SAPIENZA
UNIVERSITÀ DI ROMA

Sapienza University of Rome

Department of Computer, Control and Management Engineering
PhD in Artificial Intelligence

Thesis For The Degree Of Doctor Of Philosophy

On Core Challenges in Industrial Recommender Systems

Thesis Advisor
Prof. Fabrizio Silvestri

Candidate
Alessandro Sbandi
1551081

Academic Year 2022-2023 (XXXVIII cycle)

Try not.
Do or do not.
There is no try.

Summary

Industrial-scale recommender systems are a cornerstone of the modern digital economy, demanding solutions that are not only accurate but also scalable, efficient, and robust. This thesis, focused on the practical challenges of deploying these systems, aims to bridge the persistent gap between academic advancements and industrial requirements. This work unfolds the methodologies, innovations, and practical insights crucial for developing graph-based recommender systems that perform effectively under real-world production constraints. The following summary provides an overview of the key contributions and insights of this thesis.

The introduction provides a context for the research, clarifying the motivations and objectives driving the study. It highlights the mismatch between academic benchmarks and industrial realities, such as massive-scale graphs, real-time inference demands, and the critical cold-start problem, setting the stage for a systematic exploration.

Starting our exploration, we first ground the research in a tangible industrial context by introducing TIM-Rec, a novel and unique dataset for multi-item upselling recommendations derived from telecommunication call records. We then turn our attention to one of the most critical industrial challenges: the cold-start problem. We introduce a novel framework with a re-ranking method designed to improve recommendations for new users and items on large graphs and further investigate the crucial trade-off between diversity and coverage to ensure new content is both relevant and discoverable.

We then broaden our investigation to high-value industrial problems, introducing CRAB, a reinforcement learning-based agent for proactive customer churn reduction. This demonstrates the versatility of our data-driven approach in solving critical business challenges beyond traditional recommendation tasks within the same industrial environment.

The core of this thesis directly confronts the challenge of scalability. We propose efficient recommendation methods based on graph coarsening and label propagation, which simplify massive user-item graphs while preserving their essential properties. Furthermore, we challenge the prevailing trend of increasing model complexity by demonstrating that learning an optimal graph structure allows simpler, more scalable models to achieve competitive or even superior performance.

The final part of this thesis summarizes the key findings and contributions to the field of industrial recommender systems. It shows how we achieved the objectives outlined in the introduction, advancing the understanding and practical implementation of scalable, graph-based systems. It also offers insights into future research directions, emphasizing the ongoing need for solutions that balance theoretical soundness with deployability.

In conclusion, this thesis represents a significant step in the ongoing pursuit of practical and effective recommender systems. It stands as a valuable resource for researchers and practitioners striving to build systems that meet the rigorous demands of industrial-scale deployment. With the principles of scalability, efficiency, and real-world applicability, this research contributes to the collective effort of ensuring that recommender systems deliver tangible and reliable value in complex business environments.

Keywords: Recommender Systems, Graph Neural Networks, Reinforcement Learning, Cold Start, Simulated annealing, Learning to Rank, Industry, Telecommunications.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Nowadays, recommender systems are one of the most critical components of modern industries [96, 297, 315]. They influence how users discover products on e-commerce platforms [194], how contents are viewed on media streaming services [115], and how information propagates on social networks [126]. By personalizing the user experience, recommendations not only improve user engagement but also have a direct impact on business outcomes such as retention [268], upselling [186], and company revenues [5]. The effectiveness of these systems is thus not only of academic interest but a strategic importance in industry [130]. Companies invest heavily in recommendation technologies because the quality of the recommendations delivered to users can significantly determine the competitiveness of their platforms [132, 323, 324].

However, these systems have a fundamental computational challenge [219, 326]: matching users with items that are most relevant to them, based on large volumes of historical interactions and contextual signals. Industrial-scale recommender systems must handle datasets of enormous size [159].

To capture the structure in this relational data, interactions between users and items can naturally be represented as graphs [56], where nodes correspond to entities such as users, items, or categories, and edges capture their relationships. In real-world settings, these graphs easily grow to hundreds of thousands or even millions of nodes, connected by tens or hundreds of millions of edges [47]. Computation on such structures is far from trivial: simple algorithms often reach memory problems, and even well-designed methods must be carefully engineered to meet the computational demands [220].

Moreover, industrial constraints introduce additional layers of complexity. Recommendations must be generated in real time, typically within a few tens of milliseconds, so that user interfaces remain responsive [168, 203]. This rules out the use of methods that are computationally expensive at inference time [171]. Furthermore, infrastructure costs place strict limits on the memory and processing power. Unlike in purely academic contexts, where training time or resource usage may be secondary concerns, industrial systems operate under budget and engineering constraints that demand efficient, reliable, and maintainable solutions [103, 188, 189]. Thus, the practical value of a method is not determined solely by its accuracy but by its ability to integrate smoothly into a production pipeline with limited resources [312].

In recent years, the academic community has produced a wide array of advanced algorithms for recommendation, particularly with the rise of graph-based methods and graph neural networks

(GNNs) [73, 301]. These approaches have demonstrated strong potential by explicitly modeling the structure of user–item interactions [129, 250] and by capturing high-order relationships [226]. On benchmark datasets, such methods often report impressive gains in predictive accuracy [198]. Moreover, these methods are usually developed under assumptions that do not reflect the realities of industrial data [326]. Academic evaluations typically involve static graphs [274], balanced datasets [13], or homogeneous node distributions [279], while production data is dynamic, evolving over time, and characterized by a small subset of nodes that are of primary business interest [306].

A further fundamental challenge in industrial recommendation is the cold-start problem [257]. Unlike academic benchmarks, which often assume dense historical interactions, industrial platforms continuously face situations where either new users join the system or new items are introduced into the catalog. In both cases, little to no interaction history is available, making it difficult for the system to generate reliable recommendations.

The cold-start problem is especially critical in dynamic industries such as e-commerce [179] or media streaming [115], where new products, movies [166], or songs are added daily [193], and users may interact with the platform for the first time at any moment. Failure to provide relevant recommendations in these early interactions can lead to user disengagement and lost revenue opportunities. Similarly, if new items are not surfaced quickly and accurately, they may never reach their potential audience, reducing the overall value of the catalog [257]. Therefore, addressing cold start goes beyond accuracy: it directly impacts user satisfaction and the serendipity of new content [149].

Graph-based approaches offer a promising direction for tackling cold start by allowing signals to be propagated from well-connected nodes to new or sparsely connected ones [138]. For example, information about item attributes or user profiles can be incorporated into the graph and spread through label propagation, enabling the system to make meaningful inferences even when direct interaction data is missing. In this way, methods developed in this thesis address not only scalability and efficiency but also one of the most pressing industrial challenges: delivering high-quality recommendations in cold-start scenarios.

Another key challenge in industrial recommender systems is the temporal mismatch between interactions and their realization [308]. Most models prioritize immediate proxy metrics, such as instantaneous acceptance or click-through rates [199]. However, in telecommunication market the true measure of success, long-term business objectives like maximizing customer lifetime value [248], arises over a sequence of interactions and is often highly delayed [308]. Models focused only on instantaneous prediction fundamentally struggle to capture this necessary temporal causality [290].

In the following chapters, we will explore many of those problems in real-world datasets, based on the company's use cases, from cold-start recommendations to scalability and efficiency-focused tasks, to sequential long-term recommendations. This research aims to provide practical insights and solutions for the development and deploy in production of recommender systems in a telecommunication company.

Objectives of the Thesis

The mismatch between academic advances and industrial requirements creates a persistent gap. On the one hand, industries need models that are scalable, efficient, and resilient to noisy, dynamic data. On the other hand, the most advanced academic models often prioritize expressiveness and

theoretical novelty over deployability. Bridging this gap requires approaches that are grounded in both perspectives: methods that draw from state-of-the-art theory but are simplified, adapted, or re-engineered to meet production constraints. For instance, lightweight graph-based methods that approximate the behavior of deeper models, or coarsening techniques that reduce data complexity without sacrificing predictive power, are particularly attractive. Such strategies aim not only to advance academic understanding but also to provide practical value for real-world systems.

Within this landscape, three key problems emerge in industrial recommender systems. First, scalability bottlenecks are prevalent in large-scale recommendation pipelines. Even with distributed infrastructure, certain algorithms cannot be applied at scale without unacceptable slowdowns or prohibitive costs. Second, accuracy and efficiency are entangled: methods that push for higher accuracy often do so at the expense of inference speed, while approaches optimized for efficiency may sacrifice recommendation quality. Achieving a balance between these competing objectives is one of the central design challenges. Finally, there are fundamental limitations in how current graph-based methods handle real-world data. Industrial graphs are not static but evolve as users interact and new items are introduced. They are heterogeneous, with different types of nodes and edges, and their structure is often highly imbalanced, with some nodes attracting far more attention than others. Existing methods struggle to adapt to such conditions, leading to degraded performance in production settings despite strong results in controlled academic benchmarks.

Addressing these issues is not simply a matter of optimizing existing algorithms but requires rethinking how recommendation problems are formulated and solved in industrial contexts. This thesis is motivated by the need to develop methods that are not only theoretically sound but also practically deployable at scale. By focusing on graph coarsening techniques and efficient recommendation strategies, it seeks to bridge the gap between academic research and industrial application, providing solutions that are both scalable and effective for real-world recommendation pipelines.

Thesis Outline

This thesis is structured to progressively address the key challenges of industrial recommender systems, starting from fundamental industrial problems like cold start and moving toward scalability and graph structure learning. The thesis is structured to reflect both the research contributions and their industrial context, as follows:

Chapter 2: Multi-Item Upselling Dataset in Telecommunications

Here, we present an industrial dataset. This chapter introduces TIM-Rec [231], a unique dataset for recommender systems from telecommunication call records. We demonstrate its main features: sparsity and explicit feedbacks in multi-item upselling recommendations.

- A. Sbandi, F. Siciliano, and F. Silvestri, (2025, September). TIM-Rec: Explicit Sparse Feedback on Multi-Item Upselling Recommendations in an Industrial Dataset of Telco Calls. In *Proceedings of the 19th ACM Conference on Recommender Systems*, RecSys '25 (pp. 865-873). doi: 10.1145/3705328.3748150

Chapter 3: Mitigating the Cold-Start Challenge

This chapter tackles one of the most persistent problems in industrial recommendation: the

cold-start scenario. We introduce a framework with a novel re-ranking method to improve recommendations for new users and items on large graphs [230]. We then extend this work by analyzing the critical trade-off between diversity and coverage, providing a balanced solution that ensures new content is both relevant and discoverable.

- A. Sbandi, F. Siciliano, and F. Silvestri, (2024, October). Mitigating Extreme Cold Start in Graph-based RecSys through Re-ranking. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management* (pp. 4844-4851). doi: 10.1145/3627673.3680069

- A. Sbandi, F. Siciliano, and F. Silvestri, Balancing the Diversity-Coverage Trade-off in Graph-based Recommendations for Cold-Start Industrial Applications. In *Submitted to ACM Transactions on Recommender Systems.*

Chapter 4: Proactive Churn Reduction via Reinforcement Learning

Expanding beyond traditional recommendations, this chapter explores the related industrial problem of customer churn. We introduce CRAB, a reinforcement learning-based agent designed to proactively identify and retain at-risk customers. This work showcases the versatility of our data-driven approach in solving a business problem within the same industrial context.

- A. Sbandi, F. Siciliano, and F. Silvestri, CRAB: Churn Reduction Agent Based via Reinforcement Learning. In *Submitted to ACM Transactions on Recommender Systems.*

Chapter 5: Optimizing Graph Representations for Efficient Industrial Recommendation

This chapter directly confronts the challenge of scalability. We propose a novel method for efficient recommendation based on graph coarsening and label propagation. By simplifying massive user-item graphs while preserving their essential structural properties, our approach enables fast and accurate recommendations. Moreover, we challenge the prevailing trend of using increasingly complex models. We investigate whether learning an optimal graph structure, through techniques like simulated annealing, can allow simpler and more scalable models to achieve competitive or superior performance.

- A. Sbandi, F. Siciliano, and F. Silvestri, Efficient Recommendations via Graph Coarsening and Label Propagation. In *RS4SD – 1st Workshop on Recommender Systems for Sustainable Development using Responsible Nudging, co-located with CIKM 2025. Seoul, South Korea*, In *Submitted to the ACM Web Conference 2026. Dubai, United Arab Emirates.*

- A. Sbandi, F. Siciliano, and F. Silvestri, Do We Need Deep Models or Just Better Graphs? Scalable Link Prediction through Simulated Annealing in Graph Structure Learning. In *Submitted to the ACM Web Conference 2026. Dubai, United Arab Emirates.*

Chapter 6: Conclusions

The thesis concludes by summarizing the key findings and contributions. We reflect on the lessons learned from bridging academic research with industrial practice, discuss the limitations of our work, and propose promising directions for future research in building practical, scalable, and effective recommender systems.

Research Focus

The central research focus of this thesis is the development and adaptation of graph-based recommender systems to overcome the practical challenges of industrial-scale deployment. Recognizing the persistent gap between theoretical advances and production requirements, this work moves away from a sole emphasis on predictive accuracy on benchmarks. Instead, it concentrates on creating methods that are inherently scalable, efficient, and robust enough to handle the complexities of real-world, dynamic data from the telecommunications industry. Throughout this thesis, significant contributions have been made to these areas, resulting in novel insights and approaches. Our research is guided by three primary pillars:

- **Solving the Cold-Start Problem in Practice**: We investigate and develop graph-based strategies specifically designed to provide meaningful recommendations for new users and items. The focus is not just on improving accuracy but on practical solution in re-ranking method in order to manage the diversity-coverage trade-off, which is critical for user engagement and catalog value in live industrial systems.
- **Achieving Scalability and Efficiency**: We explore alternatives to computationally expensive deep learning models for large-scale graph recommendations. This research pillar is dedicated to techniques such as graph coarsening, lightweight propagation, and graph structure learning. The objective is to demonstrate that efficient algorithms, when paired with well-engineered graph representations, can meet or exceed the performance of more complex models while adhering to strict constraints on inference time and infrastructure costs.
- **Addressing High-Value Industrial Use Cases**: We ground our research in tangible business problems faced by a telecommunication company. This involves not only creating a unique, real-world dataset for multi-item upselling but also adapting recommendation and learning frameworks to solve adjacent challenges like proactive customer churn reduction. This focus ensures that the developed methods provide direct and measurable business value beyond conventional recommendation scenarios.

In conclusion, the research presented in this thesis has been characterized by a diverse array of research contributions spanning cold-start problems, graph coarsening, and graph structure learning, and more. These endeavors collectively exemplify a dedication to advancing the frontiers of graph recommendation systems' research and its practical applications.

Chapter 2

Multi-Item Upselling Dataset in Telecommunications

The benchmark datasets used for academic research often fail to capture the complexities of real-world data from production systems. They typically lack the unique characteristics of industrial applications. This discrepancy can lead to a performance gap where models that excel in offline evaluations fail to deliver practical value when deployed.

This chapter is dedicated to bridging this gap. We present the **TIM-Rec** dataset, which is built from internal, real-world customer interactions. This work forms the basis of this entire thesis. It fulfills this foundational role by providing a comprehensive analysis of the dataset's statistical properties, highlighting the key challenges. Furthermore, it establishes a rigorous set of baseline performance metrics by evaluating a diverse set of existing recommendation algorithms, thereby creating a benchmark against which new models can be measured.

Crucially, the dataset introduced here will be used in the subsequent chapters to validate the novel methodologies proposed in this thesis. Therefore, this chapter should be viewed not only as a self-contained contribution to the research community but as the essential first step in the research narrative.

The TIM-Rec dataset is the foundational milestone for all the work because its specific characteristics directly define the discussed industrial problems. The pronounced sparsity and the existence of many new items define the cold-start problem that drives the solution in Chapter 3, where we tackle the most extreme version of it. The delayed, long-term records of customer history, which capture decisions over time, and outcomes like customer churn, provide the unique data needed for the RL agent in Chapter 4. Finally, the large scale of the interaction graph itself establishes the critical efficiency and computational bottlenecks addressed by the graph coarsening and structure learning methods in Chapter 5.

2.1 TIM-Rec: Explicit Sparse Feedback on Multi-Item Upselling Recommendations in an Industrial Dataset of Telco Calls

Upselling recommendations play a critical role in improving customer engagement and maximizing revenue in the telecommunications industry. However, real-world data on such interactions often presents unique challenges, including multiple recommendations per call and sparse customer feedback, which complicates the evaluation of recommender systems. Our review of the existing literature reveals a critical gap in publicly available datasets that reflect these challenges, limiting progress in developing and evaluating upselling strategies.

This work introduces a novel dataset that captures these complexities, offering valuable insights into customer behavior and recommendation effectiveness. The dataset, derived from real-world interactions between customers and service providers, contains multiple recommendations provided in individual calls and sparse feedback, reflecting typical user behavior where interest may be low or unrecorded.

To aid in the development of more effective recommendation systems, we provide detailed statistics on recommendation distributions, user engagement, and feedback patterns. Furthermore, we benchmark various recommendation models, from classical approaches to state-of-the-art neural networks, allowing for a comprehensive assessment of their recommendation accuracy in this challenging setting.

TIM-Rec

Dataset Link: `https://huggingface.co/datasets/shba93/tim-rec`

Github repository: `https://github.com/ashba93/tim-rec`

Industry: Telecommunications

Dataset statistics:
 # interactions: 1.5M
 # users: 370k
 # items: 388
Temporal coverage: 6 months

Recommendation setting: Multiple offers presented simultaneously

Maintenance status: Static dataset (no updates planned); technical issues will be addressed.

Figure 2.1: Overview of TIM-Rec

2.1.1 Introduction

Upselling is a widely used sales and marketing technique [200] designed to encourage customers to purchase higher-value products, upgraded versions, or additional enhancements beyond their initial selection. Unlike cross-selling [144], which focuses on suggesting complementary products or services, the primary goal of upselling is to maximize transaction value while offering solutions that better align with the customer's needs or preferences. When implemented effectively, upselling increases company revenue while enhancing customer satisfaction and loyalty by addressing their evolving requirements [218]. This strategy is often employed across several industries, including retail [35, 200], telecommunications [18, 151], e-commerce [144, 242], and hospitality [16].

In the telecommunications industry [150], upselling plays a key role in increasing the provider's average revenue per user (ARPU). Upselling in telecommunications often revolves around tariff structures [151], where customers are encouraged to opt for higher-tier service plans with more comprehensive benefits, such as increased data allowances or faster internet speeds. Telco providers strategically transition customers from basic plans to premium bundles by incorporating value-added services such as international call options, enhanced security solutions, or streaming subscriptions [206]. This approach enriches the customer experience and strengthens the provider's competitive position in a saturated market.

The success of upselling strategies in telecommunications relies on both timing and personalization. Research [67] highlights the importance of customer insights derived from real-time usage data and historical behavior patterns. By leveraging recommender systems, providers can identify optimal moments for engagement, such as when customers are approaching their data limits or during contract renewal. This targeted approach ensures that upselling offers are relevant and compelling, thereby increasing the likelihood of acceptance. However, the ethical execution of these strategies is critical, as customers respond positively when they perceive genuine value in the upselling offers, rather than viewing them as mere profit-driven tactics [16].

Despite the importance of upselling in telecommunications, the absence of publicly available datasets that accurately reflect real-world upselling interactions poses a significant challenge for researchers. Existing recommendation research datasets often focus on e-commerce [60, 317] or media consumption [25, 106], while telco-specific datasets [11, 72, 210] are either proprietary [230] or limited in scope. This gap hinders the development and evaluation of advanced recommendation models tailored for upselling in telecommunications.

To address this limitation, we introduce a novel real-world dataset designed to studying upselling recommendations in telecommunications. This dataset has several unique characteristics.

Unlike traditional recommendation datasets that track single-item interactions, our dataset captures multiple recommendations occurring on the same day. This is because recommendations are provided through human phone calls, and during a single call, customer care agents may suggest more than one item to the customer. This aspect represents one of the dataset's key novelties, as it captures the dynamics of real-world upselling scenarios where multiple interactions occur within a single customer service engagement.

Another notable feature of the dataset is the presence of sparse customer feedback. Often, users are not interested in the recommendations provided during the call, or they choose to end the call without clear feedback. This can result in a significant number of interactions without explicit acceptance or rejection. As a result, the dataset contains a relatively low proportion of accepted offers compared to the total number of recommendations made. This property reflects a common challenge in sales and customer service environments, where the effectiveness of recommendations is influenced by factors such as customer interest, timing, and the ability of customer care agents to effectively communicate the value of the suggested offers. This sparsity in feedback introduces complexities in evaluating upselling strategies and underscores the importance of understanding customer behavior and engagement patterns.

A major advantage of the dataset is that it is based on real industry data. It was collected from actual interactions between users and customer care agents within a real telecommunication company, offering high authenticity and relevance; the data captures customer behavior, decision-

making processes, and the upselling dynamics made by agents in a live environment.

This dataset provides a new resource for evaluating machine learning models in the context of real-world customer interactions in the telecommunications industry. Given its characteristics, this dataset is intended to serve as a valuable benchmark for future research, enabling the design and testing of more effective upselling strategies, recommendation systems, and customer engagement models.

The main contributions of this paper are:

- We introduce a new dataset with unique characteristics, including multiple recommendations in a single call and sparse feedback, features that together are not present in other datasets.
- We focus on the evaluation of Learning-to-Rank (LTR) models, which are critical for aligning user intent with item selection. We provide several baseline benchmarks to facilitate the future comparison and evaluation of different models within this context.

The remainder of this paper is organized as follows: Section 2.1.2 reviews related work on upselling and currently used datasets. Section 2.1.3 provides the dataset overview, including its construction and main statistics. Section 2.1.4 outlines the experimental setup, including models used, training and evaluation procedures. Section 2.1.5 presents benchmark results, followed by discussions and conclusions.

2.1.2 Related Work

In recent years, there has been a growing interest in recommender system datasets due to their central role in the development of personalized user experiences across various domains [9, 212], including e-commerce [10, 233, 271], streaming services [142], and social media platforms [77]. Access to diverse and high-quality datasets allows researchers to benchmark models, identify challenges, and advance the field of recommendation.

Several public datasets are available for evaluating LTR models [6, 40, 43, 282], each addressing specific research needs. Among the most widely used are the MovieLens [106] and Amazon [197] datasets, both of which incorporate explicit user feedback in the form of ratings (and reviews), making them a popular choice for recommender systems research. The Netflix Prize dataset [25], released in 2006, served as the foundation for the Netflix Prize competition, which challenged researchers to develop algorithms that could improve the accuracy of Netflix's movie recommendations. This dataset, along with Microsoft's LETOR [213] and Yahoo's LTR Challenge [43] datasets, have contributed significantly to LTR research by providing structured benchmarks for evaluating ranking models. For implicit feedback settings, the Taobao UVA dataset [317] captures user-item interactions without explicit ratings. Yelp's dataset provides a diverse set of user interactions, spanning multiple domains. Despite the availability of these datasets, most focus on standard recommendation tasks, with limited applicability to telecommunications upselling scenarios, where multi-item recommendations [209] and sparse feedback are key challenges.

Recent research has emphasized the need for sequential recommendation datasets [32] that better reflect real-world interactions by capturing time-sensitive dependencies. In fact, many commonly used datasets lack strong sequential structures [141], which can limit the development and evaluation of sequential recommender systems.

Dataset	Interactions	Time Range	Dataset Type	Acceptance Sparsity (%)	# Rec.	Public	Dynamic
Amazon [197]	∼571M	27 years	Cross-selling, Upselling	Not Applicable	∼ 1.0	✓	×
Movielens100K [106]	0.1M	7 months	Recommendation	Not Applicable	∼ 2.0	✓	×
Movielens25M [106]	25M	24 years	Recommendation	Not Applicable	∼ 1.2	✓	×
LETOR [213]	<1M	Not specified	Learning-to-Rank (LTR)	Not Applicable	NA	✓	×
Taobao [317]	∼100M	9 days	Upselling	∼ 2% (implicit)	∼ 1.0	✓	✓
Yelp [293]	∼7M	8 years	Cross-selling, Retention	Not Applicable	1	✓	✓
Netflix Prize [25]	∼100M	7 years	Recommendation	Not Applicable	∼ 5.8	✓	×
MIND [276]	∼17.7M	6 weeks	Recommendation	∼11% (implicit)	∼ 37	✓	✓
Iranian Churn [1]	∼3k	1 year	Customer Churn	Not Applicable	1	✓	×
Dookeram et al. [72]	∼3k	1 year	Upselling	Not Applicable	?	×	✓
Alves et al. [11]	∼8M	?	Recommendation	Not Applicable	?	×	×
Pham et al. [210]	∼80k	6 months	Recommendation	Not Applicable	?	×	✓
Ours	∼1.5M	6 months	Upselling	**∼5% (explicit)**	**∼2.4**	✓	✓

Table 2.1: Comparison of public and private datasets based on key metrics such as number of interactions, time range, dataset type, acceptance sparsity, and average recommendations per session (# Rec.). The datasets are grouped into two categories: general recommendation datasets and datasets in telecommunications, separated by a horizontal line in the table for clarity. "Public" column indicates if the dataset is publicly available (✓) or not (×). The "Dynamic" column indicates whether interactions were recorded in real time (✓) or collected retrospectively (×).

Traditional recommendation datasets, such as MovieLens, capture user preferences in a static manner, akin to a preference questionnaire completed in a short period [80]. Since interactions in these datasets are collected without a temporal dimension, they represent an aggregated summary of past preferences rather than evolving behaviors. In contrast, real-world telecommunications environments exhibit dynamic user behavior, where customers make decisions in response to changing needs, promotions, and contextual factors.

Unlike e-commerce and entertainment domains, publicly available datasets for recommendation tasks in telecommunications are scarce. Lian-Ying et al. [165] analyze upselling in telecom using Support Vector Machines and employ the Iranian Churn dataset from the UCI Machine Learning Repository[1]. While the dataset spans 12 months, the first 9 months are provided as aggregate statistics, meaning that individual customer interactions are not available. In addition, the dataset contains only churn related information, making it unsuitable for evaluating upselling strategies. Dookeram et al. [72] also study upselling in telecommunications, but their dataset is both small (3K observations) and not publicly available, further limiting its impact on the research community. Alves et al. [11] and Pham et al. [210] employ large telecom datasets - 8M interactions, 800K users, 177 items in [11] and 80M customers, 2K items over 6 months in [210]. However, both datasets are proprietary, limiting their reproducibility and use for benchmarking in academic research.

In contrast, the publicly available dataset presented in this paper overcomes these limitations by providing real-world, fine-grained customer interaction data with multiple recommendations per session and explicit upselling attempts. A comparison of our dataset with other datasets in the literature is presented in Table 2.1.

2.1.3 Dataset Overview

In this section, we introduce the dataset and its relevance to LTR task. We first present detailed statistics of our real-world dataset, emphasizing its distinctive characteristics, such as multiple recommendations within a single interaction and sparse user feedback. We then provide a comprehensive analysis of the dataset's structure and features, and formally define the research objectives it is designed to support, with a focus on advancing recommendation strategies.

Dataset Construction

Our dataset was collected from real-world telecommunications interactions over a 180-day period and contains 370,314 users, 388 items, and approximately 1.5 million interactions. To ensure a focus on upselling, we applied several preprocessing steps:

- Retention and cross-selling instances, aimed at preventing customer churn or offering extra packages, were filtered out to focus only on upselling and inbound contacts.
- Interactions are categorized into three distinct types: (1) accepted offers, (2) refused offers, and (3) abandoned calls. Due to the large portion of abandoned calls, they were filtered out to avoid skewing the dataset statistics.
- To optimize storage and enhance the dataset's usability, under-sampling was used to balance the dataset while preserving its real-world characteristics.

[1] `https://archive.ics.uci.edu/dataset/563/iranian+churn+dataset`

In the telco domain, upselling is temporal: customers typically maintain one or more relatively stable subscriptions, which can be considered their "initial items". Due to the variety of these services, they are not explicitly listed as an "initial item". Instead, it is captured and embedded within our PCA feature vectors. Our dataset is constructed around interactions where specific items are proactively offered during evolving upselling campaigns. Upselling thus occurs over time, rather than contemporaneously with the purchase of an "initial item". Consequently, the dataset therefore primarily contains records of these campaign-specific offered items and the customer's explicit response, rather than being structured as pairs of (initial item, upsold item) within each record.

The decision to filter out the abandoned calls was carefully considered and driven by critical data characteristics. Abandoned calls constituted over 99% of the raw interactions. Including these would have introduced even more extreme sparsity into the dataset. More significantly, the reasons for abandonment are highly ambiguous: they could be unanswered calls, dropped connections, or hang-ups occurring before an upsell offer could even be fully presented, rather than necessarily indicating dissatisfaction with a potential offer. Treating this vast volume of ambiguous abandoned calls as negative interactions would have skewed any recommendation model, likely leading to overly conservative or broadly negative suggestions for nearly all users. By focusing on completed interactions, where an offer was actually presented and a response recorded, we aimed to capture a clearer, more reliable signal of user response to specific upselling attempts. We believe this approach is more relevant for interpreting genuine upselling interactions and for developing effective recommendation strategies based on explicit feedback.

Preliminaries

In a telecommunications recommendation system, each data sample is structured as $(x_i, y_i, d_i, f_i, v_i)$ where x_i is a user identifier, y_i a recommended item identifier, d_i the date of contact and f_i the user's feedback. v_i represents the contextual characteristics of the user at the time of contact. To ensure privacy and dimensionality reduction, user characteristics are represented using Principal Component Analysis (PCA). We label our principal components variables as $PCAFeat\_0$ through $PCAFeat\_63$ ordered from most to least important. These 64 PCA components summarize an embedding built from 550 raw user features across several domains:

- *Demographics:* current and past residence, age, gender etc.
- *ARPU:* average revenue per user for fixed phone, mobile, TV services, and all the company extra packages.
- *Customer care interactions:* number of calls, call reasons, and support history.
- *Billing details:* total charges and how they are distributed across services.
- *Ticket issues:* number of reported technical problems and their underlying causes.
- *Sales potential:* products or services the user could buy.
- *Network usage:* upload/download volumes of data.

Each feature vector combines static attributes, values at the time of the call, with temporal behavior, activity over previous days, weeks, and months. The items y_i offered to customers include telecommunication services (e.g., voice and Wi-Fi plans), TV and streaming subscriptions (such as DAZN, Netflix, Amazon Prime, Disney+, and TIMVISION), promotional discounts, smart home devices (e.g., Google Nest), and network equipment such as routers. Also, each user x_i and item y_i is assigned an integer ID to maintain anonymity. Comprehensive details about the dataset's schema are provided in Table 2.2.

The dataset consists of the following key attributes:

- *user_id*: unique identifier assigned to each user.
- *contact_date*: The date when the interaction took place (formatted as YYYY-MM-DD).
- *category*: Category or classification of the recommended item.
- *contact_outcome*: Result of the interaction ("accepted" or "refused").
- *PCA_features*: this contains the PCA features derived from the user data. These features capture the most important telecommunications and demographic attributes, reducing data complexity while preserving the core characteristics for downstream analysis or modeling.
- *item_id*: Unique identifier assigned to each item or product involved in the interaction.

Table 2.2: An example of an instance of the dataset.

Field	Description	Symbol	Example
user_id	User identifier	x_i	314
contact_date	Day of contact	d_i	2024-04-16
category	Offer category	-	Options
contact_outcome	Outcome of a contact	f_i	Accepted
PCA_features	User PCA features	$\vec{v_i}$	[-1.241,...,0.728]
item_id	Item identifier	y_i	72

Data Desensitization and License.

To ensure the protection of user privacy, the dataset underwent a rigorous desensitization process. To maintain anonymity, each user and item in the dataset has been assigned a unique securely encrypted identifier, de-linked from the original production system. User-related contextual features were reduced to 64 dimensions using PCA, preserving essential behavioral signals while removing personally identifiable information. These anonymization steps ensure that the dataset complies with privacy standards, while preserving the integrity of the dataset for research purposes. Moreover, we assess customer consent regarding data sensitivity and collect information exclusively from those who provide explicit agreement at the time of subscription. No data are recorded for customers who do not grant consent.

This dataset is provided under the terms of the Creative Commons Attribution Non Commercial 4.0 International (CC-BY-NC 4.0) license. It fully permits and encourages academic research, including publication, modification, and the sharing of derivative works, provided these also adhere to non-commercial terms.

Dataset Statistics

The processed dataset is tailored for upselling recommendations in the telecommunications domain, as summarized in Table 2.3. To enhance the usability of our dataset, we further investigate its characteristics by conducting an in-depth analysis.

Table 2.3: Summary of Dataset Statistics

Unique calls	613,449
Unique recommendations	1,494,061
Unique users	370,314
Unique items	388
Average recommendations per call	2.428
Average acceptations per call	0.153
% of calls with ≥ 1 accepted recommendation	5.286%
% of calls with ≥ 2 accepted recommendations	1.373%
% of calls with ≥ 3 accepted recommendations	0.233%

Data Sparsity.

The dataset reveals key patterns in customer interactions:

- On average, 2.428 recommendations are made per call, with an average acceptance rate per call of 0.153
- 5.286% of calls lead to at least one acceptance, 1.373% result in two or more acceptances, and only 0.233% see three or more recommendations accepted.

Those statistics underscore the challenges of upselling in real-world scenarios, characterized by sparse positive feedback and the rarity of multiple acceptances in a single interaction.

The Fig. 2.2 further illustrates the monthly distribution of acceptance and rejection interactions, revealing clear seasonal trends and a significant imbalance between the number of rejected and accepted recommendations. This imbalance reflects the inherent sparsity of positive interactions in upselling recommendations, as acceptance rates remain consistently low, ranging from 2.81% to 20.90%. Another noteworthy observation is the gradual increase in acceptance rates in June and July 2024, reaching 13.21% and 20.90%, respectively. This suggests seasonal or temporal trends in user behavior, which may correspond to targeted marketing campaigns, changes in offer quality, or shifts in customer preferences.

These factors the need for robust recommendation models that can learn from sparse positive interactions and handle unbalanced data.

User and Item Intercontact Patterns.

Figs. 2.3 and 2.4 illustrate the distribution of days between consecutive interactions for users and items, on a logarithmic and a linear scale respectively. These plots offer valuable insights into the temporal dynamics of interactions within the dataset.

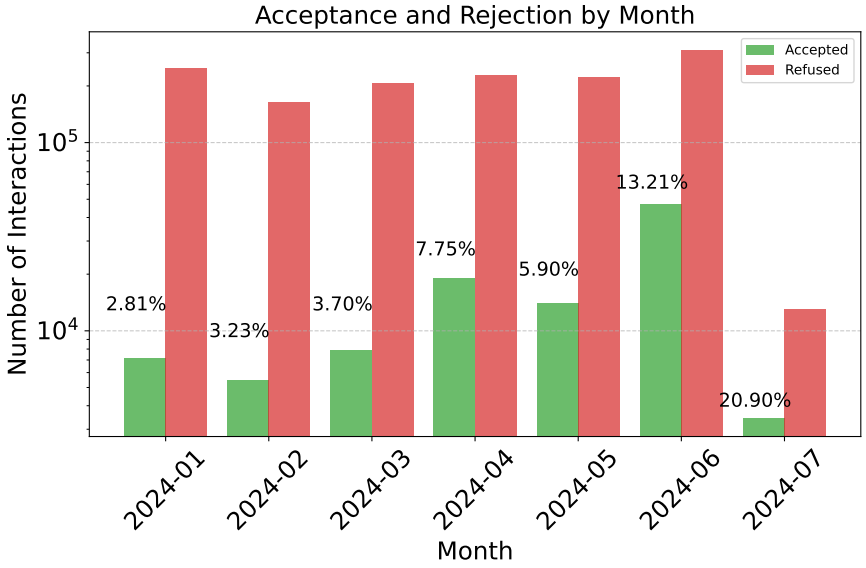


Figure 2.2: Seasonal variations in acceptance sparsity.

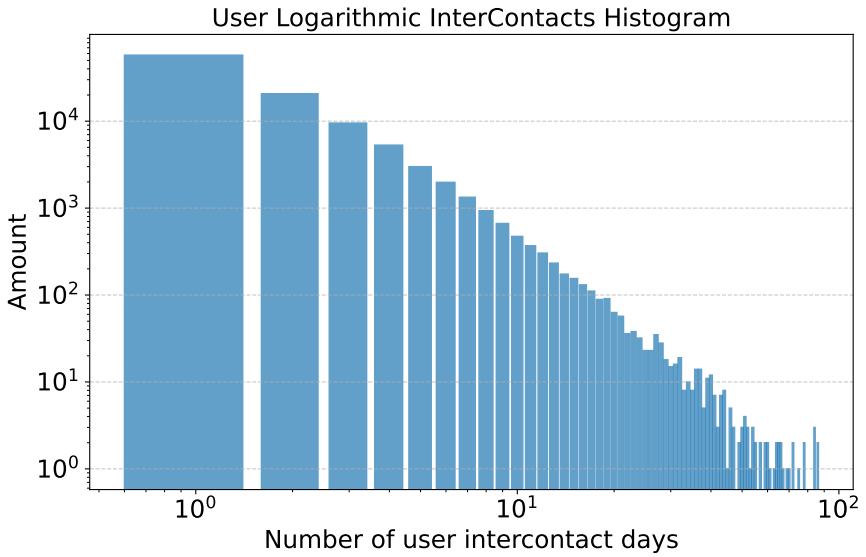


Figure 2.3: Distribution of user intercontact days on a logarithmic scale.

For users, the intercontact histogram has a strong leftward skew, indicating that most interactions occur within very short time intervals. This indicates that a significant fraction of users are contacted almost daily or within just a few days. As the intercontact period increases, the frequency of interactions decreases sharply, forming a heavy-tailed distribution. This suggests that the majority of user engagement occurs in bursts, likely driven by periodic promotions or targeted upselling strategies.

In contrast, the distribution of item intercontact days exhibits greater variability. While many items are engaged within short intervals, the distribution shows sporadic peaks at longer intervals, indicating that these items are periodically reintroduced into recommendations after extended periods. These peaks may correspond to strategic re-engagement cycles or seasonal marketing shifts.

Overall, the user interactions are denser and more frequent, highlighting the importance of maintaining regular engagement, whereas item interactions exhibit a less concentrated pattern.

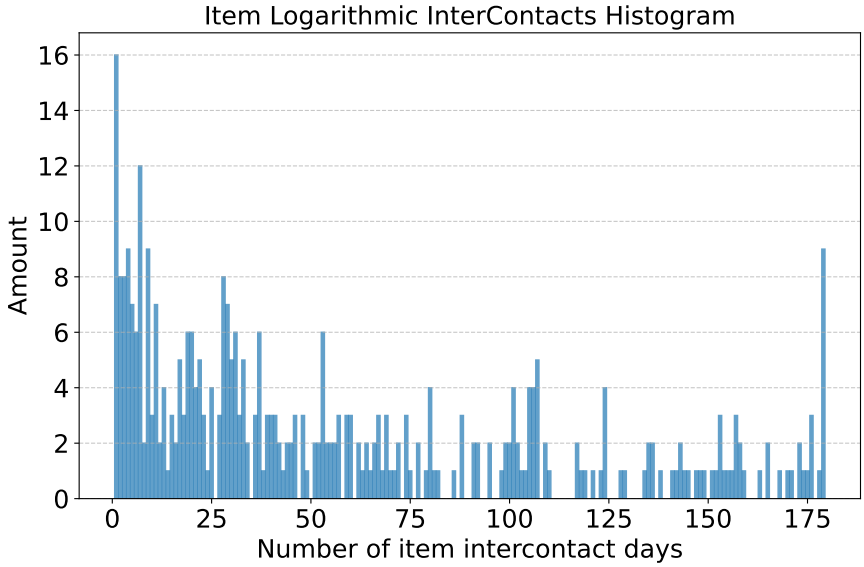


Figure 2.4: Distribution of item intercontact days.

These observations have significant implications for model design, emphasizing the need to account for dense and frequent user interactions, while effectively capturing the periodic fluctuations in item re-engagements. Additionally, the temporal sparsity and variability in both user and item interaction patterns must be carefully addressed to optimize recommendation performance.

User and Item Acceptance Distributions.

The distribution of item acceptance frequency, shown in Fig. 2.5, offers valuable insights into engagement patterns within the dataset. The number of acceptances per item is represented on a logarithmic scale (y-axis), while the corresponding number of items is shown on a linear scale (x-axis). The plot follows a highly skewed distribution, with most items receiving minimal acceptances. There's also a subset of items that are moderately accepted (i.e. 5 to 10 acceptances). After this cluster, the distribution skews heavily, with the frequency of items declining as the number of acceptances increases. Beyond this point, many acceptance counts correspond to only a single item, suggesting that while a small subset of items receives moderate acceptance levels, the majority are either lightly accepted or represent outliers with high engagement.

The highly skewed nature of this distribution underscores the sparsity of item acceptances, with most items either infrequently accepted or uniquely prominent. This imbalance highlights the challenge of accurately modeling item relevance and acceptance, as systems must balance the predominant low-acceptance items with the minority of highly accepted items.

The frequency distribution of user acceptance, shown in Fig. 2.6 provides insights into how users respond to offers. The plot shows the number of users (on the y-axis) who accepted a certain number of offers (on the x-axis). The distribution reveals a steep decline, with most users accepting only a few offers, while a small subset accepts a much larger number.

The overall distribution highlights the sparsity of offer acceptance, with most users engaging with a small number of offers. This imbalance suggests that systems need to consider how to manage the varying levels of user engagement, as the majority of users are likely to be less active, while a minority of users will have a more significant influence on acceptance patterns.

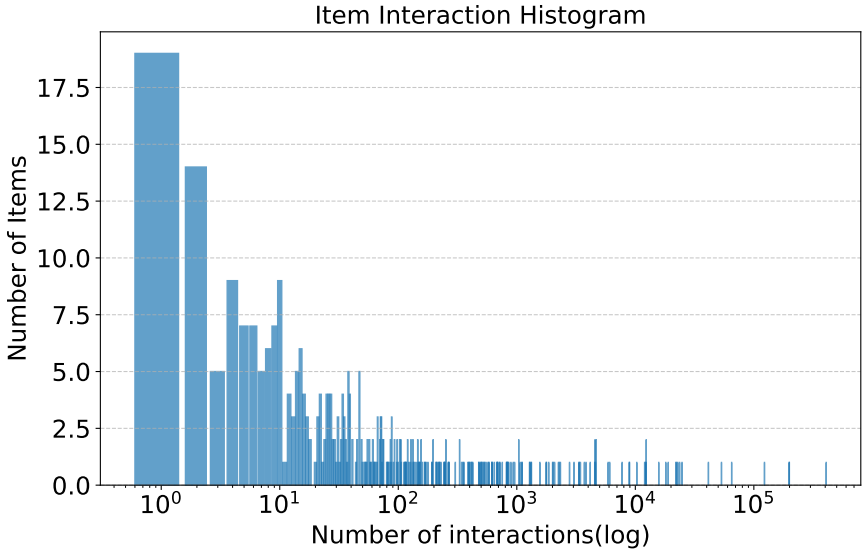


Figure 2.5: Histogram of item contact frequency, showing how often items are proposed and how many different items are involved.

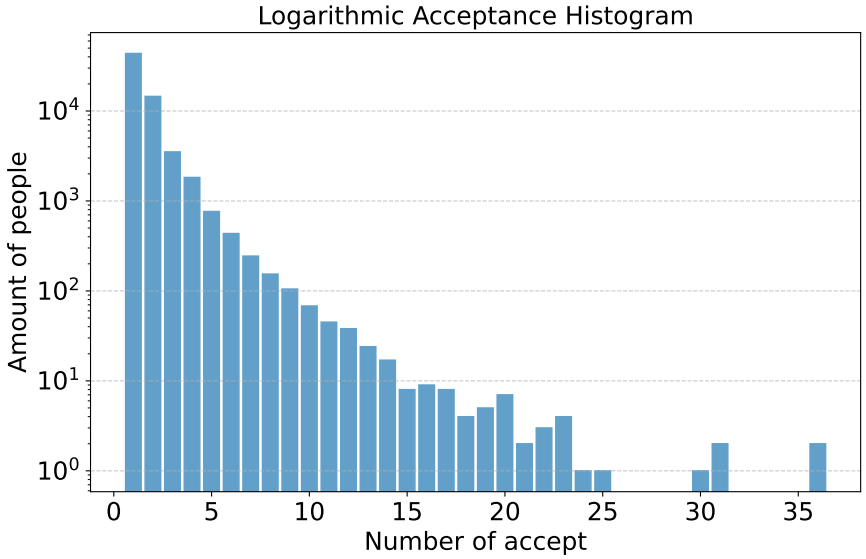


Figure 2.6: Histogram of user acceptance amounts, illustrating the distribution of the number of accepted offers per user.

