37650–37663, 2019. ISSN 21693536.

CHEN, C. et al. Efficient Neural Matrix Factorization without Sampling for Recommendation. **ACM Transactions on Information Systems (TOIS)**, v. 38, n. 2, p. 1–28, feb 2020. ISSN 1046-8188.

CHEN, J. et al. Collaborative filtering recommendation algorithm based on user correlation and evolutionary clustering. **Complex & Intelligent Systems**, Springer International Publishing, v. 6, n. 1, p. 147–156, 2020. ISSN 2199-4536.

CHEN, L. et al. Heterogeneous neural attentive factorization machine for rating prediction. **International Conference on Information and Knowledge Management, Proceedings**, p. 833–842, 2018.

CHEN, R. et al. A survey of collaborative filtering-based recommender systems: from traditional methods to hybrid methods based on social networks. **IEEE Access**, IEEE, v. 6, p. 64301–64320, 2018. ISSN 21693536.

CHEN, S. et al. An improved optimum-path forest clustering algorithm for remote sensing image segmentation. **Computers and Geosciences**, Elsevier Ltd, v. 112, n. December 2017, p. 38–46, 2018. ISSN 00983004.

CHEN, W. et al. Joint Neural Collaborative Filtering for Recommender Systems. **ACM Transactions on Information Systems**, v. 37, n. 4, p. 1–30, aug 2019. ISSN 10468188.

CHEN, X. et al. Adversarial distillation for efficient recommendation with external knowledge. **ACM Transactions on Information Systems**, v. 37, n. 1, p. 1–28, 2019. ISSN 15582868.

CHEN, Y. et al. Collaborative filtering grounded on knowledge graphs. **Pattern Recognition Letters**, v. 151, p. 55–61, 2021. ISSN 0167-8655.

CHIN, J. Y. et al. ANR: Aspect-based Neural Recommender. **International Conference on Information and Knowledge Management, Proceedings**, p. 147–156, 2018.

COSTA, K. A. et al. A nature-inspired approach to speed up optimum-path forest clustering and its application to intrusion detection in computer networks. **Information Sciences**, Elsevier, v. 294, p. 95–108, 2015.

COVINGTON, P.; ADAMS, J.; SARGIN, E. Deep neural networks for youtube recommendations. In: **Proceedings of the 10th ACM Conference on Recommender Systems**. New York, NY, USA: Association for Computing Machinery, 2016.

Da Costa, F. S.; DOLOG, P. Collective embedding for neural context-aware recommender systems. **RecSys 2019 - 13th ACM Conference on Recommender Systems**, p. 201–209, 2019.

DACREMA, M. F.; CREMONESI, P.; JANNACH, D. Are we really making much progress? A worrying analysis of recent neural recommendation approaches. In: **Proceedings of the 13th ACM conference on recommender systems**. New York, NY, USA: Association for Computing Machinery, 2019. (RecSys '19), p. 101–109. ISBN 9781450362436.

DAVIES, D. L.; BOULDIN, D. W. A cluster separation measure. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, PAMI-1, n. 2, p. 224–227, 1979.

de Rosa, G. H.; PAPA, J. P. Opfython: A python implementation for optimum-path forest. **Software Impacts**, p. 100113, 2021. ISSN 2665-9638.

DEVLIN, J. et al. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. n. Mlm, 2018.

DO, P. M. T.; NGUYEN, T. T. S. Semantic-enhanced neural collaborative filtering models in recommender systems. **Knowledge-Based Systems**, v. 257, p. 109934, 2022. ISSN 0950-7051.

DUA, D.; GRAFF, C. **UCI Machine Learning Repository**. 2019.

DUAN, R.; JIANG, C.; JAIN, H. K. Combining review-based collaborative filtering and matrix factorization: A solution to rating's sparsity problem. **Decision Support Systems**, v. 156, p. 113748, 2022. ISSN 0167-9236.

EKSTRAND, M. D.; RIEDL, J. T.; KONSTAN, J. A. Collaborative filtering recommender systems. **Foundations and Trends® in Human–Computer Interaction**, v. 4, n. 2, p. 81–173, 2011. ISSN 1551-3955.

ELAHI, M.; RICCI, F.; RUBENS, N. A survey of active learning in collaborative filtering recommender systems. **Computer Science Review**, v. 20, n. Supplement C, p. 29–50, 2016. ISSN 1574-0137.

ERTÖZ, L.; STEINBACH, M.; KUMAR, V. Finding clusters of different sizes, shapes, and densities in noisy, high dimensional data. In: **Proceedings of the 2003 SIAM international conference on data mining**. San Franscisco, California, USA: Society for Industrial and Applied Mathematics, 2003. p. 47–58.