User Contact Frequency

The distribution of user contact frequency, shown in Fig. 2.7, has on the x-axis the number of times users have been contacted, while the y-axis, presented on a logarithmic scale, indicates the corresponding number of users.

The distribution is highly skewed, with a significant concentration of users receiving only few contacts. Beyond this initial peak, the number of users gradually declines as the number of contacts increases. Notably, in the tail of the distribution, a few users have been contacted over 100 times, suggesting outlier behavior in which a small subset of users experiences exceptionally high interaction rates. This observation might reflect specific targeted retention strategies or individual engagement patterns.

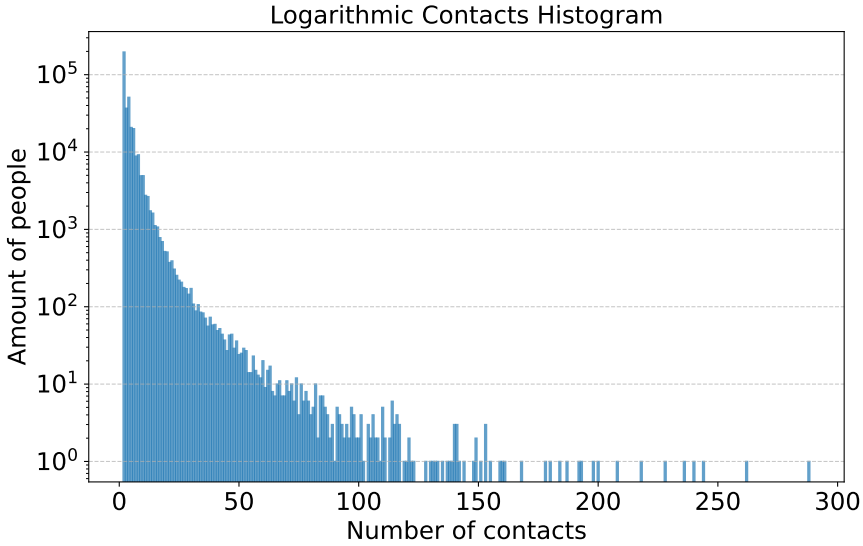


Figure 2.7: Histogram of user contact frequency, illustrating the distribution of the number of contacts per user.

The skewness of the distribution underscores a key challenge for models: balancing learning from a large pool of minimally contacted users while also effectively modeling the behaviour of the highly engaged minority.

Multi-Item Recommendations

A distinctive feature of our dataset is the presence of multiple recommendations within each customer care interaction. As illustrated by the histogram in Fig. 2.8, each call includes a minimum of two recommended items, with the majority of calls containing exactly two recommendations. This characteristic highlights the multi-item nature of the dataset, setting it apart from typical recommendation datasets that focus on single-item interactions.

This multi-item structure reflects real-world customer care practices, where agents propose multiple offers in a single interaction. This characteristic enriches the dataset by providing a nuanced perspective on user behavior when presented with multiple options simultaneously. It also opens opportunities for studying complex decision-making processes, such as how the composition of offers influences customer acceptance and how recommendations can be optimized in multi-item scenarios.

2.1.4 Experimental Setup

This section describes the experimental setup designed to evaluate the performance of various recommendation models on our dataset. First, we outline the dataset splitting strategy, followed by a discussion of the evaluation metrics used to assess model effectiveness. Finally, we introduce the baseline models selected for comparison.

Dataset Splitting

We decide to split the dataset both by user and time. For each user, we select their last 20% of interactions and make this the test set. The previous 10% are used for validation, while the remaining 70% are kept for training. By employing a random user-based split, we aim to distribute

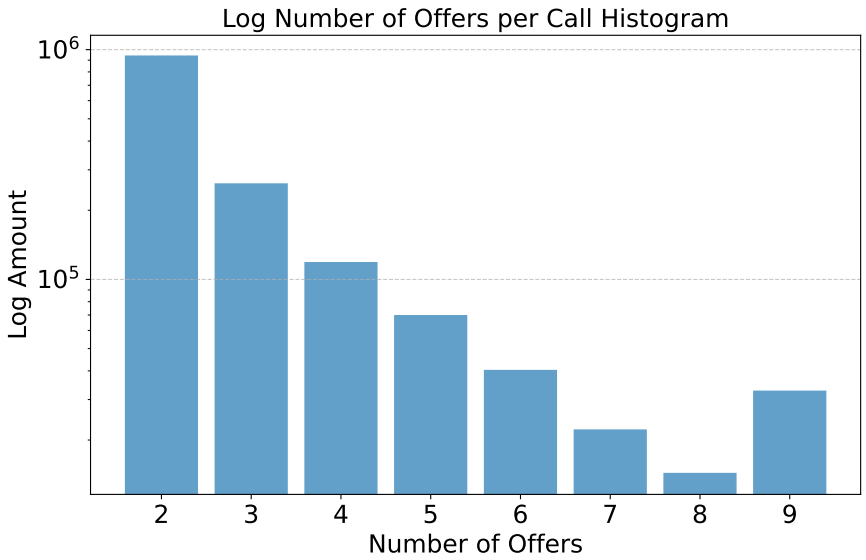


Figure 2.8: Multi-Item recommendations. Each interaction includes a minimum of two recommended items.

the user interactions across the subsets in a way that avoids bias and preserves the diversity of user behavior in each split. This data split should ensure a balanced and representative distribution of the dataset.

Method	NDCG@5	Precision@5	Recall@5	NDCG@10	Precision@10	Recall@10	NDCG@20	Precision@20	Recall@20
SVD	0.073	0.022	0.083	0.074	0.012	0.089	0.075	0.006	0.091
ALS-WR	0.150	<u>0.049</u>	<u>0.219</u>	**0.205**	**0.043**	**0.388**	**0.220**	**0.024**	**0.444**
NCF	**0.164**	**0.053**	**0.246**	<u>0.173</u>	<u>0.029</u>	**0.388**	<u>0.192</u>	<u>0.019</u>	<u>0.347</u>
Caser	0.091	0.032	0.145	0.122	0.026	0.238	0.135	0.016	0.287
Gru4Rec	0.036	0.016	0.071	0.062	0.017	0.155	0.107	0.018	0.326
SASRec	<u>0.154</u>	0.047	0.218	0.169	0.029	<u>0.264</u>	0.184	0.018	0.325
LightGCN	0.080	0.028	0.131	0.102	0.021	0.198	0.118	0.014	0.260

Table 2.4: Results of the models on the dataset. **Bold** indicates the best results, while the second best for each metric is <u>*underlined*</u>.

Evaluation Metrics

To assess the performance of the models, we rely on well-established evaluation metrics in the recommendation system domain: Precision@K, Recall@K and NDCG@K, which assess the relevance and quality of the ranking. We compute these metrics for K values of 5, 10, and 20, to capture performance variations across different recommendation list lengths.

A key challenge in our dataset is the imbalance between accepted and rejected interactions. Since only a small subset of users provide explicit feedback, usual metrics, such as NDCG@K may not fully capture the effectiveness of recommendations in the real world. To address this, we modify how relevance is assigned in NDCG@K to account for both accepted and rejected offers, as well as unseen items. This adjustment allows us to evaluate model performance across the entire user base, rather than just those who have provided explicit feedback. The standard NDCG@k formula is:

$$NDCG@k = \frac{\sum_{i=1}^{K} \frac{rel(i)}{log_2(i+1)}}{IDCG@k} \tag{2.1}$$

where $rel(i)$ is the relevance score of the i-th item in the ranked list and $IDCG_K$ normalizes

the score between 0 and 1.

In our modified version, relevance score $rel(i)$ is calculated based on the type of user feedback:

$$rel(i) = \begin{cases} 0 & \text{if the offer was rejected} \\ 0.5 & \text{if the offer was never presented} \\ 1 & \text{if the offer was accepted} \end{cases}$$

By including unseen offers in the evaluation, we provide a more comprehensive measure of model effectiveness and ensure that the sparsity of user feedback does not bias the results.

For items that have never been presented to the user, the $rel(i)$ is assigned an intermediate value, reflecting uncertainty regarding user preference. This approach prevents such items from being interpreted as either explicitly preferred ($rel(i) = 1$) or explicitly refused ($rel(i) = 0$). According to the company's strategy, it encourages the model to recommend unseen items, which is preferable to recommending items that the user has already rejected.

Baseline Models

To evaluate the effectiveness of our proposed dataset in practical applications, we compare several widely used recommendation models that span different paradigms:

- **SVD**[143], Singular Value Decomposition, is a classical matrix factorization technique that decomposes the user-item interaction matrix into latent factors, capturing underlying patterns in user preferences. It is widely used for collaborative filtering and serves as a strong baseline for recommendation tasks.
- **ALS-WR** [114], Alternating Least Squares with Weighted Regularization, is a matrix factorization technique similar to SVD.
- **NCF** [108], Neural Collaborative Filtering, is a neural network-based collaborative filtering method that learns user and item embeddings to capture non-linear interactions.
- **Caser** [246], Convolutional Sequence Embedding Recommendation, uses a Convolutional Neural Network to capture sequential patterns in user-item interactions, ideal for session-based or time-sensitive recommendations.
- **GRU4Rec** [110], Gated Recurrent Unit for Recommender Systems, applies Gated Recurrent Units to model sequential dependencies in user behavior, excelling in real-time or session-based recommendations.
- **SASRec** [135], Self-Attention Sequence Recommendation, is a transformer-based model that utilizes self-attention to capture long-range dependencies in user interactions, effective for complex, long-term behavior patterns.
- **LightGCN** [109], Light Graph Convolutional Network, simplifies GCN by removing non-essential components for efficient, large-scale recommendations.

These baselines allow us to benchmark the effectiveness of our dataset on real-world recommendation tasks, covering both traditional and deep learning-based approaches.

Training Strategies

To clarify the experimental setup, we detail the input features and training procedures used for each model:

- **Matrix factorization** methods (SVD and ALS-WR) train on the most recent user-item interactions and corresponding feedback labels (accepted, non-interacted, not accepted), except for the validation and test split reserved interactions. They use stochastic gradient descent (SVD) or alternating least squares (ALS-WR). We run SVD until convergence, while ALS-WR is run for 50 iterations with a regularization coefficient of 0.001.
- **Neural Models** (NCF, Caser, SASRec, GRU4Rec, LightGCN) receive in input the most recent 45 interactions for each user, and are trained to distinguish between accepted and not-accepted items. Following the setups described in their original papers, these models optimize a binary cross-entropy loss using an Adam optimizer with a learning rate of 0.001. Hyperparameters not specified here are set to the defaults recommended in the original publications.

To make the comparison fair, we fixed the latent dimension to 64 for all models. All neural architectures are implemented in PyTorch [27]. More specifics about the training setup can be found in our code repository `https://github.com/ashba93/tim-rec`.

2.1.5 Results

The benchmark results of the selected model on our dataset are presented in Table 2.4. We analyze the experimental findings to provide insights for future research and potential improvements.

Our experiments show that ALS-WR consistently outperforms all other models on all evaluated metrics. This suggests that matrix factorization with weighted regularization is particularly well suited to this recommendation task, effectively capturing user-item interactions despite the feedback imbalance of the dataset. NCF emerges as a strong competitor to ALS, demonstrating competitive performance in most evaluation settings. Its neural network-based approach enables it to learn non-linear user-item interactions, although it does not surpass ALS in overall effectiveness. On the other hand, SVD struggles to achieve competitive results, probably due to its limited ability to learn expressive user and item representations. The model may not be flexible enough to capture the nuanced feedback patterns present in the dataset.

The performance of deep learning based models varies considerably. GRU4Rec performs poorly, despite being designed for sequential recommendations. This suggests that the model may not effectively capturing patterns in the dataset, possibly due to its reliance on implicit feedback structures. Both LightGCN and Caser perform comparably to SVD, suggesting that their graph-based and convolutional architectures, respectively, do not offer significant advantages in this setting. In contrast, SASRec achieves performance on par with ALS-WR, confirming its status as a state-of-the-art model for sequential recommendation tasks. Its self-attention mechanism allows it to capture long-term dependencies in user interactions, which appears to be particularly beneficial for our dataset.

Apart from SASRec, deep learning-based models generally underperform. A key reason for this may be that these models, in their standard implementations, are primarily designed for implicit feedback scenarios. As a result, they do not explicitly incorporate the nuanced user feedback

available in our dataset. However, given their competitive performance in certain cases, there is likely room for improvement through modifications that better integrate explicit feedback signals.

2.1.6 Conclusion

In this work, we present a novel dataset designed to capture the complexity of recommendation scenarios where multiple items are presented simultaneously and user feedback is sparse. Furthermore, a distinctive feature of the presented dataset is the presence of explicit negative feedback, which influences the validation of the models. This dataset fills a gap in existing benchmarks by combining these unique features, allowing for a more realistic evaluation of recommendation models in the telecommunications domain.

A major focus of our study was also the evaluation of LTR models to establish a set of baselines for future research. Our experimental results showed that ALS-WR outperformed other methods, closely followed by NCF. Deep learning models struggled in this setting, probably due to their reliance on implicit feedback assumptions.

By releasing the dataset and benchmarking multiple recommendation approaches, we provide a solid foundation for future studies aimed at improving LTR models in scenarios with sparse feedback and multiple concurrent recommendations. Future work could explore hybrid approaches that combine the strengths of different model architectures, or refine deep learning-based ranking techniques to better handle explicit user interactions.

Chapter 3

Mitigating the Cold-Start Challenge

Recommender systems are essential tools in the modern digital landscape, guiding users through vast catalogs of content and products. Graph Neural Networks (GNNs) have achieved state-of-the-art performance due to their ability to model complex, high-order relationships in user-item interaction graphs. However, the effectiveness of GNNs drops in cold-start scenarios, where new users or items lack the interaction data necessary for the model to learn meaningful representations. In these settings, nodes appear isolated in the graph, and the message-passing mechanism that is central to GNNs becomes ineffective, leaving the system unable to generate relevant suggestions. This is a common issue; it is a persistent and costly problem in large-scale industrial applications, where user bases and item catalogs are always changing.

This chapter addresses this critical challenge by presenting a comprehensive, multi-stage framework designed to generate meaningful recommendations even in settings of extreme data sparsity. The core of our contribution is a novel pipeline that transforms a sparse user-item graph into a dense, information-rich structure. This approach enables the system to produce diverse and valuable recommendations—including long-tail and previously unsold items—for new and established users alike.

Enhancing recommendation diversity often comes at the cost of reduced item coverage or a slight degradation in personalization accuracy. Recognizing this, this chapter extends the initial work with a thorough investigation into this diversity-coverage trade-off. Through extensive hyperparameter analysis and a series of ablation studies, we dissect the framework's components to provide actionable insights for practitioners.

The chapter is structured to guide the reader from the foundational concept to its practical implications. Initially, we introduce the core methodology and validate its effectiveness on a real-world industrial dataset. Subsequently, we provide an expanded analysis that explores the framework's robustness to different GNN architectures, the specific impact of the graph augmentation step, and the relationship between hyperparameter settings and performance metrics.

Ultimately, this chapter not only proposes a novel solution to the cold-start problem but also provides a practical guide for practitioners seeking to balance the competing objectives of accuracy, diversity, and coverage in real-world recommender systems.

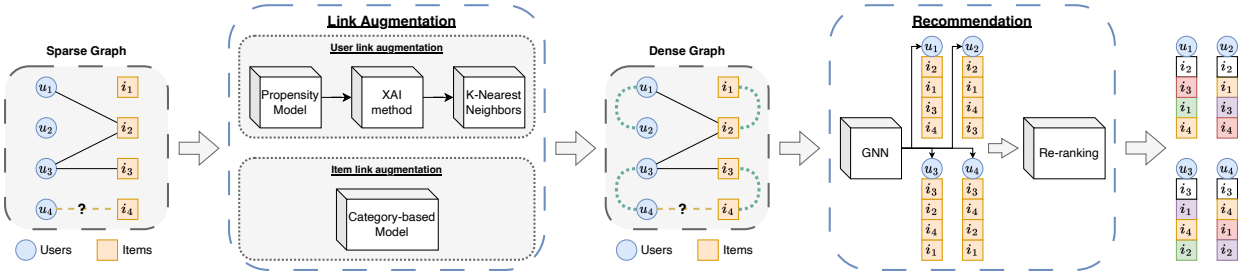


Figure 3.1: Overview of the proposed framework. The input sparse graph is augmented with edges derived from user similarities and item categories, resulting in an enriched dense graph. Subsequently, a Graph Neural Network model is trained on this augmented graph. Finally, re-ranking algorithm is applied to the model predictions to increase recommendation diversity.

3.1 Mitigating Extreme Cold Start in Graph-based RecSys through Re-ranking

Recommender systems based on Graph Neural Networks (GNN) have become the state-of-the-art approach in recommendation, but they struggle with in extreme cold-start settings, where most users or items lack interaction data. This paper proposes a novel framework to address this challenge in four steps: (i) a propensity model to predict item purchase behaviour, with associated explainability to identify the most relevant features, (ii) a link augmentation module to connect users based on previously obtained similarities, (iii) a GNN-based link prediction step on the obtained dense graph and (iv) a final re-ranking stage to increase diversity in predictions leveraging users embeddings. By exploiting the enriched graph structure, the framework generates embeddings for cold-start users and items, enabling diverse recommendations, containing long tail and unsold items, for both established and new users. We validate the framework's effectiveness on real-world industrial data from TIM S.p.A.

3.1.1 Introduction

Recommender systems aim to suggest items to users that are most likely to appeal to them. With the rise of the Internet and the availability of huge amounts of data, these algorithms have become essential. Traditional methods like collaborative filtering rely on user-item interaction similarity [245]. However, these approaches remain significantly limited as their prediction and training stages overlook high-order structural information present in observed data [19, 161].

The emergence of deep learning has led to Graph Neural Networks (GNNs) outperforming traditional methods. GNNs overcome the aforementioned problems by exploiting embedding propagation, iteratively aggregating information from neighboring nodes in a graph structure [139]. Through the stacking of propagation layers, each node gains access to information beyond immediate neighbours, surpassing the limitations of traditional methods that only consider first-order neighbors. The strength of GNNs lies in their ability to learn good user and item representations (embeddings) and generate high-quality recommendations, especially for established users and items (warm scenarios) [212, 304].

Nevertheless, GNNs face challenges in extreme cold-start scenarios, where most users and items lack interaction data. These users and items appear as isolated nodes in the user-item bipartite graph, hindering information propagation and embedding updates.

Many existing approaches [118, 267] have used side information to alleviate this problem. [104] propose pre-trained GNNs and neighbor sampling for cold-start items. More recently, [46] presents a two-step framework to treat warm and cold items differently.

Our paper proposes a practical solution to the extreme cold-start problem in a real large-scale industrial setting, focusing on mitigating the problem for both new users (no purchase history) and new items (minimal or missing sales data). Our contributions include:

- **Identifying User Similarities with Explainable AI (XAI):** We leverage XAI to identify characteristics of warm users. This allows us to connect users with similar behavior patterns, even if they haven't interacted with the same items.
- **Building a Dense Graph for Information Propagation:** We enrich the initial bipartite heterogeneous user-item graph by linking cold nodes to their nearest warm counterparts, according to the XAI patterns, enabling information flow across the entire graph.
- **GNN-based Recommendation with Re-ranking:** Finally we implement a re-ranking algorithm to promote recommendation diversity while simultaneously maintaining the quality of the previous GNN predictions.

Our approach demonstrates success in generating diverse and accurate recommendations in cold-start scenarios: for new items - with minimal, or no sales history or where sales history is missing - and new users - those who have never made a purchase.

3.1.2 Background

XAI for Feature Selection

Understanding the inner workings of a predictive model is crucial for trusting and effectively utilizing its outputs. In recent years, a large number of XAI methodologies have emerged to tackle the complexities inherent in addressing this specific problem [181, 240]. Often, a simpler *explanation model* is used [222]. This simpler model is essentially a more straightforward version designed to help you understand the complexities of the original, more "complicated" model. Shapley values are also a popular XAI technique to explain the contribution of each feature to the output of a Machine Learning (ML) model. The goal is to decompose the model prediction and assign importance scores (Shapley values) to individual features [244]. The explanation method SHAP [178] is a common tool to compute Shapley values. In the context of feature selection, Shapley values can be used to identify the most relevant features of an ML model. In this situation, the features themselves become the *players* in a cooperative selection game, and model performance acts as the *payoff* [84, 99]. Formally, the feature selection game is defined as follows [224]:

Feature selection game.

Let the players' set be $\mathcal{N} = \{1, ..., n\}$, where n is the total number of features. A feature subset $\mathcal{S} \subseteq \mathcal{N}$ represents a specific combination (coalition) of features. For each coalition, the train and test feature vector sets are defined as $X_{\mathcal{S}}^{Train} = \{\mathbf{x}_i^{Train} | i \in S\}$ and $X_{\mathcal{S}}^{Test} = \{\mathbf{x}_i^{Test} | i \in \mathcal{S}\}$. Let $f_{\mathcal{S}}(\cdot)$ be a machine learning model trained using $X_{\mathcal{S}}^{Train}$ as input, then the payoff is $v(\mathcal{S}) = g(\mathbf{y}, \hat{\mathbf{y}}_{\mathcal{S}})$, where $g(\cdot)$ is a goodness of fit function, $\mathbf{y}$ and $\hat{\mathbf{y}}_{\mathcal{S}} = f_{\mathcal{S}}(X_{\mathcal{S}}^{Test})$ are the ground truth and predicted targets.

Shapley values have been studied as a measure of feature importance in various contexts [58] and applied in several tasks, including human action recognition [97] and natural language processing [204]. Model features can be ranked, selected, or removed by exploiting the Shapley importance.

Graph Construction and Link Augmentation

We consider an unweighted heterogeneous bipartite graph $G = (U, I, E, X)$, where U is the set of user nodes, I is the set of item nodes, E is the set of edges connecting users and items based on past interactions, and $X \subseteq \mathbb{R}^{d \times |U|}$ is a matrix of d user features. The task is then formalized as a link prediction task [167]. In a cold-start scenario, many users and items lack interaction data, resulting in isolated nodes that hinder information propagation to the local neighbors, making it impossible to capture graph topology and obtain recommendations. To overcome this problem, Gupta and Shrinath [98] leverage the K-Nearest Neighbors (KNN) algorithm to identify similar users and items based on their features and connect the closest ones. This approach has been shown to outperform traditional recommender systems when applied to huge sparse networks. We employ a similar technique to perform link augmentation over the graph, connecting user-user and item-item nodes based on their similarity.

Figure 3.2: A schematic representation of the proposed framework for graph construction. (A) The initial sparse graph. (B) A propensity model is trained, then (C) Shapley values are computed to select the most representative features. (D) A bipartite graph is built using interaction data, with link augmentation for users and items. Finally, a dense graph (E) is obtained.

GNNs for Recommendation

GNNs are powerful tools that can capture the structural information of a graph and learn node representations. This operation is performed by message passing, where nodes aggregate and update information from their neighbours in iterative steps. The feature matrix X can either represent existing node features or be randomly initialized, which significantly increases the expressiveness at the cost of slower model convergence [3].

The message-passing paradigm has been widely used in user-item GNNs [109, 266]. The messages are iteratively aggregated and updated to obtain informative node embeddings. A single message passing iteration is written as:

$$\mathbf{h}_u^{(k+1)} = UPD^k(\mathbf{h}_u^{(k)}, AGG^k(\{\mathbf{h}_v^{(k)}, \forall v \in N(u)\})) \tag{3.1}$$

where $\mathbf{h}_u^{(k)}$ is node u embeddings at the k-th step, with $\mathbf{h}_u^{(0)} = X_u$, i.e. the u-th row of the feature

matrix X. $N(u)$ denotes the set of neighbors of node u and UPD and AGG are the update and aggregation permutation-invariant operation, respectively. Many differentiable approaches exist for the aggregation operation [139, 254]. After multiple iterations, each node j obtains a final embedding h_j that captures its local and global context within the graph.

Finally, the predicted likelihood of u buying item i is obtained using the inner product of their embeddings:

$$\hat{y}_{ui} = \mathbf{h}_u^T \mathbf{h}_i \tag{3.2}$$

Those obtained likelihoods allow us to rank the items for each user u, recommending the ones most likely relevant for him.

Re-ranking for Recommendation Diversity

As the field of Recommender systems continues to evolve, various critical issues have been identified in the literature [28, 111], like the *Matthew effect* [173, 316] and *filter bubble* [196].

The *Matthew effect*, which can be summarized as "*the rich get richer and the poor get poorer*", occurs in recommendation when popular items or categories have increased chances of being recommended, while the less popular ones are overlooked. This is a critical concern to users seeking diverse recommendations, negatively impacting user satisfaction, even if they initially favoured such content. This lack of diversity can lead to the *filter bubble* phenomenon, where individuals are repeatedly exposed to content that reinforces their existing preference, isolating them from diverse perspectives and information. As a consequence, it fortifies previous preferences and forces the user to delve deeper into them. This situation is also called "group polarization" [91].

Extremely personalized recommendations can exacerbate these issues. To avoid this situation, it is imperative to introduce diversity into them. One way to do so is by solving a multi-objective optimization problem, balancing personalization and diversity.

3.1.3 Methodology

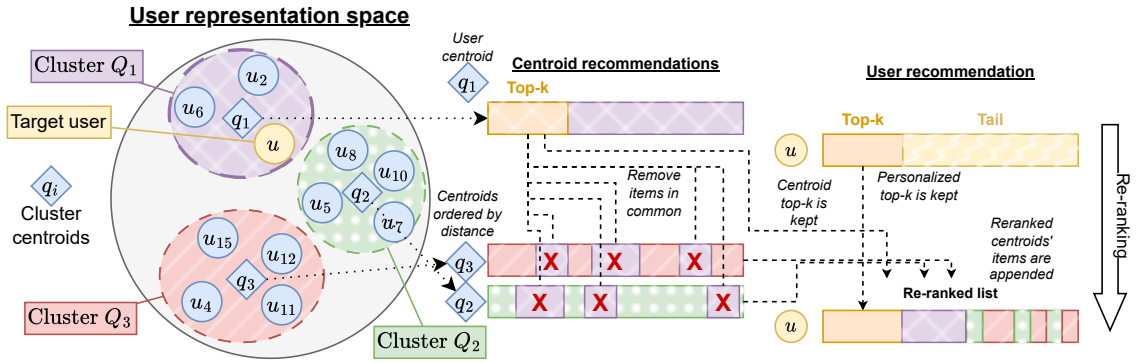


Figure 3.3: Proposed re-ranking algorithm. In each of the three clusters, the closest user to the centroid is considered as the centroid recommendation list. For each user, the top recommendations are frozen, and the others are re-ranked. Its cluster is not taken into consideration; items already present are excluded, while those not present in the intersection remain relevant; finally, in addition to the initially frozen ones, the remaining candidates are concatenated, and sorted by descending probability.

Graph construction

Interaction data. The first step involves building a bipartite heterogeneous graph, where nodes represent users and items, and edges connect users to items they have purchased.

In extreme cold-start scenarios, most nodes are isolated, due to the lack of interaction data for most users and items. We need to address this sparsity in order to make a viable prediction for isolated nodes. To do so, we employ link augmentation to create additional edges. As discussed in Section 3.1.2, the main idea behind our approach is to connect similar users and items. While for items we can just use their categorization, for users we need a way to describe them based on the most relevant factors influencing purchasing behavior.

Propensity model. Each user is represented by 600 features. Due to computational complexity, which is a major concern in industrial applications, we decide to reduce the number of features to 150 by performing PCA. This results in capturing over 80% of the explained variance, but with a 75% reduction of space.

A *propensity model* is trained to simulate a user's likelihood to purchase items: users who have bought at least one item are labeled as positive, while all the others are considered negative. For the actual algorithm, we decided to use an XGBoost binary classifier [53], a scalable end-to-end tree boosting model. Regarding the data, "no purchase" users significantly outnumber purchasing users, leading to a strongly unbalanced dataset. To avoid data splitting and overfit the model for retrieving the most relevant features, we address the imbalance by undersampling the majority class ("no purchase"). We then perform a grid search to optimize the XGBoost hyperparameters: maximum tree depth, learning rate and regularization. We consider optimal the model that achieves the highest F1 score.

Once the propensity model is trained, we compute its Shapley values. Typically used in XAI, they quantify the contribution of each feature to the model's prediction. This allows us to identify the top-10 most influential features for explaining user purchase behavior. Ultimately, this method reduces the computational complexity of similarity by 60 times, making the solution highly valuable for industrial applications where efficiency and interpretability are crucial.

Link augmentation

The key features of the *propensity model* identified via the Shapley values are utilized to connect users with similar characteristics in the graph construction stage. Users with similar behaviour are linked via Approximate Nearest-Neighbor (ANN). Connecting users with similar profiles is beneficial as they tend to exhibit similar purchasing patterns. This strategy helps predict new interactions based on established user behaviour. Especially in cold-start scenario, a user might influence other users connected to him, even though these haven't made a purchase yet.

We employ a similar approach connecting item nodes based on their categorization. This is based on the assumption that a user who already expressed interest in one item is more likely to buy another item in the same category. Linking isolated nodes within the same category facilitates information propagation between them, enabling the GNN model to capture latent relationships and similarities among items, even if initially missing direct connections. In this way, every item, even the unsold ones, is linked to the final graph.

Several open-source Python libraries like FAISS [61] and Hnswlib [183] offer GPU-accelerated ANN implementations. However, for industrial tasks, scalability becomes challenging. We leverage the Annoy library [26] which, thanks to its optimized memory scaling, allows us to augment hundreds of billions of edges.

In Fig. 3.2 are summarized these steps to achieve a denser graph.

Recommendation model

GNN model. We adopt a 2-layer GraphSAGE [101] model for our recommender systems. In GraphSAGE, the update function (UPD) combines concatenation and a linear layer, while the aggregation function (AGG) is a mean-pooling. According to Eq. (3.2), we compute the dot product between the user and item embeddings obtained from the GNN, providing a score for each item-user pair. Finally, for each user, the items sorted based on their predicted score give a ranked recommendation list. We randomly split the edges of the graph into training, validation and test sets with a 70%-20%-10% ratio. During training, we mask a portion of the edges for link prediction. For mini-batch training on the large-scale graph, we choose batches containing equal numbers of positive (real) edges and negative (non-existing), which are randomly sampled.

We utilize a weighted binary cross-entropy loss function for link prediction, as shown in the equation below:

$$\mathcal{L}(x, y) = -\frac{1}{N}\sum_{n=1}^{N}[y_n \log(x_n) + w_{neg}(1 - y_n)\log(1 - x_n)] \tag{3.3}$$

where N is the batch size, x_n is the model prediction for the $n - th$ element in the batch, y_n the corresponding target label (0 or 1), and w_{neg} is the parameter to weight the negative loss term.

To identify the best configuration for our GNN model, we perform hyperparameter tuning of input feature vector node dimension, number of neighbours considered in message-passing, model hidden space dimensions, optimizer used in training, and negative weight w_{neg} in the loss function. The objective of hyperparameter tuning is to minimize the loss function on the validation set.

Re-ranking

The GNN produces user and item embeddings, which are vector representations of user preferences and item characteristics, as well as their connections within the graph. To promote recommendation diversity and user serendipity (discovering unexpected but interesting items), we use a re-ranking approach.

Our re-ranking strategy leverages the user embeddings learned by the GNN. We proceed as follows:

1. **Community Detection:** We aim to identify behavioral communities in the user's embeddings latent space. We achieve this by clustering the users using k-means‖ [21]. The optimal number of clusters is determined by maximizing the Silhouette score [223], a metric for evaluating clustering quality. Let $Q = \{Q_1, \ldots, Q_K\}$ be the set of detected clusters and $q = \{q_1, ..., q_K\}$ their respective centroids.

2. **Distance to Other Centroids:** We compute the Euclidean distance between each user embedding h_u and cluster centroid q_k: $d = \sqrt{(h_u - q_k)^2}$.

3. **Cluster Centroid Recommendations:** For each cluster centroid q_k, we determine the closest user u. This user's recommendation item list is considered as the base recommendation for all users in the same cluster Q_k.

4. **Distance between centroids:** For each cluster centroid q_k, we compute the Euclidean distance among all of them q_k: $d = \sqrt{(q_u - q_k)^2}$. Then we choose the furthest as good candidates for the centroid re-ranking, $\hat{q_k}$, where $\hat{q_k} = \{q_1, ..., q_F\}$ is the set of candidates for q_k.

5. **Re-ranking Centroid Tail Recommendations:** We freeze the top K recommendations for each centroid and the tails are re-ranked. First, we remove from each centroid candidate q_f, the top recommendations of the centroid q_k. We consider for re-ranking those items that remain both in the recommended tail list of centroid q_k, Y_k, and the other centroids lists candidates, Y_f. This set is formally indicated as $Y_{k,f} = Y_k \cap Y_f$, $f \in \{1, \ldots, F\}$, with f the set of furthest centroids. The selected items $Y_{k,f}$ are sorted based on importance (position) in other lists, ordered from the furthest centroid list to the closest, and appended to the top K frozen centroids recommendations to form the final centroid recommendation list $\hat{Y_k}$.

6. **Re-ranking User Tail Recommendations:** Once the centroid lists have been re-ranked, we re-rank user recommendations based on his cluster centroid. In order to maintain the user personalization provided by the GNN, we decided to freeze the top-K user recommendations and re-rank the user tails based on their re-ranked cluster. To achieve this, once again, we initially removed the top-K when comparing user recommendations with those of their centroid, and then appended the remaining ones following the cluster recommendation's order.

This re-ranking approach preserves the quality of GNN results by retaining the first K recommendations. Simultaneously, it increases diversity by taking into account other communities' interests, potentially leading to serendipitous discoveries for users. Additionally, it helps reducing the group polarization problem. A potential drawback of this approach is that re-ranking all the tail items in the same order within a cluster might lead to some homogeneity. To bypass the problem, we create a large number of micro-clusters. By grouping users into smaller communities, the overlap in tail recommendations among users is reduced. Essentially. the tail recommendations are identical only for users belonging to the same micro-cluster. Re-ranking based only on centroids offers a significant computational advantage compared to using individual user recommendations. It reduces the cost from $|U| * K$ to K^2. Additionally, pre-computing tail recommendations for each cluster allows for faster response times for new users. This translates to substantial efficiency gains, especially for large user bases, making the solution highly scalable for industrial deployments. The described re-ranking algorithm is visualized in Fig. 3.3.

3.1.4 Experimental Results

Experimental Setup

Research Questions. We conduct a series of experiments to evaluate the effectiveness of our proposed approach, particularly focusing on its ability to handle cold-start and long-tail item recommendations. Our evaluation aims to answer the following research questions.

- **RQ1:** does the re-ranking algorithm improve recommendation diversity?
- **RQ2:** does the re-ranking algorithm promote long-tail item recommendations?
- **RQ3:** is the re-ranking algorithm robust to hyperparameter variations?

Table 3.1: The description of the dataset.

Dataset	Industrial			
	# tot	# warm	# cold	# extremely cold
items	693	29	412	252
users	1080681	13622	169505	897554

Dataset. TIM S.p.A. supplies various commodities to the customers, generally called IT items. However, the dataset provided by the company exhibits a significant cold-start challenge, as only 17% of the customer base has purchased any item. Among those, 93% have only bought one item. Approximately 40% of the items have never been sold, and the top three best-selling items collectively account for half of all sales. This extreme cold-start scenario presents a significant challenge for recommender systems. A detailed summary of the data is shown in Table 3.1.

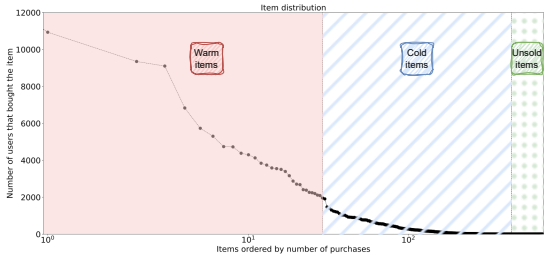


Figure 3.4: Internal dataset distribution: warm, cold, and unsold items are highlighted.

Given the dataset's characteristics, we deviate from the traditional approach of splitting items into warm and cold categories according to the Pareto Principle [15]. Instead, as presented in Table 3.1, we define user and item categories based on purchase behaviour:

warm users have purchased more than one item; **cold users** have only one purchase; **extremely cold users** have no purchases. **warm items** represent more than 1% of the total stock; **cold items** represent less than 1% of the total stock; **unsold items** have never been purchased. The item distribution is illustrated in Fig. 3.4.

Evaluation Metrics. We employ several metrics to assess the performance of our method.

Intra-List Distance (ILD) measures the average pairwise distance between items in a user's recommendation list.

$$ILD(R_u) = \frac{1}{|R_u|(|R_u|-1)} \sum_{i \in R_u} \sum_{j \in R_u \setminus i} d(i,j) \tag{3.4}$$

where $d(i,j)$ is the Euclidean distance between item embeddings derived from the graph, and $|R_u|$ is the cardinality of the set of items.

Table 3.2: ILD results of GNN, Random Shifted GNN and re-ranking algorithm. Experiments are repeated five times with a different weight initialization; mean and variance are reported. **Bold** indicates the best result, while our metrics surpassing the baselines are underlined. * denotes statistically significant ($p<0.01$) improvements over the top-performing baseline.

Method	ILD@K				
	K=5	K=10	K=20	K=50	K=100
GNN	6.5249 ± 1.9822	6.4746 ± 1.7203	6.6528 ± 1.7000	6.6797 ± 1.7088	7.1063 ± 1.8448
Random Shifted GNN	7.0976 ± 4.2660	1.5773 ± 0.9480	0.3735 ± 0.2270	0.0580 ± 0.0348	0.0143 ± 0.0086
Re-ranked F@5	-	**16.9174 ± 3.5631***	**22.2285 ± 1.4671***	**22.1086 ± 0.5082***	**21.9093 ± 0.2157***
Re-ranked F@10	-	-	10.8625 ± 1.8004*	10.3264 ± 0.9937*	10.1021 ± 0.4554*
Re-ranked F@20	-	-	-	11.0949 ± 0.6530*	12.6525 ± 0.2326*
Re-ranked F@50	-	-	-	-	15.2856 ± 1.5690*

Table 3.3: Coverage results of GNN results and after re-ranking. **Bold** indicates the best result, while second-best is underlined.

Method	Coverage@K									
	K=5	K=10	K=20	K=50	K=100	K=200	K=300	K=400	K=500	K=600
GNN	**0.3255**	**0.3967**	**0.4877**	**0.6582**	**0.8172**	**0.9389**	**0.9895**	**1.0000**	**1.0000**	**1.0000**
Re-ranked F@5	-	0.3578	0.3839	0.4176	0.4688	0.5679	0.6510	0.7471	0.8323	0.9248
Re-ranked F@10	-	-	0.4294	0.4825	0.4983	0.5788	0.6488	0.7312	0.8323	0.9219
Re-ranked F@20	-	-	-	0.5004	0.5365	0.6137	0.6941	0.7890	0.8699	0.9364
Re-ranked F@50	-	-	-	-	0.7036	0.7736	0.8255	0.8800	0.9205	0.9624

Table 3.4: Type-Coverage improvement results for different K. Experiments are repeated freezing at different ranges in re-ranking algorithm. The coverage improvements are presented for warm (W), cold (C) and unsold (U) items. Bold indicates the type of item that has lower decreases for a specific K, underlining represents the general best results for the type of item.

Method	Type-Coverage@K					
	W@100	C@100	U@100	W@200	C@200	U@200
GNN	0.8534	0.7384	0.9385	0.9493	0.9017	0.9921
Improvements F@5	-0.5557	-0.6589	**-0.1204**	-0.4828	-0.5730	**-0.1191**
Improvements F@10	-0.4500	-0.5876	**-0.1623**	-0.3461	-0.5309	**-0.1841**
Improvements F@20	**-0.2501**	-0.3799	-0.3018	**-0.2414**	-0.3873	-0.2939
Improvements F@50	-0.0833	-0.1931	**-0.0736**	-0.0812	-0.2434	**-0.0755**

Coverage@K: This metric calculates the proportion of unique items appearing in the top-K recommendations across all users:

$$Coverage@K = \frac{|\cup_{u \in U} Y_K(u)|}{|R_u|} \tag{3.5}$$

where $Y_K(u) = [i_1, ..., i_k]$ represent the top-K recommendation list for user u and R_u the set of all items.

Type-Coverage@K: Similarly, this metric focuses on the coverage given to long-tail items:

$$Type - Coverage@K = \frac{|\cup_{u \in U} Y_K^{Type}(u)|}{|R_u^{Type}|} \tag{3.6}$$

where $Y_K^{Type}(u)$ and $|R_u^{Type}|$ refer to a specific item type.

We evaluate the Type-Coverage@K differentiating between warm, cold and unsold items.

Baselines. Currently, TIM S.p.A. is not using any machine learning algorithm to recommend IT items. In this extreme cold-start setting, good performances would correspond to overfitting on the

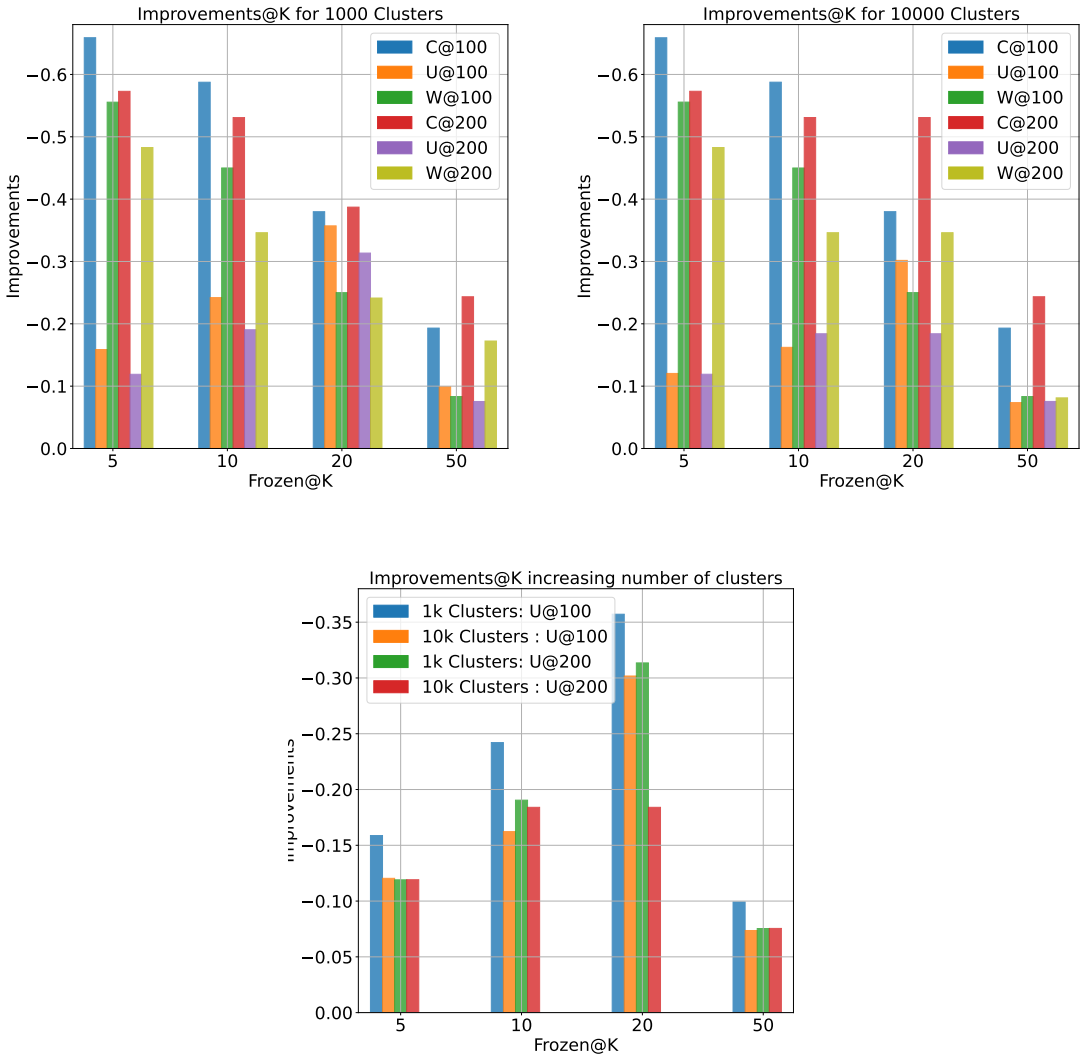


Figure 3.5: From left to right: Type-Coverage@K improvements for a low number of clusters, Type-Coverage@K improvements for a high number of clusters, and Unsold-Coverage@K improvements between low and high number of clusters.

few items sold. This is opposite to the idea of pushing a limited amount of items previously decided in every campaign. All these reasons add up to why we consider the GNN as one of the baselines against which to compare our re-ranking approach.

Additionally, we decide to compare our results with a random baseline. For each user, we randomly shuffle the recommended item list. In Table 3.2, we refer to it as Random Shifted GNN, and the ILD results are calculated based on the obtained GNN item embeddings.

Results

This section presents the evaluation results for our proposed approach, addressing the research questions outlined earlier.

Diversity. To answer **RQ1**, we evaluate the impact of re-ranking on recommendation diversity by comparing the ILD obtained by the GNN with that resulting from the re-ranking algorithm.

The results presented in Table 3.2 show that re-ranking methods increase the ILD of the recommendations by **+126.7%** in average, across list lengths ranging from 10 to 100. This improvement is attributed to the re-ranking strategy. By freezing the top-K recommendations and incorporating items from the furthest cluster centroids, the model injects items with a lower user-community overlap, leading to more diverse recommendations.

Long tails. However, altering the user tails using those of the centroids, introduces a trade-off. Since users within the same cluster probably receive similar tail recommendations, the overall item coverage is expected to decrease. Table 3.3 summarize the coverage results to answer **RQ2**. As expected, overall coverage decreases due to re-ranking. However, since our focus is on long-tail items, we should turn our attention to Type-Coverage@K, shown in Table 3.4. The coverage for unsold items shows the smallest decrease, while cold items show less coverage than warm items. This suggests that the re-ranking algorithm is effective at promoting long-tail recommendations, particularly for items that have never been purchased before.

Hyperparameter sensitivity. To assess the robustness of our approach to hyperparameter variations **RQ3**, we present the effect of four hyperparameters in re-ranking algorithm: the number of clusters, the number of candidates for each centroid, the number of items to freeze in the centroid-centroid and in the user-centroid re-ranking step. In order to reduce the number of experiments, we set the same number of frozen items in centroid re-ranking and user re-ranking, reducing to three hyperparameters.

We use the top 10 and top 50 furthest centroids in re-ranking approach and we notice that the number of furthest centroids considered is not impacting the ILD and type-coverages up to one hundred frozen items. So, we go deep into the variations in the number of clusters and the amount of items to freeze.

The number of frozen recommendations determines how much GNN's initial predictions are preserved. As shown in Tables 3.2 to 3.4, a higher number of frozen items produces both a higher ILD and a higher coverage for all three item types. As shown in Fig. 3.5, the coverages decrease more as the number of frozen items increases.

Furthermore, the deterioration in coverage for unsold items increases with each K up to 20, and at 50, they have the lowest value overall. At 50, the degradation for cold items is at its highest. This is because the number of warm items is 29, and for low K values, the re-ranking algorithm places tail items (unsold) in the top positions. Once the number of warm items is exceeded, these unsold items end up disadvantaging the cold items, lowering their type coverage.

Moreover, when the number of clusters increases, the amount of unsold items is always decreasing, for each number of frozen items, @100 and @200. This happens because having a larger number of clusters allows for the identification of more behaviors within the micro-clusters, thereby favoring the unsold items more.

Finding the optimal hyperparameter configuration requires careful consideration of the desired balance between diversity and coverage within the specific application context.

The results presented in RQ1 and RQ2 are obtained with one thousand clusters and ten furthest centroid candidates.

3.1.5 Conclusions

This paper addressed the extreme cold-start problem for new users and items in a recommender system. We proposed a novel end-to-end framework designed to generate recommendations in an extreme cold-start scenario. The framework utilizes relevant customer features to connect users, aiming to prevent isolated nodes in the graph and facilitate information flow. Additionally, we introduced a re-ranking algorithm to enhance user diversity.

We conducted experiments to evaluate the effectiveness of our approach on a real-world dataset with extreme cold-start characteristics. The results show that our method significantly improves recommendation diversity compared to the baseline GNN model. Additionally, it shows promising results in promoting long-tail item recommendations, especially for unsold items.

For future work, we plan to incorporate additional information sources, such as external surveys, to improve GNN embeddings' quality. The algorithm has also been deployed in production and we will monitor user feedback to refine the framework. Finally, we aim to investigate state-of-the-art neural multi-objective re-ranking models for potentially even better performance and flexibility.

3.2 Balancing the Diversity-Coverage Trade-off in Graph-based Recommendations for Cold-Start Industrial Applications

In extreme cold-start scenarios, where users and items have little or no interaction history, traditional recommender systems struggle to generate diverse and meaningful suggestions. This work introduces a unified framework designed to address this challenge using two complementary techniques: (i) a graph augmentation module which densifies the user-item graph using feature-based link prediction, and (ii) a re-ranking algorithm which increases recommendation diversity via centroid-aware sampling across distant user clusters. The resulting architecture also combines graph neural networks for representation learning.

We empirically evaluate the framework using a real-world industrial dataset from the telecommunications sector, which is characterized by severe data sparsity and item imbalance. A comprehensive exploration of the possible hyperparameter configurations is conducted, allowing for the assessment of diversity-coverage trade-off and enabling the selection of a configuration that meets specific objectives.

The results provide actionable insights for practitioners, demonstrating how to optimize the framework according to specific goals, whether prioritizing diversity, coverage, or computational efficiency. This work serves as a resource for deploying scalable high-quality recommendation systems in real-world industrial settings, addressing the unique challenges posed by extreme cold-start scenarios.

3.2.1 Introduction

Recommender systems are critical components of modern digital platforms, driving user engagement and business revenues by providing personalized suggestions for products, services, or contents. With the proliferation of user-item interactions in areas such as e-commerce [8, 233], entertainment [30, 48, 157], and social networking [14, 214, 237], recommender systems have evolved significantly, using sophisticated models to deliver highly accurate predictions [227].

Despite these advances, the challenge of dealing with cold-start scenarios - where new users or items lack sufficient interaction data - remains a significant barrier to their effectiveness [117, 288]. In industrial environments, where systems must support millions of users and dynamically evolving item catalogs, the extreme cold-start problem is exacerbated, having a direct impact on user satisfaction and business outcomes [12, 230].

The extreme cold start scenario is particularly challenging due to its inherent sparsity [39]. New users often join the platform with no prior history of interaction, and new items may be added to the catalog with limited or no contextual information. Traditional collaborative filtering methods [245] struggle in this environment because they rely heavily on historical data to establish user-item relationships. While content-based and hybrid approaches can partially address the problem by incorporating ancillary information such as user demographics or item metadata [118, 177, 267], these solutions often fail to capture the complex and evolving preferences reflected in high-order graph structures [37]. Graph Neural Networks (GNNs) have emerged as a promising avenue for recommender systems [88, 212] by effectively encoding such structural information [139, 161, 304]. However, even state-of-the-art GNN-based recommenders show limitations when confronted with the extreme sparsity of cold-start scenarios [46, 104].

This paper addresses the extreme cold-start problem by investigating how to balance three core objectives in industrial recommender systems: accuracy, item coverage, and list diversity. These goals often conflict with each other: models that maximize accuracy tend to favor popular, well-known items, which limits long-tail exposure. Conversely, increasing diversity can reduce personalization, catalog coverage, and short-term precision. This raises the following research question: *how can we balance the trade-off between item coverage and recommendation diversity in extreme cold-start settings while minimizing the accuracy degradation?*

To address this issue, our previous work [230] introduced a novel framework that combines graph link augmentation and re-ranking strategies to address extreme cold-start challenges. Using Explainable AI (XAI) techniques, the framework identifies user similarities to augment a sparse graph, allowing for improved information propagation through a GNN. The subsequent re-ranking stage enhances the diversity of recommendations, addressing well-documented issues such as *filter bubbles* [196] and the *Matthew effect* [173, 316]. This framework is designed to scale to real-world industrial settings and is empirically validated on a large proprietary dataset from TIM S.p.A., a leading telecommunications company.

Building on this foundation, this paper presents an extended and more comprehensive evaluation of the proposed framework. Specifically, we provide:

- Extended related work: A deeper analysis of the state of the art in cold-start recommendations, GNNs, and re-ranking strategies, highlighting how our approach compares to and contributes to these advances.
- Expanded insights into methodology and scalability: We provide a more detailed explanation of the framework's design, including the theoretical justifications for its components and the choices made in its implementation. In addition, we discuss scalability in depth, exploring optimizations and architectural improvements that enable the framework to efficiently handle large-scale industrial deployments.
- Ablation studies: We analyze the impact of key parameters such as the number of augmented

links and the number of selected features, providing insights into the robustness and scalability of the framework. Moreover, we evaluate results on a GNN architecture variant, the graph augmentation impact, the influence in accuracy for the proposed re-ranking approach, and the choice of farthest centroids in the re-ranking step.

- Additional baselines: To contextualize our results, we introduce and evaluate an additional baseline method, highlighting the unique strengths and weaknesses of our approach.

By addressing these aspects, this paper bridges the gap between industrial relevance and academic rigor, contributing valuable insights to both communities. From an academic perspective, it advances the understanding of cold-start mitigation techniques in graph-based recommender systems. From an industrial perspective, it provides scalable, interpretable, and effective solutions applicable to large-scale deployments.

The rest of this paper is structured as follows: Section 3.2.2 provides a comprehensive review of related work, situating our approach within the broader landscape of recommender systems. Section 3.2.3 describes the methodology, detailing the core components of the framework and the design rationale behind each. Section 3.2.4 outlines the experimental setup, including datasets, evaluation metrics, and baseline configurations. Section 3.2.5 presents the results, analyzing the performance of the framework across experimental conditions. Section 3.2.6 discusses an ablation study of the framework, evaluating a different GNN architecture, highlighting both the strengths and limitations of the graph augmentation step, describing the re-ranking impact on accuracy, and assessing the choice of farthest cluster selection in the re-ranking step. Finally, Section 3.2.7 concludes the paper and outlines directions for future research.

3.2.2 Related Work

Recommender systems have made significant progress, with GNNs emerging as powerful tools for encoding high-order structural information in user-item interactions. However, challenges such as cold-start scenarios, recommendation diversity, and scalability remain, especially in industrial applications. In this section, we review recent advances in GNN-based recommendation, strategies for addressing cold-start challenges, re-ranking algorithms, and their industrial relevance. In addition, we highlight gaps in the existing literature and position our approach as a solution to these issues.

GNN-based Recommender Systems

Early recommendation approaches, such as collaborative filtering [245], relied on matrix factorization techniques to capture latent relationships. While effective in many scenarios, these methods often overlooked the structural richness inherent in user-item graphs. For example, indirect relationships, such as users with overlapping neighbors or items that are frequently purchased together, were often neglected. Graph Neural Networks (GNNs) overcome these limitations by employing message-passing mechanisms that iteratively update node embeddings based on the features of their neighbors. This ability to capture both local and global graph structures has made GNNs a natural fit for recommendation tasks, where user preferences and item characteristics are often linked by complex networks.

GNNs provide a powerful framework for learning on graph-structured data, with methods tailored to different applications. Graph Convolutional Networks (GCNs)[139] use spectral convolutions to efficiently aggregate and propagate node information, but face scalability challenges in large graphs. To address this, GraphSAGE[101] introduces an inductive learning approach that allows dynamic generation of embeddings for unseen nodes while reducing computational overhead by sampling a fixed number of neighbors. Further advances, such as Graph Attention Networks (GATs)[254], incorporate attention mechanisms to prioritize the most relevant connections, enhancing personalization and adaptability. These general GNN techniques have laid the groundwork for applying graph-based learning to various domains, including recommender systems and social networks.

In the realm of recommender systems, GNNs have shown significant promise in capturing both direct and high-order relationships within user-item bipartite graphs. LightGCN[109], a widely cited work in this domain, simplifies traditional GCN architectures by removing non-linear activation functions and focusing solely on linear aggregation and embedding propagation. This design choice not only reduces computational overhead but also improves performance for recommendation tasks, where the key is to learn effective user and item embeddings. Neural Graph Collaborative Filtering (NGCF)[266] builds upon traditional collaborative filtering by integrating graph-structured embeddings, enabling the model to capture multi-hop relationships that go beyond direct interactions.

Other efforts to integrate GNNs into recommender systems have included various domain-specific adaptations, such as incorporating side information [118, 267], leveraging hybrid models that combine GNNs with matrix factorization[187, 265], and designing scalable training algorithms to handle large datasets[170, 302].

Despite their success, most GNN-based recommenders assume sufficient connectivity in the interaction graph. In extreme cold-start settings, where nodes are isolated, the effectiveness of GNN embeddings decreases significantly. Methods such as GPatch [46] and pretrained GNNs [104] attempt to address this issue through architectural changes or transfer learning; however, these approaches often introduce significant computational overhead or require auxiliary interaction data. In contrast, our method employs a lightweight augmentation strategy to enhance graph connectivity before GNN training without requiring model pre-training

Cold-Start Challenges

The cold-start problem, where new users or items lack sufficient interaction data, remains a major obstacle in recommender systems. This issue disrupts the ability of algorithms to establish meaningful connections between users and items, resulting in suboptimal recommendations. Addressing cold-start scenarios is particularly critical in dynamic environments, such as e-commerce platforms and streaming services, where catalogs and user bases continuously evolve.

Traditional recommendation approaches have long attempted to address cold-start scenarios using various strategies, primarily focusing on content-based filtering (CBF) [211, 273]. CBF leverages descriptive information about users and items, such as demographics or textual metadata [30, 48]. By training models on feature representations, CBF methods can provide recommendations based on similarities in content. While effective in some cases, these approaches depend heavily on the availability and quality of metadata, which can vary widely across domains. Recent advancements have shown potential for mitigating cold-start issues. LLMs, for example, can leverage pretrained

embeddings to encode semantic relationships based on item metadata or language features. This capability allows them to provide competitive recommendations even in sparse settings [227]. Similarly, generative models [288] create synthetic interactions to enrich training datasets, addressing both data sparsity and recommendation diversity. These approaches exemplify the growing trend of leveraging external knowledge and synthetic data to improve cold-start performance.

GNNs, despite their success in capturing high-order relationships in user-item graphs, face unique challenges in cold-start scenarios due to their reliance on graph connectivity. Sparse or missing connections for new nodes significantly hinder the information propagation necessary for generating high-quality recommendations. Several methods have been proposed to adapt GNNs to these scenarios by enriching the graph structure or enhancing the embedding process.

Cai et al. [37] address user cold-start recommendations by introducing the Inductive Heterogeneous Graph Neural Network (IHGNN), which captures rich relational information across multiple modalities. To mitigate the sparsity of user attributes, they use a random walk-based strategy for collecting associated neighbors, followed by a hierarchical attention mechanism to aggregate meaningful connections. This approach effectively enriches sparse user attributes and generates high-quality representations for cold-start users.

Building on the use of external data and attributes, Cao et al. [39] propose ColdGPT, a multi-task framework designed for extreme cold-start scenarios, where user-item interactions are entirely unavailable. ColdGPT constructs a fine-grained item-attribute graph using data from various sources, such as item contents, purchase sequences, and review texts. This approach enhances the embeddings for cold items by enabling the model to infer item relationships through attribute propagation, achieving significant improvements over state-of-the-art methods on public datasets.

Pretraining techniques have also emerged as a promising direction to enhance graph-based recommendations in cold-start scenarios. Hao et al. [104] propose a pretrained GNN framework that simulates cold-start conditions by reconstructing embeddings from users/items with sufficient interactions. This process explicitly optimizes embeddings for cold-start scenarios, allowing the model to generalize effectively to unseen users/items while reducing reliance on cold-start neighbors during training.

Innovative architectural adaptations further contribute to solving the cold-start challenge. Lu et al. [177] introduce HyperRS, a hypernetwork-based recommender system that does not rely on demographic information. A hypernetwork generates all weights in the underlying recommender system, enabling the weights to adapt quickly to capture user interests. This approach ensures high adaptability and robust performance in cold-start settings, even when user demographic data is unavailable or sparse.

GPatch[46] presents a dual-component framework to handle both warm and cold-start users/items. It employs an efficient GNN architecture for warm scenarios, and a Patching Network to simulate recommendations for cold-start users/items. This dual design ensures that the performance for warm users/items remains uncompromised while effectively integrating cold-start entities into the graph structure.

While these methods have significantly advanced the state-of-the-art, they are not without limitations. Many rely on the availability of extensive attribute data or external datasets for pretraining, which may not always be feasible in real-world industrial applications. Computational complexity also remains a concern, particularly for methods requiring large-scale pretraining or dynamic

model adaptations.

Our work takes a different approach. We construct feature-based user-user and item-item graphs, retaining only the most informative user attributes. Although existing methods have explored edge prediction for graph completion [68, 272, 311], they usually predict missing user-item links [71] or optimize different tasks (e.g. rating prediction) [305]. To the best of our knowledge, no prior method combines a supervised classifier for purchase probability with SHAP-informed feature selection for similarity-based augmentation

Re-ranking Algorithms

Re-ranking is increasingly recognized as a critical step in recommender systems to address challenges such as over-personalization, diversity and fairness. As recommender systems are optimized for user engagement, they often inadvertently exacerbate problems such as the *Matthew effect*. [173, 316] and the *filter bubble* [196]. The *Matthew effect* describes the tendency for popular items to become even more dominant in recommendation lists, marginalizing less popular or long-tail items. This imbalance can reduce user satisfaction by exposing users to repetitive and overly familiar content. Similarly, *filter bubbles* occur when recommendation algorithms reinforce users' existing preferences, limiting exposure to diverse or serendipitous items. Over time, these effects can lead to group polarization [91], reducing the overall utility of the system and user retention.

To mitigate these challenges, re-ranking algorithms strive to introduce diversity and novelty into recommendation lists without sacrificing relevance. Recent works have proposed innovative approaches to this problem, leveraging advanced modeling techniques.

Ren et al. [221] proposed NAR4Rec, a non-autoregressive generative model that overcomes the limitations of traditional autoregressive methods, such as slow inference and error accumulation. By generating and evaluating multiple feasible item sequences simultaneously, NAR4Rec captures intra-list dependencies efficiently, demonstrating superior performance in both offline and online tests on large-scale industrial platforms.

Similarly, Gu et al. [95] introduced a hierarchical attention-based encoder combined with SciBERT for local citation recommendation. This method improves reranking accuracy by leveraging fine-grained textual context while reducing the computational burden of candidate generation.

Li et al. [163] proposed a hybrid model for academic paper recommendations that combines user behavior and content similarity. By utilizing a knowledge graph to assess entity-level relationships, their model demonstrated significant performance improvements, emphasizing the value of recent user activity.

Despite these advancements, significant gaps remain in the re-ranking literature. These techniques are often orthogonal to the use of GNN embeddings, and few have been fully integrated into pipelines designed to mitigate cold-start. Many existing methods focus on specific metrics, such as diversity [41, 87, 289] or fairness [92, 190, 285], but fail to address scalability in real-world scenarios. Our cluster-based re-ranking strategy, on the other hand, is straightforward and scalable: it introduces diversity by recommending items from different user clusters, which can be considered a mild form of novelty injection or serendipity

Industrial Setting

The use of recommender systems in industrial settings presents unique challenges, including scalability, latency and diversity. Industrial environments demand algorithms that can process extensive user-item interactions efficiently while adhering to strict latency requirements. Furthermore, ensuring that recommendations are diverse and relevant is critical to maintaining user satisfaction and optimizing business outcomes.

In large-scale industrial environments, scalability is often hindered by the computational demands of constructing and processing massive user-item graphs. GNN-based recommenders, while powerful in leveraging structural information, face challenges in handling the complexity of graph construction and convolution operations at scale. While academic advances have proposed sophisticated algorithms to improve recommendation quality, empirical studies show that some e-commerce platforms continue to favor simpler methods, such as best-seller lists, for their reliability and ease of deployment [202]. Lastly, industrial systems often struggle with issues of diversity and fairness. Over-personalized recommendations can reduce exposure to long-tail or unsold items. This not only impacts user satisfaction but also limits the business value of underutilized inventory. Achieving a balance between relevance and diversity is thus a critical goal for industrial recommender systems.

3.2.3 Methodology

Our framework consists of three primary stages designed to address the extreme cold-start problem in recommender systems:

1. The initial user-item graph, characterized by sparse interactions, is enriched through a multi-step process, transforming it into a dense structure that facilitates effective information propagation.

2. A GNN is trained on the augmented graph to generate embeddings that capture the structural relationships between users and items.

3. To improve recommendation diversity, a centroid-based re-ranking strategy is applied to the GNN-generated predictions. This step reorders the recommendation lists by incorporating long-tail items, balancing personalization and diversity.

Together, these steps enable our framework to deliver accurate, diverse and scalable recommendations in industrial contexts. The rest of the section provides an in-depth explanation of the framework, detailing its components and the scalability enhancements that make it suitable for industrial applications.

From Sparse to Dense Graph

The starting point of our framework is a bipartite user-item graph constructed from interaction data. In this graph, each node represents either a user or an item, and the edges encode historical interactions

Sparse graphs are common in recommender systems, especially in cold-start scenarios where many users and items lack interaction data. In such cases, the limited number of edges severely limits the ability to propagate information across the graph, leading to weakly connected or even isolated

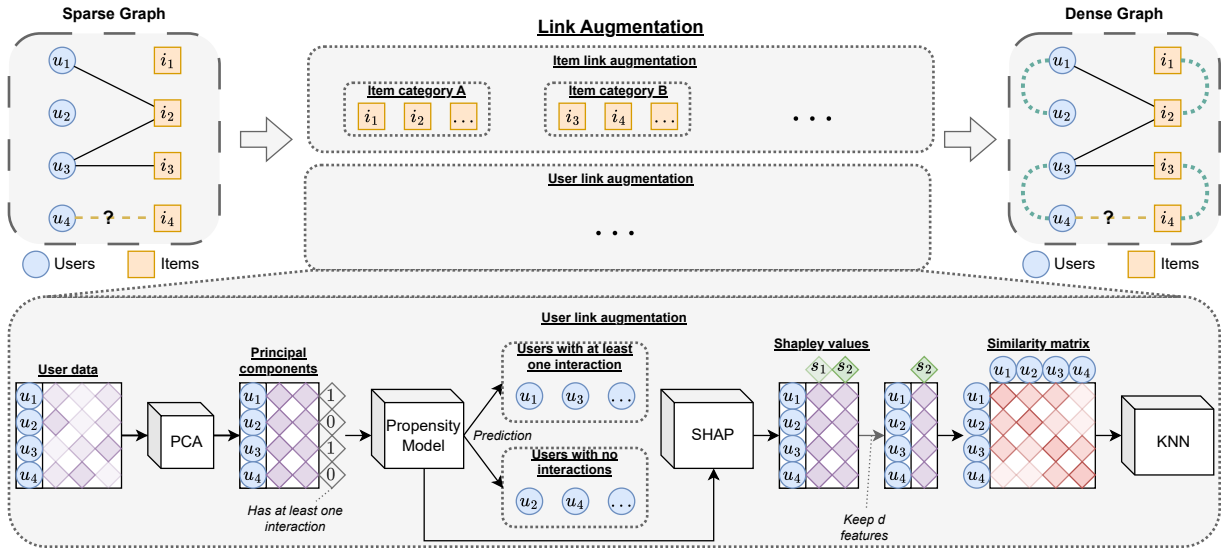


Figure 3.6: A visual depiction of the dense graph construction pipeline.

nodes. For a GNN, this results in poor representation learning, as message passing is confined to very local neighborhoods, thereby reducing both recommendation quality and the system's ability to generalize

Therefore, transforming a sparse user-item graph into a denser one is a critical step in our framework. A denser graph offers two main advantages. First, additional edges create multiple connection paths between users and items, allowing collaborative signals to travel farther and mitigating the isolation of cold-start nodes. Second, by increasing neighborhood connectivity, the GNN can aggregate richer and more diverse contextual information, leading to more stable and discriminative node representations. This is particularly important for our objectives of improving coverage, diversity, and cold-start performance

In our approach, densification is achieved through four interconnected components: Principal Components Analysis, propensity modeling, XAI-driven feature selection, and link augmentation. This enriched connectivity enables the GNN to leverage both direct and indirect relationships, unlocking the full potential of our re-ranking strategy

Principal Components Analysis

Fig. 3.6 presents an overview of the complete link augmentation procedure.

The raw user data in our framework is represented by over 600 features, capturing diverse aspects such as demographic details, browsing patterns and device usage. While this high-dimensional data provides valuable insights, it also poses significant challenges in terms of scalability and computational efficiency, especially for industrial-scale datasets such as TIM S.p.A.'s. To address these issues, we use Principal Component Analysis (PCA)[207] as a preprocessing step to reduce the feature space while retaining the most relevant information.

PCA transforms the original high-dimensional data into a smaller set of orthogonal components, prioritizing those that capture the majority of the variance in the dataset. For our framework, we select the top m principal components. This reduction results in a $\left(100 - \frac{m}{6}\right)\%$ decrease in computational complexity, making subsequent processes more efficient.

This dimensionality reduction is particularly beneficial in managing the trade-off between preserving meaningful behavioral patterns and optimizing for scalability. For example, the reduced feature set enables faster computation during the K-Nearest Neighbors step, which is the backbone of the graph augmentation process. It also minimizes the risk of overfitting in propensity modeling by focusing on the most critical features that influence user behavior.

By ensuring that only the most influential components of the data are retained, PCA contributes to the overall efficiency and scalability of the framework, making it a crucial step in preparing the data for dense graph construction.

Propensity Modeling

Propensity modeling's goal is to predict the likelihood of user-item interactions. This step is essential for understanding user behavior in the absence of sufficient interaction data, and forms the basis for connecting isolated nodes to the broader graph.

For this task, we use XGBoost [52], a gradient boosting framework well suited to tabular data. XGBoost's inherent scalability and ability to handle large data sets make it ideal for industrial applications.

The model is trained as a binary classifier, where users with at least one recorded interaction are labeled as positive and those with no interactions are labeled as negative. This approach allows the prediction of interaction probabilities for all users, even in the absence of explicit data.

The TIM dataset presents a significant challenge due to class imbalance, as non-interacting users greatly outnumber interacting users. To address this, we balance the training data by undersampling, which ensures a representative distribution of positive and negative samples while also improving training efficiency. In addition, hyperparameters such as learning rate, tree depth, and regularization are fine-tuned through grid search, optimizing the model for F1 score.

XAI-based feature selection

To ensure efficiency in graph augmentation, we integrate XAI techniques into the feature selection process. Shapley values [178, 244] provide a principled way to quantify the contribution of each feature to the output of a predictive model. By identifying and retaining only the most influential features, we significantly reduce the computational overhead while maintaining the quality of the augmentation process.

The XAI-based feature selection is applied to the output of the propensity model, where Shapley values measure the importance of individual user features in predicting interaction probabilities. This step narrows down the original m features (obtained after PCA) to the top d most influential attributes. These features capture the key drivers of user behavior, ensuring that the most relevant information is used to populate the graph.

This reduction has another significant effect on computational efficiency. Computing similarities between users or items in high-dimensional spaces can be prohibitively expensive for large datasets. By focusing on a reduced set of key features, similarity computations during the K-Nearest Neighbors step become $\frac{m}{d}$ times faster, allowing the framework to scale effectively to industrial-scale datasets.

Furthermore, this approach ensures that the augmented graph is based on meaningful behavioral patterns, improving the quality of subsequent GNN training. By using Shapley values for feature se-

lection, we strike a balance between computational efficiency and preserving meaningful information.

Link Augmentation

The final step in transforming the sparse user-item graph into a dense structure is link augmentation, which connects cold start nodes to their most similar counterparts. This step is critical for allowing information to propagate throughout the graph, ensuring that isolated users and items can contribute to and benefit from the recommendation process.

Link augmentation is based on the reduced feature set identified by Shapley values. For user-user links, the similarity is determined based on shared behavioral patterns reflected in the top d selected features. For item-item links, product type is used as the basis for similarity. We employ a similar approach connecting item nodes based on their categorization. This works by assuming that a user who has already expressed interest in an item is more likely to want to buy another in the same category. This approach ensures that even nodes with minimal interaction history are embedded in a larger, connected graph structure.

Due to the large size of the dataset, exact similarity calculations are not feasible. Instead, we use Approximate Nearest Neighbor (ANN) methods, in particular the Annoy [26] library. Annoy builds memory-efficient search trees that allow for fast similarity computations, allowing for the augmentation of hundreds of billions of edges while maintaining scalability. The choice of Annoy ensures that link augmentation can be performed efficiently even on industrial scale graphs with millions of users and items.

The augmented graph is significantly more connected than the original sparse graph. Extremely cold start users, who were previously isolated, are now connected to warm start users with similar behavioral tendencies. Similarly, unsold items are connected to related products based on shared categories, allowing them to influence and be influenced by recommendations. This enriched graph structure provides a strong foundation for the subsequent GNN training stage, where embeddings are learned for all nodes, including those that were previously unconnected.

Graph Neural Network Recommender

Once the dense graph is constructed through link augmentation, the next step in our framework is to train a GNN to learn node embeddings. These embeddings capture the relationships between users and items, allowing prediction of interactions, even for cold-start nodes. GNNs are particularly well suited to this task because they iteratively aggregate and propagate information through the graph, capturing both local and global structural patterns.

The GNN component in our framework is implemented using a 2-layer GraphSAGE[101] model, chosen for its inductive learning capabilities, which allow it to generalize to unseen nodes in dynamic graphs. The training process consists of three main steps:

Message Passing and Aggregation

The core of GNNs is the message-passing mechanism, which updates the embeddings of each node by aggregating information from its neighbors. For a given node u, the embedding at layer $k+1$ is calculated as:

$$h_u^{(k+1)} = \text{UPDATE}_k \left(h_u^{(k)}, \text{AGG}_k \left(\{ h_v^{(k)} \mid v \in N(u) \} \right) \right) \tag{3.7}$$

where $h_u^{(k)}$ is the embedding of node u in layer k, $N(u)$ denotes the set of neighbors of u, and AGG_k and UPDATE_k are the aggregation and update functions, respectively.

In our implementation, the aggregation function is sum pooling, which computes the average of the embeddings of neighboring nodes. The update function combines this aggregated information with the current embedding of the node via concatenation, a linear transformation and ReLU activation.

This iterative process ensures that node embeddings incorporate information from increasingly distant neighbors with each additional layer.

Link Prediction

The trained GNN is used for the link prediction task, where the goal is to estimate the probability that an edge exists between a user node and an item node. The prediction score for a user u and an item i is computed from the dot-product of their embeddings:

$$\hat{y}_{ui} = h_u^\top h_i \tag{3.8}$$

where h_u and h_i are the final embeddings for user u and item i, respectively. These scores are used to rank items for each user and form the basis of personalized recommendations.

Efficient link prediction in large graphs relies heavily on the dimensionality of the embeddings, which directly affects both the representational capacity of the model and its computational efficiency. By optimizing the embedding dimensionality, the framework achieves a balance between capturing rich, expressive node representations and maintaining the speed required for fast inference.

Training

The GNN is trained using a weighted binary cross-entropy loss, designed to balance the contributions of positive (present) and negative (absent) edges:

$$\mathcal{L} = -\frac{1}{N}\sum_{n=1}^{N}\left[y_n \log(\hat{y}_n) + w_{\text{neg}}(1-y_n)\log(1-\hat{y}_n)\right] \tag{3.9}$$

where N is the batch size, y_n is the ground truth label of nth edge, $\hat{y}_n$ is the predicted score, and w_{neg} is the weight for negative edges, which accounts for class imbalance.

To scale training for large datasets, we use mini-batch training, which processes smaller subgraphs rather than the entire graph in each iteration. This significantly reduces memory requirements, making it feasible to train on graphs with millions of nodes and edges. Batches are constructed with an equal number of positive and negative edges, where negative edges are dynamically sampled to ensure diversity in the training examples and to prevent overfitting.

By combining message passing, efficient link prediction and scalable training strategies, the GNN learns embeddings that accurately represent the relationships between users and items. These embeddings provide the basis for generating high quality recommendations, even for cold start nodes. The next stage of the framework builds on these embeddings to further improve recommendation diversity through re-ranking.

Re-ranking for diversity

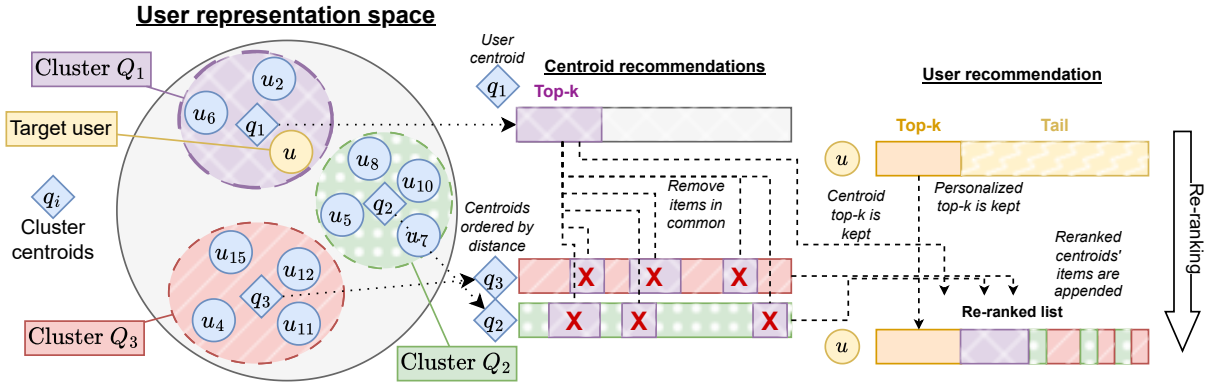


Figure 3.7: Proposed Re-Ranking Algorithm. For each of the three clusters, the user closest to the cluster centroid is selected to represent the centroid's recommendation list. During the re-ranking process, the top recommendations for each user are frozen, and the remaining items are reordered. The user's own cluster is disregarded during re-ranking, and items already included in the recommendation list are excluded. Items not in the intersection with the centroid recommendations remain relevant. Finally, the remaining items are concatenated with the initial recommendations and sorted in descending order of probability.

While the GNN-based recommendation step generates accurate and personalized predictions, its output often prioritizes popular items or items highly relevant to a user's immediate preferences. This can exacerbate problems such as *filter bubbles*, where users are repeatedly exposed to similar content, and the *Matthew effect*, which increases the visibility of already popular items at the expense of less prominent ones. To address these limitations, our framework incorporates a re-ranking mechanism designed to balance personalization with diversity, thereby improving the overall recommendation experience.

Clustering

Before the re-ranking process, we want to identify common behavioral patterns using the embeddings generated by the GNN. These embeddings, which encode user preferences in a high-dimensional latent space, are clustered into distinct communities using k-means‖ clustering [21]. This step segments users based on the similarity of their learned preferences, forming groups that show cohesive tendencies in their interaction behavior. The optimal number of clusters is determined by the Silhouette score, a metric that measures the compactness of each cluster relative to its separation from others. A high Silhouette score[223] ensures that the clusters are well defined and meaningful, minimizing overlap and maximizing intra-cluster similarity. Let $Q = \{Q_1, \ldots, Q_K\}$ be the set of detected clusters.

Once clusters are formed, the next step is to identify a representative for each cluster. The user q_i whose embedding is closest to the cluster Q_i center - calculated as the average position of all embeddings within the cluster - is selected as its representative. This user's GNN-generated recommendation list serves as the centroid recommendations, representing the dominant preferences of the group. By using centroid recommendations as a proxy for the shared behavior of the cluster, the system captures group-level preferences while reducing computational complexity.

Re-ranking

To diversify the recommendations for user u, we introduce suggestions from other clusters' centroids. The mixing of personalized and centroid recommendations follows a structured approach:

1. We identify the cluster k of user u, and its centroid recommendation list Y_k.
2. We compute the Euclidean distance between this centroid and all other cluster centroids, identifying the furthest centroids as candidates for re-ranking. Selecting the most distant centroids maximizes the diversity of recommendations
3. The top-K items in the original centroid recommendation list Y_k are frozen, ensuring that the most relevant items for the cluster are retained.
4. From the remaining items, we select those that are common to the tail list of q_k, Y_k, and the recommendation lists of the furthest centroids Y_f. These shared items are re-ordered, prioritizing items from the most distant centroid. The refined list $\hat{Y_k}$ is constructed by appending the re-ranked tail items to the frozen top-K items.

Once the centroid diversified recommendation lists have been refined, we re-rank each user's recommendations. For each user u, the re-ranking process starts with its personalized recommendations generated by GNN:

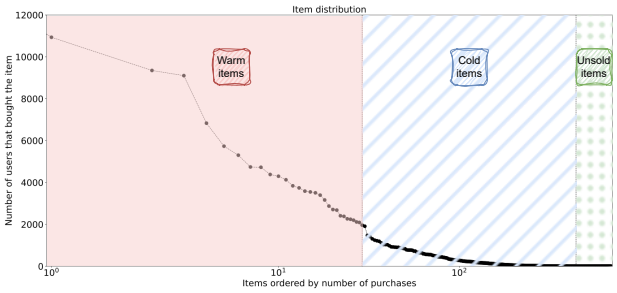
- Top K items: The most relevant items according to GNN are kept unchanged to ensure personalization.
- Remaining items: Items in the user's tail list that overlap with the centroid recommendations $\hat{Y_k}$ are appended in the order of their score.

The resulting recommendation list balances the user's immediate preferences with new, diverse suggestions. Long-tail items are strategically placed at the bottom of the list, ensuring that exploration does not interfere with the relevance of top-ranked items. By carefully balancing personalization and diversity, the re-ranking algorithm prevents over-generalization while effectively mitigating problems such as filter bubbles.

This method is computationally efficient because it pre-computes centroid recommendations for each cluster. During re-ranking, the system only needs to compare a user's list to the relevant centroid lists, reducing the complexity from processing individual users to processing a smaller number of clusters. This not only improves the user experience, but also meets the scalability requirements of the system, making it suitable for large-scale industrial applications.

3.2.4 Experimental Setup

This section describes the dataset, baseline methods, evaluation metrics, and experimental configurations used to validate our framework. Our experiments are designed to evaluate the framework's performance in addressing cold-start challenges and improving diversity in an industrial scenario.



Dataset	Industrial			
	# tot	# warm	# cold	# extremely cold
items	693	29	412	252
users	1080681	13622	169505	897554

Table 3.5: Internal dataset distribution. The figure on the left shows the warm, cold, and unsold items ordered by the number of purchases. The corresponding table and statistical summary is presented on the right.

Industrial Dataset

The primary dataset used in this study is provided by TIM S.p.A., an Italian telecommunications company. This dataset represents real-world interaction data, where users correspond to customers and items correspond to digital services. The dataset is characterized by extreme sparsity:

- Only 17% of users have recorded interactions.
- Of the users who have interacted, 93% have only interacted with a single item.
- More than 40% of the items have no interactions, posing a significant challenge for cold start.

This sparsity highlights the inherent difficulty of generating effective recommendations, especially for cold-start users and items. In addition, the dataset exhibits strong class imbalance, with a small fraction of highly popular items dominating the interaction graph.

To better analyze and address these challenges, we deviate from the traditional Pareto-based [15] approach of splitting items into warm and cold categories. Instead, we define categories for both users and items based on purchase behavior:

- Warm users: Users who have purchased more than one item.
- Cold users: Users with only a single recorded purchase.
- Extremely cold users: Users with no recorded purchases.
- Warm items: Items that represent more than 1% of the total stock.
- Cold items: Items that represent less than 1% of the total stock.
- Unsold items: Items that have never been purchased.

These categories provide a more granular view of the dataset's structure, as summarized in Table 3.5, and better reflect the unique challenges posed by extremely cold users and unsold items. The distribution of items across these categories is illustrated in Section 3.2.4, highlighting the dominance of unsold and cold items in the overall catalog.

We report the statistical properties of the dataset in Table 3.6, which reveals two key challenges that characterize our problem setting

First, on the user side, the data confirms an extreme cold-start scenario. The median of zero interactions indicates that the majority of the users have never made a purchase. Furthermore, the very low average of 0.19 interactions per user underscores the general inactivity across the customer base

Second, the item distribution exhibits a pronounced long-tail. The high difference between the high mean (678.37) and the low median (21) for item interactions reveals a heavy concentration of interactions among a small number of popular (warm) items. For instance, one item has been purchased over $11,000$ times, which significantly pushes the average up. The large standard deviation ($2,289.40$) further confirms the extreme variance in item popularity

Table 3.6: Statistical properties of the interaction data. The distributions for both users and items exhibit characteristics typical of an extreme cold-start and long-tail environment.

Interaction Distribution	**Users**	**Items**
Mean	0.19	678.37
Median	0	21
Standard Deviation	0.62	2,289.40
Minimum	0	0
Maximum	18	11,058

Baselines

To evaluate the performance of our framework, we compare it to a set of baselines designed to provide insight into how our framework improves accuracy, diversity, and overall recommendation quality.

Original GNN Model

This baseline represents the core GNN model without any re-ranking applied to its output. The GNN is trained on the augmented graph, but directly outputs recommendations based on the predicted interaction probabilities between user and item embeddings. This allows us to evaluate the raw performance of the GNN without additional diversity-enhancing mechanisms.

Random shuffling baseline

In this baseline, the recommendations generated by GNN are randomly shuffled before being presented to the user. This approach eliminates any inherent ranking produced by GNN and provides a naive comparison point to evaluate the importance of structured ranking mechanisms, especially for personalization and accuracy.

Popularity Shuffling Baseline

This baseline shuffles the recommendations generated by GNN based on item popularity. All items are ranked by their overall frequency of interaction across all users, and the GNN recommendations are reordered accordingly. This tests the framework's ability to outperform popularity-based ranking systems, which are commonly used in real-world applications but often lead to an overemphasis on popular items at the expense of diversity.

By comparing our framework to these baselines, we can isolate the impact of re-ranking strategies in balancing personalization and diversity, and demonstrate how they mitigate problems such as filter bubbles and the dominance of popular items. Each baseline is implemented using the same experimental configurations as our framework to ensure a fair comparison.

Evaluation Metrics

To comprehensively evaluate our framework, we employ a set of metrics that measure both accuracy and diversity of recommendations. These metrics capture the dual goals of providing personalized and relevant suggestions while promoting diversity and fairness, particularly in the context of cold start challenges.

Intra-List Distance (ILD)

Intra-List Distance measures the diversity of items within a single user's recommendation list by calculating the average pairwise distance between item embeddings. The embeddings, derived from the graph, represent the structural and semantic relationships between items. Formally, the ILD for a user u is defined as:

$$ILD(R_u) = \frac{1}{|R_u|(|R_u|-1)} \sum_{i \in R_u} \sum_{j \in R_u \setminus i} d(i,j) \tag{3.10}$$

where $d(i,j)$ is the Euclidean distance between the embeddings of the elements i and j, and $|R_u|$ denotes the number of items in the recommendation list R_u. Higher ILD values indicate greater diversity within the recommendation list, an important factor in mitigating filter bubbles.

Coverage@K

Coverage@K measures the breadth of recommendations across the entire item catalog by measuring the proportion of unique items that appear in the top K recommendations for all users. It is calculated as:

$$Coverage@K = \frac{|\cup_{u \in U} Y_K(u)|}{|R|} \tag{3.11}$$

where $Y_K(u) = [i_1, ..., i_k]$ are the top K recommendations for user u, and R is the set of all available items. High Coverage@K values indicate that the system is exposing a larger portion of the catalog, which is particularly important for addressing long-tail recommendations and ensuring fairness in item visibility.

Type-Coverage@K

To further explore the diversity of recommendations, Type-Coverage@K measures the coverage of items belonging to specific categories, such as warm, cold, and unsold items. This metric quantifies the representation of different item types within the top K recommendations, defined as:

$$Type - Coverage@K = \frac{|\cup_{u \in U} Y_K^{Type}(u)|}{|R^{Type}|} \tag{3.12}$$

where $Y_K^{Type}(u)$ is the subset of top-K recommendations for user u that belong to the specified item type, and R^{Type} is the total number of items in this category. By distinguishing between warm, cold, and unsold items, this metric highlights the system's ability to promote underrepresented or long-tail content, a critical aspect of fairness and diversity in recommendations.

Experimental Configuration

Our experiments are designed to evaluate the performance of the framework by systematically analyzing the impact of key parameters involved in the graph augmentation and embedding generation processes. In contrast to our previous work[230] that has focused extensively on re-ranking hyperparameters, this study emphasizes the parameters that influence the quality of dense graph construction.