ESTER, M. et al. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In: **Proceedings of the Second International Conference on Knowledge Discovery and Data Mining**. Portland, Oregon, USA: AAAI Press, 1996. (KDD'96, 34), p. 226–231.

FAN, C.; MOSTAFAVI, A. A graph-based method for social sensing of infrastructure disruptions in disasters. **Computer-Aided Civil and Infrastructure Engineering**, 2019. ISSN 14678667.

FAN, W. et al. Deep social collaborative filtering. **RecSys 2019 - 13th ACM Conference on Recommender Systems**, p. 305–313, 2019.

FENG, X.; ZENG, Y. Multi-Level Fine-Grained Interactions for Collaborative Filtering. **IEEE Access**, IEEE, v. 7, p. 143169–143184, 2019. ISSN 21693536.

_____. Neural Collaborative Embedding From Reviews for Recommendation. **IEEE Access**, v. 7, p. 103263–103274, 2019.

FU, M. et al. A Novel Deep Learning-Based Collaborative Filtering Model for Recommendation System. **IEEE Transactions on Cybernetics**, v. 49, n. 3, p. 1084–1096, 2019. ISSN 21682267.

GAO, Y.; KONG, B.; MOSALAM, K. M. Deep leaf-bootstrapping generative adversarial network for structural image data augmentation. **Computer-Aided Civil and Infrastructure Engineering**, 2019. ISSN 14678667.

GONZALEZ, T. F. Clustering to minimize the maximum intercluster distance. **Theoretical computer science**, Elsevier, v. 38, p. 293–306, 1985.

GROVER, A.; LESKOVEC, J. node2vec: Scalable feature learning for networks. In: **Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining**. New York, NY, USA: Association for Computing Machinery, 2016. p. 855–864. ISBN 9781450342322.

GUO, L. et al. Collaborative filtering recommendations based on multi-factor random walks. **Engineering Applications of Artificial Intelligence**, Elsevier, v. 123, p. 106409, 2023.

HARPER, F. M.; KONSTAN, J. A. The movielens datasets: history and context. **ACM Transactions on Interactive Intelligent Systems**, ACM New York, NY, USA, v. 5, n. 4, p. 1–19, 2015. Disponível em: <https://grouplens.org/datasets/movielens/1m/>.

HE, M.; MENG, Q.; ZHANG, S. Collaborative Additional Variational Autoencoder for Top-N Recommender Systems. **IEEE Access**, IEEE, v. 7, p. 5707–5713, 2019. ISSN 21693536.

HE, X. et al. Neural collaborative filtering. **26th International World Wide Web Conference, WWW 2017**, p. 173–182, 2017.

HERLOCKER, J. L.; KONSTAN, J. A.; RIEDL, J. Explaining collaborative filtering recommendations. In: ACM. **Proceedings of the 2000 ACM conference on Computer supported cooperative work**. New York, NY, USA: Association for Computing Machinery, 2000. p. 241–250.

HIEN, T. D. et al. Novel algorithm for non-negative matrix factorization. **New Mathematics and Natural Computation**, v. 11, n. 2, p. 121–133, 2015. ISSN 17937027.

HU, P. et al. Hybrid item-item recommendation via semi-parametric embedding. **IJCAI International Joint Conference on Artificial Intelligence**, p. 2521–2527, 2019. ISSN 10450823.

HU, Y.; VOLINSKY, C.; KOREN, Y. Collaborative filtering for implicit feedback datasets. In: **Proceedings - IEEE International Conference on Data Mining, ICDM**. Pisa, Italy: IEEE, 2008. p. 263–272. ISBN 9780769535029. ISSN 15504786.

HUA, C. et al. A Novel Method of Building Functional Brain Network Using Deep Learning Algorithm with Application in Proficiency Detection. **International Journal of Neural Systems**, 2019. ISSN 17936462.

HUANG, J. et al. Improving sequential recommendation with knowledge-enhanced memory networks. **41st International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2018**, p. 505–514, 2018.

HUANG, Y. et al. Path-enhanced explainable recommendation with knowledge graphs. **World Wide Web**, World Wide Web, v. 24, n. 5, p. 1769–1789, 2021. ISSN 15731413.

HUG, N. Surprise: A python library for recommender systems. **Journal of Open Source Software**, The Open Journal, v. 5, n. 52, p. 2174, 2020.

HYUN, D. et al. Review sentiment-guided scalable deep recommender system. **41st International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2018**, p. 965–968, 2018.

JARVIS, R.; PATRICK, E. Clustering using a similarity measure based on shared near neighbors. **IEEE Transactions on Computers**, C-22, n. 11, p. 1025–1034, 1973.