Technical setup

All experiments are implemented in Apache Spark and Python. The internal dataset is preprocessed and converted to a graph using Pyspark, leveraging its distributed high-efficiency capabilities. Pytorch is used for GNN training, XGBoost library for propensity modeling, and Annoy for approximate nearest neighbor search. Finally, the re-ranking algorithm is also performed in Spark.

Hyperparameters studied

To isolate the impact of graph augmentation on the overall performance of the framework, we vary three key parameters:

- Number of PCA components: We vary the number of PCA components from 100 to 300 to evaluate how the retained variance affects graph augmentation and GNN performance. Lower dimensionality reduces computational cost but risks discarding relevant behavioral patterns, while higher dimensionality may lead to redundancy or noise.
- Number of SHAP values selected: We experiment with selecting 5, 10 or 20 features. A smaller set of features may oversimplify user behavior, while a larger set captures more nuanced patterns.
- Number of neighbors in link augmentation: The number of nearest neighbors used in the KNN link augmentation step is varied between 5 and 20. This parameter determines the density of the augmented graph, which greatly affects the computation efficiency. Smaller k values result in sparser graphs, but may also limit the ability of the GNN to propagate information effectively.

By focusing on the parameters that influence graph augmentation, this study complements our previous work on re-ranking hyperparameters and provides a more complete evaluation of the framework. The results provide valuable insights into how different design choices in the early stages of the pipeline affect the overall recommendation performance.

3.2.5 Results

This section presents the experimental results, focusing on the performance of our framework in terms of ILD@K, Coverage@K, and Type-Coverage@K. The results are evaluated over different configurations of the framework hyperparameters. We also compare the performance of the re-ranking approaches with the baseline methods.

Diversity performance

Table 3.7: ILD results of GNN, Random Shifted GNN (RS-GNN), Popularity Shifted GNN (PS-GNN), and re-ranking algorithm (RR), with k Frozen items (F@k). Experiments are repeated five times with a different weight initialization; mean and variance are reported. **Bold** indicates the best result, while our metrics surpassing the baselines are underlined. * denotes statistically significant ($p<0.01$) improvements over the top-performing baseline. Hyperparameter configurations are shown in superscript.

	ILD@K			
Method	K=10	K=20	K=50	K=100
GNN	$11.3144 \pm 3.2548^{(100,5,10)}$	$12.7945 \pm 3.7697^{(100,20,20)}$	$14.3737 \pm 2.4674^{(100,5,10)}$	$14.6192 \pm 2.5075^{(100,5,10)}$
RS-GNN	$5.2508 \pm 1.5513^{(100,5,10)}$	$1.2436 \pm 0.3674^{(100,5,10)}$	$0.1928 \pm 0.0569^{(100,5,10)}$	$6.3802^{(150,5,10)}$
PS-GNN	$13.7762^{(100,5,10)}$	$10.8841^{(100,5,10)}$	$9.5564^{(100,5,10)}$	$14.3174^{(150,5,10)}$
RR-F@5	$\mathbf{34.1190 \pm 4.8456}^{(100,5,10)*}$	$\mathbf{31.2307 \pm 1.8181}^{(100,5,10)*}$	$\mathbf{26.8501 \pm 0.9069}^{(100,5,10)*}$	$\mathbf{25.4076 \pm 0.4595}^{(100,5,10)*}$
RR-F@10	-	$\underline{28.0319} \pm \underline{3.8241}^{(300,10,20)*}$	$\underline{23.9072} \pm \underline{2.1670}^{(300,10,20)*}$	$\underline{21.6955} \pm \underline{0.9935}^{(300,10,20)*}$
RR-F@20	-	-	$\underline{22.7998} \pm \underline{0.4614}^{(300,10,10)*}$	$\underline{21.2385} \pm \underline{0.2524}^{(300,10,10)*}$
RR-F@50	-	-	-	$\underline{21.7844} \pm \underline{1.5586}^{(300,10,10)*}$

Table 3.7 reports the ILD@K for different configurations of our framework, expressed in the form (X,Y,Z), i.e. (X PCA components, Y SHAP values, Z neighbors), evaluated across different baselines and re-ranking strategies. Several important trends emerge from the results.

General observations

Re-ranked configurations consistently achieve higher ILD@K values compared to other methods, especially at lower K values. This improvement highlights the ability of the re-ranking step to effectively diversify recommendation lists. Random shuffling produces significantly lower diversity compared to the GNN baseline, while popularity shuffling achieves values comparable to or slightly lower than GNN, showing limited effectiveness in improving diversity.

Optimal configurations

The configuration (100, 5, 10) gives the highest ILD@K across all baselines (GNN, Random and Popularity) and re-ranking strategies for Frozen@5. For Frozen@10, the best configuration shifts to (300, 10, 20). For higher K values (Frozen@20, Frozen@50), (300, 10, 10) emerges as the optimal choice.

Effect of the hyperparameters on the diversity

Increasing the number of PCA components, SHAP values and neighbors in link augmentation generally increases diversity. This trend suggests that retaining more features and creating denser graphs leads to richer embeddings that support the re-ranking step in generating more diverse recommendations.

Coverage performance

Table 3.8: Coverage results of GNN results before and after re-ranking. **Bold** indicates the best result, while second-best is underlined. Hyperparameter configurations are shown in superscript.

Method	Coverage@K					
	K=5	K=10	K=20	K=50	K=100	K=200
GNN	**0.4277**$^{(150,5,3)}$	**0.5317**$^{(150,5,3)}$	**0.6387**$^{(150,5,3)}$	**0.8106**$^{(100,10,3)}$	**0.9407**$^{(100,10,3)}$	**1.0000**$^{(100,20,3)}$
RR-F@5	-	0.4219$^{(100,10,3)}$	0.4306$^{(100,10,3)}$	0.4566$^{(100,10,3)}$	0.4971$^{(300,5,10)}$	0.5852$^{(300,5,10)}$
RR-F@10	-	-	0.5419$^{(150,5,3)}$	0.5592$^{(150,5,3)}$	0.5910$^{(150,5,3)}$	0.6777$^{(150,5,3)}$
RR-F@20	-	-	-	0.5722$^{(100,10,3)}$	0.6083$^{(100,10,3)}$	0.6835$^{(100,10,3)}$
RR-F@50	-	-	-	-	0.7731$^{(100,20,3)}$	0.8121$^{(100,20,3)}$

The table Table 3.8 shows the maximum Coverage@K achieved with different configurations. The GNN baseline achieves higher coverage than Re-Ranking, which comes very close to GNN's coverage in some configurations with higher Frozen@K values.

GNN configuration

GNN achieves the best Coverage@K for lower K-values with the configuration (150, 5, 3), while (100, 10, 3) performs better for higher K values.

Optimal configurations for re-ranking

For Frozen@5, (100, 10, 3) gives the best coverage up to K=20, while (300, 5, 10) becomes optimal at higher K values. For Frozen@10, Frozen@20 and Frozen@50, (150, 5, 3), (100, 10, 3) and (100,20,3) emerge as the best configurations, respectively.

Effect of the hyperparameters on the coverage

Increasing the number of SHAP selected features seems to improve the coverage, especially as Frozen@K increases. This trend suggests that retaining more behavioral patterns helps the framework to identify and include more unique items in the recommendation lists.

Selecting the Optimal Configuration

Table 3.9: The results for all the considered metrics are reported for the optimal configuration. **Bold** indicates the best value, while underline shows the metric we are optimizing. The selected run has parameters (100,20,3)

Method	Type-Coverage@K						Coverage@K		ILD@K
	W@100	C@100	U@100	W@200	C@200	U@200	Cov@100	Cov@200	ILD@100
GNN	0.8965	0.8325	**0.9801**	0.9655	0.9854	**1.0000**	**0.8901**	**0.9913**	6.2024 ± 2.0056
RR-F@50	0.7586	0.6601	0.9563	0.7931	0.7208	0.9603	0.7731	0.8121	$\mathbf{12.7585 \pm 2.1392}$

To identify the best overall configuration, we align the selection with our practical business objective: maximizing the coverage of unsold items while maintaining high diversity. As shown in Table 3.9, the configuration (100, 20, 3) emerges as the optimal choice. With this configuration,

Table 3.10: Summary of Dataset Statistics

	Train	Test
Unique users buying	2,552,477	736,648
Total items bought	4,818,165	1,070,679
% of users with ≥ 1 items bought	18.49%	5.34%
% of users with ≥ 2 items bought	7.34%	1.42%
% of users with ≥ 3 items bought	3.62%	0.53%

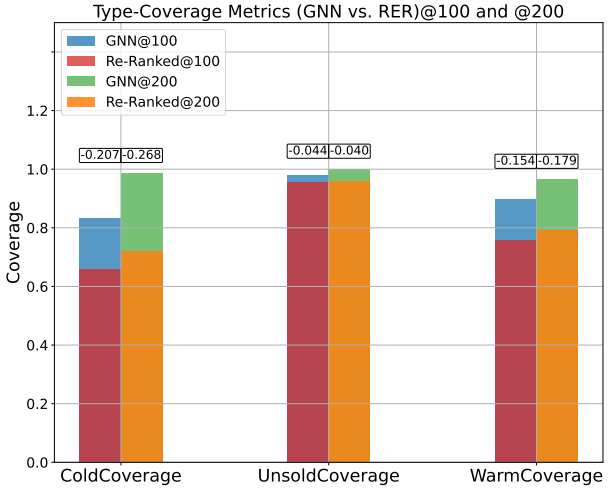


Figure 3.8: GNN and Re-Ranked Type-Coverage obtained in the optimal configuration. Overlapping bars for GNN and Re-ranked are shown, for both K=100 and K=200. The degradation is reported in rectangles at the top.

the Re-Ranking Frozen@50 achieves an Unsold-Coverage@100 of 95%, falling less than 5% short of the GNN's performance while doubling the ILD@100. These results are also presented in Fig. 3.8.

Type-Coverage performance:

The configuration achieves Type-Coverage@200 values of over 70% for warm and cold items, demonstrating its ability to maintain adequate representation for these categories while prioritizing unsold items.

Efficiency and trade-offs:

The configuration is notable for its efficiency, requiring the minimum number of PCA components and neighbors, which reduces computational overhead. Despite the focus on diversity and unsold items, the impact on warm and cold item coverage is moderate and within acceptable limits for industrial applications.

Trade-Offs Between Coverage and Diversity

Fig. 3.9 presents the Coverage@100 and ILD@100 for all tested configurations, distinguishing between models based solely on GNN (blue dots) and those with re-ranking applied afterward (red dots). The results of the two models appear to cluster into two distinct groups: GNN-based models

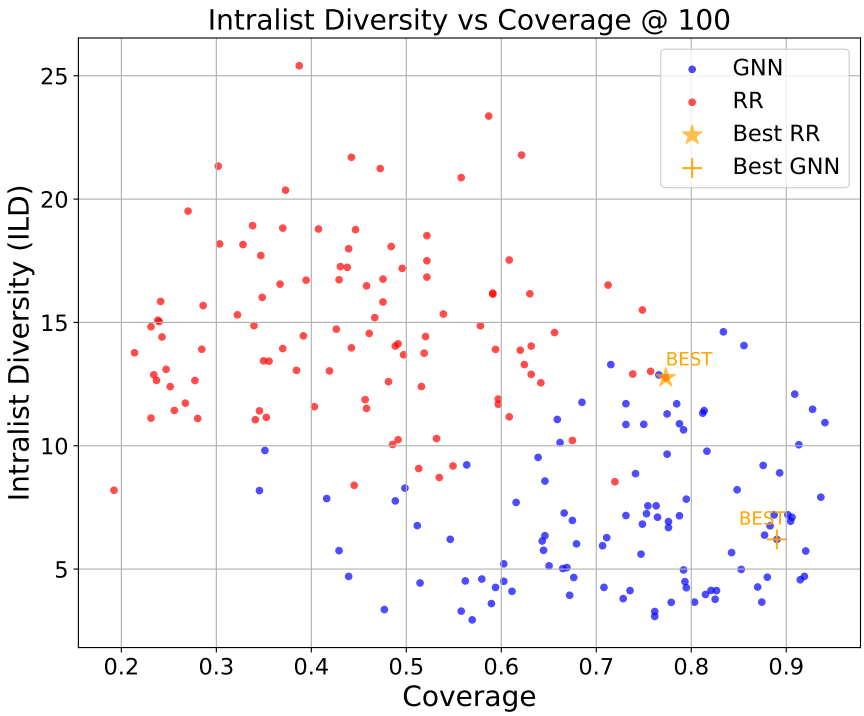


Figure 3.9: Intra-List Diversity vs Coverage @100. Distribution of all GNN-based recommendations and re-ranked results. The 'BEST' re-ranked recommendation and its corresponding GNN are highlighted.

typically achieve higher coverage but lower ILD, while re-ranked models show the opposite trend, indicating greater diversity in the recommendations and lower coverage.

The best models, as selected in Section 3.2.5, are highlighted with yellow markers and labeled "BEST". These configurations are positioned toward the right side of each cluster, indicating higher coverage values, which aligns with our business objective of maximizing Unsold Coverage. This is particularly evident for the best re-ranking configuration, which achieves the highest average coverage across all re-ranking models.

The plot also demonstrates the flexibility of our framework, allowing us to adjust hyperparameters to find the optimal balance between coverage and diversity. For instance, if maximizing diversity were the primary objective, one could select a re-ranking configuration with a higher ILD, reaching up to 25, but this would come at the cost of a substantial drop in coverage, falling below 40%.

3.2.6 Ablation Study

In this section, we conduct a detailed ablation study to systematically assess the contribution of each key component of our framework. Specifically, we investigate four main aspects: the impact of using a different GNN architecture, the role of the graph augmentation step, the effect of each model and re-ranking step on final prediction accuracy, and the influence of choosing the closest or farthest centroids in the re-ranking step

GNN Architecture Variant

To evaluate the generalisability of our framework and its sensitivity to the choice of GNN architecture, we replace GraphSAGE with LightGCN [109]. We integrate our re-ranking module with

LightGCN, keeping the previous optimal hyperparameters. The results in terms of coverage, diversity, and type-coverage are presented in Tables 3.11 to 3.13

LightGCN achieves optimal item exposure, reaching 100% catalog coverage at $K = 50$. While this demonstrates significantly better performance in ensuring items are recommended, it comes at the cost of significantly lower ILD. This aligns with previous findings [310], which show that LightGCN exhibits embeddings collapse, potentially explaining the lower ILD. When the re-ranking module is applied, we observe an enhancement of diversity for all K, as expected. However, the overall gains are notably smaller than those obtained with GraphSAGE

Table 3.11: Coverage results of LightGCN before and after re-ranking.

Method	**Cov@5**	**Cov@10**	**Cov@20**	**Cov@50**	**Cov@100**	**Cov@200**
LightGCN	0.7711	0.9139	0.9948	1.0000	1.0000	1.0000
RR-F@5	-	0.7728	0.7814	0.7917	0.8072	0.8364
RR-F@10	-	-	0.9156	0.9191	0.9259	0.9363
RR-F@20	-	-	-	0.9948	0.9948	0.9965
RR-F@50	-	-	-	-	1.0000	1.0000

Table 3.12: Intra-List Diversity (ILD) results of LightGCN before and after re-ranking.

Method	**ILD@10**	**ILD@20**	**ILD@50**	**ILD@100**
LightGCN	0.2926±0.0274	0.2878±0.0305	0.2898±0.0443	0.3075±0.0570
RR-F@5	0.3094±0.0356	0.3821±0.0246	0.3772±0.0093	0.3833±0.0047
RR-F@10	-	0.3174±0.0361	0.3656±0.0181	0.3762±0.0093
RR-F@20	-	-	0.3255±0.0311	0.3587±0.0182
RR-F@50	-	-	-	0.3175±0.0385

The most critical insight emerges from the type-coverage analysis. Unlike GraphSAGE, re-ranking LightGCN results reduces Unsold and Cold coverage. This suggests that Warm items dominate the top ranks and that the GNN overfits the most popular items in the dataset

Table 3.13: Type-Coverage results for different frozen configurations. W, C, and U stand for Warm, Cold, and Unsold coverage, respectively.

Method	Type-Coverage@100			Type-Coverage@200		
	W@100	C@100	U@100	W@200	C@200	U@200
LightGCN	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
RR-F@5	1.0000	0.8902	0.5555	1.0000	0.9052	0.6274
RR-F@10	1.0000	0.9551	0.8366	1.0000	0.9625	0.8562
RR-F@20	1.0000	1.0000	0.9803	1.0000	1.0000	0.9869
RR-F@50	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000

Fig. 3.10 presents Coverage versus ILD for all configurations, underlying the differences between the re-ranked results (red dots) and the base LightGCN outputs (blue stars). As with GraphSAGE, pure GNN models yield higher coverage and lower ILD, whereas re-ranked variants trade off some coverage for improved diversity. The framework's overall behavior remains consistent, allowing flexible selection of the best configuration depending on task-specific priorities

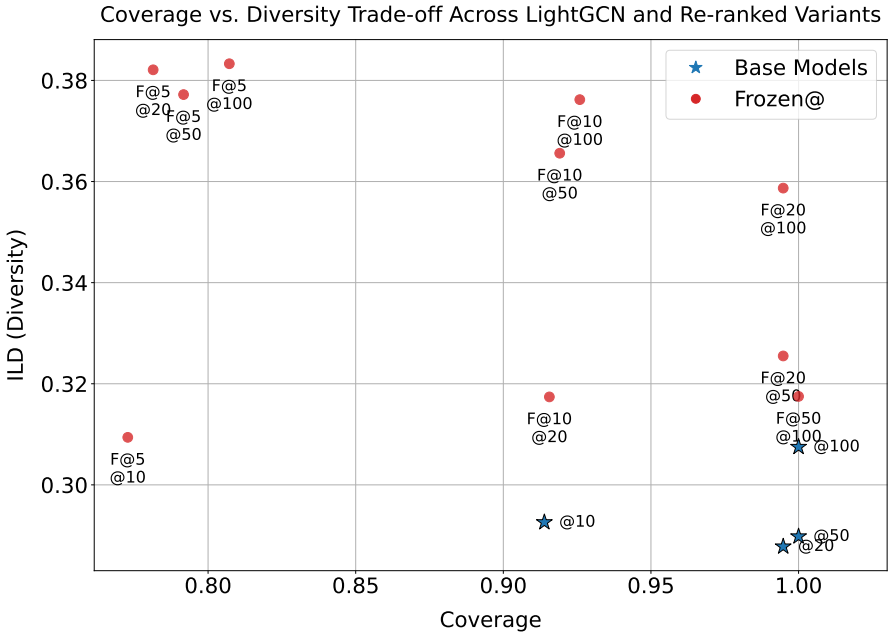


Figure 3.10: LightGCN Intra-List Diversity vs Coverage. Base models (blue stars) refer to the LightGCN metrics, frozen@ (red dots) indicate the re-ranking outcomes.

In conclusion, this analysis reveals that the optimal GNN architecture is task-dependent. While LightGCN is a good option for maximizing accuracy, its properties are not well aligned with our goals of enhancing diversity and ensuring adequate coverage of cold and unsold items

Graph Augmentation

We conduct an ablation study to quantify the impact of the graph augmentation step, specifically, the inclusion of user-user edges. As detailed in Section 3.2.3, these edges are based on user similarity in order to encode collaborative information into the graph structure. While these edges may introduce noise, we hypothesize that they are essential for improving recommendation diversity and performance for users with sparse interaction histories (i.e., cold start users); a key objective of our work

To test this, we compare two graph variants: a baseline bipartite graph, and our proposed augmented graph incorporating user-user edges. We evaluate the relative performance change resulting from the augmentation for both the GraphSAGE baseline and our complete re-ranking framework. The results are presented in Table 3.14

For the GraphSAGE baseline (without re-ranking), the graph augmentation degrades several key metrics. Item Coverage@100 drops by 9.4%, particularly for Unsold (13.3%) and Warm (24.0%) item coverage. Similarly, ranking accuracy, measured by NDCG@100, decreases by -3.0%, suggesting that user-user edges may introduce noise signals that hinder the ranking quality of standard GNNs pipelines. However, this drawback is accompanied by a significant increase ($+37.0\%$) in Intra-List Diversity (ILD@100). This confirms that user-user edges encourage the model to consider more than just a user's interaction history, promoting the exploration of a more diverse set of items from the catalog

Table 3.14: Percentage Improvement w/wo user-user edges, comparing re-ranked results to GraphSAGE baseline.

	GraphSAGE	**Reranked Improvement (%)**			
Metric	**Baseline (%)**	**RR-F@5**	**RR-F@10**	**RR-F@20**	**RR-F@50**
Cold Cov@100	-1.5	+4.8%	+10.0%	-8.3%	-10.6%
Unsold Cov@100	-24.0	-7.9%	0.0%	-17.9%	-11.1%
Warm Cov@100	-13.3	-37.5%	-33.3%	-55.6%	-18.2%
Coverage@100	-9.4	-1.5%	+5.3%	-13.5%	-11.1%
ILD@100	+37.0	+21.8%	+21.7%	+21.3%	+27.6%
NDCG@100	-3.0	-19.2%	-32.7%	-30.1%	-18.7%

The benefits of the graph augmentation become more evident when coupled with our re-ranking module. While re-ranking naturally decreases coverage by focusing on fewer warm items, it mitigates the drop in coverage for cold and unsold items. Crucially, the diversity gains are preserved: the re-ranked outputs consistently yield an improvement in ILD@100 of over +21%. Although NDCG remains lower, this aligns with our design goal of optimizing diversity and cold-start coverage over achieving pure ranking accuracy

Re-ranking Impact on Ranking Accuracy

While the primary objective of our framework is to enhance recommendation diversity, it is essential to quantify the impact of our re-ranking strategy on traditional ranking accuracy. We measure it using the percentage change in NDCG at various cutoffs ($K = 5, 10, 20, 50, 100$), as reported in Table 3.15.

Table 3.15: Percentage change in NDCG@k of re-ranking models.

Method	**NDCG@5**	**NDCG@10**	**NDCG@20**	**NDCG@50**	**NDCG@100**
GraphSAGE-RR-F@5	-	-88.57%	-98.61%	-71.88%	-38.65%
GraphSAGE-RR-F@10	-	-	-89.07%	-81.62%	-65.59%
GraphSAGE-RR-F@20	-	-	-	-5.18%	+141.53%
GraphSAGE-RR-F@50	-	-	-	-	-18.87%
LightGCN-RR-F@5	-	-24.54%	-41.70%	-51.17%	-46.20%
LightGCN-RR-F@10	-	-	-22.70%	-40.73%	-43.03%
LightGCN-RR-F@20	-	-	-	-25.86%	-29.91%
LightGCN-RR-F@50	-	-	-	-	-9.74%

As expected, the introduction of a re-ranking step optimized for diversity leads to a decrease in NDCG. This is a direct and anticipated consequence of our methodology: the re-ranking step demotes highly relevant items from top positions to accomodate more diverse and novel candidates. This effect behaves differently across the two GNN architectures. In GraphSAGE, increasing the evaluation cutoff (e.g., moving from NDCG@5 to NDCG@100) results in a smaller relative drop in NDCG. In contrast, LightGCN exhibits the opposite trend: the drop in NDCG becomes more pronounced at larger cutoffs, particularly when re-ranking a small initial set of items (e.g., F@5 or F@10). This behavior may be due to LightGCN's consistently strong performance. Since the re-

ranking step may introduce candidates that deviate substantially from the most relevant ones, it can disproportionately affect NDCG at higher K values

However, the magnitude of this accuracy degradation is not uniform and provides valuable insights. Notably, LightGCN is more robust to re-ranking: even when reranking the top 50 items (F@50), the drop in NDCG@100 is relatively modest (-9.74%), indicating that LightGCN provides a stronger set of initial candidates and allows the re-ranker to balance relevance and diversity more effectively

In summary, the observed drop in ranking accuracy reflects an intentional trade-off built into our design. This trade-off can be controlled by selecting the appropriate GNN backbone and re-ranking depth, enabling practitioners to adapt the framework to their application's specific needs

Impact of Farthest Clusters

Our re-ranking step selects other clusters based on their embeddings in the latent space. A crucial aspect of this strategy is prioritizing the most distant centroids from the selected one under the hypothesis that they will produce a more diverse set of item candidates. To validate this approach, we compare re-ranking based on the farthest versus the closest centroids. All experiments are performed using the main GraphSAGE model and are based on the best-performing hyperparameter configuration presented in the main results

Table 3.16 shows the percentage change in coverage when re-ranking is performed by selecting nodes from the closest cluster centroids. Results reveal a consistent decline in coverage across all cutoffs, confirming that close centroids provide less varied item recommendations

Table 3.16: Percentage coverage improvements from the re-ranking step using closest centroids. The re-ranking is based on node embeddings generated by the GraphSAGE model.

Method	Cov@5	Cov@10	Cov@20	Cov@50	Cov@100	Cov@200
RR-F@5	-	-1.54%	-11.77%	-20.63%	-27.56%	-15.87%
RR-F@10	-	-	-1.90%	-13.50%	-26.29%	-15.62%
RR-F@20	-	-	-	-0.60%	-17.32%	-15.40%
RR-F@50	-	-	-	-	-0.73%	-12.02%

Table 3.17 shows a similar pattern. While some configurations (e.g. RR-F@50) show marginal improvements in certain item categories, the overall pattern indicates substantial drops in all type-coverages

Table 3.17: Type-Coverage results for different frozen configurations. W, C, and U stand for Warm, Cold, and Unsold coverage, respectively.

Method	Type-Coverage@100			Type-Coverage@200		
	W@100	C@100	U@100	W@200	C@200	U@200
GraphSAGE-RR-F@5	-33.33%	-37.04%	-7.50%	-29.41%	-18.18%	-7.62%
GraphSAGE-RR-F@10	-33.33%	-36.25%	-5.06%	-29.41%	-17.82%	-7.62%
GraphSAGE-RR-F@20	-23.08%	-26.54%	+2.53%	-29.41%	-17.52%	-7.62%
GraphSAGE-RR-F@50	-8.33%	+2.19%	-1.28%	-20.00%	+12.58%	+9.01%

The impact of this strategy is also reflected in the ILD results in Table 3.18. When re-ranking

using the closest centroids, ILD scores drop consistently across all cutoffs. In contrast, using the farthest cluster centroids for re-ranking improves all key metrics. This is partially to be expected, given that ILD is calculated precisely in the latent space. By construction, items in clusters distant from those in the ranking have distant embeddings, so adding them can only increase diversity

Table 3.18: Intra-List Diversity results using farthest or closest centroids in the re-ranking step.

Method	ILD@10	ILD@20	ILD@50	ILD@100
GraphSAGE-RR-F@5	-8.95%	-3.11%	-8.77%	-4.01%
GraphSAGE-RR-F@10	-	-4.32%	-5.39%	-4.37%
GraphSAGE-RR-F@20	-	-	-1.33%	-4.30%
GraphSAGE-RR-F@50	-	-	-	-0.69%

This difference highlights the critical role of selecting the appropriate cluster centroid strategy in the re-ranking step. Our findings strongly suggest that prioritizing nodes from clusters farthest from the query embedding is a more effective approach for enhancing recommendation quality in this context

3.2.7 Conclusions

This paper presented an end-to-end framework that effectively addresses extreme cold-start scenarios by transforming a sparse user-item interaction graph into a dense graph and using GNNs to generate recommendations. The framework also includes a re-ranking algorithm designed to enhance the diversity of recommendations.

We conducted a thorough investigation of the hyperparameters within the framework, focusing on aspects that were not explored in our previous study. Specifically, we examined the number of Principal Components used for compressing input data, which improved performance efficiency, fundamental in large-scale industrial scenarios. We also analyzed the role of Shapley value-based feature selection in determining the most influential features for similarity computation. Furthermore, we evaluated the effect of the number of neighbors K used in link augmentation, identifying its impact on graph density and subsequent GNN performance. These explorations allowed us to determine the optimal configurations for creating a dense graph and achieving diverse recommendations.

Moreover, our extensive ablation study validated the key design choices of our framework and underlined the critical trade-offs involved in optimizing for recommendation diversity. We demonstrated that the GNN architecture is task-dependent, with GraphSAGE providing a more suitable foundation for our diversity-enhancing goals than the accuracy-oriented LightGCN. Furthermore, we established that augmenting the graph with user-user edges, although it decreases accuracy, is a necessary prerequisite for injecting the item diversity that our re-ranking step leverages. Crucially, our analysis provided evidence for our core re-ranking mechanism: choosing the farthest centroid proved to have superior performance in all considered metrics, as the alternative, closest one, consistently degraded coverage, diversity, and type-coverage. Together, these findings underscore that each component in our framework can be chosen to balance accuracy, diversity, and coverage performance

A central focus of our study is the relationship between hyperparameter configurations and key metrics such as Intra-List Diversity and Coverage. By evaluating these relationships, we demonstrated how parameter settings can be optimized to achieve specific goals, whether maximizing cov-

erage for unsold items or improving diversity for long-tail recommendations. By aligning these findings with the company's business priorities, we demonstrated how to choose parameter combinations that achieve high unsold item coverage without compromising diversity.

Our findings provide a detailed guide for practitioners seeking to adopt this framework. By offering insights into parameter tuning and its impact on performance, diversity, and coverage, we have enabled users to make informed decisions tailored to their specific needs. This comprehensive analysis ensures that the framework can be effectively adapted to various use cases, serving as a valuable resource for addressing the challenges of extreme cold-start scenarios while balancing organizational priorities.

3.2.8 Hyperparameter Optimization

To identify the optimal settings for both the propensity and GNN model, we employed an automated hyperparameter optimization process. We utilized Optuna [7] to efficiently explore the hyperparameter space

For the XGBoost propensity model, the primary goal was to maximize the F1-score on the validation set, as previously explained in Section 3.2.3. GNNs objective was to minimize the validation loss. The following results are obtained using the best parameters obtained in the models' optimization search spaces, as reported in Table 3.19

Table 3.19: Optimal Hyperparameter Configurations

Model	Hyperparameter	Search Space	Optimal Value
XGBoost	Learning Rate	{0.1, 1e-4}	7e-2
	Max Tree Depth	{3,8}	5
	L2 Regularization	{0,10}	0.4
	L1 Regularization	{0,10}	0.2
GraphSAGE	Neighbors	{3, 10}	8
	Mini-batch Size	{128, 2048}	1155
	Negative Sampling Ratio	{1.0, 3.0}	2.0
	Embedding Dimensionality	{8, 128}	44
	Dropout Rate	[0.1, 0.2, 0.3]	0.3
	Learning Rate	{1e-6, 1e-2}	8e-6
	Number of Layers	{1, 3}	2
	Loss Negative Weight	{1, 5}	1
LightGCN	Learning Rate	{1e-6, 1-e2}	3e-2
	Num Layers	{1, 3}	2
	Embedding Dimensionality	{8, 128}	36

Chapter 4

Proactive Churn Reduction via Reinforcement Learning

The optimization objective of most recommender system is focused on immediate, proxy metrics such as click-through rates or offer acceptances. In telecommunication industries, this approach is insufficient. The most critical long-term outcome, *customer churn*, is often the result of a sequence of interactions, and its feedback is significantly delayed. An offer that seems successful today may contribute to a customer's decision to leave months later, a connection invisible to models that do not account for such long-term consequences. This mismatch between short-term optimization and long-term business goals can lead to strategies that, while seemingly effective, ultimately harm customer retention.

Addressing this challenge requires not only a shift in methodology but also access to data that captures these temporal dynamics. The dataset introduced and analyzed in the previous chapter provides the necessary foundation for this work, containing the granular, longitudinal records of user interactions, explicit feedback, and eventual churn outcomes that are essential for training and validating a long-term-oriented model.

This chapter moves from myopic prediction tasks to long-term sequential decision-making. We introduce a framework designed for this purpose: CRAB (**C**hurn **R**eduction **A**gent-**B**ased). We leverage offline Reinforcement Learning (RL) to train an agent on this historical data, optimizing its policy not for immediate offer acceptance, but for the long-term goal of maximizing cumulative rewards and minimizing customer churn. A core contribution of this work is a practical methodology for tackling this complex problem. We detail how to engineer a reward structure that explicitly handles the delayed churn signals present in the data. Furthermore, we present a novel and highly scalable technique for enriching the agent's state representation by incorporating the predictive outputs of pre-existing, independently trained business models, providing the agent with a more holistic view of the customer without the complexity of joint training.

The resulting framework is rigorously evaluated, demonstrating superior performance in extensive offline experiments against the baselines benchmarked. More significantly, its value is confirmed through live A/B testing in a production environment, providing a robust, data-driven solution to one of the industry's most pressing industry challenges.

4.1 CRAB: Churn Reduction Agent Based via Reinforcement Learning

The application of offline reinforcement learning (RL) in recommendation systems has proven valuable in improving user engagement. However, a significant challenge in real-world scenarios is the delayed nature of user feedback, which can degrade algorithm performance and complicate the task of reducing user churn (i.e. losing clients) to competitors. In this paper, we present an end-to-end framework called Churn Reduction Agent-Based (CRAB), which is specifically designed to efficiently incorporate delayed feedback into the recommendation process. CRAB uses a conservative Q-learning model to recommend offers to customers, incorporating delayed churn feedback directly into the reward function of the RL model. This allows the RL agent to optimize recommendations not only based on immediate acceptance probabilities, but also considering long-term user retention. Our approach also uses additional auxiliary model predictions to improve the representation of customer states in the RL environment, which significantly improves the framework's performance. We validate our method using large-scale proprietary data from a real-world telecommunication company, addressing the complexities of scalability and practical implementation. Extensive offline experiments show that CRAB outperforms existing methods, including both internal company policies and state-of-the-art approaches. In addition, our online testing results confirm the framework's effectiveness in delivering superior recommendations and effectively reducing user churn by 4.95%.

4.1.1 Introduction

Recommendation systems have become an integral part of many major companies, including Google [50, 59], Facebook [192], Amazon [241], and Netflix [94]. They are used in a wide range of applications, such as e-commerce [232], entertainment [48, 157], and news [136], where they help tailor content to user preferences.

Unlike e-commerce or social media platforms, where the primary goal may be to increase engagement or sales, telecommunications companies prioritize reducing customer churn rather than actively pursuing upselling opportunities [116]. In the telecommunications industry, customer churn is a significant challenge [148]. It can be caused by dissatisfaction, increased costs, inadequate quality, missing features, and privacy issues [238]. In this context, churn refers to the phenomenon of customers switching from one service provider to another [123], resulting in substantial financial losses. In online systems, due to the data collection ingestion time, a substantial amount of user feedback may not have occurred, leading to their absence in the training data. The importance of addressing this delayed feedback is particularly pronounced in this sector, complicating the process of developing effective customer retention strategies.

Applying Reinforcement Learning to Recommender Systems (RLRS) has attracted considerable interest due to its ability to adapt to users' behaviour over time. Unlike traditional recommendation methods that may focus only on immediate interactions, RLRS can optimize the long-term user engagement by learning from sequential user-system interactions [4, 228]. This makes RLRS particularly suited to environments where user preferences are dynamic and complex, as it can provide personalized and timely recommendations.

Offline reinforcement learning has become popular in recent years [62, 158], as it allows models to be trained using historical data without the need for live interaction, making it practical for

industries where real-time user feedback is either unavailable or impractical. Despite its potential, offline RL presents its own set of challenges, such as overfitting and distributional shifts due to the static nature of the training data [145]. In addition, a significant issue in RLRS is the handling of delayed feedback [307], where there is a delay between the action taken by the system and the resulting user feedback. This delay can introduce bias and make it difficult to estimate the true value of an action.

In light of this, our primary goal is to reduce customer churn, ensuring high, consistent performance in recommendations simultaneously. In this paper, we present a novel approach to recommendation systems in the telecommunications industry, with a focus on preventing churn. We propose an end-to-end framework called **C**hurn **R**eduction **A**gent-**B**ased (CRAB), which uses offline RL to optimize the recommendation of offers to customers. By incorporating a carefully designed reward function that penalizes churn events more severely than offer refusals, our model aims to generate personalized recommendations that not only appeal to the customer, but also mitigate the risk of churn. Our method also includes the integration of additional customer state representations derived from various company models to improve the accuracy and relevance of the recommendations.

To validate the effectiveness of our approach, we conduct extensive offline and online experiments on real data from millions of users in a telecommunications company.

Our results show that CRAB outperforms existing company policy and state-of-the-art methods, and offers a promising solution for reducing customer churn (-4.95%). Moreover, we provide diverse recommendations, achieving over 50% Coverage@5.

The main contributions of this work are as follows:

- We introduce a new perspective on recommendation systems by prioritizing customer churn reduction, providing a novel application of offline RL in a critical industry context.
- We propose and validate a practical methodology for enriching RL state representations by incorporating auxiliary predictions from pre-existing, independently trained business models. This is a powerful and scalable alternative to complex joint-training, with significant implications for industrial RL applications.
- Our method is rigorously evaluated through offline experiments and an online A/B test within a major telecommunications company. This achieved a 4.95% reduction in customer churn and outperformed competitive baselines. To the best of our knowledge, this is one of the first studies to demonstrate measurable success in customer upselling and reduction in long-term churn events, providing valuable insights for real-world applications.

The rest of the paper is organized as follows: Section 2 reviews related work, Section 3 outlines the theoretical background, Section 4 details the methods and the model, and Section 5 presents the dataset, baselines, and evaluation metrics, and the choice of hyperparameters. Finally, Section 6 concludes the paper and discusses possible future work. Section 4.1.7 details the hyperparameter tuning search space for all baselines, and the best values that we found.

4.1.2 Related work

Offline RLRS

RL has become a popular approach for RLRS because it can model the sequential nature of user interactions [54, 283]. Unlike simpler models that only optimize for immediate feedback, RLRS aim to learn a recommendation policy π that maximizes long-term user engagement and satisfaction [256, 309]. RLRS can be divided into three different categories. The first one is *on-policy* RL methods that require the learning agent to interact directly with live users to collect data for policy updates. This is often not practical for large systems because it is expensive and can lead to a poor user experience [229].

The second one is *off-policy* RL methods [107], which buffer all their data in a replay buffer and then replay it from there. However, classic off-policy RL methods require additional online collection, as after we update the policy, we usually use it to obtain additional data [320]. The third is *offline RL* [169], where the agent learns from a fixed dataset of pre-collected experiences without collecting additional data during training [147, 255].

However, offline RL faces the fundamental challenge of distributional shift [158]. Since the dataset is generated by a fixed behaviour policy, it does not cover all possible actions [299]. Consequently, standard off-policy algorithms can wrongly assign high values to out-of-distribution (OOD) actions [85, 185], creating policies that appear successful during training, but fail upon deployment [146]. Several approaches have been proposed to mitigate this issue. One approach involves constraining the learned policy to stay close to the policy that generated the data, thereby limiting divergence [208, 216]. While effective, these constraints may be too limiting and may prevent the discovery of more effective strategies. An alternative approach is to change the learning objective to be cautious about OOD actions. This is the main idea behind Conservative Q-learning (CQL) [147, 239]. CQL uniformly pushes down all Q-values to guarantee a lower bound on performance.

Building on CQL, several variants have been developed to address specific limitations. Calibrated Q-Learning (Cal-QL) enables efficient online fine-tuning after offline pre-training [191]. Cal-QL learns conservative lower bounds on the learned policy's true value and reasonable upper bounds on reference policies such as the behavior policy. This calibration prevents the "unlearning" phenomenon where offline-trained policies deteriorate during online fine-tuning, enabling rapid improvement with minimal additional interaction.

Exclusively Penalized Q-Learning (EPQ) tackles the problem of unnecessary conservatism in well-covered regions of the state-action space [294]. EPQ introduces an adaptive threshold mechanism that selectively applies penalties only to state-action pairs that are genuinely prone to overestimation, thereby avoiding the excessive underestimation that can occur in dense data regions. This selective approach maintains the safety benefits of conservative estimation while reducing bias in areas where the behavioral data provides adequate coverage.

Counterfactual Conservative Q-Learning (CFCQL) adapts CQL principles to multi-agent settings by computing conservative regularization for each agent separately in a counterfactual manner [236]. This approach maintains the theoretical guarantees of CQL while ensuring that the induced regularization bounds remain independent of the number of agents, making it particularly suitable for large-scale multi-agent systems.

State-corrected algorithms, such as SCAS [185], go beyond penalizing actions and also address,

offering more holistic robustness.

Value Penalized Q-Learning (VPQ) [89] is an offline RL algorithm that addresses value overestimation differently. It captures uncertainty via an ensemble of Q-functions and uses it to penalize OOD actions. A key advantage of VPQ is that it obviates the need for explicit behavior policy estimation, a challenging task in recommender systems, thereby improving estimates of long-term rewards [303].

In light of this, we selected Conservative Q-Learning (CQL) as the foundation for our proposed method. Our primary objective is to develop a robust and reliable policy learned from a fixed, static dataset. CQL's core mechanism of penalizing OOD actions provides a safe and effective foundation against value overestimation, making it a suitable and strong choice for our high-stakes industrial application, where unreliable recommendations can directly lead to customer churn.

Delayed Feedbacks

A fundamental challenge in applying RL to real-world recommender systems is the significant delay between issuing a recommendation and observing its true outcome [133]. While immediate signals, such as clicks or ratings, are readily available, critical long-term outcomes, such as user retention, subscription renewal, or churn, may only become evident weeks or months later. This temporal gap complicates credit assignment, as the RL agent must learn to associate a delayed reward to the correct sequence of prior actions that caused it. This challenge is further exacerbated in the offline RL setting, where the agent cannot perform online exploration and must infer delayed outcomes only from historical data where such connections are not explicit [17, 102].

In recommender systems, the real reward depends on the user's interaction with the recommended item [113], an event that may require weeks to be determined.

Several strategies have been proposed to mitigate the effects of delayed feedbacks. [184] exploit the intermediate reward signals to determine future delayed feedback using a factored learning approach. [122] solve delayed reward tasks by reassigning the rewards that appear after a causative event. To do this, they employ a memory-based agent and utilize the memory retrieval system to attribute credit to events from the distant past appropriately. These events are valuable for predicting the value function at the current timestep. [17] present the RUDDER algorithm that adopts a backward-view approach for a task, creating a return-equivalent Markov Decision Process where delayed rewards are more evenly redistributed across time. More recently, [102] focus on establishing a sample-efficient off-policy RL algorithm that is adaptive to delayed environment signals.

Memory-augmented models also play a decisive role in tackling this challenge. Architectures incorporating explicit memory retrieval modules have shown strong performance in linking key past events to delayed rewards [29], facilitating more effective value estimation.

At the architectural level, deep RL methods (e.g., DDPG, SAC, TD3) have been extended to support long-horizon or delayed feedback scenarios. These adaptations often involve state augmentation [154], attention mechanisms, and sophisticated sequence modeling [49] or counterfactual reasoning [20], further enhancing robustness and interpretability [86].

While these approaches offer sophisticated mechanisms, they often assume complex temporal dependencies or require substantial architectural modifications. In the context of telco churn, the problem is both simpler and more definitive: churn is a binary, terminal event occurring after a

certain delay. To address this directly, we propose a more practical methodology. By structuring each episode with a fixed look-ahead window (see Section 4.1.3) and designing a reward function that heavily penalizes this terminal churn event (see Section 4.1.3), we align the agent's learning objective with the long-term business goal of churn reduction in a simple yet effective way, avoiding the need for complex model modifications.

Auxiliary Tasks

In deep RL, an agent's success depends on the quality of its internal representations, yet in environments characterized by high-dimensional observations and sparse rewards, learning these representations from the primary objective alone often proves inefficient. Auxiliary tasks, secondary objectives optimized alongside the main RL objective, have been introduced to provide richer, more frequent learning signals, accelerating both representation learning and policy optimization [81, 125, 180].

These extra tasks provide additional signals that help the agent develop improved representations and policies, particularly in situations where data is sparse or rewards are delayed.

Auxiliary tasks fall into several paradigms. One category is predictive learning, in which the agent learns to predict future aspects of its environment, such as next state, immediate reward, or latent dynamics. This paradigm is central to model-based RL methods, such as Dreamer [44, 100]. By learning to predict what will happen next, it integrates reconstruction and prediction losses within a latent world model, allowing the agent to predict trajectories before acting in the real environment. This driven approach dramatically improves sample efficiency by guiding exploration toward the most informative transitions.

Another key paradigm is self-supervised learning, where training signals are generated from the input data itself, without requiring external labels. Contrastive learning approaches like CURL [153] train the agent to produce similar representations for different augmented views of the same state, promoting invariance to task-irrelevant features. These techniques have been shown to improve sample efficiency in vision-based RL [76]. Other self-supervised tasks include observation reconstruction, where an autoencoder architecture is used to reconstruct observations, and inverse dynamics modeling, where the agent learns to predict the action taken between two consecutive states [205].

Auxiliary tasks can also be framed as control objectives. Diversity-maximization methods like DIAYN [75] train the agent to learn a set of skills by maximizing the mutual information between a latent skill variable and the resulting state visitation distribution. This promotes broad exploration and the acquisition of disentangled features that can benefit the main RL objective. This encourages the exploration of the state space.

In recommender systems, auxiliary tasks are typically designed to address the unique challenges of user modeling. One application is to stabilize the learning process and handle stochasticity. For instance, the Stochastic Reward Stabilization (SRS) framework [38] replaces the unreliable environment's stochastic user feedback (e.g. clicks, or likes) with a learned reward expectation, improving training stability and convergence. Similarly, augmenting a recommendation agent with a click-prediction head forces the agent to encode short-term user preferences, accelerating learning when real rewards are sparse.

Another prevalent strategy is to model immediate user feedback to create denser training signals. Chen et al. [51] augment a model-free RL agent with a task that predicts immediate user reactions (e.g., click or no-click), encouraging representations aligned with the user's short-term reaction.

In the context of news recommendation, models such as DRN [314] treat different types of user feedback (e.g., clicks, shares, read time) as separate tasks within a multi-task learning framework. This encourages the agent to learn a more holistic representation of user engagement.

These innovations in predictive, self-supervised, and control-based auxiliary learning have advanced deep RL, producing agents that learn faster, generalize better, and tackle sparse-reward challenges in both synthetic and real-world applications.

Despite their success, these methods typically involve joint-training of the primary RL agent with auxiliary objectives, which can be computationally expensive and impractical in large-scale (i.e., millions or billions of users) enterprise environments where specialized models are already developed and maintained. A key contribution of our work is moving away from this paradigm. We demonstrate that by strategically decoupling these tasks, i.e. using the outputs of pre-existing, independently trained models as direct inputs to the agent's state representation (see Section 4.1.3), we can significantly enrich the state and improve performance without the overhead and complexity of multi-task learning. This makes our approach highly practical and scalable for real-world deployment.

4.1.3 Methodology

Markov Decision Process

The recommendation task can be effectively modeled as a Markov Decision Process. In this setting, a recommender agent interacts with the environment over a series of time steps. At each time step, the agent selects an action a from a set of possible actions A, based on the current user state s, which belongs to the set of possible states S. After performing this action, the agent receives a reward $R(s, a)$, and the user state transitions to a new state s' according to the transition probability distribution $p(s'|s, a)$. The goal of the recommender agent is to learn an optimal policy $\pi_\theta : S \times A \to [0, 1]$ that maximizes the expected cumulative reward over time. The Markov Decision Process is defined by the tuple $M = (S, A, P, R, \rho)$, where:

- **States (S)**: a state $s \in S$ encapsulates the user's features and interaction history.
- **Actions (A)**: an action $a \in A$ corresponds to recommending a specific item from the dataset to the user.
- **Transition probability ($p(s'|s, a)$)**: it describes the probability of transition from state s to s' after action a.
- **Reward function ($R(s, a)$)**: this function quantifies the user's response to action a in state s.
- **Discount Rate (ρ)**: the discount rate $\rho \in [0, 1]$ measures how the value of future rewards is perceived in the present; a higher value places more emphasis on long-term rewards, encouraging the agent to consider the long-term consequences of his actions.
- **Policy π_θ**: the policy defines the agent's strategy for selecting actions given a particular state, mapping each state s to a probability distribution over actions A. The optimal policy aims to maximize the expected sum of discounted rewards, guiding the agent to make recommendations that not only meet immediate user preferences, but also promote long-term engagement.

Proposed Framework

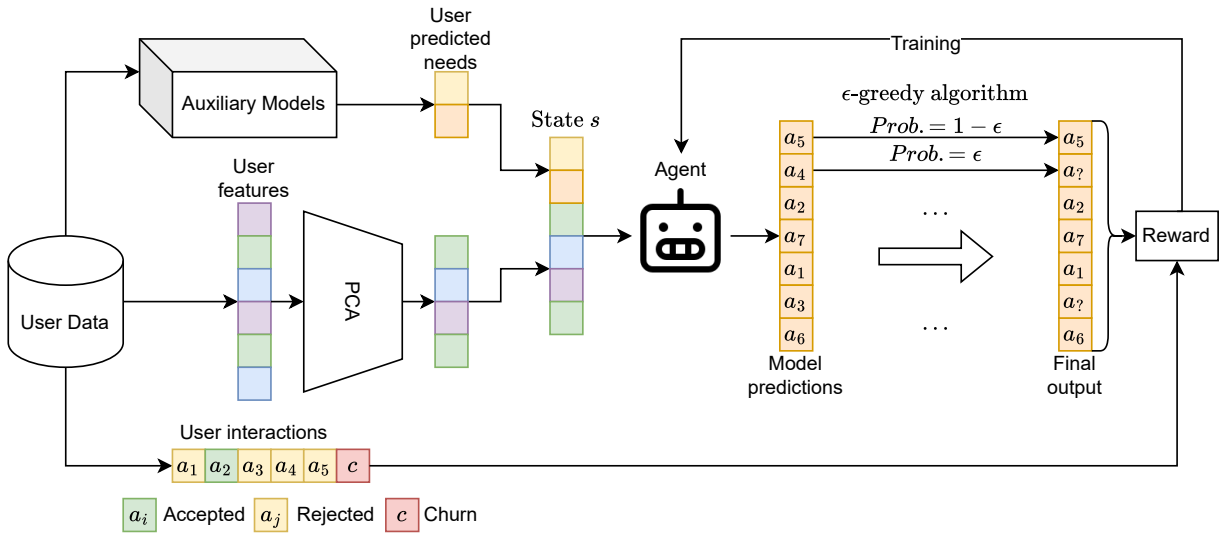


Figure 4.1: The schematic structure of the proposed CRAB framework. States are composed of the concatenation of user features and company model outputs. The agent is trained using previous user interactions, taking into account its responses and churn events. ϵ-greedy strategy is implemented during the training procedure to deal with the exploration-exploitation problem.

Our goal is to create an RLRS that provides personalized recommendations to customers, maximizing the acceptance of recommended items while minimizing customer churn.

Our proposed framework, named CRAB, is illustrated in Fig. 4.1. It consists of two main components: the environment and the agent. Raw user interaction data is pre-processed to create an appropriate RL environment. Within this environment, the RL model generates personalized recommendations for each customer, aiming to reduce churn by providing optimal action suggestions.

The implementation of CRAB starts with the creation of the RL environment, which is described in detail in the following section.

States

States of our RL environment encompass the daily characteristics of each customer. We collect approximately 350 features per customer, encompassing previous interactions, demographic attributes, billing history, call records, and navigation data.

More specifically, each state includes fundamental demographic attributes such as age, gender, region of birth, and current region of residence. It also captures the client's relationship history with the company, including client tenure, i.e the total duration the user has been a customer, and line age, i.e. the operational age of their current service line. The dataset details user service subscriptions and financial interactions, including the billing amount, which reflects the total monthly cost of all services, i.e. the composite amount of their core service plan and charges for any supplementary add-on packages. These add-ons provide insight into customer preferences for value-added services, encompassing offerings such as premium television packages, integrated smart home or domotic solutions, and traditional voice services. In addition to their commercial and demographic profiles, the dataset captures the client's direct interactions with the customer care. These records serve as a valuable proxy for customer satisfaction, allowing for the systematic analysis of prevalent

issues, including technical support needs such as connectivity problems and slow speeds, as well as administrative queries concerning billing issues, and commercial inquiries about upgrading services or activating new offers. Another set of features quantifies the client's network usage and online behavior, such as data traffic, specifically the total volume of data downloaded and uploaded by the client over specified intervals.

To reduce the computational cost by approximately three times, we use Principal Component Analysis (PCA) to project the original feature space into a 128-dimensional space, retaining over 80% of the explained variance.

Auxiliary Models

Inspired by the works presented in Section 4.1.2, we incorporate auxiliary tasks into the customer state representations to enhance agent recommendations without the need for retraining the auxiliary models, training them jointly, or training the primary model to perform multiple tasks. Specifically, we use eight pre-existing auxiliary models developed for related business tasks, including: a gradient-boosted tree (GBT) model that predicts the 60-day churn probability, a regression model that estimates the 12-month Customer Lifetime Value, and multiple GBT-based propensity models for offer acceptance across categories (e.g, entertainment, sport, fiber-to-the-home, assurance, etc.). These models are automatically retrained in the event of a performance drop, and their outputs are generated daily. Their scalar outputs are concatenated with the PCA-reduced base user feature vector.

This design choice offers two key advantages: computational efficiency and operational viability. From a computational perspective, we bypass the overhead of joint-training pipelines, which often demand massive, unified datasets and extensive hyperparameter tuning to balance the loss functions. Secondly, by treating the auxiliary models purely as feature extractors, we avoid potential optimization challenges that could affect the performance of the primary task. Furthermore, our framework aligns with the operational realities of large companies, where a set of specialized machine learning models is developed, deployed, and managed by different teams to solve specific business problems. CRAB is designed to integrate their valuable, pre-existing outcomes without requiring costly and complex changes to current systems.

Actions

In our RL setup, the action space defines the set of all possible recommendations the agent can make. These correspond to commercial offers and service upgrades designed to improve the customer experience and increase lifetime value. The catalog is highly diverse, spanning multiple product categories. It includes foundational telecommunication services (e.g., specific voice and Wi-Fi plans), a wide array of entertainment subscriptions for TV and streaming platforms (such as DAZN, Netflix, Amazon Prime, Disney+, and TIMVISION), various promotional discounts, smart home devices (e.g., Google Nest), extra services (e.g., assurance), and essential network equipment like routers. For the scope of this study, this cumulative catalog results in a discrete action space comprising over three hundred unique actions, presenting a substantial and complex decision-making challenge for the recommendation agent.

At inference time, recommendations are deployed in real-time customer service interactions.

Whenever a customer contacts the call center, for example, to cancel their service, report technical malfunctions, or request information, the RL framework estimates Q-values for all possible actions and ranks them. The top five actions are presented to the human operator, who then presents one or more to the customer. Crucially, the feedback used for training corresponds to the specific offer the human operator suggested, and the reward is based on the customer's response to that chosen offer. This process ensures our agent learns to generate rankings that are effective in a real-world, operator-assisted environment.

As only the top five items are shown in practice, our offline analysis presented in Section 4.1.5 places a significant emphasis on evaluation metrics calculated with a cutoff of $K = 5$. This ensures alignment between offline evaluation and its intended operational environment.

Episodes

In our RL formulation, an episode represents the complete interaction history of a single customer: sequence of states, actions (or recommendations), and rewards, starting from their first observed interaction and terminating either when they churn or when the study's observation period ends.

A significant challenge in modeling customer behavior, particularly churn, is the inherent latency between a customer's decision to leave and the actual service cancellation. The negative experiences or unsatisfactory offers that trigger churn may occur weeks or even months before the event is formally recorded in the system. A naive approach that only considers the churn status at the immediate end of an episode would fail to capture these delayed effects, potentially misattributing the churn event to more recent, unrelated interactions and failing to penalize the true causal actions. To address this temporal misalignment, we establish a two-month look-ahead window beyond the episode's terminal state. If churn occurs within this window, the episode's terminal state is retrospectively labeled as a churn state, and the corresponding reward is applied.

Crucially, while this window is used for outcome labeling, any actions suggested to the customer or interactions that occur within this period are excluded from the episode's state-action-reward sequence. This ensures that rewards are attributed to the sequence of recommendations made during the episode and not to events that happened after the episode had concluded. This strategy allows the agent to learn the long-term consequences of its actions more effectively, leading to a more robust and realistic recommendation policy.

Reward

The reward function is a crucial component in our framework, as it guides the agent to learn actions that not only maximize immediate acceptance rates, but also minimize long-term customer churn. The reward for each action depends on whether the action was accepted or refused by the customer, as well as the occurrence of a churn event.

To account for churn, we introduce a delayed reward component. This approach ensures that the agent prioritizes actions that both engage the customer and reduce the likelihood of churn. Specifically, the agent receives a reward or penalty in a terminal state not only based on the acceptance of the last offer, but also based on the churn outcome. The reward function $R(s, a)$, based

on the customer's response r to the suggested action and the churn event c, is defined as follows:

$$R(s,a) = \begin{cases} \alpha, & \text{if } r(s,a)=1, c(s,a)=0 \quad \textit{Accept} \\ \beta, & \text{if } r(s,a)=0, c(s,a)=0 \quad \textit{Refuse} \\ \gamma, & \text{if } r(s,a)=1, c(s,a)=1 \quad \textit{Accept but churn} \\ \delta, & \text{otherwise} \quad \textit{Refuse but churn} \end{cases} \tag{4.1}$$

where $r(s,a)$ is the customer's response (1 for accepted, 0 for refusal), and $c(s,a)$ indicates the occurrence of a churn event (1 for churn, 0 for no churn). α, β, γ and δ are constants chosen according to the following criteria:

- $\alpha, \beta > 0, \alpha > \beta$ representing positive rewards for non-churning users.
- $\gamma, \delta < 0$ to reflect a penalty for churning.
- $\alpha > \beta, \gamma > \delta$ to prioritize action acceptance over rejection.
- $\alpha + \delta < 0$ to discourage the event of a user accepting a recommendation and churning after the first refusal.
- $\alpha + \gamma > 0$ to accept the event of a user accepting two recommendations before churning

The churn-related rewards apply only in the terminal state, while for the rest of the episode, only cases with $c = 0$ are considered.

The reward is constructed this way to emphasize the importance of preventing the churn of customers. Even if the reward considers whether the action is accepted or refused, it gives more weight to whether this action leads to the churn of the customer or not. In an ideal scenario, with this reward, the agent, by interaction with the environment, should learn the long-term consequences of the actions to understand which action leads to churn.

Recommender System

To optimize the policy, we employ a discrete *Conservative Q-Learning* (CQL) model [146], a state-of-the-art approach in offline reinforcement learning. CQL is designed to learn conservative, lower-bound estimates of the value function by incorporating regularization into the Q-values during training, thereby mitigating the risks associated with overestimation and distributional shift. Formally, the model is trained to minimize the loss:

$$L(\theta) = \phi\, \mathbb{E}_{s_t \sim D}[\log \sum_a exp Q_\theta(s_t, a) - \mathbb{E}_{a \sim D}[Q_\theta(s,a)]] + L_{DoubleDQN(\theta)} \tag{4.2}$$

In the CQL algorithm, multiple Q-functions can be updated, instead of one, with independently sampled experience replay memory, and then the action selected by the ensemble is taken. The number of these Q-functions is referred to as the number of critics. Previous studies, such as [258], have shown that this ensemble approach surpasses non-ensemble methods in both performance and stability. Utilizing an ensemble of Q-functions effectively reduces decision variance, leading to more efficient training in specific tasks. However, this approach also introduces a potential drawback: increased computational time for each reinforcement learning step.

To address the exploration-exploitation dilemma, we adopted the ϵ-greedy approach during the training phase. Specifically, with some probability ϵ, the agent selects a random action, while with probability $(1-\epsilon)$, it chooses the best action according to the Q-function. This method allows the agent to explore, during training, different actions with probability ϵ and exploit the action that maximizes the Q-value with probability $(1-\epsilon)$. This balance between exploration and exploitation is crucial for effectively learning an optimal policy. During inference and deployment, ϵ is *set* to 0, and the agent recommends the actions with the highest learned Q-values.

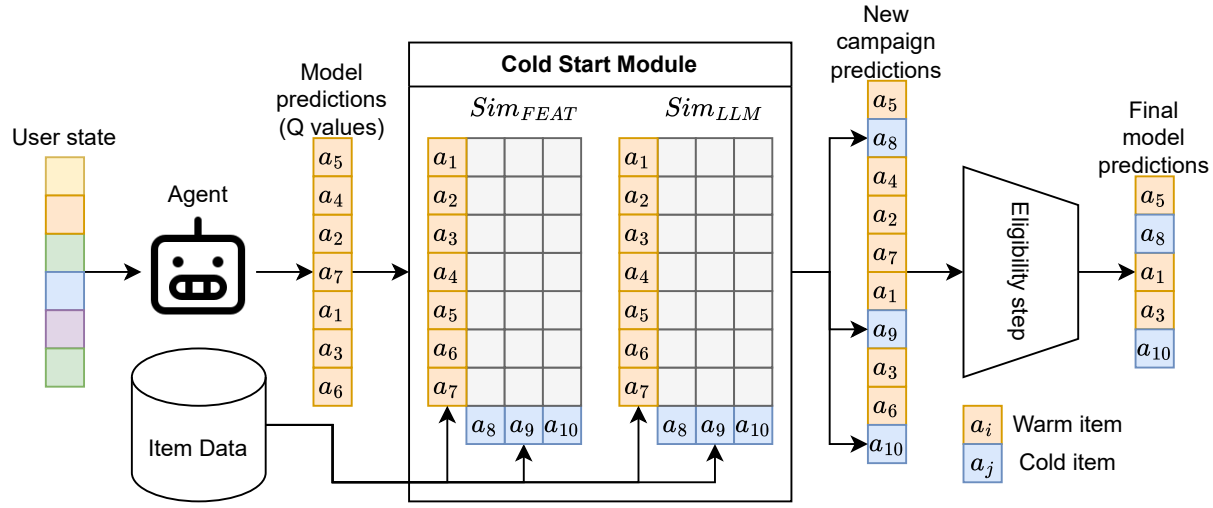


Figure 4.2: CRAB functioning during inference. The cold-start module uses the agent predictions and similarities of cold start items to obtain the new campaign ranks. The eligibility step filters out items that are not eligible for selection.

Cold start campaigns

When launching a new campaign, the company may introduce a new offer set that has not been previously tested or recommended to users. This situation leads to the cold start problem, because our reinforcement learning model has no historical performance data from which to derive Q-values for these new items. To address this, we leverage the similarities between the new offers and the existing ones, along with the rankings provided by the trained model, to generate a ranking for the campaign offers. This ranking takes into account several aspects, such as price, offer composition by using the various components (including components like data, voice, and extra packages), and the textual description of the offer.

To estimate the new Q-values, $Q(s,\hat{a})_{\pi_\theta}$, which represent the long-term expected reward of taking action $\hat{a}$ in state s under policy π_θ, we use the following formula:

$$Q(s,\hat{a})_{\pi_\theta} = \sum_{a}^{A'} Q(s,a)_{\pi_\theta} * \frac{(Sim_{FEAT}(a,\hat{a}) + Sim_{LLM}(a,\hat{a}))}{2} \tag{4.3}$$

where $\hat{a}$ represents one of the cold actions in the set $\hat{A}$ and A' are the warm actions with top-5 associated similarities. This equation combines three elements:

- **The old Q-values $Q(s,a)_{\pi_\theta}$:** these are provided by the trained model on the warm items.

- **Similarity of tabular features Sim_{FEAT}:** this represents the cosine similarity between the new and existing tabular features. These attributes include quantitative features like offer price and data allowance.
- **NLP similarity Sim_{LLM}:** this is the cosine similarity of offer description embeddings. The embeddings are generated using a pre-trained Italian sentence-BERT-based model [64]. It captures the similarity in the offers' textual descriptions.

By averaging these two similarity measures, we adjust the old Q scores to provide reasonable values for the new cold start items. This process, illustrated in Fig. 4.2, ensures that new campaigns can be launched with an effective initial ranking, mitigating the cold start problem.

Eligibility step

After the agent generates a complete, ranked list of all actions, including values for new items from the cold-start procedure, the framework proceeds to the eligibility step, a critical phase that allows to recommend real-world offers. Each recommended action is mapped to a specific set of offers from the catalog. This process involves filtering out offers that are ineligible due to various constraints, such as business price constraints, connection availability, customer accessibility reasons, or company rules. This step ensures that the recommendations presented to customers are practical and actionable.

4.1.4 Experimental Setup

In this section, we outline the setup of our offline and online experiments, designed to test the performance of the proposed CRAB framework against appropriate baselines. In addition, we conduct ablation studies to identify the key factors that contribute to its effectiveness.

Dataset

The company dataset regards upselling recommendations, and it has several unique characteristics. Unlike traditional recommendation datasets that track single-item interactions, our dataset captures multiple recommendations occurring on the same day. This is because recommendations are provided through human phone calls, and during a single call, customer care agents may suggest multiple items to the customer. Moreover, the company recommender system suggests the top five sorted offers to the customer care agent, but the items may not be suggested in order. It can happen because the customer explicitly requests a specific item during the call, or simply because the customer care agent decides not to recommend it in that order, based on how the call proceeds.

We use thirteen months of historical daily customer data provided by the company. The data is split temporally: the first eight months serve as training and validation sets, while the last five months are used for offline evaluation (test set). The validation set consists of a random 10% sample of episodes from the first eight months.

The dataset contains daily customer information, including their status, actions proposed by the company, and customers' responses (accepted or rejected). Actions have been generated by the company's baseline recommender system, with varying frequency across customers.

The dataset used for training comprises records from over 7 million users, of whom 1.7 million exhibited activity within the specified time period. A total of 9.4 million actions were recorded

during this period, with only 370 thousand actions meeting the acceptance criteria, yielding an acceptance rate below 4%. The mean episode length was 5.65 ± 9.09. Furthermore, 190 thousand customers churned during the considered timeframe.

We use this offline dataset to benchmark the CRAB framework against the company's recommender system over the last five months.

Testing our method on a public dataset is currently not feasible because, to the best of our knowledge, no available dataset provides the essential combination of features. Specifically, our approach requires temporal data on users, detailed churn information, and a rich history of user interactions, including, in particular, user acceptance or rejection of proposed offers.

Baselines

We evaluate our proposed CRAB framework against a diverse set of representative baseline models.

- **Multinomial Logistic Regression (MLR)** [78]: a linear model that classifies each possible offer and sorts the output odds to generate the list of recommended offers.
- **Alternating-Least-Squares with Weighted λ Regularization (ALS-WR)** [318]: a matrix factorization algorithm, known for its efficiency in handling sparse matrices, making it suitable for large-scale recommendation tasks.
- **LambdaMART** [34]: a learning-to-rank model that uses gradient boosting to produce personalized rankings.
- **LightGCN** [109]: a graph neural network performing link prediction on the bipartite graph made by users and items, where a link connects a user to an item if he bought it.
- **SASRec** [135]: a sequential model leveraging self-attention to capture users' sequential behavior patterns, enabling personalized next-item recommendations by modeling short and long term user preferences.
- **Company baseline**: the current non-deep learning recommender system used by the company.

All the baselines are trained on the first eight months of data and tested on the last five months. For MLR, ALS-WR, and LambdaMART, offers are rated as 1 (accepted), 0 (rejected), or 0.5 (not offered).

All models are trained on the same data, enriched with the same auxiliary information used in CRAB, ensuring a fair evaluation.

To ensure a fair and robust comparison, all baseline models were tuned using Optuna [7]. Search spaces and final hyperparameters are detailed Section 4.1.7.

Company Baseline

The company's recommender system used in production is fundamentally different from our approach. Its primary objective is to optimize for immediate, short-term outcomes by predicting the likelihood of offer acceptance at each interaction. It does not employ a sequential decision-making

framework and is not explicitly trained to minimize long-term churn, making it a suitable benchmark for a short-term focus recommendation strategy. Due to the company's privacy policy and business confidentiality, further information about the model cannot be made publicly available.

Evaluation Metrics

We evaluate the models using Recall@K, Coverage@K, Normalized Discounted Cumulative Gain@K (NDCG@K) and Precision@K (P@K). Metrics are computed using relevant actions generated by the original policy that were accepted.

Moreover, we use specific RL metrics to compare models with each other, performing a grid search to find the best hyperparameter set. We decide to use Average TD error and Soft Off-Policy Classification (SOPC), evaluating the difference for each time-step t between the estimate for V_t and the discounted value estimate of V_{t+1} plus the actual reward gained from transitioning between states s_t and s_{t+1}, and gaps of action-value estimation between the successful episodes and all episodes respectively.

Hyperparameters Tuning

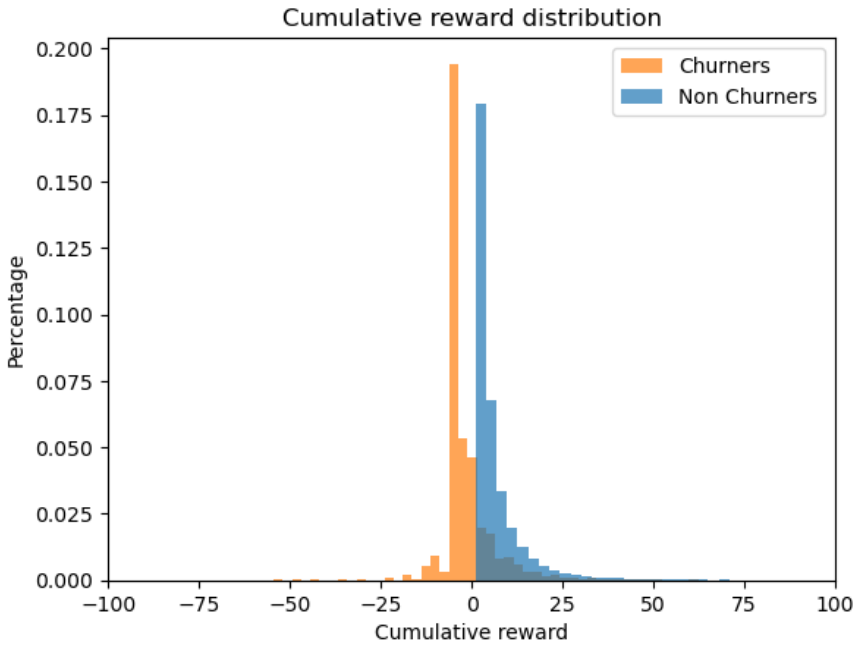


Figure 4.3: Cumulative reward distribution for churners (left) and non-churners (right) customers.

To decide the values for the parameters α,β,γ and δ of the reward Eq. (4.1), we focus on obtaining opposite distributions between churning and non-churning customers. In particular, the company requires that a churning customer who has accepted two offers is still regarded as a favorable outcome. This imposes the additional constraint $\alpha + \gamma > 0 \wedge 2\alpha + \delta > 0$. To satisfy this and the other constraints introduced in Section 4.1.3, we select the following parameter values: $\alpha = 4, \beta = 1, \gamma = -3, \delta = -6$. Using these values, the cumulative reward distributions for clients who churn and clients who do not churn throughout an episode are shown in Fig. 4.3. The chosen values ensure that the cumulative reward is negative for most churning customers, while always being positive for non-churning customers, effectively aligning with the company's requirements.

We conduct a grid search to optimize the following hyperparameters:

- batch size: $\{256, 512, 1024\}$
- the number of critics: $\{2, 3\}$
- Activation function: $\{Swish, ReLU\}$
- the tradeoff factor in the iterative update for training the Q-function in CQL algorithm ϕ: $\{0.01, 0.1, 1\}$
- probability of ϵ-greedy ϵ: $\{0, 0.1, 0.2, 0.4\}$
- discount factor ρ: $\{0.1, 0.5, 0.7, 0.99\}$.
- learning rate: $\{6.25\text{e}^{-6}, 0.01\}$.

Fitted Q Evaluation (FQE) [155] is employed for offline policy evaluation. We use it to retrain a critic for each policy generated by the offline RL algorithms. From all the obtained models, we select the one that maximizes Recall@K on the validation set.

The best model has the following hyperparameters:

- batch size: 1024
- the number of critics: 3
- activation function: *swish*
- tradeoff factor ϕ: 0.01
- probability of ϵ-greedy ϵ: 0.2
- discount factor ρ: 0.99.
- learning rate: 6.25e^{-6}

All subsequent results are based on this model configuration.

Code and privacy restrictions

For the implementation of the discrete CQL model, we use the Python *d3rlpy* library[234]. The original implementation only returns the best action for each state according to the Q-values. However, to use the ϵ-greedy algorithm, our RLRS model must provide a list of all possible actions, sorted and ranked by their Q-values. To achieve this, we modify the library to output the top k actions for a given user in a specific state. The updated version of the d3rlpy library can be found on our private Git repository[1] and installed for further development and experimentation.

Due to the company's privacy policy and business confidentiality, neither the trained model nor the dataset can be made publicly available.

[1] https://anonymous.4open.science/r/crab/

4.1.5 Results

In this section, we present the results of our extensive experiments, which demonstrate the superiority of the proposed CRAB framework. Additionally, we conduct ablation studies to identify the key factors contributing to its effectiveness.

Offline Comparison

Table 4.1: Performance comparison between baseline models and CRAB for NDCG@K, Precision@K and Mean Average Precision (MAP). **Bold** indicates the best result, while second-best is underlined. †indicates a statistically significant result according to a Friedman test with a significance level of 0.05.

Model	NDCG@5	NDCG@10	Precision@5	Precision@10	MAP
MLR	0.4041	0.4041	0.0844	0.0422	0.3990
ALS-WR	0.3015	0.3283	0.0802	0.0511	0.3146
LambdaMART	0.4073	0.4079	0.0923	0.0469	0.3353
SASRec	0.4191	**0.4691**	0.0990	0.0527	0.4242
LightGCN	0.4183	0.4218	0.0904	0.0559	0.4219
Company Baseline	0.3154	0.4163	0.0836	0.0544	0.3996
CRAB	**0.4233**†	0.4479†	**0.0994**†	**0.0582**†	**0.4246**†

As discussed in Section 4.1.3, we focus on $K = 5$ because customer service agents suggest the top 5 recommendations returned by the recommendation model. Our primary goal is to provide a desirable item in the top-5 recommendations. We also include results for $K = 10$ to assess broader ranking effects.

In Table 4.1, we compare all tested models by analyzing recommended offers for all customers. The results show how CRAB can consistently outperform all competitors on metrics calculated with $K = 5$, with the sequential model SASRec achieving the second best results. At $K = 10$, SASRec achieves better performance in NDCG, while CRAB is the second-best. This indicates that while other models might be tuned for optimizing longer lists, CRAB is suited for short-list recommendations. Moreover, CRAB's goal is multi-objective and not only focused on upselling.

Table 4.2: CRAB offline performance against the company baseline. The average rank of accepted action increases.

	Accepted actions average rank ↓
Company baseline	4.57
CRAB	**3.73**

Table 4.2 shows that CRAB also has a higher average rank of accepted actions (**3.73**) compared to the company baseline model **4.57**. This difference indicates that CRAB's accepted actions are, on average, roughly one rank higher, providing a better customer experience.

Online Performance

Table 4.3: CRAB online performance in A/B test. During the month experimenting A/B test, average acceptance increased and churn ratio decreased.

Offer Acceptance ↑	*Customer Churn* ↓
+3.51%	**-4.95%**

To validate our approach, we conduct a series of A/B experiments on a live system serving millions of users. Experiments last one month. Our focus is centered on two key metrics: offer acceptance and churn.

Table 4.3 shows that our method improves the offer acceptance rate by **+3.51%** over the baseline. There is also a significant reduction in the user churn rate of **-4.95%** compared to the baseline. This substantial reduction can be attributed to CRAB's strategy of incorporating a strong penalty for customer churn events, facilitated by a high discount rate ρ. This approach prioritizes long-term rewards, enabling the agent to recommend actions that are less likely to result in churn. This result is fundamentally important because it validates the effectiveness of the recommendation model in an online scenario and confirms that CRAB's recommendations achieve a lower churn rate.

Model Analysis and Hyperparameter Tuning

RLRS Performance and Diversity. In Table 4.4, we present additional metrics for the RL model of the CRAB framework: *Recall*@K and *Coverage*@K. The distribution of performance of models with different hyperparameters, based on the metric Recall@K averaged over all considered Ks is shown in Fig. 4.4. We notice that, on average, models have a Recall@K of more than 0.3.

We also notice from Table 4.4 that Coverage@K is sufficiently large, which indicates that recommended actions are diverse, as different actions are recommended for customers.

Table 4.4: Performance of the best CRAB model obtained after hyperparameter tuning on the test set. It provides diverse and good recommendations, based on *Coverage*@K and *Recall*@K.

K	*Recall*@K	*Coverage*@K
1	0.3713	0.2681
3	0.4209	0.4181
5	0.4623	0.5045
7	0.4995	0.5954
10	0.5107	0.7000

Impacts of the Number of Critics. In our experiments, we tested models with two and three critics. As explained in Section 4.1.3, we refer to critics as the number of Q functions for the ensemble. The results, as illustrated in Fig. 4.4, indicate that using three critics yields significantly higher performance compared to employing only two. However, due to constraints in computational resources, we could not test configurations with more than three critics, as it would considerably increase the algorithm's time. In particular, training with three critics necessitates approximately three times the time per epoch compared to training with two critics.

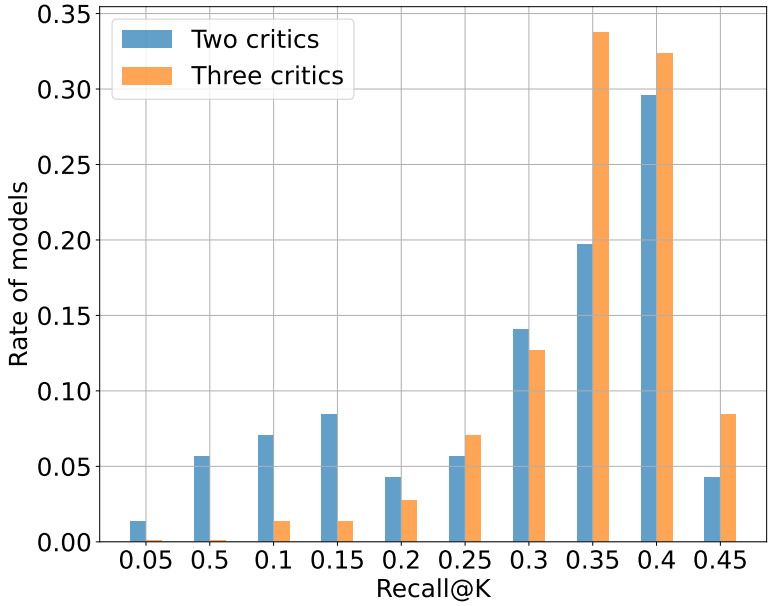


Figure 4.4: Distribution of Recall@K within all models in grid search with two critics and three critics.

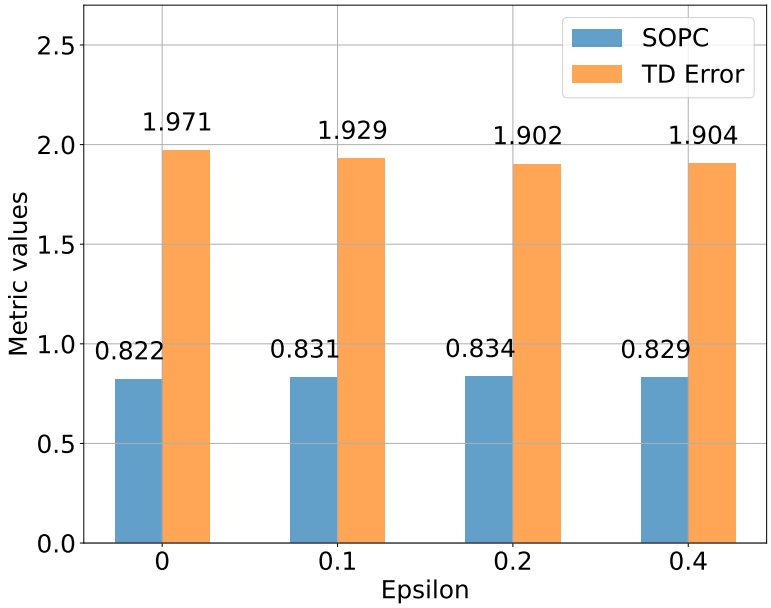


Figure 4.5: RL metrics results obtained for different ϵ.

Impacts of the ϵ-Greedy Probability. In order to optimize the balance between exploration and exploitation, the efficacy of the model was evaluated using TD error and SOPC metrics, with values of $\epsilon \in \{0, 0.1, 0.2, 0.4\}$. The results are shown in Fig. 4.5. As illustrated, the model exhibits optimal performance with $\epsilon = 0.2$, indicating a good exploration rate while not preventing exploitation as much as higher values do. Action values in successful episodes are higher than in the others. The plot also illustrates that the lowest performance is attained when $\epsilon = 0$ values are not equal to zero, proving the ϵ-greedy approach to be beneficial.

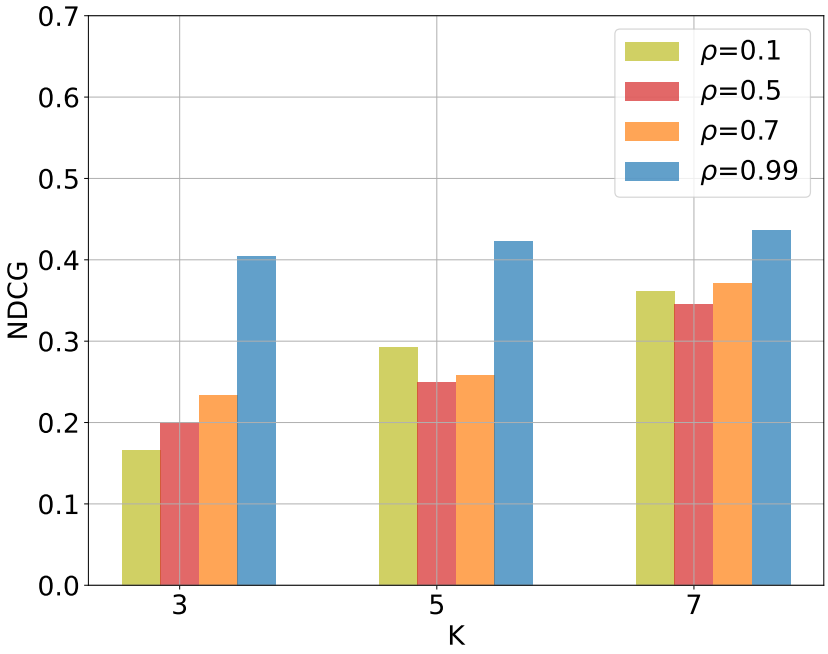


Figure 4.6: Results obtained with different ρ values.

Impacts of the Discount Factor - Delayed Feedback Analysis. We investigated the influence of delayed feedback of customer churn on model performance by varying the discounted factor ρ. Preventing churn is one of the main company's objectives, and churn events typically mark the conclusion of a customer episode, so it is crucial to consider high values of ρ.

In Fig. 4.6, we present the NDCG values obtained by different models trained with ρ equal to $\{0.1, 0.5, 0.7, 0.99\}$. The outcomes demonstrate that a high discount factor ($\rho = 0.99$) markedly enhances the model's efficacy, suggesting that prioritizing long-term rewards facilitates the model's capacity not only to more accurately account for churn, but also to provide better recommendations.

Ablation Study: Impact of Auxiliary Model Outputs. To assess the impact of auxiliary models, we compared the NDCG of CRAB variants: the first variant, CRAB-1, incorporates auxiliary customer predictions into the state representation. In contrast, the second variant, CRAB-2, does not. Finally, to investigate whether performance scales with the amount of information provided, a third variant, CRAB-3, includes a random 50% subset of the auxiliary predictions. As shown in Fig. 4.7, CRAB-2 exhibits performance comparable to that of the company baseline model at K=5. CRAB-1 demonstrates superior performance compared to both CRAB-2 and CRAB-3, thereby underscoring the added value of incorporating auxiliary model outputs into the state representation. This comparison not only highlights the benefit of this enrichment but also suggests that performance scales with the amount of auxiliary information provided, as CRAB-3 consistently places between the other variants.

The critical importance of this step is illustrated when contextualizing these results with our main Table 4.1. The performance of the CRAB-2 variant, which lacks auxiliary data, drops significantly. This means that without this information, our model would perform worse than top baseline competitors. Therefore, this is the key reason for CRAB's better performance.

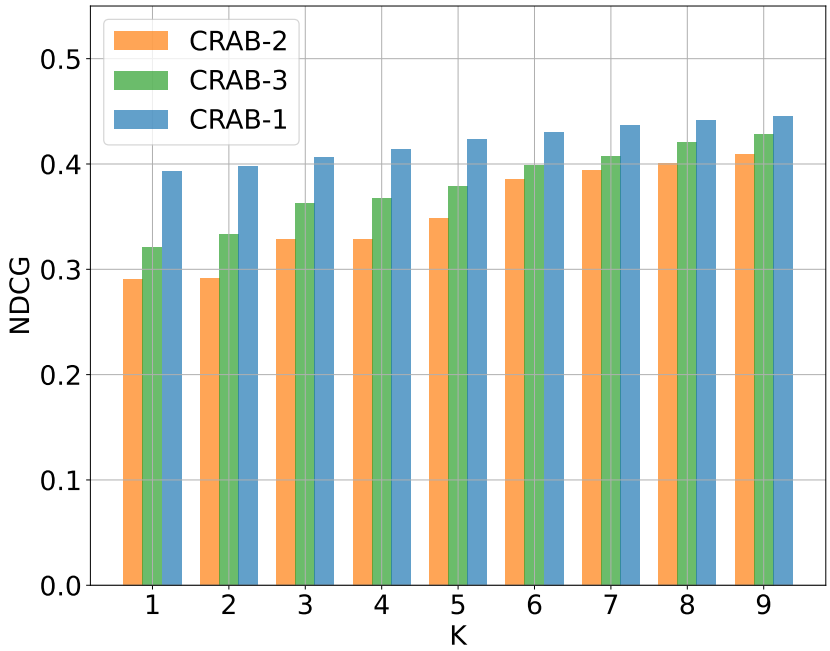


Figure 4.7: Results comparing CRAB with (CRAB-1), without (CRAB-2), and using 50% random (CRAB-3) auxiliary model outputs.

4.1.6 Conclusions and Future Work

In this paper, we addressed the critical challenge of customer churn in the telecommunications industry, a problem compounded by the delayed nature of customer feedback. We introduced CRAB (Churn Reduction Agent-Based) , a recommender system designed to mitigate delayed feedback churn cases through the utilization of offline RL. This approach employs historical customer interactions, product usage patterns, and user feedback to train the model, without requiring real-time user interactions. Unlike traditional recommender systems that optimize short-term metrics such as immediate clicks or offer acceptances, CRAB is explicitly designed to learn a policy that balances the immediate goal of upselling with the crucial long-term objective of reducing churn.

Our primary contribution is a practical and effective methodology for integrating delayed feedback into the RL training process. By defining episodes that extend into a future look-ahead window, and by designing a reward function that heavily penalizes future churn events, CRAB learns to avoid recommendations that seem effective in the short term, but carry a high risk of leading to customer dissatisfaction and churn. Our offline results show that the CRAB framework generates more diverse and relevant recommendations than the original company policy. Findings demonstrate the consistent superiority of CRAB over other baseline models. In particular, we demonstrated a novel and highly effective method for enriching the agent's state representation. By incorporating the predictive outputs of pre-existing, independently trained auxiliary models, we provide the agent with a richer, more holistic view of the customer. Our ablation studies confirmed that this enrichment is fundamental to CRAB's superior performance.

The empirical validation of our framework, conducted on a large-scale proprietary dataset from a major telecommunications company, yielded compelling results. In extensive offline experiments, CRAB consistently outperformed a wide range of state-of-the-art baselines and the existing company

policy. More importantly, subsequent live A/B testing in a real-world production environment confirmed these findings, demonstrating a 4.95% reduction in customer churn alongside a 3.51% increase in offer acceptance.

In addition to improving performance, the ability of offline RL to operate without direct interaction with the customer can reduce the risks associated with online RL, such as privacy concerns and ethical issues. Our findings provide a strong case for adopting offline RL in the recommendation systems domain. Following the successful online testing, CRAB has been deployed to all users.

Nonetheless, there are various directions for future work.

- Currently, CRAB handles the cold start problem for new items using similarity-based Q-value estimation. However, recommendations for new users are missing. A method to address the cold-start issue for users needs to be designed.
- The current framework is applied only to upsell offers. We plan to extend its application beyond upselling offers and also to include retention strategies.
- Exploring more advanced offline RL algorithms beyond CQL could yield further performance gains.

In conclusion, CRAB shows that offline RL is an efficient way to solve difficult, real-world business problems. By effectively managing delayed rewards and leveraging auxiliary models for richer state representations, CRAB offers a scalable solution for improving upselling and reducing customer churn.