JHAMB, Y.; EBESU, T.; FANG, Y. Attentive contextual denoising autoencoder for recommendation. **ICTIR 2018 - Proceedings of the 2018 ACM SIGIR International Conference on the Theory of Information Retrieval**, p. 27–34, 2018.

KIM, D. et al. Convolutional matrix factorization for document context-aware recommendation. In: **RecSys 2016 - Proceedings of the 10th ACM Conference on Recommender Systems**. New York, NY, USA: Association for Computing Machinery, 2016. ISBN 9781450340359.

KIPF, T. N.; WELLING, M. Semi-Supervised Classification with Graph Convolutional Networks. p. 1–14, 2016.

KITCHENHAM, B. **Procedures for Performing Systematic Reviews**. Department of Computer Science, Keele University, UK, 2004.

KOREN, Y. Collaborative filtering with temporal dynamics. In: **Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining**. New York, NY, USA: ACM, 2009. (KDD '09), p. 447–456. ISBN 978-1-60558-495-9.

_____. Factor in the neighbors. **ACM Transactions on Knowledge Discovery from Data**, v. 4, n. 1, p. 1–24, jan 2010. ISSN 15564681.

KOREN, Y.; BELL, R.; VOLINSKY, C. Matrix factorization techniques for recommender systems. **Computer**, IEEE, v. 42, n. 8, 2009.

KOREN, Y.; RENDLE, S.; BELL, R. Advances in Collaborative Filtering. In: _____. **Recommender Systems Handbook**. New York, NY: Springer US, 2022. p. 91–142. ISBN 978-1-0716-2197-4.

KUNCHEVA, L. I. Fusion of Continuous-Valued Outputs. In: **Combining Pattern Classifiers**. Hoboken, NJ: John Wiley & Sons, Ltd, 2004. cap. 5, p. 151–188. ISBN 9780471660262.

LAVANYA, R.; BHARATHI, B. Movie recommendation system to solve data sparsity using collaborative filtering approach. **ACM Trans. Asian Low-Resour. Lang. Inf. Process.**, Association for Computing Machinery, New York, NY, USA, v. 20, n. 5, jul 2021. ISSN 2375-4699.

LECUN, Y.; BENGIO, Y.; HINTON, G. Deep learning. **Nature**, Nature Publishing Group, a division of Macmillan Publishers Limited. All Rights Reserved., v. 521, p. 436, may 2015.

LEE, W.; SONG, K.; MOON, I. C. Augmented variational autoencoders for collaborative filtering with auxiliary information. **International Conference on Information and Knowledge Management, Proceedings**, Part F1318, p. 1139–1148, 2017.

LEE, Y. et al. How to Impute Missing Ratings?: Claims, Solution, and Its Application to Collaborative Filtering. **Proceedings of the 2018 World Wide Web Conference**, v. 2, p. 783–792, 2018.

LI, M.; WEN, L.; CHEN, F. A novel Collaborative Filtering recommendation approach based on Soft Co-Clustering. **Physica A: Statistical Mechanics and its Applications**, Elsevier B.V., v. 561, p. 125140, 2021. ISSN 03784371.

LI, S.; ZHAO, X.; ZHOU, G. Automatic pixel-level multiple damage detection of concrete structure using fully convolutional network. **Computer-Aided Civil and Infrastructure Engineering**, 2019. ISSN 14678667.

LI, Z. et al. Learning from history and present: Next-item recommendation via discriminatively exploiting user behaviors. **Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining**, p. 1734–1743, 2018.

LIANG, D. et al. Variational Autoencoders for Collaborative Filtering. In: **Proceedings of the 2018 World Wide Web Conference on World Wide Web - WWW '18**. New York, New York, USA: ACM Press, 2018. p. 689–698. ISBN 9781450356398.

_____. Variational Autoencoders for Collaborative Filtering. In: **Proceedings of the 2018 World Wide Web Conference on World Wide Web - WWW '18**. New York, New York, USA: ACM Press, 2018. p. 689–698. ISBN 9781450356398.

LIANG, X. Image-based post-disaster inspection of reinforced concrete bridge systems using deep learning with Bayesian optimization. **Computer-Aided Civil and Infrastructure Engineering**, 2019. ISSN 14678667.

LIU, D. et al. DAML: Dual attention mutual learning between ratings and reviews for item recommendation. **Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining**, p. 344–352, 2019.

LIU, J.; WANG, D.; DING, Y. PhD: A probabilistic model of hybrid deep collaborative filtering for recommender systems. **Journal of Machine Learning Research**, v. 77, p. 224–239, 2017. ISSN 15337928.