4.1.7 Baselines Hyperparameters Tuning

To ensure a fair comparison among all baseline methods, we performed hyperparameter optimization using Optuna [7]. All models were trained and validated using the same data splitting strategy, applied uniformly across all methods. Specifically, for each user, we sorted their interactions chronologically and split them into three disjoint sets, as described in Section 4.1.4: the last five months are used exclusively for test evaluation, while the first eight months were used for train and validation (split 90%-10%). Model selection and optimization decisions were based solely on performance on the validation set.

In the tuning process, we optimized NDCG@5 as the primary metric. This choice reflects our interest in accurately ranking the top few items, which is particularly important in our recommendation context. As discussed in Section 4.1.4, focusing on NDCG@5 aligns with our goal, where only top results are visible to the customer care agent during call recommendations.

In each Optuna study, we fixed a number of trials (up to 20), with pruning enabled to discard poorly performing configurations early. We report the hyperparameters tuned for each baseline with the final values selected through the optimization process.

MLR:

- L2 regularization : $\in [1e-4, 1e2]$
- Solver: $\{lbfgs, saga\}$
- Max iterations: $\in [100, 1000]$

ALS-WR:

- Latent factors: $\in [10, 200]$
- Regularization: $\in [1e-4, 10]$
- Number of iterations: $\in [10, 100]$

LambdaMART:

- Number of trees: $\in [100, 1000]$
- Max tree depth: $\in [3, 10]$
- Learning rate: $\in [1e-3, 0.1]$
- Subsample ratio $\in [0.6, 1]$

LightGCN:

- learning rate: $\in [1e-4, 0.1]$
- embedding dimension: $\in [16, 128]$
- number of layers: $\in [1, 3]$

SASRec:

- Attention blocks: $\in [1, 3]$
- Attention heads: $\in [1, 3]$
- Batch size: $\in [128, 2048]$
- Learning rate : $\in [1e-4, 1e-1]$
- Embedding size : $\in [16, 1024]$.

Following the optimization procedure, we identified the best-performing hyperparameter set for each baseline model on the validation set. The final values used for the test set evaluation are detailed in Table 4.5.

This unified tuning approach ensures that all models were evaluated under their best configurations according to the same protocol, making our baseline comparisons robust.

Table 4.5: Best hyperparameter configurations for baseline models.

Model	**Hyperparameter**	**Best Value**
MLR	L2 regularization	4e-3
	Solver	*lbfgs*
	Max iterations	346
ALS-WR	Latent factors	152
	Regularization	0.1
	Number of iterations	73
LambdaMART	Number of trees	497
	Max tree depth	7
	Learning rate	0.05
	Subsample ratio	0.8
LightGCN	Learning rate	3e-3
	Embedding dimension	54
	Number of layers	2
SASRec	Attention blocks	2
	Attention heads	2
	Batch size	985
	Learning rate	5e-3
	Embedding size	464

Chapter 5

Optimizing Graph Representations for Efficient Industrial Recommendation

The graph-based models central to this thesis, while powerful, confront a significant barrier when transitioning from academic benchmarks to live, industrial environments: the dual challenge of computational scale and information quality. Real-world user-item interaction graphs are not only massive, containing millions of nodes and edges, but are also inherently sparse and noisy. This combination creates a significant bottleneck. On one hand, the huge size of the graph makes state-of-the-art models like deep Graph Neural Networks (GNNs) computationally infeasible, leading to out-of-memory errors and prohibitive inference latency. On the other hand, the sparse and incomplete nature of the graph structure limits the effectiveness of the message-passing mechanisms that these models rely on, exacerbating issues like over-smoothing and degrading recommendation accuracy.

This chapter presents two distinct but complementary research directions designed to overcome this dual challenge. These studies change direction from the pursuit of ever-deeper model architectures and instead focus on a more fundamental principle: intelligently engineering the graph itself to make it more flexible to efficient and effective learning.

The first part tackles the scalability problem through graph coarsening. We introduce a framework that leverages business-driven heuristics to aggregate users into meaningful communities, effectively reducing the graph's size. We demonstrate that a two-stage propagation—first across this efficient, coarsened graph and then a refinement within local communities—provides a robust architectural solution for low-latency production systems.

The second study addresses the problem from the perspective of graph structure learning. It poses the question: "Do we need complex, deep models, or can we achieve better performance by fundamentally improving the graph's topology?" Here, we propose a novel framework that uses Simulated Annealing to learn an optimal set of user-user connections, iteratively building a graph that balances recommendation accuracy with strict memory constraints. Our findings reveal that a shallow, one-layer GNN trained on this optimized graph significantly outperforms deeper models trained on the raw, sparse data. This demonstrates that a superior graph structure can eliminate the need for costly deep architectures, simultaneously solving for both accuracy and efficiency. By either simplifying the graph's scale or optimizing its structure, we can build systems that are not only more scalable and efficient but also deliver more accurate and relevant recommendations.

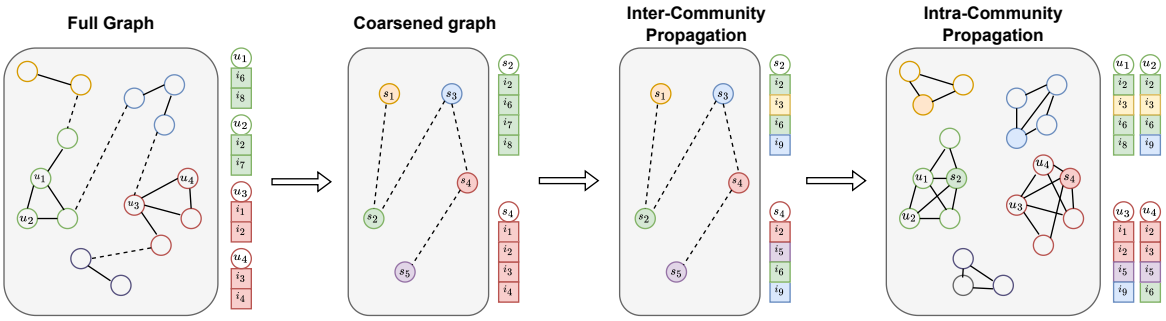


Figure 5.1: Overview of the proposed framework. On the left, the original full graph, where colors illustrate the business-driven rules used to group nodes into communities, while dashed lines highlight inter-community interactions. In the middle is shown the coarsened graph and inter-community propagation. On the right, the intra-community refinement stage is shown.

5.1 Efficient Recommendations via Graph Coarsening and Label Propagation

Graph-based recommendations are widely adopted in real-world industrial applications. However, graphs in these systems often reach a massive scale, posing notable scalability and efficiency challenges. This requires techniques that can effectively balance predictive quality with computational cost. One promising approach is graph coarsening, an adaptive graph reduction technique that offers a way to systematically construct smaller, yet structurally representative, versions of the original large-scale graphs. In this work, we propose a flexible two-stage diffusion framework that combines graph coarsening with multi-step label propagation in the telecommunications domain. Domain-specific heuristics are applied to first aggregate nodes into meaningful communities, reducing graph size while preserving essential business-relevant relationships. An initial diffusion process done by a Label Propagation Algorithm (LPA) or a Graph Neural Network (GNN) propagates labels across the coarsened graph to produce coarse-grained predictions. Finally, a second LPA within subgraphs generates the final recommendations for individual users. On a real-world telecommunications dataset, when using LPA in both stages, our method achieves up to +24% NDCG@5 over the full-graph LPA baseline. Incorporating a lightweight GNN in the first stage further boosts NDCG@5 by more than 50%, but requires substantial training and inference time. Through extensive experiments and a detailed ablation, we quantify these trade-offs and demonstrate that our coarsening-driven approach delivers an optimal balance between scalability, latency, and recommendation quality.

5.1.1 Introduction

Graph-structured data has become widespread across various domains, such as social networks [79, 237], biological networks [259], financial transactions [260], e-commerce platforms [120, 286] because it naturally represents complex relationships between entities. In recommendation systems [88, 280], user-item interactions form large bipartite graphs [79], whose rich connectivity enables machine learning models to uncover latent patterns and deliver personalized recommendations [201]. However, as industrial-scale datasets grow to millions or billions of nodes and edges [65], this expressiveness comes at a steep computational cost: memory demands increase, training and infer-

ence times become prohibitively long, and graph algorithms face scalability bottlenecks [131, 225].

Graph Neural Networks (GNNs) [70, 182] have emerged as a powerful paradigm for recommendation, learning node embeddings through repeated message exchanges across edges. Yet even state-of-the-art large-scale GNNs rely on sampling techniques to handle these large-scale graphs, require heavy computational resources, [300] and struggle with problems like over-smoothing [45] and capturing long-range dependencies [162]. These factors make handling truly massive graphs challenging in low-latency production settings and highlight the need for more scalable alternatives that maintain high-quality recommendations [131].

Lightweight alternatives such as the Label Propagation Algorithm (LPA) [284] bypass expensive gradient computations by diffusing information across the graph by iteratively updating node labels based on its neighbors. Compared to GNNs, LPA is computationally efficient and achieves competitive accuracy [119]. However, directly applying LPA to industrial-scale graphs still involves iterating over every edge multiple times, resulting in significant runtime and memory requirements. Naïve down-sampling of the graph can reduce costs, but it often severs critical links, destroying community structures or isolating hubs, which in turn degrades recommendation quality.

Graph coarsening [36] offers a systematic approach to reduction, involving the aggregation of nodes to create a smaller, more computationally efficient representation of the original graph. While coarsening can improve scalability [121] and reduce noise [112], it may lead to information loss, potentially impacting effectiveness. In fact, a propagation algorithm can diffuse labels efficiently on a coarsened graph, but recommendations become too coarse for fine-grained personalization. In our industrial telecommunications setting, offers (e.g. additional services, bundled products) must be tailored to implicit user groups, that are not explicitly encoded in the raw data, and only made to users who do not already have these services.

To address these challenges, we propose a framework that integrates business-driven graph coarsening with at two-step propagation process. Our approach is designed for large-scale, real-world datasets and follow these steps: (i) users are grouped into super nodes that approximate real families by applying community detection techniques driven by business-specific domain rules, e.g shared billing and call frequency; (ii) either LPA or a lightweight GNN is applied on the coarsened graph to efficiently generate initial recommendations; (iii) a parallel label propagation step is performed within each subgraph to refine recommendations back down to the individual user level. This design combines the efficiency of reduction with the accuracy of localized propagation.

Offline experiments demonstrate the superiority of our proposed framework, achieving sub-second inference times and improving NDCG@5 by up to 24% (LPA) and over 90% (GNN) over a full-graph LPA baseline. Our contributions are:

- We formalize a business-driven coarsening technique that uses business rules to uncover implicit family structures, preserving essential relationships and reducing the original graph size by over 70%.
- We have designed an efficient two-step propagation scheme that combines coarse-graph diffusion with fine-grained refinement to ensure low latency and high accuracy.
- We empirically validate our approach against full-graph LPA and alternative coarsening methods, demonstrating significant improvements in terms of scalability and recommendations quality.

5.1.2 Methodology

Fig. 5.1 presents our framework for scalable offer recommendations, which consists of three steps: 1) coarsening the interaction graph into communities, 2) propagating labels across communities, and 3) intra-community propagation.

Our data can be represented as an undirected user graph $G = (V, E, \mathbf{Y})$, where $V = \{1, \ldots, N\}$ is the set of nodes (users), $E = \{1, \ldots, M\}$ is the set of edges (calls between users) and $\mathbf{Y} \in \{0, 1\}^{N \times L}$ is the label matrix, with L denoting the number of unique labels (offers) and $\mathbf{Y}_{ij} = 1$ if user i has accepted marketing offer j, 0 otherwise.

Graph Coarsening

A coarsened graph $G' = (V', E')$ contains $V' = \{1, \ldots, N'\}$ super nodes and $E' = \{1, \ldots, M'\}$ super edges, with $N' < N$. The coarsening process is represented by a surjective mapping $\mathbf{C} : V \to V'$, where $\mathbf{C} \in \{0, 1\}^{N \times N'}$, ensuring each node i in V is mapped to exactly one super node in V' (i.e. $\sum_{j=1}^{N'} \mathbf{C}_{ij} = 1$).

To ensure that the detected communities align with business goals, we incorporate domain-specific rules. The following heuristics indicate which users can be clustered together to capture implicit family and household structure:

Interaction Frequency: users who communicate more frequently based on call duration and number of calls per week.

Surname: users with the same surname.

Caring: users who frequently contacts customer support on behalf of someone else.

Balance: financial dependencies (e.g. topping up another user's mobile balance, charging someone else bills), especially for younger users

Number of lines: lines associated with a single user are redundant for personalized recommendation.

Users who satisfy at least one of these conditions and who comply with established marketing constraints (e.g. node degree and community size), are merged together to create a super node.

To create the new label matrix $\mathbf{Y'}$, for each super node, we concatenate all label vectors $\mathbf{y} \in \mathbf{Y}$ of the nodes part of that community. Formally, given the mapping $\mathbf{C}$, the new label $\mathbf{Y'}_j$ of community j is computed as $\mathbf{Y'}_j = \mathbb{1}\left(\sum_{i=1}^{N} \mathbf{C}_{ij}\mathbf{Y}_j \geq 1\right)$. This results in each community label $\mathbf{Y'}_j$ containing all the offers activated by all of its users. At this stage, super edges between communities are created based on inter-community interactions (e.g. calls between families).

Inter-Community Propagation

After graph coarsening, a first propagation is performed. We explore two algorithms: LPA and GNNs.

LPA offers an efficient method for information diffusion. It iteratively updates super nodes $v \in V'$ labels based on predominant labels within their local neighborhoods. The LPA is defined as:

$$\hat{\mathbf{Y}} = \alpha \cdot \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}\mathbf{Y} + (1 - \alpha)\mathbf{Y}$$

where α is the propagation hyperparameter, $\mathbf{A}$ is the adjacency matrix of the graph, and $\mathbf{D}$ is the

diagonal degree matrix with $\mathbf{D}_{ii}=\sum_j \mathbf{A}_{ij}$.

When employing a GNN for the first propagation step, we adopt a more expressive graph representation to model interactions more accurately. Specifically, we transform the coarsened community-community graph $G' = (V', E')$ into a heterogeneous graph, denoted as G_h. This graph consists of two distinct node types: communities (the super nodes from V') and items (the L unique offers available in the catalog). The connectivity of G_h is defined by two types of edges:

- **Community-Community** edges, which correspond to the super-edges in E'.
- **Community-Item** edges, which capture the user-item interactions at the community level.

To learn representations on this heterogeneous graph, we utilize the GraphSAGE architecture [101]. Let $\mathbf{h}_v^{(0)}$ be the initial representations of super node $v \in V'$. In each layer l, an aggregated message $a_v^{(l)}$ is computed by applying an AGG function over the representations $\mathbf{h}_u^{(l-1)}$ of v neighboring super nodes $\mathbf{u} \in N(v)$. v representation is updated through $\mathbf{h}_v^{(l)} = \sigma\left(\mathbf{W}^{(l)} \cdot \mathbf{a}_v^{(l)} + b^{(l)}\right)$, where $\mathbf{W}^{(l)}$ and $b^{(l)}$ are learnable parameters. After L layer, the final representation $\mathbf{h}_v^{(L)}$ is obtained.

Intra-Community Propagation

Although the coarsened graph, or its heterogeneous counterpart, efficiently propagates information across large communities, it misses finer intra-community relationships. The recommendations generated from the inter-community step are uniform for all users within the same super-node, lacking the personalization required for individual level predictions. The intra-community step is designed to address this by refining these coarse-grained predictions and reintroducing user personalization.

This refinement is performed independently and in parallel for each community c identified during the coarsening phase. We augment the original user-level subgraph of each community adding a community representative node r_c, containing the aggregated label information from the previous propagation. By connecting this representative node to all users in the community, we build a subgraph $G_{c,aug}$ that integrates both local (user-level) and global (community-level) information. Finally, we apply LPA over each subgraph to refine the recommendations, ensuring a more accurate and personalized output $\hat{\mathbf{Y}}_{final}$. The presented framework is detailed in Algorithm 1.

5.1.3 Experiments

We conduct comprehensive experiments to demonstrate the effectiveness of the proposed method. Specifically, we aim to answer the following research questions:

- **Effectiveness (RQ1):** How does our graph coarsening, compare in terms of recommendation quality, to (i) full-graph label propagation and (ii) alternative coarsening methods?
- **Structural integrity (RQ2):** To what extent do different coarsening strategies preserve the topological properties (e.g. community structure, clustering and degree distribution) of the original graph?
- **Efficiency and Scalability (RQ3):** How does the proposed approach perform in terms of computational efficiency and scalability when compared to full-graph propagation and other coarsening approaches?

Algorithm 1 Efficient Recommendations via Graph Coarsening and Label Propagation

Input: Full user graph $G = (V, E, \mathbf{Y})$; Coarsening business rules $\mathcal{R}$; Propagation algorithm $PA_{1st} \in \{\text{LPA, GNN}\}$
Output: Personalized recommendation scores $\hat{\mathbf{Y}}_{final}$ for all users in V
Stage 1: Graph Coarsening
$E_{aux} \leftarrow \emptyset$ {Initialize auxiliary edges for coarsening}
for all user pair $(u, v) \in V \times V$ **do**
  if SatisfiesRules$((u, v), \mathcal{R})$ **and** MeetsConstraints$((u, v))$ **then**
    $E_{aux} \leftarrow E_{aux} \cup \{(u, v)\}$
  end if
end for
$\mathbf{C} \leftarrow$ ConnectedComponents(V, E_{aux}) Generate community mapping matrix
$G' = (V', E', \mathbf{Y'}) \leftarrow$ CoarsenGraph$(G, \mathbf{C})$ {Create coarsened graph}
Stage 2: Inter-Community Propagation
if PA_{inter} is GNN **then**
  $G_h \leftarrow$ CreateHeterogeneousGraph(G')
  $\mathbf{h}^{(L)}_{comm}, \mathbf{h}^{(L)}_{item} \leftarrow$ GraphSAGE(G_h)
  $\hat{\mathbf{Y}}'_{scores} \leftarrow$ ComputeScores$(\mathbf{h}^{(L)}_{comm}, \mathbf{h}^{(L)}_{item})$ {Scores from GNN embeddings}
else
  $\hat{\mathbf{Y}}'_{scores} \leftarrow$ LPA$(G', \mathbf{Y'})$ {Scores from LPA}
end if
Stage 3: Intra-Community Propagation (in parallel)
for all community c with super-node $v'_c \in V'$ **do**
  $G_c \leftarrow$ ExtractSubgraph(G, c) {Get original users and edges for community c}
  $r_c \leftarrow$ CreateRepresentativeNode$(\hat{\mathbf{Y}}'_{scores}[c])$ {Node with global info}
  $G_{c,aug} \leftarrow$ AugmentSubgraph(G_c, r_c) {Connect r_c to all users in G_c}
  $\hat{\mathbf{Y}}_{final} \leftarrow$ LPA$(G_{c,aug})$ {Refine recommendations locally}
end for
return $\hat{\mathbf{Y}}_{final}$

- **Ablation (RQ4):** Which LPA design choices (number of propagation layers L and propagation hyperparameter α) drive performance and efficiency gains?

Experimental Setup

Dataset. We employ a real-world industrial dataset, containing more than 13 million users, collected from marketing campaigns run by TIM, an Italian telecommunications company, containing more than 13 million users. Specifically, it captures instances where customer care agents, during phone calls, suggest multiple products or services to a customer in a single engagement. This characteristic is central in the evaluation process, as discussed in Section 5.1.3, because agents usually present a list of recommendations.

The dataset spans from January to September 2024 and captures user interactions with promotional offers. To ensure a realistic evaluation, we split the data chronologically: the last month is used as test set, and the previous months as training set. Table 5.1 provides comprehensive details about train and test split statistics.

The item catalog within our industrial recommendation dataset covers a diverse range of offers:

Table 5.1: Summary of Dataset Statistics

	Train	Test
Unique users buying	2,552,477	736,648
Total items bought	4,818,165	1,070,679
% of users with ≥ 1 items bought	18.49%	5.34%
% of users with ≥ 2 items bought	7.34%	1.42%
% of users with ≥ 3 items bought	3.62%	0.53%

1st PA	Model	MAP	@3			@5			@10			@20		
			P	NDCG	R	P	NDCG	R	P	NDCG	R	P	NDCG	R
LPA	*Users*	0.0220	0.0155	0.0260	0.0275	0.0112	0.0275	0.0318	0.0062	0.0282	0.0339	0.0031	0.0282	0.0341
	Unique	0.0181	0.0105	0.0191	0.0207	0.0080	0.0211	0.0260	0.0051	0.0236	0.0330	0.0028	0.0243	0.0357
	Louvain	0.0234	0.0134	0.0270	0.0268	0.0084	0.0270	0.0281	0.0043	0.0271	0.0285	0.0021	0.0271	0.0286
	Ours	**0.0281**	**0.0182**	**0.0313**	**0.0332**	**0.0138**	**0.0341**	**0.0405**	**0.0084**	**0.0367**	**0.0481**	**0.0043**	**0.0372**	**0.0498**
GNN	*Users*	OOM	OOM	OOM	OOM	OOM	OOM	OOM	OOM	OOM	OOM	OOM	OOM	OOM
	Unique	0.0305	0.0172	0.0318	0.0321	0.0124	0.0352	0.0393	0.0078	0.0382	0.0455	0.0043	0.0394	0.0491
	Louvain	0.0390	0.0181	0.0373	**0.0489**	0.0121	0.0392	**0.0532**	0.0095	0.0498	**0.0859**	**0.0058**	**0.0543**	**0.1026**
	Ours	**0.0434**	**0.0312**	**0.0552**	0.0461	**0.0193**	**0.0528**	0.0471	**0.0098**	**0.0524**	0.0478	0.0049	0.0524	0.0478

Table 5.2: Performance metrics averaged over 3 runs for baselines. Standard deviation is not reported due to space constraints, but is always less than 0.001. PA = Propagation Algorithm, OOM = Out Of Memory. **Bold** indicates the best metric with those parameters, underlined are second best.

- **Telecommunications Services:** including voice and Wi-Fi plans.
- **TV & Streaming Subscriptions:** such as DAZN, Netflix, Amazon Prime, Disney+, and TIMVISION.
- **Promotional Discounts:** such as percentage-off deals or limited-time offer.
- **Smart Home Devices:** for example, Google Nest or security cameras.
- **Network Equipment:** including routers and related hardware.

The nature of offers allowing for collective access or use by a group (such as a family) has a dual implication for our recommendation task. Firstly, it can lead to lower individual adoption rates and contribute to dataset sparsity. Secondly, this makes identifying family members (or communities) particularly valuable for tailoring recommendations, as it's often redundant or ineffective to recommend the same item to multiple people in the same family.

Each of the 561 unique items represents a distinct promotional offer or service package. We focus on purchases as the primary signal of user intent. This naturally leads to a sparse dataset, with an overall interaction density of 0.06% in the training set. This sparsity highlights one of the challenges of industrial recommendation.

Baseline Models. We compare the proposed framework with three coarsening baselines:

- *Users*: the full user's graph, where each node represents an individual user and edges denote direct interactions.
- *Unique*: a coarsened version where super nodes are created based on unique label combinations.

- *Louvain*: a coarsened version where the Louvain algorithm [31] is used to partition the original graph into communities.

To ensure consistency in the evaluation, for all models, we follow the same methodology as in the proposed framework: PA is first applied to the coarsened graph, then LPA within each community.

Evaluation Metrics. To evaluate the effectiveness of our model, we investigate three areas. First, we measure the quality of the recommendation using standard ranking metrics: Precision@K, NDCG@K, Recall@K and Mean Average Precision (MAP). To evaluate the structural preservation of the coarsened graphs, we measure Degree Similarity (DS), Average Clustering Coefficient (AC), Connected Components Similarity (CCS), and Edge Density (ED), along with a comparison of the node count. This helps us determine if the coarsening process retains the key features of the original graph. The DS and CCS are computed using the formula $1/(1+d(x,y))$, where d is the Wasserstein distance, and x and y are the lists of node degrees or connected component sizes for the original and coarsened graphs, respectively. Finally, we compare runtime efficiency by recording LPA execution time, to show how scalable and fast our approach is when handling large graphs.

Given that customer care agents typically propose at least five items during these interactions, evaluating the effectiveness of their call recommendations requires to consider and focus on K=5. This is precisely why NDCG@5 is our primarily evaluation metric. It allows us to measure how well the most relevant items are prioritized within the top 5 positions, which is crucial in a scenario where an agent offers a limited set of options during a brief interaction.

Effectiveness (RQ1)

Table 5.2 reports the results for the complete set of metrics of our proposed framework against several baselines. The comparison includes our coarsening business-driven strategy and alternative methods, each evaluated with LPA or GNN as the first propagation algorithm, and the full-graph LPA baseline. A critical initial observation is the Out Of Memory (OOM) encountered when attempting to apply the GNN to the original full graph, underlining the necessity of graph coarsening. All of these use the best configuration found (see Section 5.1.3 for LPA). Since customer agents typically recommend up to five offers from the recommendation list, we set the primary evaluation goal to $K = 5$.

When LPA is applied as the first PA, our approach achieves the highest recommendation quality consistently across all metrics. Notably, our method improves NDCG@5 by 24% over the full-graph propagation baseline.

Using a GNN as the first PA notably enhances performance, particularly when combined with our coarsening strategy. Our approach achieves overall best results across several metrics. It improves the $NDCG@5$ by 54% over its LPA counterpart and by 35% over the GNN trained on *Louvain* coarsened graph. The latter generally stands as the second best method, achieving always highest recall and best metrics @20.

In summary, these results confirm that graph coarsening is indispensable for applying GNNs to large industrial-scale graphs. Moreover, *Ours* business-driven coarsening approach is consistently having high performance with both (LPA and GNN) PA. GNN in the first propagation step achieves the highest recommendation quality, significantly outperforming LPA approaches.

Structural Integrity (RQ2)

Graph	DS	AC	CCS	ED	NN
Users	-	$6.287 \cdot 10^{-7}$	-	$2.839 \cdot 10^{-7}$	13.80
Unique	0.263	0.327	$2.820 \cdot 10^{-4}$	$3.312 \cdot 10^{-5}$	0.14
Louvain	**0.587**	0.984	$\mathbf{5.152 \cdot 10^{-2}}$	$1.976 \cdot 10^{-6}$	0.49
Ours	0.485	$\mathbf{1.070 \cdot 10^{-6}}$	$1.647 \cdot 10^{-2}$	$\mathbf{9.105 \cdot 10^{-7}}$	4.27

Table 5.3: Comparison of graph properties across different methods: degree similarity (DS), average clustering (AC), connected components similarity (CCS), edge density (ED), and number of nodes (NN) in millions.

Coarsening		First Propagation				Second LPA (per community)		
Approach	Runtime	Nodes (M)	Edges (M)	LPA Runtime	GNN Runtime	Nodes	Edges	LPA Runtime
Users	**0s**	13.80	9.65	17m41s±30s	*OOM*	-	-	-
Unique	3min53s±40s	0.14	0.27	**5m07s±50s**	**10m39s±24s**	99.57	463.57	4457ms±264ms
Louvain	2h12min±2m	0.49	0.26	6m47s±30s	11m02s±35s	29.16	32.65	1571ms±423ms
Ours	**34s±5s**	4.27	6.36	9m01s±47s	12*m*59*s*±14s	4.23	7.68	**358ms±24ms**

Table 5.4: Comparison of graph statistics across the three steps, including number of nodes, edges and runtime. The best runtime for each step is highlighted in **bold**.

Table 5.3 reports the five structural metrics for each coarsening strategy compared to the original user–item graph. Together, these metrics help quantify how faithfully each method preserves global and local structural properties.

Our approach yields an AC that most closely matches that of the original graph. The low clustering coefficient similarity reflects the sparse and loosely connected nature of user-item interactions within the generated communities, which is consistent with typical user behavior on large-scale platforms. This low clustering coefficient aligns with our goal of preserving realistic connectivity patterns without introducing artificial density. Similarly, the edge density (ED) of our coarsened graph remains low, indicating that the compression process preserves the original structure's natural sparsity, avoiding excessive over-connection among super nodes. At the same time, our method achieves moderate compression (in terms of node count), avoiding the extreme reductions that may compromise structural fidelity.

The *Louvain* method produces a more compact graph than ours, and it exhibits the highest DS and CCS scores, indicating that it doesn't distort the original component structure and degree distribution heavily. Despite a low ED value — suggesting nodes with poor connectivity — the AC deviates greatly from the original graph's, limiting *Louvain* effectiveness.

On the other hand, the *Unique* coarsening strategy achieves the most aggressive reduction in node count, resulting in a very compact graph. It yields the lowest DS and CCS, indicating that it doesn't preserve some macro-structural properties of the original graph. However, this comes at the cost of excessive edge density and lowest clustering coefficient similarity, both of which suggest overly tight communities that may inhibit meaningful diffusion and lead to unstable performance (as observed in RQ1).

Overall, our coarsening framework strikes the best balance between graph compression and structural fidelity.

Efficiency and Scalability (RQ3)

Table 5.4 summarizes, for each method, the graph size (nodes and edges) at each propagation step and the corresponding runtimes for both coarsening and propagation steps. The full-graph baseline (*Users*), which involves performing a single propagation on an original graph comprising roughly 14 million nodes and 10 million edges, taking significantly longer (>17 minutes) than all the other methods. Crucially, attempting to run GNN propagation on this full graph results in Out Of Memory (OOM), highlighting the necessity of coarsening for GNN-based approaches.

By contrast, our business-driven coarsening achieves a minimal coarsening time (34s), thanks to lightweight heuristics that can be precomputed offline. The resulting coarsened graph contains 4 million nodes and 6 million edges, and undergoes two LPA propagation steps in nine minutes: the first on the reduced graph and the second in parallel across smaller user-level subgraphs. On the other side, GNN propagation is the slowest with approximately 13 minutes.

The *Unique* strategy completes the coarsening process in around 4 minutes), producing the smaller graph (140 thousand nodes and 270 thousand edges). While its first propagations are the fastest (∼5 minutes LPA, ∼10 minutes GNN), this advantage is reversed in the second step, where it takes over 4 seconds per graph, the longest of all methods, due denser communities. While Louvain coarsening produces well-balanced partitions, it is the slowest to run, taking over two hours. Though its propagation steps are marginally faster than ours, the upfront cost dominates its total runtime and, ultimately, our approach remains competitive in terms of total LPA and GNN runtime while achieving superior recommendation performance, as shown in Table 5.2.

From a theoretical point of view, from [215, 251, 264], LPA algorithm has a running time of $\mathcal{O}(TM)$, where T is the number of iterations and M is the number of edges in the graph. The number of iterations can either be fixed or left to convergence. However, no theoretical proof of LPA convergence time has been found [251] except for special cases. Empirical studies suggest that a few iterations are usually sufficient to stabilize the labels [215]. Reported complexity also depends on LPA variant, with studies citing $\mathcal{O}(Nlog^2N)$ [57] and $\mathcal{O}(M+N)$ [275]. GNNs typically exhibit higher computational complexity: with L layers, a graph with N nodes, and M edges, the complexity is $\mathcal{O}(L\cdot(M+N))$ [281]. This factor inherently make GNNs more resource-intensive compared to LPA.

In our case, the first step, whether LPA or GNN, is expected to be the most computationally intensive due to the large scale of the coarsened graph, which contains millions of nodes and edges. The GNN runtimes clearly demonstrate a higher computational demand than LPA on the same coarsened graph. Conversely, the second LPA step operates on much smaller community graphs, containing only tens or hundreds of nodes, making it computationally light. An additional advantage of our approach is that the second LPA step can be efficiently parallelized. As it operates at the user level, it can be performed online at inference time, while the first propagation step can be precomputed offline. This design ensures that our method remains scalable and operates within sub-second inference times.

Ablation (RQ4)

To analyze the effect of the main hyperparameters on recommendation accuracy, we perform an ablation study by varying the number of propagation layers $L \in \{1, 3, 5\}$, and damping parameter $\alpha \in \{0.1, 0.5, 0.9\}$. Due to the computational complexity associated with GNN training, we conduct

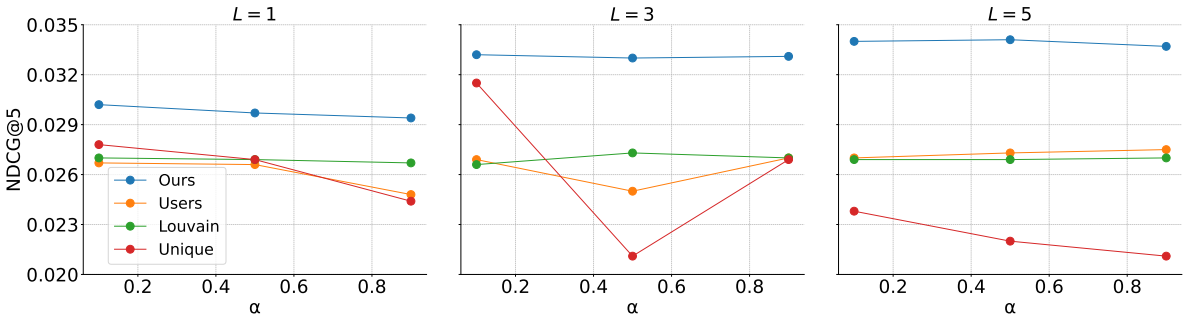


Figure 5.2: NDCG@5 values for all coarsening methods, varying the number of layers L, with corresponding α values indicated.

this ablation study specifically using LPA as our propagation algorithm. For each configuration, we evaluate all coarsening strategies against the full graph baseline.

As shown in Table 5.5, our coarsening approach consistently outperforms all baselines across different hyperparameter settings. In particular, it shows a clear upward trend in performance as the number of propagation layers L increases, while the performance of the other models remain largely stable. This is also observable in Fig. 5.2. Moving from a single propagation step to five, yields and average NDCG@5 gain of over 17%, regardless of the chosen α.

In contrast, the alternative coarsening methods and full-graph propagation remain largely insensitive to changes in L or α. As can be seen in Fig. 5.2, their NDCG@5 curves are essentially flat, indicating that deeper or more aggressive propagation does not result in significant improvements to either the original or differently coarsened graphs. The only exception is *Unique*, which shows a high sensitivity to hyperparameter variation - its performance deteriorates at $L = 5$ and fluctuates unpredictably with α when $L = 3$.

Based on these results, we set the optimal hyperparameters to $L = 5$ and $\alpha = 0.5$, as this configuration gives the highest NDCG@5.

5.1.4 Conclusion

In this paper, we have introduced a novel graph coarsening framework for industrial-scale recommendation systems that incorporates business-driven rules to improve scalability. The framework involves a two-stage propagation: inter-community propagation (LPA or GNN-based) is applied to the coarsened graph to efficiently spread information across communities, and intra-community propagation refines the initial recommendations within each community. This method effectively utilizes user interactions to enhance the recommendation of family offers and provides more scalable recommendation updates, which is crucial when user and community interactions are subject to frequent changes.

Through extensive evaluation, we have shown that our approach outperforms baseline models in terms of recommendation quality, particularly for key industrial metrics at rank $K = 5$. Additionally, it achieves improved scalability while preserving important topological properties of the original full graph. These results highlight the practical applicability of our method in real-world marketing campaigns, where maintaining both high quality recommendations and scalability are critical. The LPA-only pipeline is immediately deployable in low-latency environments, while the GNN-enhanced option suits settings where higher predictive performance justifies extra training

L	α	Model	MAP	@3			@5			@10			@20		
				P	NDCG	R	P	NDCG	R	P	NDCG	R	P	NDCG	R
1	0.1	*Users*	0.0213	0.0152	0.0256	0.0267	0.0109	0.0267	0.0304	0.0059	0.0272	0.0319	0.0030	0.0272	0.0320
		Unique	0.0240	0.0134	0.0243	0.0266	0.0109	0.0278	**0.0354**	**0.0069**	**0.0309**	**0.0442**	**0.0036**	**0.0315**	**0.0462**
		Louvain	0.0235	0.0132	0.0270	0.0269	0.0084	0.0270	0.0282	0.0043	0.0270	0.0287	0.0021	0.0271	0.0287
		Ours	**0.0243**	**0.0168**	**0.0287**	**0.0302**	**0.0121**	**0.0302**	0.0347	0.0066	**0.0309**	0.0368	0.0033	0.0309	0.0369
	0.5	*Users*	0.0212	0.0152	0.0255	0.0266	0.0109	0.0266	0.0302	0.0060	0.0271	0.0317	0.0029	0.0271	0.0319
		Unique	0.0164	0.0158	0.0245	0.0263	0.0116	0.0269	0.0327	0.0054	0.0295	0.0297	0.0029	0.0301	0.0421
		Louvain	0.0234	0.0132	0.0269	0.0268	0.0084	0.0269	0.0282	0.0043	0.0270	0.0286	0.0021	0.0270	0.0286
		Ours	**0.0239**	**0.0166**	**0.0282**	**0.0297**	**0.0120**	**0.0297**	**0.0342**	**0.0065**	**0.0304**	**0.0363**	**0.0033**	**0.0304**	**0.0364**
	0.9	*Users*	0.0215	0.0122	0.0248	0.0244	0.0077	0.0248	0.0256	0.0039	0.0248	0.0261	0.0019	0.0248	0.0261
		Unique	0.0138	0.0120	0.0218	0.0239	0.0095	0.0244	0.0303	0.0063	0.0272	0.0333	0.0027	0.0278	0.0405
		Louvain	0.0233	0.0132	0.0269	0.0264	0.0082	0.0267	0.0274	0.0041	0.0267	0.0278	0.0021	0.0268	0.0279
		Ours	**0.0236**	**0.0165**	**0.0282**	**0.0294**	**0.0118**	**0.0294**	**0.0333**	**0.0064**	**0.0299**	**0.0349**	**0.0032**	**0.0299**	0.0350
3	0.1	*Users*	0.0215	0.0152	0.0257	0.0269	0.0110	0.0269	0.0307	0.0060	0.0275	0.0326	0.0030	0.0275	0.0327
		Unique	0.0266	0.0149	0.0274	0.0298	0.0122	0.0315	**0.0399**	**0.0079**	**0.0353**	**0.0505**	**0.0040**	**0.0358**	**0.0521**
		Louvain	0.0232	0.0131	0.0266	0.0264	0.0082	0.0266	0.0277	0.0042	0.0266	0.0282	0.0021	0.0266	0.0282
		Ours	**0.0271**	**0.0179**	**0.0307**	**0.0326**	**0.0135**	**0.0332**	0.0393	0.0078	0.0351	0.0448	**0.0040**	0.0354	0.0458
	0.5	*Users*	0.0215	0.0123	0.0248	0.0247	0.0078	0.0250	0.0263	0.0040	0.0251	0.0268	0.0020	0.0251	0.0269
		Unique	0.0175	0.0105	0.0184	0.0205	0.0085	0.0211	0.0273	0.0054	0.0237	0.0345	0.0028	0.0242	0.0363
		Louvain	0.0238	0.0134	0.0273	0.0269	0.0084	0.0273	0.0283	0.0043	0.0273	0.0289	0.0022	0.0273	0.0289
		Ours	**0.0270**	**0.0179**	**0.0308**	**0.0325**	**0.0132**	**0.0330**	**0.0386**	**0.0077**	**0.0348**	**0.0437**	**0.0039**	**0.0350**	**0.0445**
	0.9	*Users*	0.0216	0.0153	0.0257	0.0270	0.0110	0.0270	0.0309	0.0060	0.0276	0.0327	0.0030	0.0276	0.0329
		Unique	0.0231	0.0125	0.0238	0.0262	0.0098	0.0269	0.0339	0.0062	0.0298	0.0421	0.0032	0.0303	0.0438
		Louvain	0.0234	0.0135	0.0271	0.0268	0.0085	0.0270	0.0281	0.0043	0.0271	0.0285	0.0022	0.0271	0.0286
		Ours	0.0182	**0.0179**	**0.0308**	**0.0327**	**0.0134**	**0.0331**	**0.0391**	**0.0078**	**0.0350**	**0.0446**	**0.0040**	**0.0352**	**0.0454**
5	0.1	*Users*	0.0215	0.0153	0.0257	0.0270	0.0110	0.0269	0.0309	0.0060	0.0276	0.0328	0.0030	0.0276	0.0329
		Unique	0.0199	0.0120	0.0211	0.0234	0.0095	0.0240	0.0308	0.0059	0.0264	0.0378	0.0031	0.0272	0.0403
		Louvain	0.0235	0.0133	0.0270	0.0269	0.0084	0.0270	0.0283	0.0043	0.0270	0.0287	0.0021	0.0271	0.0288
		Ours	**0.0282**	**0.0182**	**0.0314**	**0.0334**	**0.0138**	**0.0341**	**0.0405**	**0.0084**	**0.0368**	**0.0481**	**0.0043**	**0.0373**	**0.0499**
	0.5	*Users*	0.0219	0.0155	0.0260	0.0274	0.0112	0.0273	0.0315	0.0062	0.0280	0.0335	0.0031	0.0281	0.0336
		Unique	0.0194	0.0111	0.0205	0.0221	0.0086	0.0227	0.0280	0.0054	0.0252	0.0349	0.0029	0.0260	0.0378
		Louvain	0.0234	0.0132	0.0269	0.0264	0.0083	0.0268	0.0276	0.0042	0.0268	0.0280	0.0021	0.0269	0.0281
		Ours	**0.0283**	**0.0182**	**0.0314**	**0.0333**	**0.0138**	**0.0342**	**0.0406**	**0.0084**	**0.0368**	**0.0482**	**0.0043**	**0.0373**	**0.0499**
	0.9	*Users*	0.0218	0.0154	0.0259	0.0273	0.0111	0.0273	0.0315	0.0062	0.0279	0.0335	0.0031	0.0280	0.0337
		Unique	0.0175	0.0101	0.0184	0.0202	0.0077	0.0204	0.0253	0.0050	0.0231	0.0330	0.0027	0.0238	0.0357
		Louvain	0.0233	0.0132	0.0268	0.0267	0.0083	0.0268	0.0280	0.0042	0.0269	0.0284	0.0021	0.0269	0.0285
		Ours	**0.0282**	**0.0181**	**0.0312**	**0.0332**	**0.0138**	**0.0340**	**0.0406**	**0.0085**	**0.0368**	**0.0487**	**0.0044**	**0.0374**	**0.0505**

Table 5.5: Performance metrics with baselines for different layers and alpha values. **Bold** and *underlined* indicate the best and second-best metric with those parameters, respectively.

cost. Future work will investigate adaptive coarsening techniques to further optimize performance in dynamic environments.

5.1.5 Related Work

Graph Coarsening

Graph coarsening is widely used across industries primarily because it offers a practical way to distill massive graphs into manageable representations, enabling faster computations and clear understanding without losing the underlying connections.

Foundational to this is the multi-level paradigm posed by METIS [137], which uses techniques like heavy-edge matching to recursively simplify graphs for high-quality partitioning. The Louvain method [31] performs coarsening by grouping densely connected nodes into communities. This approach has been further refined by algorithms like the Leiden method [252]. Lean Algebraic Multigrid (LAMG) [175] leverages advanced numerical techniques for coarsening that maintains spectral fidelity, while [176] introduced frameworks for spectrally approximating large graphs with smaller counterparts.

However, the most recent advancements are concentrated in learnable graph coarsening for

Graph Neural Networks (GNNs). [296] introduced a differentiable method to learn hierarchical cluster assignments for pooling nodes. Starting from this, many alternative approaches such as the node selection strategy [90], Self-Attention Graph Pooling (SAGPool) [156] has been implemented.

Label Propagation Algorithms

LPAs are extensively utilized in industry because they offer a remarkably efficient and intuitive method for inferring missing information or uncovering latent structure within large-scale graphs. In original LPA, [215], nodes in the network keep copying the most common label from their neighbors.

More recently, GNNs have become very popular. GNNs can be seen as a smart, learnable way to do label propagation. The general idea of message passing [93], or attention-based GNN [254], also work by passing information between neighbors. These deep learning models have achieved top results on many graph tasks. However, in [119], it is demonstrated that classical label propagation can work as well as, or even better than, complex GNNs. This is especially true when there isn't much labeled data, or when speed is fundamental. This key finding shows that simple propagation methods are still very powerful and efficient.

GNNs in Industrial Recommender Systems

GNNs have become fundamental in modern industrial recommender systems. Pinterest's Pinsage [295] was a seminal work demonstrating how GNNs, leveraging random walks and graph convolutions, could learn recommendations for billions of items. Alibaba further pushed the boundaries with algorithms [261, 262] designed to handle large-scale heterogeneous graphs. Furthermore, foundational advancements in GNN architectures, such as Graph Attention Networks (GAT) [254], have provided powerful mechanism for modeling complex relational data, significantly influencing the subsequent design of GNN-based recommendations systems across the industry. Leveraging the power of such graph learning architectures, Linkedin has developed models like LiGNN [33], which refines recommendation within its vast graph.

5.2 Do We Need Deep Models or Just Better Graphs? Scalable Link Prediction through Simulated Annealing in Graph Structure Learning

Graph neural networks (GNNs) have emerged as a highly effective approach for recommender systems thanks to their ability to leverage the relational structure between users and items to propagate preference signals. However, in applied recommendation settings, the interaction data is scarce, leading to a sparse, noisy and incomplete graph. This exacerbates phenomena such as over-squashing and over-smoothing, thereby reducing the quality of GNN recommendations.

To address this issue, we propose a scalable structure learning algorithm, tailored for production environments, that optimizes graph topology to amplify weak signals while minimizing memory budget. Our method, Simulated Annealing for Graph Structure Learning `SA-GSL`, partitions the full graph into random subgraphs and employs s Simulated Annealing (SA) to simultaneously refine adjacency matrices during GNN training. At each epoch, the SA step generates candidate edge sets

that balance two objectives: maximizing NDCG, efficiently estimated by label propagation, and minimizing global and local connectivity. The GNN is trained end-to-end on each subgraph evolving structure, leveraging the SA-refined edges without requiring deep architectures, thereby mitigating over-smoothing and improving scalability.

On a large-scale industrial dataset, `SA-GSL` improves NDCG@5 by 57% compared to the raw bipartite graph, and by 25% over other structure learning baselines. Ablation studies demonstrate the critical roles of graph density augmentation, number of GNN layers, and the scalability of the proposed algorithm. Lastly, we evaluate `SA-GSL` on several benchmark datasets, and the code implementation is publicly available on our GitHub repository (`https://github.com/ashba93/sa-gsl`).

5.2.1 Introduction

Recommender systems (RSs) play a central role in modern digital platforms, influencing user experience and engagement across domains such e-commerce [269], social media [247, 249], and content streaming [63, 217].

The increase in large-scale interaction logs has led to the adoption of graph-based models, particularly Graph Neural Networks (GNNs) [88], which are highly effective in a variety of tasks such as node classification [139], graph classification [287], and link prediction [304]. GNNs have also emerged as a dominant paradigm in recommendation tasks due to their ability to propagate preference signals over graph-structured data [280].

However, real-world interaction data is often sparse and noisy, resulting in graphs that hinder the effectiveness of GNNs' message passing mechanism [281]. In real-world scenarios lacking an explicit graph, structure must be inferred from raw interaction data [83], exacerbating the issue in highly sparse user-item graphs [109].

To compensate for poor connectivity, one might stack many GNN layers; however, deeper architectures can lead to over-smoothing, where node embeddings become indistinguishable [45], and over-squashing, where distant signals are compressed and lost [66]. Furthermore, scalability is another critical constraint: the receptive field expands with each additional layer, increasing memory

and runtime requirements, especially for graphs containing millions of nodes and edges, which can quickly exhaust memory and computational resources [23, 69, 235].

These limitations raise a key question: *do we need ever-deeper GNNs, or simply better graphs?* We hypothesize that an optimized graph topology, one that amplifies weak signals while preserving sparsity, can enable shallow, scalable GNNs to match or exceed deep models. To this end, we reframe the problem of recommendation as graph structure learning and introduce a novel, scalable algorithm based on Simulated Annealing (SA). Unlike prior GSL approaches that rely on continuous relaxations of the adjacency matrix [55, 127, 313], SA operates directly in the discrete space of edge additions and removals, naturally enforcing sparsity and degree constraints while retaining the ability to escape local minima. This makes it particularly well-suited for refining sparse, noisy user–item graphs in recommendation. Moreover, SA is well-suited for adjacency matrix completion, as it effectively handles optimization problems with large and complex search spaces, particularly when numerous local optima are present and traditional gradient-based methods might get stuck.

Our approach, Simulated Annealing for Graph Structure Learning `SA-GSL`, partitions the full graph into random subgraphs to limit memory usage and enable parallel optimization. Within each epoch, SA is used to iteratively select edges that optimize graph sparsity and recommendation effectiveness. The latter is efficiently evaluated via Label Propagation Algorithm [215]. After each step of SA, a GNN is continuously trained on each subgraph. This process is repeated for a predefined number of steps.

On a large-scale industrial dataset, `SA-GSL` achieves a 57% improvement in NDCG@5 compared to the raw interaction graph and a 25% improvement compared to existing structure-learning baselines. Ablation studies emphasise the critical roles of density augmentation and shallow design. This helps avoid deep architectures and mitigate over-smoothing and over-squashing, achieving high accuracy within tight computational budgets.

In summary, our contributions are:

- To the best of our knowledge, we propose the first SA-based algorithm that directly optimizes graph topology for recommendation under memory constraints.
- We demonstrate that shallow GNNs, combined with well-learned structures, outperform their deeper counterparts, avoiding over-smoothing and over-squashing.
- We present a scalable, parallelizable end-to-end algorithm that integrates fast SA search, label propagation evaluation, and shallow GNN training, validated on a massive real-world dataset.

We organize the paper as follows: Section 5.2.2 presents background on graph theory, SA, and GNNs for recommendations; Section 5.2.3 reviews the related work; Section 5.2.4 describes our methodology; Section 5.2.5 details the experimental setup; and Section 5.2.6 concludes with a summary of our findings and future directions. Appendix comprehends the pseudocode and the benchmark with state-of-the-art algorithms on common benchmarks.

5.2.2 Background and Notation

Graphs in Recommendation

Let $G{=}(V,E)$ be an undirected graph, where V is a finite set of nodes (or vertices) and $E \subseteq V \times V$ is the set of edges. In recommender systems, $V = U \cup I$ is the union of user nodes U and item

nodes I, while an edge $(u, i) \in E \subseteq U \times I$ corresponds to an observed interaction (e.g. click, rating, or purchase). We represent a bipartite graph G by its adjacency matrix $\mathbf{A} \in \mathbb{R}^{|U|\times|I|}$:

$$\mathbf{A}_{uv} = \begin{cases} 1 & \text{if } (u,v) \in E, \\ 0 & \text{otherwise} \end{cases} \tag{5.1}$$

The degree of a node $v \in V$ is defined as the number of neighbours of v:

$$d(v) = \sum_{u \in V} \mathbf{A}_{vu} \tag{5.2}$$

The density of a graph measures how many edges exist relative to the total number of possible edges. For a bipartite undirected graph, with no self-loops, density is defined as:

$$density(G) = \frac{2|E|}{|U||I|} \tag{5.3}$$

In real-world recommendation systems, there can also be (u_1, u_2) edges that capture similarities between users u_1 and u_2.

Simulated Annealing

SA is a global optimisation meta-heuristic inspired by the metallurgical annealing process in solid-state physics [42], where a material is heated to allow its particles to move freely among high-energy states and then slowly cooled so they can settle into a low-energy configuration. SA translates this physical intuition into an algorithm for global optimization over a discrete solution space.

Let $\mathcal{X}$ be a finite set of candidate solutions (states), and let $E : \mathcal{X} \to \mathbb{R}$ be an energy (cost) function to maximize (or minimize). Starting from an initial state x_0, SA performs the following steps at each iteration t: (1) samples a neighbor $x' \in N(x_t)$, where $N : \mathcal{X} \to 2^{\mathcal{X}}$ defines the neighborhood structure; (2) computes the energy difference $\Delta E = E(x') - E(x_t)$; (3) accepts x' as the new state x_{t+1} with probability:

$$P(x_t \to x') = \begin{cases} 1 & \text{if } \Delta E \geq 0, \\ \exp(\frac{\Delta E}{T_t}) & \text{otherwise} \end{cases} \tag{5.4}$$

where $T_t > 0$ is the temperature at iteration t, which is gradually decreased according to a cooling schedule. This probability defines a time-inhomogeneous Markov chain over $\mathcal{X}$, where transitions are biased toward lower-cost states, but suboptimal moves are still possible. This probabilistic behaviour, controlled by the Boltzmann factor $\exp(\frac{\Delta E}{T_t})$, allows the algorithm to escape local minima. Under appropriate cooling schedules, such as logarithmic decay, the Markov chain converges in distribution to the global minimum of E.

GNNs for Recommendation

A GNN learns node embeddings by iteratively propagating and aggregating features from neighbours. Let $\{\mathbf{h}_v^0 = \mathbf{x}_v \in \mathbb{R}^d : v \in V\}$ be the initial node features and $\mathbf{h}_v^l$ be the embedding of v at layer l. In heterogeneous recommendation graphs, edges have types (e.g. user–item, user–user). We

write each edge as $e = (u, \tau, v)$, where τ indexes the relation, and define a message function as:

$$m_{u \to v}^{(l,\tau)} = W_{\tau}^{(l)} h_u^l \tag{5.5}$$

where $\mathbf{W}_{\tau}^{(l)} \in \mathbb{R}^{d \times d}$ is a learnable matrix. These messages are then aggregated at the target node v via a permutation-invariant operator (e.g sum or mean):

$$\hat{\mathbf{h}}_v^l = AGG(\{m_{u \to v}^{(l,e)} : (u, \tau, v) \in E\}). \tag{5.6}$$

Then a non-linear operation function σ is applied to the aggregated message and the node's previous state:

$$\mathbf{h}_v^{(l+1)} = \sigma(\mathbf{W}_0^{(l)} \mathbf{h}_v^{(l)} + \hat{\mathbf{h}}_v^{(l)}), \tag{5.7}$$

where $\mathbf{W}_0^{(l)} \in \mathbb{R}^{d \times d}$ is another learnable matrix. After L layers, each user u and item i obtains final embeddings $\mathbf{h}_u^L$ and $\mathbf{h}_i^L$. Finally, a link prediction score is computed for each candidate (u, i):

$$y_{u,i} = \mathbf{h}_u^{(L)\top} \cdot \mathbf{h}_i^{(L)}, \tag{5.8}$$

These scores are then ranked to produce the top-K recommendations.

5.2.3 Related Work

Graph Structure Learning

Graph Structure Learning (GSL) has emerged as a critical research area focused on inferring, refining, or generating optimal graph topologies directly from data, especially when the observed graph is incomplete, noisy, or sub-optimal for downstream tasks [321, 322]. The premise is that the performance of GNNs is intrinsically tied to the quality of the input graph structure; an inadequate graph can significantly hinder the propagation of information and the learning of effective node representations [281]. Early GSL approaches often relied on constructing graphs from feature similarities using techniques like k-Nearest Neighbors (k-NN) or Gaussian kernels, as seen in manifold learning methods like Laplacian Eigenmaps [24], or by learning graph Laplacians that promote signal smoothness across the graph [134]. More recent advancements have shifted towards end-to-end GSL, where the graph structure is learned jointly with the GNN parameters. Franceschi et al. (2019) [83] introduced a pioneering framework for learning discrete graph structures through bi-level optimization, allowing gradients to flow back to the adjacency matrix. Other notable works include iterative methods that alternate between refining the graph structure and training the GNN [127]. In the context of recommender systems, where user-item interaction graphs are often extremely sparse and potentially noisy, GSL offers a promising avenue to densify relevant connections or prune misleading ones. For instance, Wang et al. [266] in their work on Neural Graph Collaborative Filtering (NGCF) implicitly learn edge importance through message passing, while other works explicitly try to learn an underlying item-item or user-user graph to augment the sparse bipartite graph [277].

Simulated Annealing

SA, introduced in [140], is a versatile and powerful probabilistic optimization algorithm designed for finding near-global optima in large, often discrete and non-convex, search spaces. SA iteratively ex-

plores the solution space, accepting not only improvements but also, with a decreasing probability controlled by a temperature parameter, solutions that worsen the objective function. This characteristic allows SA to escape local optima, a common pitfall for many greedy optimization algorithms. SA has a rich history of application to classic combinatorial optimization problems on graphs. Its efficacy has been demonstrated for the Traveling Salesperson Problem [42, 253] and graph partitioning [128]. Beyond these canonical problems, SA has been employed for more direct graph structure optimization tasks. For example, in network design, SA can be used to search for network topologies that optimize for robustness, cost, or efficiency [2]. In bioinformatics, SA has been used to infer the structure of gene regulatory networks by optimizing a scoring function over a vast space of possible network configurations [124]. The core idea involves defining a state as a candidate graph structure, moves as operations like adding, deleting, or re-weighting edges, and an energy function that quantifies the quality of the current graph based on desired properties or its fit to observed data.

GNNs in Industrial Recommendation Systems

GNNs have become fundamental in modern industrial recommendation systems. Pinterest's PinSage [295] demonstrated how GNNs could leverage random walks and how graph convolutions could learn recommendations for billions of items. Alibaba developed a GNN-based recommender system for its e-commerce platforms. EGES (Enhanced Graph Embedding with Side Information) [261] and M2GRL (Multi-task Multi-view Graph Representation Learning) [262] were designed to handle extremely large-scale heterogeneous graphs on Taobao. Amazon utilizes GNNs for various product recommendation scenarios. For instance, P-Companion [105] models complementary product relationships using graph-based approaches to enhance recommendation diversity and utility. Google has a long history of using deep learning for recommendations [59], and subsequent advancements have seen the integration of GNNs to better model the complex relationships between users, videos, and channels, thereby improving candidate generation and ranking in their massive recommendation pipeline [164].

5.2.4 Methodology

Figs. 5.3 and 5.4 give an overview of our Simulated Annealing for Graph Structure Learning, `SA-GSL` algorithm, which is explained in more detail in the following sections and presented in detail in Section A. A key design choice in `SA-GSL` is to focus the SA-based structure search on the user–user subgraphs rather than the entire bipartite graph. In many applications, but especially for telecom and subscription services, user behaviour and preferences tend to remain stable over time, whereas the catalogue of items or offers can change rapidly (new plans, promotions or content being added or removed). By identifying robust and dense user similarities, we can amplify persistent preference signals and then reconnect the current item set via the original user–item edges. This separation reduces the search space by focusing on $|U|^2$ rather than $(|U| + |I|)^2$, and yields a topology that generalises across catalogue updates.

Graph Partitioning

To enable scaling to industrial-size graphs, we first partition the global graph $G = (U \cup I, E)$ into M disjoint user-induced subgraphs $\{G_k\}_{k=1}^{M}$. Each subgraph $G_k = (U_k \cup I_k, E_k)$ is constructed to

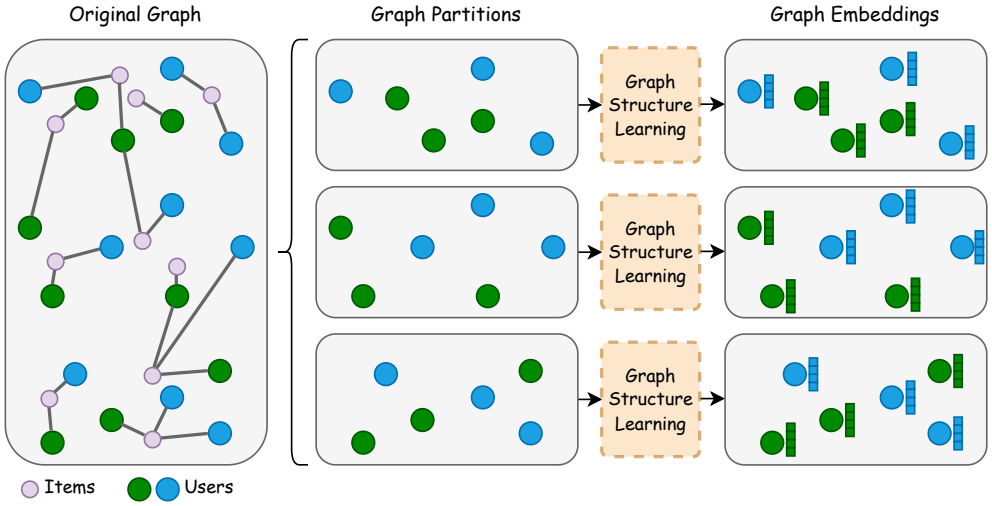


Figure 5.3: The figure illustrates the key components of the proposed algorithm. The initial set of users and their purchased items, i.e. the complete graph, is partitioned into user-subgraphs. Each subgraph is optimized separately via Graph Structure Learning. The resulting embeddings are used to produce recommendations.

contain approximately $\frac{|U|}{M}$ nodes, ensuring that any given user node resides in only one subgraph, i.e. $U_j \cap U_k = \emptyset \ \forall j, k = 1, \ldots, M, j \neq k$. We employ a stratified random allocation strategy to ensure that the global item popularity distribution is preserved within each subgraph. In this way, items that might be popular within one partition are also well-represented within others. This ensures that each subgraph remains representative of the full graph, allowing each GNN to recommend such items to users within its own subgraph while reducing memory and computing requirements per worker.

Simulated Annealing for GSL

Within each subgraph G_k user-user connections are represented by the adjacency matrix $\mathbf{A}_k \in \{0,1\}^{|U_k|\times|U_k|}$. Initially, $\mathbf{A}_k$ is a zero matrix. Considering one subgraph, we define the space of possible adjacency matrix as all binary symmetric adjacency matrices with at most $d_{\max}$ edges per user: $\mathcal{A}_k = \{\mathbf{A} \in \{0,1\}^{|U_k|\times|U_k|} : \mathbf{A}_{ij} = \mathbf{A}_{ji} \wedge \sum_{u\in U_k} d(u) \leq d_{\max}\}$.

The goal of SA is to discover an optimal adjacency $\mathbf{A}_k^*$ that maximizes a downstream task-specific loss function $E : \mathcal{A} \to \mathbb{R}$: $\mathbf{A}_k^* = \arg\max_{\mathbf{A}\in\mathcal{A}} E(\mathbf{A})$ The SA process defines a non-homogeneous Markov chain over $\mathcal{A}$ that explores the space of adjacency matrices using stochastic perturbations and a Boltzmann-inspired acceptance rule. At each iteration t, we perturb the current adjacency matrix $\mathbf{A}_t$ (losing the k subscript for simplicity), to generate a neighbor $\mathbf{A}_{t+1} \in N(\mathbf{A}_t)$, obtained by adding $k_{add} \in \mathbb{N}$ edges. We then evaluate an energy function:

$$E = \alpha \cdot r + \beta \cdot s - \gamma \cdot D - \lambda \cdot A_{avg} - \delta \cdot V, \tag{5.9}$$

where:

- $r = currAcc$ is the supervised training accuracy,
- $s = NDCG@5$ is the validation ranking quality computed on the recommendations produced by LP [215],

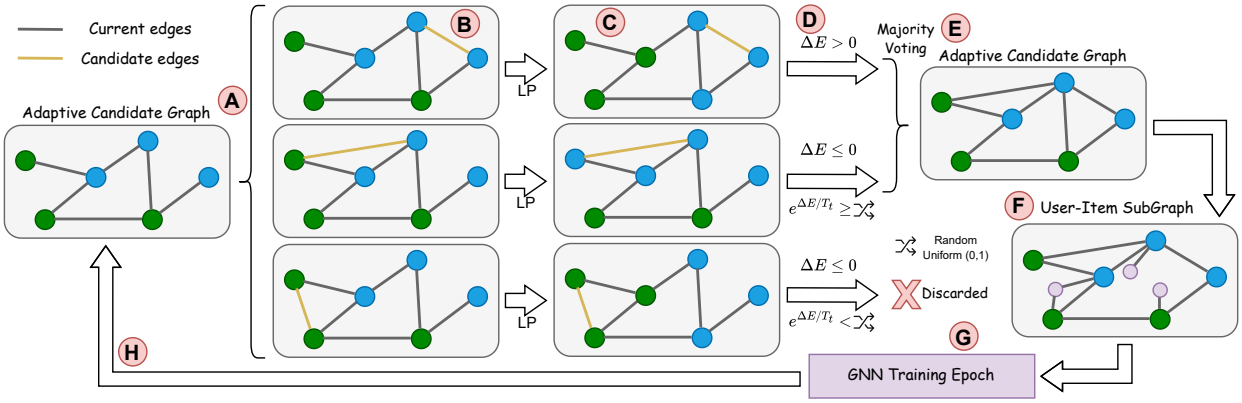


Figure 5.4: One step of the iterative SA-GNN optimization process is shown for a single graph partition: (A) First, an ensemble of graph views is created in parallel from the current candidate (sub)graph; (B) For each view, edges are added between the most similar nodes; (C) A Label Propagation (LP) step is performed; (D) The change in energy (ΔE) is evaluated, and edge modifications are accepted or rejected based on the ΔE and Metropolis-Hastings criterion; (E) A final optimized graph is constructed via majority voting across the ensemble; (F) The graph is restored as heterogeneous and the adjacency matrix is updated; (G) The GNN is trained for one epoch; (H) The entire procedure is repeated until convergence.

- $D = \frac{m}{n^2}$ is the edge density,
- $A_{avg} = \frac{m}{n}$ is the average degree,
- $V = \sum_{u \in U_k} \max(0, d(u) - d_{\max})$ penalizes degree violations,
- $m = \sum A(i,j)$ is the total number of edges, n the number of nodes, and $d(u)$ is the degree of node u
- $\alpha, \beta, \gamma, \lambda, \delta$ are energy hyperparameters
- $d_{\max}$ is an hyperparameter.

This energy function encodes a multi-objective trade-off between recommendation quality (rewarded by r and s) and graph sparsity/degree constraints (penalized by D, A_{avg}, V). A new configuration $\mathbf{A}'$ with energy E' is accepted with probability:

$$P(t, \Delta E) = \begin{cases} 1 & \text{if } \Delta E \geq 0, \\ \exp(\frac{\Delta E}{T_t}) & \text{otherwise} \end{cases} \tag{5.10}$$

where $\Delta E = E' - E_t$, with E_t being the energy of the current state and E' the energy of the proposed neighbor, and T_t is the temperature at iteration t, defined by cooling schedule $T_t = \frac{1}{1+t/t_{max}}$, with t_{max} equal to the maximum number of epochs. If the proposed configuration is rejected, the algorithm reverts to the previous adjacency and removes $k_{remove} \in \mathbb{N}$ edges. This iterative procedure is described in detail in Algorithm 2 in Section A.

Graph Neural Network

In our approach, the user-user adjacency matrices $\mathbf{A}_k$ are not precomputed but are iteratively refined via the SA process simultaneously with GNN training. At each epoch, the current adjacency for each subgraph G_k is updated by the SA step, generating a set of candidate adjacency matrices that

balance prediction accuracy, ranking quality (NDCG@5), sparsity, and degree constraints. These matrices are aggregated via an ensemble and combined with the original user-item interaction data R to construct the full adjacency matrix used by the GNN:

$$\mathbf{A}_{\text{GNN}} = \begin{bmatrix} \mathbf{A}_k & \mathbf{R} \\ \mathbf{R}^\top & \mathbf{0}_{|I|\times|I|} \end{bmatrix}, \tag{5.11}$$

where $\mathbf{A}_k$ is the current SA-refined adjacency for subgraph G_k, incorporating edge additions/removals according to the energy function.

The GNN is trained end-to-end on this evolving adjacency structure; node embeddings are updated using the refined graph at each training iteration, and the GNN outputs user and item embeddings for recommendation. Recommendation scores for each user are computed via the dot product between the user embedding and item embeddings. The training loss is defined as:

$$\mathcal{L}_{\text{train}} = \mathcal{L}_{\text{task}}(\mathbf{Z}, \mathbf{Y}) + \lambda \mathcal{L}_{\text{reg}}(\mathbf{A}, \mathbf{Z}, \mathcal{G}) \tag{5.12}$$

where $\mathcal{L}_{\text{task}}$ is the binary cross-entropy loss, $\mathcal{L}_{\text{reg}}(\mathbf{A}, \mathbf{Z}, \mathcal{G})$ imposes constraints on the learned graph structure and representations, $\mathcal{L}_{\text{reg}} = \text{D}(\mathbf{A}_{\text{refined}}) + \text{V}(\mathbf{A}_{\text{refined}})/N$, and λ_{reg} controls the strength of the regularization.

This joint optimization allows the GNN to adapt to the evolving subgraph structures, effectively leveraging the SA-refined user-user edges while simultaneously learning high-quality embeddings for downstream recommendation. Validation NDCG@5 is monitored during training, guiding both early stopping and hyperparameter selection via Bayesian optimization [7]. After training, the final embeddings are used to generate predictions and evaluate the model according to standard metrics.

5.2.5 Experimental Results

Research Questions.

- **RQ1:** Can graph structure be learned to improve recommendation performance in sparse, real-world settings?
- **RQ2:** What are the trade-offs between graph connectivity and recommendation accuracy when building user-user graphs?
- **RQ3:** Is model depth necessary in GNN-based recommender systems, or can carefully learned shallow graphs outperform deeper architectures?
- **RQ4:** How does the proposed algorithm perform in terms of scalability, resource consumption, and runtime efficiency on large graph datasets?

Dataset. The experiments are conducted on a large-scale, proprietary dataset. While the raw data cannot be publicly released due to its proprietary nature, we present its key statistical characteristics. This approach aims to provide sufficient context for our findings and enable researchers working with similarly structured graph-based interaction data.

The overall dataset comprises interactions of $4,060,619$ unique users and 547 unique items. A summary of the statistics for the different temporal splits used in our experiments, as described in Section 5.2.5, is presented in Table 5.6.

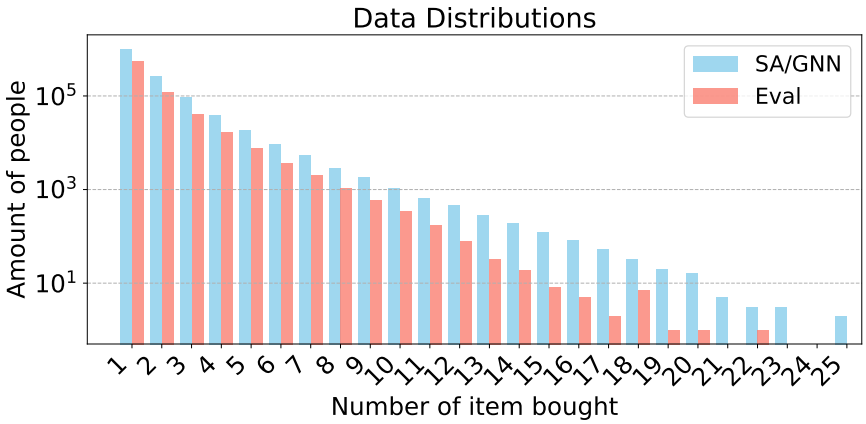


Figure 5.5: Long-tail distribution of item bought per user, in SA train and test datasets. The y-axis represents the number of users on a log-scale.

Table 5.6: Dataset Statistics Across Temporal Splits

Temporal Split	$\%U_{1-buy}$	$\%U_{cold-start}$	$\#I_{boughts}$	$\#U_{buying}$
SA/GNN	24.93	64.26	2,240,071	1,450,952
Eval	13.32	81.85	1,070,679	736,648

In Table 5.6, $\%U_{1-buy}$ quantifies the percentage of users within each split who made only a single purchase. The $\%U_{cold-start}$ measures the proportion of users who have no interaction. The table also reports the total number of item purchases $\#I_{boughts}$ and the count of unique purchasing users $\#U_{buying}$ for each split.

In Fig. 5.5 is shown the number of users (on a log y-axis) and the number of items they have bought (on the x-axis) for all the temporal splits.

Evaluation Metrics. We evaluate recommendation quality using standard ranking metrics computed at cutoff levels $K \in \{5, 10, 20\}$: Mean Average Precision (MAP), Precision@K (P@K), Recall@K (R@K), and Normalized Discounted Cumulative Gain (NDCG@K).

Evaluation Protocol. We used item purchase data from the past 9 months (m) to evaluate the performance of our proposed method. The data was divided in:

- *SA/GNN* (m=1 to m=8): the data from m=1 to m=8 were randomly split into training, validation, and test sets using a 70-20-10 distribution, respectively. These splits were applied to each subgraph to ensure the models were trained, validated, and tested on distinct subsets of the data.
- *Eval* (m=9): Finally, the last month's data (m=9) was used exclusively for evaluating the results. This evaluation step allowed us to assess the generalization capability of the models and the effectiveness of the graph learning methods over time.

Implementation Details. To ensure a rigorous and fair comparison across all evaluated approaches, several key implementation choices were consistently applied.

Table 5.7: Recommendation performance across different graph structures. Mean and variance obtained over five experiments are reported. For each metric, **bold** represents the best value obtained, and *underline* the second best.

Model	MAP	NDCG@5	R@5	NDCG@10	R@10	NDCG@20	R@20
Bipartite	0.0086 ± 0.0009	0.0063 ± 0.0008	0.0076 ± 0.0005	0.0112 ± 0.0014	0.0266 ± 0.0021	0.0215 ± 0.0018	0.0608± 0.0044
Erdős–Rényi	0.0098 ± 0.0008	0.0072 ± 0.0005	0.0108 ± 0.0010	0.0117 ± 0.0011	0.0250 ± 0.0023	0.0217 ± 0.0021	0.0604 ± 0.0058
Barabási–Albert	0.0102 ± 0.0009	0.0075 ± 0.0006	0.0112 ± 0.0011	0.0124 ± 0.0012	0.0262 ± 0.0024	0.0232 ± 0.0023	0.0631 ± 0.0059
Watts–Strogatz	*0.0107 ± 0.0009*	*0.0079 ± 0.0005*	*0.0119 ± 0.0010*	*0.0131 ± 0.0012*	*0.0278 ± 0.0025*	*0.0241 ± 0.0023*	0.0650 ± 0.0061
Configuration Model	0.0089 ± 0.0011	0.0065 ± 0.0007	0.0099 ± 0.0012	0.0103 ± 0.0014	0.0223 ± 0.0027	0.0194 ± 0.0022	0.0566 ± 0.0064
Random Regular	0.0091 ± 0.0012	0.0066 ± 0.0006	0.0117 ± 0.0013	0.0108 ± 0.0014	0.0275 ± 0.0028	0.0227 ± 0.0023	*0.0665 ± 0.0066*
SeGSL	OOM	OOM	OOM	OOM	OOM	OOM	OOM
CoGSL	OOM	OOM	OOM	OOM	OOM	OOM	OOM
SUBLIME	OOM	OOM	OOM	OOM	OOM	OOM	OOM
SA-GSL (Ours)	**0.0119 ± 0.0011**	**0.0099 ± 0.0006**	**0.0130 ± 0.0006**	**0.0153 ± 0.0010**	**0.0299 ± 0.0009**	**0.0253 ± 0.0015**	**0.0670 ± 0.0009**

Firstly, both our proposed method and all baseline models were trained and evaluated using the exact same sets of pre-generated subgraphs. It eliminates variability arising from different data partitions or subgraph sampling strategies, ensuring that observed performance differences are directly attributable to the inherent characteristics and efficacy of the models themselves rather than discrepancies in the input data.

Secondly, for a fair comparative evaluation, an equivalent percentage of edges was generated across all baseline approaches.

Baselines. The baselines used in our experiments represent different approaches to construct the user-user graph, specifically how the adjacency matrix is generated. Our focus here is on the effect of graph construction, rather than on evaluating or optimizing state-of-the-art GNN architectures. The selected baselines are:

- **Erdős-Rényi Graph** [74]: edges are formed between nodes randomly, with a fixed probability.
- **Barabási-Albert** [22]: scale-free network model, where new nodes are more likely to connect to existing nodes with higher degrees.
- **Configuration Model** [195]: edges are assigned randomly while maintaining the degree sequence for each node.
- **Random Regular** [243]: edges are placed randomly while ensuring uniformity in node degrees.
- **Watts–Strogatz** [270]: characterized by high local clustering and short average path lengths between any two nodes.
- **SEGSL** [325]: a GSL method that uses the one-dimensional structural entropy maximization strategy to extend the information embedded in the topology.
- **CoGSL** [172]: a GSL framework that learns a compact graph structure by compressing mutual information.
- **SUBLIME** [174]: an unsupervised method that uses contrastive learning to maximize the agreement between the learned topology and a self-enhanced learning target.

Each of these baseline models fills the user-user adjacency matrix differently, capturing diverse structural properties. All models are then trained using a GNN to predict recommendations, allowing us to assess their relative performance under the same experimental conditions.

Hyperparameter tuning. We integrate Optuna [7] for Bayesian hyperparameter optimization, maximizing validation metric NDCG@5 as the objective. We define the following search space:

- SA energy hyperparameters α, β, γ, δ, $\lambda \in [0.1, 100.0]$
- Maximum allowed degree $\in [3, 50]$
- Fraction of adding edges $\in [0.1, 0.3]$
- Number of ensemble SA matrices per step $\in [3, 7]$
- Number of hidden dimensions, $d \in [8, 256]$
- Number of GraphSAGE layers, $L \in \{1, 2, 3\}$
- Dropout rate, $p \in [0, 1)$
- Learning rate, $\eta \in [5 \cdot 10^{-7}, 5 \cdot 10^{-1}]$
- Weight decay, $\in [10^{-5}, 10^{-1}]$
- Regularization decay λ_{reg}, $\in [5 \cdot 10^{-5}, 5 \cdot 10^{-1}]$
- Neighbor sampling size, $n \in [5, 30]$
- Negative sampling ratio, $r \in [1, 5]$
- Aggregator type, $AGG \in [sum, min, max, mean]$

RQ1: Recommendation Quality

To evaluate the effectiveness of our graph structure learning algorithm, we compared its recommendation performance against the other graph generation baselines. The key results are summarized in Table 5.7. The results show that our proposed method, labeled as "Ours", achieves a superior performance across all evaluated metrics. Notably, several of the competing structure-learning algorithms could not run on our dataset due to memory constraints; this is indicated as OOM (Out of Memory) in the table.

These findings strongly suggest that the graph structures derived from our partitioning and SA-based optimization process are significantly more effective for GNN recommendations. The substantial gains highlight the value of targeted structural refinement in uncovering more meaningful user-item relationships, ultimately leading to enhanced recommendation quality, especially in the sparse data settings characteristic of our problem domain.

RQ2: Bipartite Graph

Our experimental results highlight the efficacy of enriching the standard graph representation for recommendation. We focused on learning and adding connections directly between users, enriching the basic user-item graph. Our approach consistently has higher performance across key metrics. Improvements are evident for NDCG@5: our method performed approximately 27% better than the simpler bipartite graph. This suggests that by understanding and incorporating user-user links, our system can more effectively discover and recommend relevant items, especially in situations where user-item interaction data might be sparse.

Graph Density

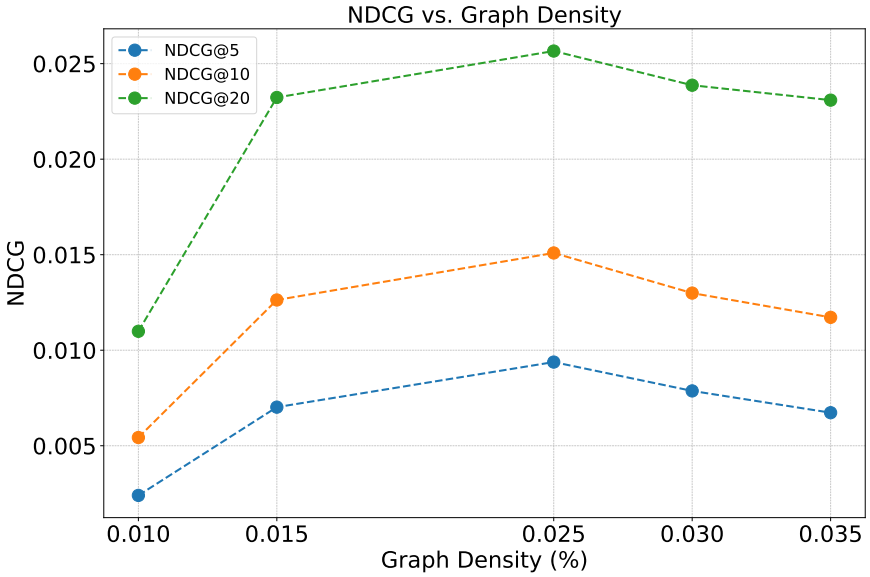


Figure 5.6: Impact of graph density on NDCG recommendation performances.

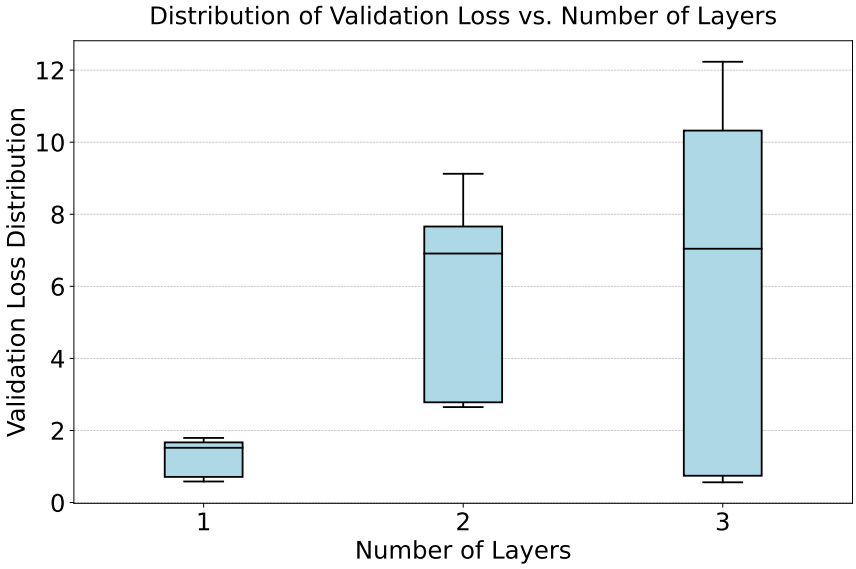
A key consideration when constructing user-user graphs for recommendation is understanding the trade-offs between the graph's connectivity and the resulting recommendation accuracy. Higher connectivity, often correlated with graph density, can provide richer information pathways but may also introduce noise or increase computational demands.

We systematically varied the density of the user-user interaction graph across five levels: 0.01%, 0.015%, 0.025%, 0.030%, and 0.035%. NDCG@K was used to evaluate the results on the test set. The results, illustrated in Fig. 5.6, reveal a distinct trade-off. Initially, increasing graph connectivity (from 0.01% to 0.025% density) yields significant benefits in recommendation accuracy. All NDCG metrics (NDCG@5, NDCG@10, and NDCG@20) demonstrate substantial improvement. This suggests that a certain level of connectivity is crucial. However, the trade-off becomes apparent as connectivity increases further. Beyond a density of 0.025%, we observe a decrease in recommendation accuracy.

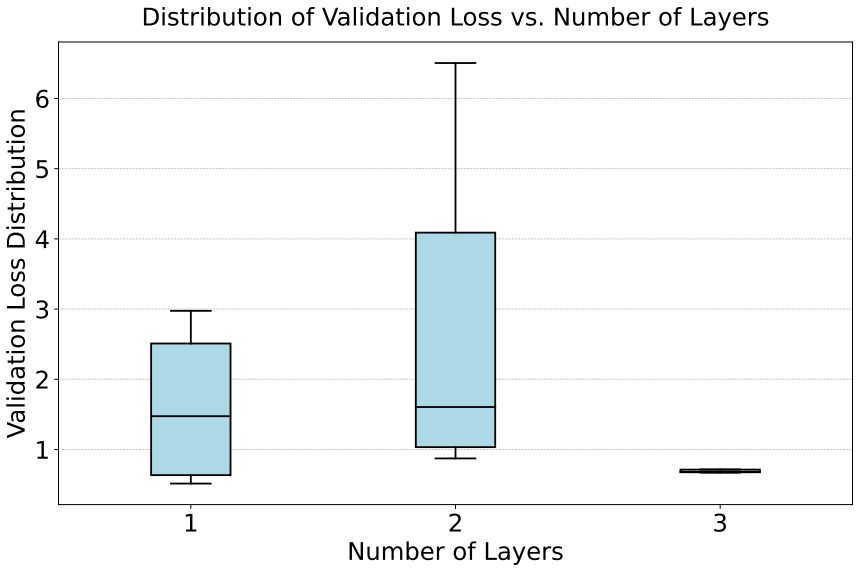
This study indicates that the relationship between graph connectivity and recommendation accuracy is not linear: there is an optimal range of connectivity, around 0.025% density in this case, where accuracy is maximized.

RQ3: Over-smoothing and Over-squashing Effects

The depth of a GNN, defined by its number of layers, critically impacts performance. While deeper models can access larger receptive fields, they risk over-smoothing and over-squashing. We analysed validation losses from our GNN trainings across varying layer depths (1, 2, and 3 layers) to observe these effects, as shown in Fig. 5.7a.



(a) 'Ours' graph construction method.



(b) Baselines methods.

Figure 5.7: Validation loss distribution for 1,2,and 3 GNN layers on (a) our method and (b) baseline methods.

Fig. 5.7a shows that models with 1 layer consistently achieve the lowest median validation loss, indicating robust performances. 2-layer models lead to a higher median loss and greater variability, while 3-layer models' performance degrades substantially. The median loss is highest, and the distribution shows high-loss outliers.

This pronounced deterioration in performance and instability with increased depth beyond one layer are characteristic of over-smoothing. Over-squashing may contribute, as deeper models struggle to compress exponentially growing neighbourhood information into fixed-size embeddings without loss.

The empirical evidence strongly suggests that for our task and dataset, shallow architectures (primarily 1 layer) are optimal, mitigating these detrimental effects. This observation guided our GNN trainings, which favoured shallower models to balance performance and scalability.

RQ4: Scalability Benchmarks and Runtime Analysis

To assess scalability, we ran our method on a Google Cloud `e2-highmem-16` instance equipped with 16 vCPUs and 128 GiB of memory.Table 5.8 reports the peak memory usage and runtime. The algorithm demonstrates excellent scalability: despite the large graph sizes, the memory footprint remained within the available resources, and the training time per run was under one hour. This scalability is achieved through the parallel training of subgraphs, which efficiently distributes the workload across computational resources, and by imposing a maximum degree per node, which limits the overall graph size.

Furthermore, Fig. 5.8 shows the convergence behavior: the energy consistently increases over timesteps and stabilizes, demonstrating that the algorithm reliably reaches convergence across all trained subgraphs. Across all trainings, we observe a $77.2\% \pm 0.7$ acceptance rate. The plot reports the mean trend over the subgraphs, with the shaded area representing the 95% confidence interval.

Table 5.8: Scalability benchmarks on the internal dataset using an `e2-highmem-16` (16 vCPUs, 128 GiB RAM).

Metric	Value	Notes
Memory Usage	113.8 GiB	88.90% GiB usage
Runtime	$46m08s \pm 1m24s$	Average over runs

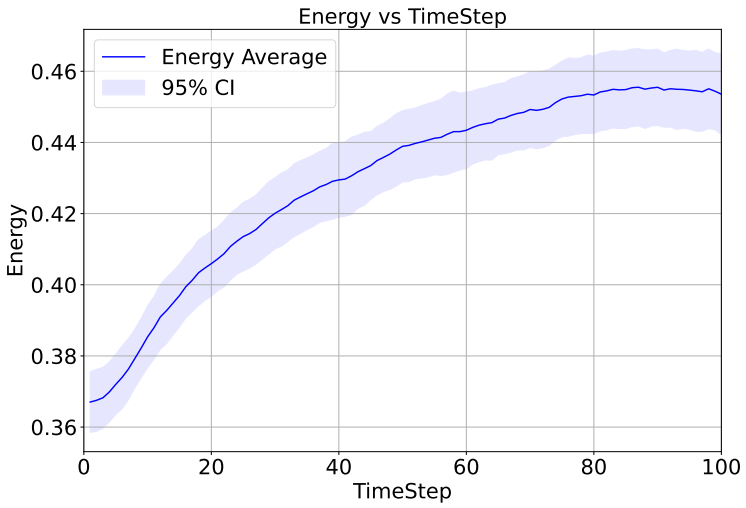


Figure 5.8: Convergence plot of the energy function through the epochs. The blue line represents the energy average across all subgraphs, the shaded area the 95% confidence interval.

5.2.6 Conclusion

This work investigated the potential of learning optimal graph structures to advance GNN-based recommendation systems, particularly in challenging sparse data scenarios. Our research answers key questions regarding the efficacy of learned graph topologies, the relationship between connectivity and accuracy, and the necessity of model depth.

We demonstrated that graph structure can be learned to significantly improve recommendation performance in sparse, real-world settings. This was achieved through our novel contribution of `SA-GSL`, a scalable structure learning algorithm that optimizes the underlying user-user graph topology via SA. In sparse recommendation graphs, we effectively navigate the critical trade-offs in graph connectivity by selectively identifying high-utility edges rather than merely increasing density.