LIU, X.; CHEUNG, Y.-M. On incremental collaborative appearance model and regional particle filtering for lip region tracking. **Integrated Computer-Aided Engineering**, v. 25, p. 63–80, 12 2018. ISSN 10692509.

LUO, H.; PAAL, S. G. A locally weighted machine learning model for generalized prediction of drift capacity in seismic vulnerability assessments. **Computer-Aided Civil and Infrastructure Engineering**, 2019. ISSN 14678667.

LYDIA, M. et al. Temporal dengue outbreak prediction from climatic variables using finite element machines for regression. In: IEEE. **2023 30th International Conference on Systems, Signals and Image Processing (IWSSIP)**. Ohrid, North Macedonia: IEEE, 2023. p. 1–5.

MAEDA, H. et al. Road Damage Detection and Classification Using Deep Neural Networks with Smartphone Images. **Computer-Aided Civil and Infrastructure Engineering**, 2018. ISSN 14678667.

MAEDA, K. et al. Convolutional sparse coding-based deep random vector functional link network for distress classification of road structures. **Computer-Aided Civil and Infrastructure Engineering**, 2019. ISSN 14678667.

MAGRON, P.; FÉVOTTE, C. Neural content-aware collaborative filtering for cold-start music recommendation. **Data Mining and Knowledge Discovery**, Springer US, v. 36, n. 5, p. 1971–2005, 2022. ISSN 1573756X.

MALKOV, Y. A.; YASHUNIN, D. A. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. **IEEE transactions on pattern analysis and machine intelligence**, IEEE, v. 42, n. 4, p. 824–836, 2018.

MANOTUMRUKSA, J.; MACDONALD, C.; OUNIS, I. A deep recurrent collaborative filtering framework for venue recommendation. **International Conference on Information and Knowledge Management, Proceedings**, Part F1318, p. 1429–1438, 2017.

MANZANERA, O. M. et al. Scaled Subprofile Modeling and Convolutional Neural Networks for the Identification of Parkinson's Disease in 3D Nuclear Imaging Data. **International Journal of Neural Systems**, 2019. ISSN 17936462.

MARLIN, B. Collaborative filtering: A machine learning perspective. **Master Thesis**, p. 137, 2004. ISSN 10414347.

MAROTO, J.; VIGNAC, C.; FROSSARD, P. Modurec: Recommender systems with feature and time modulation. **ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings**, v. 2021-June, p. 3615–3619, 2021. ISSN 15206149.

MARTINS, G. B.; PAPA, J. P. Collaborative filtering matches decision templates: A practical approach to estimate predictions. In: **2022 35th SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)**. Natal, Brazil: IEEE, 2022. v. 1, p. 186–191.

_____. How to proper initialize gaussian mixture models with optimum-path forest. In: **2022 35th SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)**. Natal, Brazil: IEEE, 2022. v. 1, p. 127–132.

MARTINS, G. B.; PAPA, J. P.; ADELI, H. Deep learning techniques for recommender systems based on collaborative filtering. **Expert Systems**, Wiley, 2020.

MARTINS, G. B. et al. Opfsumm: On the video summarization using optimum-path forest. **Multimedia Tools Appl.**, Kluwer Academic Publishers, USA, v. 79, n. 15–16, p. 11195–11211, apr 2020. ISSN 1380-7501.

MIKOLOV, T. et al. Distributed representations of words and phrases and their compositionality. **Advances in Neural Information Processing Systems**, p. 1–9, 2013. ISSN 10495258.

MOLAEI, S. et al. Collaborative Deep Forest Learning for Recommender Systems. **IEEE Access**, v. 9, p. 22053–22061, 2021. ISSN 21693536.

MOLINA-CABELLO, M. A. et al. Vehicle type detection by ensembles of convolutional neural networks operating on super resolved images. **Integrated Computer-Aided Engineering**, IOS Press, NLD, 2018. ISSN 18758835.

NAHTA, R. et al. Two-step hybrid collaborative filtering using deep variational Bayesian autoencoders. **Information Sciences**, Elsevier Inc., v. 562, p. 136–154, 2021. ISSN 00200255.

NAKAGAWA, E. et al. **Revisão Sistemática da Literatura em Engenharia de Software: Teoria e Prática**. Rio de Janeiro, Brasil: Elsevier Brasil, 2017. ISBN 9788535285970.

NGUYEN, T. et al. Deep neural network with high-order neuron for the prediction of foamed concrete strength. **Computer-Aided Civil and Infrastructure Engineering**, 2019. ISSN 14678667.

NI, J.; LI, J.; MCAULEY, J. Justifying recommendations using distantly-labeled reviews and fine-grained aspects. In: **Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)**. Hong Kong, China: Association for Computational Linguistics, 2019. p. 188–197.

NIKZAD–KHASMAKHI, N.; BALAFAR, M.; FEIZI–DERAKHSHI, M. R. The state-of-the-art in expert recommendation systems. **Engineering Applications of Artificial Intelligence**, v. 82, p. 126 – 147, 2019. ISSN 0952-1976.

NISHA, C. C.; MOHAN, A. A social recommender system using deep architecture and network embedding. **Applied Intelligence**, Applied Intelligence, v. 49, n. 5, p. 1937–1953, 2019. ISSN 15737497.

OTUNBA, R.; RUFAI, R. A.; LIN, J. Deep stacked ensemble recommender. **ACM International Conference Proceeding Series**, p. 197–201, 2019.

PAI, M. et al. Systematic reviews and meta-analyses: an illustrated, step-by-step guide. **The National medical journal of India**, v. 17 2, p. 86–95, 2004.

PAPA, J. P.; AO, A. X. F. A new variant of the optimum-path forest classifier. **International Symposium on Visual Computing**, v. 47, n. 1, p. 935–944, 2008.

PAPA, J. P. et al. Efficient supervised optimum-path forest classification for large datasets. **Pattern Recognition**, Elsevier Science Inc., New York, NY, USA, v. 45, n. 1, p. 512–520, 2012.

PAPA, J. P.; FALCÃO, A. X.; SUZUKI, C. T. N. Supervised pattern classification based on optimum-path forest. **International Journal of Imaging Systems and Technology**, John Wiley & Sons, Inc., New York, NY, USA, v. 19, n. 2, p. 120–131, 2009. ISSN 0899-9457.

PASSOS, L. A. et al. Handling imbalanced datasets through optimum-path forest. **Knowledge-Based Systems**, Elsevier, v. 242, p. 108445, 2022.

PEDREGOSA, F. et al. Scikit-learn: Machine learning in Python. **Journal of Machine Learning Research**, v. 12, p. 2825–2830, 2011.

PEREIRA, D. R. et al. FEMa: a finite element machine for fast learning. **Neural Computing and Applications**, 2020. ISSN 14333058.

PORTUGAL, I.; ALENCAR, P.; COWAN, D. The use of machine learning algorithms in recommender systems: A systematic review. **Expert Systems with Applications**, v. 97, p. 205 – 227, 2018. ISSN 0957-4174.

POUDEL, S.; BIKDASH, M. Collaborative Filtering system based on multi-level user clustering and aspect sentiment. **Data and Information Management**, Wuhan University, v. 6, n. 4, p. 100021, 2022. ISSN 2543-9251.

QIAN, T. et al. Attribute Graph Neural Networks for Strict Cold Start Recommendation. **IEEE Transactions on Knowledge and Data Engineering**, IEEE, v. 34, n. 8, p. 3597–3610, 2022. ISSN 15582191.

QIANG, J. et al. Snapshot ensembles of non-negative matrix factorization for stability of topic modeling. **Applied Intelligence**, Springer, v. 48, n. 11, p. 3963–3975, 2018.

QIU, L.; FANG, F.; YUAN, S. Improved density peak clustering-based adaptive Gaussian mixture model for damage monitoring in aircraft structures under time-varying conditions. **Mechanical Systems and Signal Processing**, Elsevier Ltd, v. 126, p. 281–304, 2019. ISSN 10961216.

RAFIEI, M. H.; ADELI, H. A New Neural Dynamic Classification Algorithm. **IEEE Transactions on Neural Networks and Learning Systems**, 2017. ISSN 21622388.

RENDLE, S. et al. Bpr: Bayesian personalized ranking from implicit feedback. **arXiv preprint arXiv:1205.2618**, 2012.

REYES, O.; VENTURA, S. Performing Multi-Target Regression via a Parameter Sharing-Based Deep Network. **International Journal of Neural Systems**, 2019. ISSN 17936462.

RICCI, F.; ROKACH, L.; SHAPIRA, B. **Recommender systems Handbook**. Third. New York, NY: Springer US, 2021. ISBN 9781071621974.

ROCHA, L. M.; CAPPABIANCO, F. A. M.; FALCÃO, A. X. Data clustering as an optimum-path forest problem with applications in image analysis. **International Journal of Imaging Systems and Technology**, Wiley Periodicals, v. 19, n. 2, p. 50–68, 2009.

ROSA, G. H. et al. On the training of artificial neural networks with radial basis function using optimum-path forest clustering. **Proceedings - International Conference on Pattern Recognition**, p. 1472–1477, 2014. ISSN 10514651.

ROSA, G. H. de; PAPA, J. P. **Speeding Up OPFython with Numba**. 2021.

ROSENBERG, A.; HIRSCHBERG, J. V-measure: A conditional entropy-based external cluster evaluation measure. In: **Proceedings of the 2007 joint conference on empirical methods in natural language processing and computational natural language learning (EMNLP-CoNLL)**. Prague, Czech Republic: Association for Computational Linguistics, 2007. p. 410–420.

RUOCCO, M.; SKREDE, O. S. L.; LANGSETH, H. Inter-session modeling for session-based recommendation. **ACM International Conference Proceeding Series**, Part F1301, p. 24–31, 2017.

SALAKHUTDINOV, R.; MNIH, A. Probabilistic matrix factorization. **Advances in Neural Information Processing Systems 20 - Proceedings of the 2007 Conference**, p. 1–8, 2009.

SARWAR, B. et al. Item-based collaborative filtering recommendation algorithms. **Proceedings of the 10th International Conference on World Wide Web, WWW 2001**, v. 1, p. 285–295, 2001. ISSN 09501991.

SEDHAIN, S. et al. Autorec: Autoencoders meet collaborative filtering. In: **Proceedings of the 24th international conference on World Wide Web**. New York, NY, USA: Association for Computing Machinery, 2015. p. 111–112.

SHEN, J. et al. A convolutional neural-network-based pedestrian counting model for various crowded scenes. **Computer-Aided Civil and Infrastructure Engineering**, 2019. ISSN 14678667.

SHI, J.; MALIK, J. Normalized cuts and image segmentation. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, IEEE Computer Society, Washington, DC, USA, v. 22, n. 8, p. 888–905, 2000. ISSN 0162-8828.

SHI, Y.; LARSON, M.; HANJALIC, A. Collaborative Filtering beyond the User-Item Matrix. **ACM Computing Surveys**, v. 47, n. 1, p. 1–45, 2014. ISSN 03600300.

SHIREMAN, E.; STEINLEY, D.; BRUSCO, M. J. Examining the effect of initialization strategies on the performance of gaussian mixture modeling. **Behavior research methods**, Springer, v. 49, n. 1, p. 282–293, 2017.

SIDANA, S. et al. User preference and embedding learning with implicit feedback for recommender systems. **Data Mining and Knowledge Discovery**, Springer US, v. 35, n. 2, p. 568–592, 2021. ISSN 1573756X.

SONG, B. et al. Neural collaborative ranking. **International Conference on Information and Knowledge Management, Proceedings**, p. 1353–1362, 2018.

SONG, Q.; CHANG, S.; HU, X. Coupled Variational Recurrent Collaborative Filtering. n. 61, p. 2000–2025, 2019.

SU, X.; KHOSHGOFTAAR, T. M. A survey of collaborative filtering techniques. **Advances in Artificial. Intelligence.**, Hindawi Publishing Corp., New York, NY, United States, v. 2009, 2009. ISSN 1687-7470.

SUH, S. et al. Localized user-driven topic discovery via boosted ensemble of nonnegative matrix factorization. **Knowledge and Information Systems**, Springer, v. 56, n. 3, p. 503–531, 2018.

SUN, K. et al. What can history tell us? Identifying relevant sessions for next-item recommendation. **International Conference on Information and Knowledge Management, Proceedings**, p. 1593–1602, 2019.

SUN, P.; WU, L.; WANG, M. Attentive recurrent social recommendation. **41st International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2018**, p. 185–194, 2018.

TANUMA, I.; MATSUI, T. Variational Autoencoder-Based Hybrid Recommendation With Poisson Factorization for Modeling Implicit Feedback. **IEEE Access**, IEEE, v. 10, p. 60696–60706, 2022. ISSN 21693536.

TAY, Y.; Anh Tuan, L.; HUI, S. C. Latent Relational Metric Learning via Memory-based Attention for Collaborative Ranking. p. 729–739, 2018.

TORRES, J. F. et al. A scalable approach based on deep learning for big data time series forecasting. **Integrated Computer-Aided Engineering**, 2018. ISSN 18758835.

VERA-OLMOS, F. J. et al. DeepEye: Deep convolutional network for pupil detection in real environments. **Integrated Computer-Aided Engineering**, 2018. ISSN 18758835.

WANG, C. et al. A light heterogeneous graph collaborative filtering model using textual information. **Knowledge-Based Systems**, Elsevier B.V., v. 234, p. 107602, 2021. ISSN 09507051.

WANG, J.; CAVERLEE, J. Recurrent recommendation with local coherence. **WSDM 2019 - Proceedings of the 12th ACM International Conference on Web Search and Data Mining**, p. 564–572, 2019.

WANG, M. et al. A collaborative session-based recommendation approach with parallel memory modules. **SIGIR 2019 - Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval**, p. 345–354, 2019.

WANG, P.; BAI, X. Regional parallel structure based CNN for thermal infrared face identification. **Integrated Computer-Aided Engineering**, IOS Press, NLD, 2018. ISSN 18758835.

WANG, Q. et al. Enhancing collaborative filtering with generative augmentation. **Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining**, p. 548–556, 2019.

_____. DCCR: Deep Collaborative Conjunctive Recommender for Rating Prediction. **IEEE Access**, IEEE, v. 7, p. 60186–60198, 2019. ISSN 21693536.

WANG, R. et al. TDCF: A two-stage deep learning based recommendation model. **Expert Systems with Applications**, Elsevier Ltd, v. 145, p. 113116, 2020. ISSN 09574174.

WANG, X. et al. Exploiting Social Review-Enhanced Convolutional Matrix Factorization for Social Recommendation. **IEEE Access**, IEEE, v. 7, p. 82826–82837, 2019. ISSN 21693536.

WEBB, A. R.; COPSEY, K. D. **Statistical Pattern Recognition: Third Edition**. New York, USA: Wiley, 2011. v. 9. 0–470 p. ISBN 9780470682289.

WILCOXON, F. Individual comparisons by ranking methods. **Biometrics Bulletin**, v. 1, p. 80–83, 1945.

WITTEN, I. H.; FRANK, E.; Mark A. Hall. **Data Mining: Practical machine learning tools and techniques**. Third. Amsterdam, The Netherlands: Elsevier Science, 2011. 630 p. ISBN 9780123748560.

WU, C. Y. et al. Recurrent recommender networks. **WSDM 2017 - Proceedings of the 10th ACM International Conference on Web Search and Data Mining**, p. 495–503, 2017.

WU, L. et al. A neural influence diffusion model for social recommendation. **SIGIR 2019 - Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval**, p. 235–244, 2019.

WU, X. et al. Neural tensor factorization for temporal interaction learning. **WSDM 2019 - Proceedings of the 12th ACM International Conference on Web Search and Data Mining**, p. 537–545, 2019.

XUE, F. et al. Deep Item-based Collaborative Filtering for Top-N Recommendation. v. 37, n. 3, 2019.

YANG, T. et al. Multi-object tracking with discriminant correlation filter based deep learning tracker. In: **Integrated Computer-Aided Engineering**. NLD: IOS Press, 2019. ISSN 18758835.

YE, W. et al. Nonnegative matrix factorization for clustering ensemble based on dark knowledge. **Knowledge-Based Systems**, Elsevier, v. 163, p. 624–631, 2019.

YEHUDA, K.; ROBERT, B.; CHRIS, Y. Matrix Factorization Techniques for Recommender Systems. **IEEE Computer Society**, p. 1–8, 2009.

YI, B. et al. Deep Matrix Factorization With Implicit Feedback Embedding for Recommendation System. **IEEE Transactions on Industrial Informatics**, v. 15, n. 8, p. 4591–4601, aug 2019. ISSN 1551-3203.

ZHANG, A. et al. Automated Pixel-Level Pavement Crack Detection on 3D Asphalt Surfaces with a Recurrent Neural Network. **Computer-Aided Civil and Infrastructure Engineering**, 2019. ISSN 14678667.

ZHANG, B. et al. Integrating an attention mechanism and convolution collaborative filtering for document context-aware rating prediction. **IEEE Access**, IEEE, v. 7, p. 3826–3835, 2019. ISSN 21693536.

ZHANG, S. et al. Learning from Incomplete Ratings Using Non-negative Matrix Factorization. In: **Proceedings of the 2006 SIAM International Conference on Data Mining**. Philadelphia, PA: Society for Industrial and Applied Mathematics, 2006. p. 549–553. ISBN 978-0-89871-611-5.

_____. Deep learning based recommender system: A survey and new perspectives. **ACM Comput. Surv.**, Association for Computing Machinery, New York, NY, USA, v. 52, n. 1, fev. 2019. ISSN 0360-0300.

ZHANG, Y. et al. Joint Representation Learning for Top-N Recommendation with Heterogeneous Information Sources. In: **Proceedings of the 2017 ACM on Conference on Information and Knowledge Management - CIKM '17**. New York, New York, USA: ACM Press, 2017. Part F1318, p. 1449–1458. ISBN 9781450349185.

ZHANG, Y.; CHENG, T.; REN, Y. A graph deep learning method for short-term traffic forecasting on large road networks. **Computer-Aided Civil and Infrastructure Engineering**, 2019. ISSN 14678667.

ZHANG, Y. et al. Deep Pairwise Hashing for Cold-Start Recommendation. **IEEE Transactions on Knowledge and Data Engineering**, v. 34, n. 7, p. 3169–3181, 2022. ISSN 15582191.

ZHANG, Y.; WANG, J.; LUO, J. Knowledge graph embedding based collaborative filtering. **IEEE Access**, IEEE, v. 8, p. 134553–134562, 2020.

ZHANG, Y. et al. Integrating stacked sparse auto-encoder into matrix factorization for rating prediction. **IEEE Access**, IEEE, v. 9, p. 17641–17648, 2021.

ZHAO, Y. et al. Modeling User Reviews Through Bayesian Graph Attention Networks for Recommendation. **ACM Transactions on Information Systems**, 2022. ISSN 1046-8188.

ZHENG, L. et al. Spectral collaborative filtering. **RecSys 2018 - 12th ACM Conference on Recommender Systems**, p. 311–319, 2018.

ZHENG, L.; NOROOZI, V.; YU, P. S. Joint Deep Modeling of Users and Items Using Reviews for Recommendation. In: **Proceedings of the Tenth ACM International Conference on Web Search and Data Mining - WSDM '17**. New York, New York, USA: ACM Press, 2017. p. 425–434. ISBN 9781450346757.

ZHOU, J. et al. From Content Text Encoding Perspective: A Hybrid Deep Matrix Factorization Approach for Recommender System. **Proceedings of the International Joint Conference on Neural Networks**, IEEE, v. 2019-July, n. July, p. 1–8, 2019.

UNIVERSIDADE FEDERAL DE SÃO CARLOS

Centro de Ciências Exatas e de Tecnologia
Programa de Pós-Graduação em Ciência da Computação

Relatório de Defesa de Tese
Candidato: Guilherme Brandão Martins

Aos 07/12/2023, às 14:00, realizou-se na Universidade Federal de São Carlos, nas formas e termos do Regimento Interno do Programa de Pós-Graduação em Ciência da Computação, a defesa de tese de doutorado sob o título: Exploring Collaborative Filtering through Machine Learning Techniques, apresentada pelo candidato Guilherme Brandão Martins. Ao final dos trabalhos, a banca examinadora reuniu-se em sessão reservada para o julgamento, tendo os membros chegado ao seguinte resultado:

Participantes da Banca	Função	Instituição	Resultado	Resultado Final
Prof. Dr. João Paulo Papa	Presidente	UFSCar	Aprovado	Aprovado
Prof. Dr. Alexandre Luis Magalhães Levada	Titular	UFSCar	Aprovado	
Prof. Dr. Diego Furtado Silva	Titular	UFSCar	Aprovado	
Prof. Dr. Aparecido Nilceu Marana	Titular	UNESP	Aprovado	
Prof. Dr. Marcelo Garcia Manzato	Titular	ICMC/USP	Aprovado	

Parecer da Comissão Julgadora*:

O candidato foi aprovado por unanimidade, respondeu às perguntas da banca com clareza e o seu trabalho possui a qualidade de um doutorado em Ciência da Computação.

Encerrada a sessão reservada, o presidente informou ao público presente o resultado. Nada mais havendo a tratar, a sessão foi encerrada e, para constar, eu, Ivan Rogério da Silva, representante do Programa de Pós-Graduação em Ciência da Computação, lavrei o presente relatório, assinado por mim e pelos membros da banca examinadora.

Prof. Dr. João Paulo Papa

Representante do PPG: Ivan Rogério da Silva

Prof. Dr. Alexandre Luis Magalhães Levada

Prof. Dr. Diego Furtado Silva

Prof. Dr. Aparecido Nilceu Marana

Prof. Dr. Marcelo Garcia Manzato

Certifico que a defesa realizou-se com a participação à distância do(s) membro(s) Alexandre Luis Magalhães Levada, Diego Furtado Silva, Aparecido Nilceu Marana, Marcelo Garcia Manzato e, depois das arguições e deliberações realizadas, o(s) participante(s) à distância está(ao) de acordo com o conteúdo do parecer da banca examinadora redigido neste relatório de defesa.

gov.br Documento assinado digitalmente
JOAO PAULO PAPA
Data: 07/12/2023 18:43:27-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. João Paulo Papa

( ) Não houve alteração no título (X) Houve alteração no título. O novo título passa a ser:
Floresta de Caminhos Ótimos no Auxílio a Filtragem Colaborativa

Observações:
a) Se o candidato for reprovado por algum dos membros, o preenchimento do parecer é obrigatório.
b) Para gozar dos direitos do título de Mestre ou Doutor em Ciência da Computação, o candidato ainda precisa ter sua dissertação ou tese homologada pelo Conselho de Pós-Graduação da UFSCar.