Furthermore, our research presents evidence that model depth is not always a prerequisite for superior performance in GNN-based recommender systems. We showed that carefully learned shallow graph structures, when processed by shallow GNN models, can achieve state-of-the-art results. It highlights that the quality of the input graph can often be more impactful than architectural complexity, leading not only to better accuracy but also to significantly enhanced computational efficiency.

Lastly, our end-to-end pipeline efficiently handles graph optimization and GNN training for large, real-world datasets. This allows for more effective and faster graph-based recommendations.

Future work will explore several promising directions. Firstly, we plan to extend the SA optimization to incorporate item-item interactions, which could particularly benefit recommendations for cold-start items. Secondly, we aim to evaluate our learned graph structures by integrating them with state-of-the-art GNN architectures. Finally, a key long-term goal is to transition this algorithm into a production environment to assess its real-world impact through online A/B testing.

A Algorithm Details

This section provides a detailed description of our proposed algorithm, Simulated Annealing for Graph Structure Learning, `SA-GSL`. We present the full pseudocode and a step-by-step explanation of its core components.

Algorithm 2 Simulated Annealing for Graph Structure Learning

Input: feature matrix X, similarity matrix $S = XX^\top$, maximum iterations $t_{\max}$, add/remove counts $(k_{\text{add}}, k_{\text{remove}})$, train labels L_{train}, validation labels L_{val}, coefficients $\alpha, \beta, \gamma, \lambda, \delta$, max allowed out-degree $d_{\max}$, density threshold D_{thresh}, early stopping patience $p_{\max}$

Output: adjacency matrix $A^* \in \{0,1\}^{n \times n}$

$A \leftarrow \mathbf{0}_{n \times n}$
prevE $\leftarrow -\infty$, prevAcc $\leftarrow$ `None`, $t \leftarrow 0$, earlystop $\leftarrow 0$
$A^* \leftarrow A$, bestE $\leftarrow -\infty$
while $t < t_{\max}$ **do**
  $T \leftarrow \max(0.1, 1 - t/t_{\max})$
  $A_{\text{copy}} \leftarrow A$
  if prevAcc is None or currAcc $>$ prevAcc $+ \varepsilon$ **then**
    select k edges (i, j) according to S_{ij} and set $A[i, j] \leftarrow 1$
  else
    $\mathcal{E} = \{(i, j) \mid A_{ij} = 1\}$
    $k' = \min(k_{\text{remove}}, |\mathcal{E}|)$
    randomly select k' edges $(i, j) \in \mathcal{E}$ and set $A[i, j] \leftarrow 0$
  end if
  $\hat{Y} \leftarrow \text{LabelPropagation}(A, L_{\text{train}})$
  currAcc $\leftarrow \text{accuracy}(\hat{Y}, L_{\text{val}})$
  NDCG $\leftarrow \text{meanNDCG@5}(\hat{Y}, L_{\text{val}})$
  $m \leftarrow \sum_{i,j} A_{ij}$ $D \leftarrow m/n^2$, $A_{\text{avg}} \leftarrow m/n$
  $d_i \leftarrow \sum_j A_{ij}$ $V \leftarrow \sum_i \max(0, d_i - d_{\max})$
  $E \leftarrow \alpha \cdot \text{currAcc} + \beta \cdot \text{NDCG} - \gamma \cdot D - \lambda \cdot A_{\text{avg}} - \delta \cdot V$
  $\Delta E \leftarrow E - \text{prevE}$ $d_{\max}^{(\text{obs})} \leftarrow \max_i d_i$
  if $E <$ prevE **then**
    $p_{\text{accept}} \leftarrow \exp(\Delta E / (T + \varepsilon))$
    if `rand()` $\geq p_{\text{accept}}$ **then**
      $A \leftarrow A_{\text{copy}}$ revert
      $E \leftarrow$ prevE
    end if
  end if
  if $D > D_{\text{thresh}}$ **or** $d_{\max}^{(\text{obs})} \geq d_{\max}$ **then**
    $A \leftarrow A_{\text{copy}}$ revert
    $E \leftarrow$ prevE
  end if
  if $E >$ bestE **then**
    bestE $\leftarrow E$; $A^* \leftarrow A$; earlystop $\leftarrow 0$
  else
    earlystop $\leftarrow$ earlystop $+ 1$
  end if
  prevE $\leftarrow E$; prevAcc $\leftarrow$ currAcc; $t \leftarrow t + 1$
  if earlystop $\geq p_{\max}$ **then**
    break
  end if
end while
return A^*

A.1 Pseudocode

Algorithm 2 outlines the complete procedure for `SA-GSL`. The algorithm iteratively refines a graph structure, represented by an adjacency matrix A, to maximize a carefully designed energy function.

A.2 Explanation of the Algorithm

- **Lines 10-17 (Adaptive Candidate Generation):** This is the core proposal mechanism. The strategy is adaptive: if the validation accuracy from the previous step improved, the algorithm optimistically adds k_{add} new edges based on the highest feature similarity scores. If accuracy decreases, it pessimistically prunes k_{remove} random edges to escape a local optimum.
- **Lines 19-24 (Energy Function):** The energy function E balances multiple objectives:
  - $\alpha \cdot \text{currAcc} + \beta \cdot \text{NDCG}$: These terms reward graph structures that lead to high node classification performance, as evaluated by a simple message-passing model (Label Propagation) on the validation set.
  - $\gamma \cdot D + \lambda \cdot A_{\text{avg}}$: These terms act as regularizers, penalizing dense graphs to encourage sparsity.
  - $\delta \cdot V$: This term penalizes graphs where nodes exceed the maximum allowed out-degree $d_{\max}$, encouraging a more uniform degree distribution.
- **Lines 27-32 (Acceptance & Constraints):** We use the standard Metropolis-Hastings acceptance criterion. Moves that increase energy are always accepted. Moves that decrease energy are accepted with a probability $\exp(\Delta E/T)$, allowing the search to escape local minima. Furthermore, any move that violates the hard constraints on graph density or maximum degree is immediately rejected.
- **Lines 34-39 (Updating):** The algorithm tracks the best-scoring graph found so far (A^*). If the best energy score does not improve for $p_{\max}$ consecutive iterations, the search terminates early.

B Experimental Setup

B.1 Datasets

To assess the generalization of our method, we evaluate on six public benchmark datasets. Table 5.9 provides a summary of their statistics. Cora, Citeseer, and Pubmed are citation networks with strong homophily. Minesweeper, BlogCatalog, and Flickr are more diverse, with Minesweeper and BlogCatalog exhibiting notable heterophilous characteristics.

C Additional Experimental Results

To situate our method within the broader GSL landscape, we compare it against a comprehensive set of state-of-the-art models on the benchmark datasets, as presented in OpenGSL library [319]

Table 5.9: Statistics for benchmark datasets.

Dataset	# Nodes	# Edges	# Features	# Classes
Cora	2,708	5,278	1,433	7
Citeseer	3,327	4,552	3,703	6
Minesweeper	10,000	39,402	7	2
BlogCatalog	5,196	171,743	8,189	6
Flickr	7,575	239,738	12,047	9

Table 5.10: Node classification accuracy (%) on benchmark datasets (mean ± std over 10 runs). **Underlined bold** indicates the best model for each dataset, the second best is represented in **bold**, and the third best is *underlined*. "–" denotes out of memory or time limit of 24 hours exceeded.

Model	Cora	Citeseer	Minesweeper	BlogCatalog	Flickr
GCN [139]	81.95 ± 0.62	71.34 ± 0.48	*78.28 ± 0.44*	76.12 ± 0.42	61.60 ± 0.49
LDS [83]	84.13 ± 0.52	**75.16 ± 0.43**	–	77.10 ± 0.27	–
ProGNN [127]	80.27 ± 0.48	71.35 ± 0.42	51.43 ± 2.22	73.38 ± 0.30	52.88 ± 0.76
IDGL [55]	*84.19 ± 0.61*	*73.26 ± 0.53*	50.00 ± 0.00	89.68 ± 0.24	*86.03 ± 0.25*
GRCN [298]	**84.61 ± 0.34**	72.34 ± 0.73	72.57 ± 0.49	76.08 ± 0.27	59.31 ± 0.46
GAug [313]	83.43 ± 0.53	72.79 ± 0.86	77.93 ± 0.64	76.92 ± 0.34	61.98 ± 0.67
SLAPS [82]	72.29 ± 1.01	70.00 ± 1.29	50.89 ± 1.72	*91.73 ± 0.40*	83.92 ± 0.63
WSGNN [152]	83.66 ± 0.30	71.15 ± 1.01	67.91 ± 3.11	**92.30 ± 0.32**	**89.90 ± 0.19**
Nodeformer [278]	78.81 ± 1.21	70.39 ± 2.04	77.29 ± 1.71	44.53 ± 22.62	67.14 ± 6.77
GEN [263]	81.66 ± 0.91	73.21 ± 0.62	**79.56 ± 1.09**	90.48 ± 0.99	84.84 ± 0.81
CoGSL [172]	81.46 ± 0.88	72.94 ± 0.71	–	83.96 ± 0.54	75.10 ± 0.47
SEGSL [325]	81.04 ± 1.07	71.57 ± 0.40	–	75.03 ± 0.28	60.59 ± 0.54
SUBLIME [174]	83.33 ± 0.73	72.44 ± 0.89	49.93 ± 1.36	**95.29 ± 0.26**	**88.74 ± 0.29**
STABLE [160]	83.25 ± 0.86	70.99 ± 1.19	70.78 ± 0.27	71.84 ± 0.56	51.36 ± 1.24
`SA-GSL` (Ours)	**86.06 ± 0.31**	**74.30 ± 0.65**	**82.61 ± 0.30**	70.88 ± 0.45	76.31 ± 0.24

Evaluation Protocol. We follow the standard GSL evaluation protocol as established in the OpenGSL library [319], reporting the mean and standard deviation of node classification accuracy over 10 runs. The results for baseline models are cited directly from the OpenGSL benchmark where available. It is important to note that many of these SOTA methods are computationally intensive; several failed to run on larger graphs due to out-of-memory (OOM) errors or exceeded a 24-hour time limit. This limitation makes them unsuitable as baselines for the large-scale industrial datasets in our main experiments but warrants this separate, direct comparison on standard academic benchmarks.

Implementation Differences. We introduce some practical adaptations for the presented graphs. First, if the graph size permits, the partitioning step is avoided and the whole graph is used to perform GSL. Moreover, if the graph is homogeneous, all nodes are used instead of just one type. Finally, instead of replacing the original graph structure entirely, we create $\mathbf{A}_{\text{GNN}}$, presented in Eq. (5.11), through a linear combination of the original adjacency matrix $\mathbf{A}$ and the newly annealed one, $\mathbf{A}^{annealed}$. Formally:

$$\mathbf{A}_{\text{GNN}} = \alpha \cdot \mathbf{A} + (1 - \alpha) \cdot \mathbf{A}^{annealed} \tag{5.13}$$

where:

$$\alpha = \alpha_{start} - (\alpha_{start} - \alpha_{end}) \cdot \frac{t}{t_{max}}, \tag{5.14}$$

t is the current iteration, t_{max} is the maximum number of iterations, and $(\alpha_{start}, \alpha_{end})$ are two hyperparameters tuned in the ranges $[0.7, 0.99]$ and $[0.01, 0.7]$, respectively.

Results and Discussion. Table 5.10 summarizes the node classification accuracy. Our method achieves state-of-the-art performance on the homophilous datasets (Cora, Citeseer [292], and the Minesweeper dataset), outperforming the next-best model by a significant margin of +2.1% on Cora, +2.36% on Citeseer, and +4.09% on Minesweeper. This success can be attributed to our candidate generation strategy, which proposes edges based on feature similarity ($S = XX^\top$). This is a highly effective heuristic when connected nodes are expected to have similar features, as is the case in homophilous graphs.

Conversely, on the heterophilous social networks (BlogCatalog, Flickr [291]), our performance is not competitive with the top models. In these graphs, connections frequently form between nodes with dissimilar features, rendering our feature-similarity-based edge proposal mechanism less effective. This highlights a clear limitation and an avenue for future work: incorporating more sophisticated, learnable edge proposal mechanisms could improve performance on heterophilous graphs.

D Reproducibility

All code, model configurations, and instructions required to reproduce the experiments presented in this paper are publicly available in our GitHub repository: `https://github.com/ashba93/sa-gsl`.

Chapter 6

Conclusions

This thesis began on a journey to bridge the persistent gap between the theoretical advancements in academic recommender systems and the practical necessities of industrial deployment. Motivated by the real-world challenges faced within a large telecommunications company, this research has focused on developing and adapting graph-based methods that are not only accurate but also scalable, efficient, and robust to the complexities of production environments. The introduction laid out a landscape defined by massive datasets, strict latency requirements, and pressing business problems like cold start and customer churn. This concluding chapter summarizes how the research presented has addressed these challenges, outlines the key contributions, and proposes directions for future investigation.

Summary of key findings and contributions

The central contribution of this thesis is a portfolio of practical, graph-based solutions designed to overcome the primary bottlenecks in industrial recommender systems. Moving beyond a singular focus on predictive accuracy, this work has consistently prioritized deployability, efficiency, and direct business value. The findings are organized around the three research pillars established in the introduction.

First, we developed practical solutions for the cold-start problem. Acknowledging that new users and items are a constant reality in industrial settings, we introduced a novel end-to-end framework to generate meaningful recommendations in extreme cold-start scenarios. Our work in Chapter 3 demonstrated that by augmenting the user-item graph with feature-based user similarities and applying a diversity-aware re-ranking algorithm, we can significantly improve the discoverability of new and long-tail items. Further investigation revealed the critical trade-offs between diversity, coverage, and accuracy, providing a clear methodology for tuning the system to meet specific business objectives, such as maximizing the visibility of unsold products.

Second, we advanced the state-of-the-art in achieving scalability and efficiency in large-scale graph recommendations. Confronting the computational cost of deep learning models, this research explored two complementary paths toward efficiency. In Chapter 5, we proposed a novel graph coarsening framework that simplifies massive user-item graphs by grouping users into communities based on business rules. This method, combined with a two-stage label propagation process, enables rapid and scalable recommendations while preserving the essential structure of the original graph.

Moreover, we challenged the "deeper is better" paradigm by demonstrating that learning an optimal graph structure via simulated annealing allows simpler, shallower GNNs to achieve state-of-the-art performance. This finding underscores a core theme of the thesis: good data representation can be more impactful and far more efficient than huge model complexity.

Third, we grounded our research in high-value industrial use cases, delivering tangible business impact. This work was directly tied to the needs of a telecommunications provider. In Chapter 2, we introduced TIM-Rec, a unique public dataset for multi-item upselling recommendations characterized by sparse, explicit feedback, providing a valuable new benchmark for the research community. Furthermore, in Chapter 4, we extended our focus beyond traditional recommendation to tackle the critical problem of customer churn. We developed CRAB, an offline reinforcement learning agent that learns to balance short-term upselling goals with the long-term objective of customer retention. Its successful deployment and validated impact—reducing churn by 4.95% in a live A/B test—stand as a testament to the practical value of the methods developed.

Collectively, these contributions provide a comprehensive guide for practitioners seeking to build and deploy effective, scalable, and business-aware recommender systems. This research demonstrates that by thoughtfully adapting graph-based techniques, it is possible to create solutions that are both theoretically sound and industrially viable.

Future Research Directions

While this thesis provides solutions to several pressing challenges, it also opens up numerous avenues for future research. Based on the findings and limitations of our work, we propose the following directions:

- **Enhancing Graph Representation Learning:** Our work highlighted the importance of the underlying graph structure. Future research could extend our simulated annealing algorithm, `SA-GSL`, to jointly optimize user-user and item-item connections, potentially improving recommendations for cold-start items.
- **Advanced Modeling for Complex Industrial Objectives:** The re-ranking and reinforcement learning frameworks presented in this thesis can be extended further. Exploring state-of-the-art neural multi-objective re-ranking models could offer greater flexibility in balancing competing metrics like diversity, coverage, and novelty. Similarly, investigating more advanced offline RL algorithms beyond CQL could unlock further gains in long-term value optimization for problems like churn reduction and customer lifetime value.
- **Broadening Applications and Dynamic Adaptation:** The methods developed here have proven effective in the telecommunications sector but could be adapted for other domains. A key challenge is to create systems that adapt to dynamic environments where user preferences and item catalogs evolve rapidly. Future work could focus on developing adaptive graph coarsening techniques that dynamically adjust community structures or incremental graph structure learning methods that update the graph topology efficiently as new data arrives.

- **Addressing the User Cold-Start in Complex Scenarios:** While our frameworks have made significant strides in addressing the item cold-start problem, the user cold-start challenge, particularly within the reinforcement learning context of CRAB, remains an open area. Developing methods to generate reliable initial Q-values or policies for new users without historical interaction data is a critical next step for deploying such systems more broadly.

Closing Remarks

The doctoral journey of Alessandro Sbandi, reported in this thesis, was driven by a commitment to bridging the gap between academic theory and industrial practice. The research has consistently demonstrated that the most effective solutions in real-world settings are often not the most complex ones, but rather those that are intelligently designed to respect the constraints of scalability, latency, and business impact. From mitigating cold start and reducing customer churn to enabling efficient recommendations on massive graphs, the contributions illustrate a pragmatic approach to recommender systems research.

Ultimately, this thesis argues that the value of a recommendation algorithm is not measured solely by its performance on a static benchmark, but by its ability to integrate into a production pipeline, adapt to dynamic data, and deliver measurable value. The insights, methods, and datasets presented here are offered as a step toward building the next generation of recommender systems, that are not only useful theoretically, but also practical and impactful in real-world cases.

Bibliography

[1] Iranian Churn. UCI Machine Learning Repository, 2020. DOI: https://doi.org/10.24432/C5JW3Z.

[2] Emile Aarts and Jan Korst. *Simulated annealing and Boltzmann machines: a stochastic approach to combinatorial optimization and neural computing*. John Wiley & Sons, Inc., 1989.

[3] Ralph Abboud, Ismail Ilkan Ceylan, Martin Grohe, and Thomas Lukasiewicz. The surprising power of graph neural networks with random node initialization, 2020.

[4] Mohammad Mehdi Afsar, Trafford Crump, and Behrouz H. Far. Reinforcement learning based recommender systems: A survey. *CoRR*, abs/2101.06286, 2021. URL `https://arxiv.org/abs/2101.06286`.

[5] Shaghayegh Agah, Shaun Schaeffer, Maria Peifer, Neeraj Sharma, Ankit Maheshwari, and Sardar Hamidian. Pareto-optimal solution: Optimizing engagement and revenue. In *Proceedings of the Nineteenth ACM Conference on Recommender Systems*, pages 1034–1037, 2025.

[6] Qingyao Ai, Xuanhui Wang, Sebastian Bruch, Nadav Golbandi, Michael Bendersky, and Marc Najork. Learning groupwise multivariate scoring functions using deep neural networks. In *Proceedings of the 2019 ACM SIGIR international conference on theory of information retrieval*, pages 85–92, 2019.

[7] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 2623–2631, 2019.

[8] Pegah Malekpour Alamdari, Nima Jafari Navimipour, Mehdi Hosseinzadeh, Ali Asghar Safaei, and Aso Darwesh. A systematic study on the recommender systems in the e-commerce. *Ieee Access*, 8:115694–115716, 2020.

[9] Yousef H Alfaifi. Recommender systems applications: Data sources, features, and challenges. *Information*, 15(10):660, 2024.

[10] Rand Jawad Kadhim Almahmood and Adem Tekerek. Issues and solutions in deep learning-enabled recommendation systems within the e-commerce field. *Applied Sciences*, 12(21):11256, 2022.

[11] Diogo Alves, Bruno Valente, Ricardo Filipe, Maria Castro, and Luís Macedo. A recommender system for telecommunication operators' campaigns. In *Progress in Artificial Intelligence: 19th EPIA Conference on Artificial Intelligence, EPIA 2019, Vila Real, Portugal, September 3–6, 2019, Proceedings, Part II 19*, pages 83–95. Springer, 2019.

[12] Xavier Amatriain and Justin Basilico. Past, present, and future of recommender systems: An industry perspective. In *Proceedings of the 10th ACM conference on recommender systems*, pages 211–214, 2016.

[13] Vineeta Anand and Ashish Kumar Maurya. A survey on recommender systems using graph neural network. *ACM Trans. Inf. Syst.*, 43(1), November 2024. ISSN 1046-8188. doi: 10.1145/3694784. URL `https://doi.org/10.1145/3694784`.

[14] Anitha Anandhan, Liyana Shuib, Maizatul Akmar Ismail, and Ghulam Mujtaba. Social media recommender systems: review and open research issues. *IEEE Access*, 6:15608–15628, 2018.

[15] Chris Anderson. Why the future of business is selling less of more, 2006.

[16] Chris K Anderson and Xiaoqing Xie. Improving hospitality industry sales: Twenty-five years of revenue management. *Cornell Hospitality Quarterly*, 51(1):53–67, 2010.

[17] Jose A Arjona-Medina, Michael Gillhofer, Michael Widrich, Thomas Unterthiner, Johannes Brandstetter, and Sepp Hochreiter. Rudder: Return decomposition for delayed rewards. *Advances in Neural Information Processing Systems*, 32, 2019.

[18] Goker Aydin and Serhan Ziya. Pricing promotional products under upselling. *Manufacturing & Service Operations Management*, 10(3):360–376, 2008.

[19] Andrea Bacciu, Federico Siciliano, Nicola Tonellotto, and Fabrizio Silvestri. Integrating item relevance in training loss for sequential recommender systems. In *Proceedings of the 17th ACM Conference on Recommender Systems*, RecSys '23, page 1114–1119, New York, NY, USA, 2023. ACM. ISBN 9798400702419. doi: 10.1145/3604915.3610643. URL `https://doi.org/10.1145/3604915.3610643`.

[20] Nasim Baharisangari, Yash Paliwal, and Zhe Xu. Counterfactually-guided causal reinforcement learning with reward machines. In *2024 American Control Conference (ACC)*, pages 522–527. IEEE, 2024.

[21] Bahman Bahmani, Benjamin Moseley, Andrea Vattani, Ravi Kumar, and Sergei Vassilvitskii. Scalable k-means++, 2012.

[22] Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *science*, 286(5439):509–512, 1999.

[23] Shivam Barwey, Riccardo Balin, Bethany Lusch, Saumil Patel, Ramesh Balakrishnan, Pinaki Pal, Romit Maulik, and Venkatram Vishwanath. Scalable and consistent graph neural networks for distributed mesh-based data-driven modeling. In *SC24-W: Workshops of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1058–1070. IEEE, 2024.

[24] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural computation*, 15(6):1373–1396, 2003.

[25] James Bennett, Stan Lanning, et al. The netflix prize. In *Proceedings of KDD cup and workshop*, volume 2007, page 35. New York, 2007.

[26] Erik Bernhardsson. Annoy: Approximate nearest neighbors in c++/python, 2018. URL `https://pypi.org/project/annoy/`. Python package version 1.13.0.

[27] Filippo Betello, Antonio Purificato, Federico Siciliano, Giovanni Trappolini, Andrea Bacciu, Nicola Tonellotto, and Fabrizio Silvestri. A reproducible analysis of sequential recommender systems. *IEEE Access*, 2024.

[28] Filippo Betello, Federico Siciliano, Pushkar Mishra, and Fabrizio Silvestri. Investigating the robustness of sequential recommender systems against training data perturbations. In *Advances in Information Retrieval*, pages 205–220, Cham, 2024. Springer Nature Switzerland. ISBN 978-3-031-56060-6.

[29] Debmalya Biswas. Delayed rewards in the context of reinforcement learning based recommender systems. In *AAI4H@ ECAI*, pages 49–53, 2020.

[30] Yolanda Blanco-Fernandez, Jose J Pazos-Arias, Alberto Gil-Solla, Manuel Ramos-Cabrer, and Martin Lopez-Nores. Providing entertainment by content-based filtering and semantic reasoning in intelligent recommender systems. *IEEE Transactions on Consumer Electronics*, 54(2):727–735, 2008.

[31] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment*, 2008(10):P10008, 2008.

[32] Tesfaye Fenta Boka, Zhendong Niu, and Rama Bastola Neupane. A survey of sequential recommendation systems: Techniques, evaluation, and future directions. *Information Systems*, page 102427, 2024.

[33] Fedor Borisyuk, Shihai He, Yunbo Ouyang, Morteza Ramezani, Peng Du, Xiaochen Hou, Chengming Jiang, Nitin Pasumarthy, Priya Bannur, Birjodh Tiwana, et al. Lignn: Graph neural networks at linkedin. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 4793–4803, 2024.

[34] Christopher Burges, Robert Ragno, and Quoc Le. Learning to rank with nonsmooth cost functions. *Advances in neural information processing systems*, 19, 2006.

[35] Raymond R Burke. Technology and the customer interface: what consumers want in the physical and virtual store. *Journal of the academy of Marketing Science*, 30(4):411–432, 2002.

[36] Chen Cai, Dingkang Wang, and Yusu Wang. Graph coarsening with neural networks. *arXiv preprint arXiv:2102.01350*, 2021.

[37] Desheng Cai, Shengsheng Qian, Quan Fang, Jun Hu, and Changsheng Xu. User cold-start recommendation via inductive heterogeneous graph neural network. *ACM Transactions on Information Systems*, 41(3):1–27, 2023.

[38] Tianchi Cai, Shenliao Bao, Jiyan Jiang, Shiji Zhou, Wenpeng Zhang, Lihong Gu, Jinjie Gu, and Guannan Zhang. Model-free reinforcement learning with stochastic reward stabilization for recommender systems. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2179–2183, 2023.

[39] Yuwei Cao, Liangwei Yang, Chen Wang, Zhiwei Liu, Hao Peng, Chenyu You, and Philip S Yu. Multi-task item-attribute graph pre-training for strict cold-start item recommendation. In *Proceedings of the 17th ACM Conference on Recommender Systems*, pages 322–333, 2023.

[40] Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. Learning to rank: from pairwise approach to listwise approach. In *Proceedings of the 24th international conference on Machine learning*, pages 129–136, 2007.

[41] Diego Carraro and Derek Bridge. Enhancing recommendation diversity by re-ranking with large language models. *ACM Transactions on Recommender Systems*, 2024.

[42] Vladimír Černỳ. Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. *Journal of optimization theory and applications*, 45:41–51, 1985.

[43] Olivier Chapelle and Yi Chang. Yahoo! learning to rank challenge overview. In *Proceedings of the learning to rank challenge*, pages 1–24. PMLR, 2011.

[44] Chang Chen, Yi-Fu Wu, Jaesik Yoon, and Sungjin Ahn. Transdreamer: Reinforcement learning with transformer world models. *arXiv preprint arXiv:2202.09481*, 2022.

[45] Deli Chen, Yankai Lin, Wei Li, Peng Li, Jie Zhou, and Xu Sun. Measuring and relieving the over-smoothing problem for graph neural networks from the topological view. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 3438–3445, 2020.

[46] Hao Chen, Zefan Wang, Yue Xu, Xiao Huang, and Feiran Huang. Gpatch: Patching graph neural networks for cold-start recommendations, 2022.

[47] Hao Chen, Yuanchen Bei, Qijie Shen, Yue Xu, Sheng Zhou, Wenbing Huang, Feiran Huang, Senzhang Wang, and Xiao Huang. Macro graph neural networks for online billion-scale recommender systems. In *Proceedings of the ACM web conference 2024*, pages 3598–3608, 2024.

[48] Hung-Chen Chen and Arbee L. P. Chen. A music recommendation system based on music data grouping and user interests. In *Proceedings of the Tenth International Conference on Information and Knowledge Management*, CIKM '01, page 231–238, New York, NY, USA, 2001. ACM. ISBN 1581134363. doi: 10.1145/502585.502625. URL `https://doi.org/10.1145/502585.502625`.

[49] Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems*, 34:15084–15097, 2021.

[50] Minmin Chen, Alex Beutel, Paul Covington, Sagar Jain, Francois Belletti, and Ed H. Chi. Top-k off-policy correction for a REINFORCE recommender system. *CoRR*, abs/1812.02353, 2018. URL `http://arxiv.org/abs/1812.02353`.

[51] Minmin Chen, Bo Chang, Can Xu, and Ed H Chi. User response models to improve a reinforce recommender system. In *Proceedings of the 14th ACM international conference on web search and data mining*, pages 121–129, 2021.

[52] T Chen. Xgboost: extreme gradient boosting. *R package version 0.4-2*, 1(4), 2015.

[53] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, page 785–794, New York, NY, USA, 2016. ACM. ISBN 9781450342322. doi: 10.1145/2939672.2939785. URL `https://doi.org/10.1145/2939672.2939785`.

[54] Xiaocong Chen, Siyu Wang, and Lina Yao. On the opportunities and challenges of offline reinforcement learning for recommender systems. *arXiv preprint arXiv:2308.11336*, 2023.

[55] Yu Chen, Lingfei Wu, and Mohammed Zaki. Iterative deep graph learning for graph neural networks: Better and robust node embeddings. *Advances in neural information processing systems*, 33:19314–19326, 2020.

[56] Yunfei Chu, Jiangchao Yao, Chang Zhou, and Hongxia Yang. Graph neural networks in modern recommender systems. In *Graph Neural Networks: Foundations, Frontiers, and Applications*, pages 423–445. Springer, 2022.

[57] Aaron Clauset, Mark EJ Newman, and Cristopher Moore. Finding community structure in very large networks. *Physical Review E—Statistical, Nonlinear, and Soft Matter Physics*, 70 (6):066111, 2004.

[58] Shay Cohen, Gideon Dror, and Eytan Ruppin. Feature selection via coalitional game theory. *Neural Computation*, 19(7):1939–1961, 2007.

[59] Paul Covington, Jay Adams, and Emre Sargin. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM conference on recommender systems*, pages 191–198, 2016.

[60] CIKM Cup. Track 2: Personalized e-commerce search challenge, 2016.

[61] Dimitrios Danopoulos, Christoforos Kachris, and Dimitrios Soudris. Approximate similarity search with faiss framework using fpgas on the cloud. In Dionisios N. Pnevmatikatos, Maxime Pelcat, and Matthias Jung, editors, *Embedded Computer Systems: Architectures, Modeling, and Simulation*, pages 373–386, Cham, 2019. Springer International Publishing. ISBN 978-3-030-27562-4.

[62] Romain Deffayet, Thibaut Thonet, Jean-Michel Renders, and Maarten de Rijke. Offline evaluation for reinforcement learning-based recommendation: A critical issue and some alternatives, 2023.

[63] Yashar Deldjoo, Markus Schedl, Paolo Cremonesi, and Gabriella Pasi. Recommender systems leveraging multimedia content. *ACM Computing Surveys (CSUR)*, 53(5):1–38, 2020.

[64] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[65] Laxman Dhulipala, Jakub Łącki, Jason Lee, and Vahab Mirrokni. Terahac: Hierarchical agglomerative clustering of trillion-edge graphs. *Proceedings of the ACM on Management of Data*, 1(3):1–27, 2023.

[66] Francesco Di Giovanni, T Konstantin Rusch, Michael M Bronstein, Andreea Deac, Marc Lackenby, Siddhartha Mishra, and Petar Veličković. How does over-squashing affect the power of gnns? *arXiv preprint arXiv:2306.03589*, 2023.

[67] Janis Dietz. Complaint management: The heart of crm. *Journal of Consumer Marketing*, 23 (1):50–51, 2006.

[68] Kaize Ding, Zhe Xu, Hanghang Tong, and Huan Liu. Data augmentation for deep graph learning: A survey. *ACM SIGKDD Explorations Newsletter*, 24(2):61–77, 2022.

[69] Mucong Ding, Tahseen Rabbani, Bang An, Evan Wang, and Furong Huang. Sketch-gnn: Scalable graph neural networks with sublinear training complexity. *Advances in Neural Information Processing Systems*, 35:2930–2943, 2022.

[70] Jialin Dong, Da Zheng, Lin F Yang, and George Karypis. Global neighbor sampling for mixed cpu-gpu training on giant graphs. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 289–299, 2021.

[71] Kaiwen Dong, Zhichun Guo, and Nitesh V Chawla. Core: Data augmentation for link prediction via information bottleneck. *arXiv preprint arXiv:2404.11032*, 2024.

[72] Navin Dookeram, Zahira Hosein, and Patrick Hosein. A recommender system for the upselling of telecommunications products. In *2022 24th International Conference on Advanced Communication Technology (ICACT)*, pages 66–72. IEEE, 2022.

[73] Tanishq Dubey, Sidharth Agarwal, Shubham Gupta, and Srikanta Bedathur. Mintt: Memory inductive transfer for temporal graph neural networks. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 750–760, 2025.

[74] Paul Erdos, Alfréd Rényi, et al. On the evolution of random graphs. *Publ. math. inst. hung. acad. sci*, 5(1):17–60, 1960.

[75] Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. Diversity is all you need: Learning skills without a reward function. *arXiv preprint arXiv:1802.06070*, 2018.

[76] Benjamin Eysenbach, Tianjun Zhang, Sergey Levine, and Russ R Salakhutdinov. Contrastive learning as goal-conditioned reinforcement learning. *Advances in Neural Information Processing Systems*, 35:35603–35620, 2022.

[77] Mohammed FadhelAljunid and DH Manjaiah. A survey on recommendation systems for social media using big data analytics. *International Journal of Latest Trends in Engineering and Technology*, pages 48–58, 2017.

[78] Morten W Fagerland, David W Hosmer, and Anna M Bofin. Multinomial goodness-of-fit tests for logistic regression models. *Statistics in medicine*, 27(21):4238–4253, 2008.

[79] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. Graph neural networks for social recommendation. In *The world wide web conference*, pages 417–426, 2019.

[80] Yu-chen Fan, Yitong Ji, Jie Zhang, and Aixin Sun. Our model achieves excellent performance on movielens: What does it mean? *ACM Transactions on Information Systems*, 42(6):1–25, 2024.

[81] Ching Fang and Kimberly L Stachenfeld. Predictive auxiliary objectives in deep rl mimic learning in the brain. *arXiv preprint arXiv:2310.06089*, 2023.

[82] Bahare Fatemi, Layla El Asri, and Seyed Mehran Kazemi. Slaps: Self-supervision improves structure learning for graph neural networks. *Advances in Neural Information Processing Systems*, 34:22667–22681, 2021.

[83] Luca Franceschi, Mathias Niepert, Massimiliano Pontil, and Xiao He. Learning discrete structures for graph neural networks. In *International conference on machine learning*, pages 1972–1982. PMLR, 2019.

[84] Daniel Fryer, Inga Strümke, and Hien Nguyen. Shapley values for feature selection: The good, the bad, and the axioms. *Ieee Access*, 9:144352–144360, 2021.

[85] Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. In *International conference on machine learning*, pages 2052–2062. PMLR, 2019.

[86] Jasmina Gajcin and Ivana Dusparic. Redefining counterfactual explanations for reinforcement learning: Overview, challenges and opportunities. *ACM Computing Surveys*, 56(9):1–33, 2024.

[87] Lu Gan, Diana Nurbakova, Léa Laporte, and Sylvie Calabretto. Enhancing recommendation diversity using determinantal point processes on knowledge graphs. In *Proceedings of the 43rd international ACM SIGIR conference on research and development in information retrieval*, pages 2001–2004, 2020.

[88] Chen Gao, Yu Zheng, Nian Li, Yinfeng Li, Yingrong Qin, Jinghua Piao, Yuhan Quan, Jianxin Chang, Depeng Jin, Xiangnan He, et al. A survey of graph neural networks for recommender systems: Challenges, methods, and directions. *ACM Transactions on Recommender Systems*, 1(1):1–51, 2023.

[89] Chengqian Gao, Ke Xu, Kuangqi Zhou, Lanqing Li, Xueqian Wang, Bo Yuan, and Peilin Zhao. Value penalized q-learning for recommender systems. In *Proceedings of the 45th international ACM SIGIR conference on research and development in information retrieval*, pages 2008–2012, 2022.

[90] Hongyang Gao and Shuiwang Ji. Graph u-nets. In *international conference on machine learning*, pages 2083–2092. PMLR, 2019.

[91] Alireza Gharahighehi and Celine Vens. Diversification in session-based news recommender systems. *Personal and Ubiquitous Computing*, 27(1):5–15, 2023.

[92] Nemat Gholinejad and Mostafa Haghir Chehreghani. Heterophily-aware fair recommendation using graph convolutional networks. *arXiv preprint arXiv:2402.03365*, 2024.

[93] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International conference on machine learning*, pages 1263–1272. PMLR, 2017.

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

[95] Nianlong Gu, Yingqiang Gao, and Richard HR Hahnloser. Local citation recommendation with hierarchical-attention text encoder and scibert-based reranking. In *European conference on information retrieval*, pages 274–288. Springer, 2022.

[96] Yang Gu, Caroline Zhou, Qiao Zhang, Scott Wang, Yongzhe Wang, Li Zhang, Nikos Parotsidis, Cj Carey, Ashkan Fard, Mingyan Gao, et al. Streaming trends: A low-latency platform for dynamic video grouping and trending corpora building. In *Proceedings of the Nineteenth ACM Conference on Recommender Systems*, pages 1091–1094, 2025.

[97] Ritam Guha, Ali Hussain Khan, Pawan Kumar Singh, Ram Sarkar, and Debotosh Bhattacharjee. Cga: A new feature selection model for visual human action recognition. *Neural Computing and Applications*, 33:5267–5286, 2021.

[98] Anshul Gupta and Pravin Shrinath. Link prediction based on bipartite graph for recommendation system using optimized svd++. *Procedia Computer Science*, 218:1353–1365, 2023.

[99] Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. *Journal of machine learning research*, 3(Mar):1157–1182, 2003.

[100] Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. *arXiv preprint arXiv:1912.01603*, 2019.

[101] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.

[102] Beining Han, Zhizhou Ren, Zuofan Wu, Yuan Zhou, and Jian Peng. Off-policy reinforcement learning with delayed rewards. In *International Conference on Machine Learning*, pages 8280–8303. PMLR, 2022.

[103] Ruidong Han, Qianzhong Li, He Jiang, Rui Li, Yurou Zhao, Xiang Li, and Wei Lin. Enhancing ctr prediction through sequential recommendation pre-training: Introducing the srp4ctr framework. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, pages 3777–3781, 2024.

[104] Bowen Hao, Jing Zhang, Hongzhi Yin, Cuiping Li, and Hong Chen. Pre-training graph neural networks for cold-start users and items representation. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, WSDM '21, page 265–273, New York, NY, USA, 2021. ACM. ISBN 9781450382977. doi: 10.1145/3437963.3441738. URL https://doi.org/10.1145/3437963.3441738.

[105] Junheng Hao, Tong Zhao, Jin Li, Xin Luna Dong, Christos Faloutsos, Yizhou Sun, and Wei Wang. P-companion: A principled framework for diversified complementary product recommendation. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 2517–2524, 2020.

[106] F Maxwell Harper and Joseph A Konstan. The movielens datasets: History and context. *Acm transactions on interactive intelligent systems (tiis)*, 5(4):1–19, 2015.

[107] Matthew Hausknecht, Peter Stone, and Off-policy Mc. On-policy vs. off-policy updates for deep reinforcement learning. In *Deep reinforcement learning: frontiers and challenges, IJCAI 2016 Workshop*. AAAI Press New York, NY, USA, 2016.

[108] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Neural collaborative filtering. In *Proceedings of the 26th international conference on world wide web*, pages 173–182, 2017.

[109] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. Lightgcn: Simplifying and powering graph convolution network for recommendation. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*, pages 639–648, 2020.

[110] B Hidasi. Session-based recommendations with recurrent neural networks. *arXiv preprint arXiv:1511.06939*, 2015.

[111] Yassine Himeur, Abdullah Alsalemi, Ayman Al-Kababji, Faycal Bensaali, Abbes Amira, Christos Sardianos, George Dimitrakopoulos, and Iraklis Varlamis. A survey of recommender systems for energy efficiency in buildings: Principles, challenges and prospects. *Information Fusion*, 72:1–21, 2021.

[112] Manel Hmimida and Rushed Kanawati. A graph-coarsening approach for tag recommendation. In *Proceedings of the 25th International Conference Companion on World Wide Web*, WWW '16 Companion, page 43–44, Republic and Canton of Geneva, CHE, 2016. International World Wide Web Conferences Steering Committee. ISBN 9781450341448. doi: 10.1145/2872518.2889415. URL https://doi.org/10.1145/2872518.2889415.

[113] Yan-e Hou, Wenbo Gu, WeiChuan Dong, and Lanxue Dang. A Deep Reinforcement Learning Real-Time Recommendation Model Based on Long and Short-Term Preference. *International Journal of Computational Intelligence Systems*, 16(1):4, January 2023. ISSN 1875-6883. doi: 10.1007/s44196-022-00179-1. URL https://doi.org/10.1007/s44196-022-00179-1.

[114] Yifan Hu, Yehuda Koren, and Chris Volinsky. Collaborative filtering for implicit feedback datasets. In *2008 Eighth IEEE international conference on data mining*, pages 263–272. Ieee, 2008.

[115] Yunan Hu, Mark Thornburg, Mario Garcia Armas, Vito Ostuni, Anne Cocos, Kriti Kohli, Christoph Kofler, and Rob Saltiel. Cold starting a new content type: A case study with netflix live. In *Proceedings of the Nineteenth ACM Conference on Recommender Systems*, pages 927–930, 2025.

[116] Bingquan Huang, Mohand Tahar Kechadi, and Brian Buckley. Customer churn prediction in telecommunications. *Expert Systems with Applications*, 39(1):1414–1425, 2012.

[117] Feiran Huang, Zefan Wang, Xiao Huang, Yufeng Qian, Zhetao Li, and Hao Chen. Aligning distillation for cold-start item recommendation. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1147–1157, 2023.

[118] James Huang, Stephanie Rogers, and Eunkwang Joo. Improving restaurants by extracting subtopics from yelp reviews. *iConference 2014 (Social Media Expo)*, 1, 2014.

[119] Qian Huang, Horace He, Abhay Singh, Ser-Nam Lim, and Austin R Benson. Combining label propagation and simple models out-performs graph neural networks. *arXiv preprint arXiv:2010.13993*, 2020.

[120] Zan Huang, Wingyan Chung, and Hsinchun Chen. A graph model for e-commerce recommender systems. *Journal of the American Society for information science and technology*, 55 (3):259–274, 2004.

[121] Zengfeng Huang, Shengzhong Zhang, Chong Xi, Tang Liu, and Min Zhou. Scaling up graph neural networks via graph coarsening. In *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*, pages 675–684, 2021.

[122] Chia-Chun Hung, Timothy Lillicrap, Josh Abramson, Yan Wu, Mehdi Mirza, Federico Carnevale, Arun Ahuja, and Greg Wayne. Optimizing agent behavior over long time scales by transporting value. *Nature communications*, 10(1):5223, 2019.

[123] Shin-Yuan Hung, David C Yen, and Hsiu-Yu Wang. Applying data mining to telecom churn management. *Expert Systems with Applications*, 31(3):515–524, 2006.

[124] Dirk Husmeier. Sensitivity and specificity of inferring genetic regulatory interactions from microarray experiments with dynamic bayesian networks. *Bioinformatics*, 19(17):2271–2282, 2003.

[125] Max Jaderberg, Volodymyr Mnih, Wojciech Marian Czarnecki, Tom Schaul, Joel Z Leibo, David Silver, and Koray Kavukcuoglu. Reinforcement learning with unsupervised auxiliary tasks. *arXiv preprint arXiv:1611.05397*, 2016.

[126] Amit Jaspal, Kapil Dalwani, and Ajantha Ramineni. Socripple: A two-stage framework for cold-start video recommendations. In *Proceedings of the Nineteenth ACM Conference on Recommender Systems*, pages 1082–1085, 2025.

[127] Wei Jin, Yao Ma, Xiaorui Liu, Xianfeng Tang, Suhang Wang, and Jiliang Tang. Graph structure learning for robust graph neural networks. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 66–74, 2020.

[128] David S Johnson, Cecilia R Aragon, Lyle A McGeoch, and Catherine Schevon. Optimization by simulated annealing: An experimental evaluation; part i, graph partitioning. *Operations research*, 37(6):865–892, 1989.

[129] Antonin Joly and Nicolas Keriven. Graph coarsening with message-passing guarantees. *Advances in Neural Information Processing Systems*, 37:114902–114927, 2024.

[130] Clark Mingxuan Ju, Leonardo Neves, Bhuvesh Kumar, Liam Collins, Tong Zhao, Yuwei Qiu, Qing Dou, Yang Zhou, Sohail Nizam, Rengim Aykan Ozturk, et al. Learning universal user representations leveraging cross-domain user intent at snapchat. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 4345–4349, 2025.

[131] Wei Ju, Siyu Yi, Yifan Wang, Zhiping Xiao, Zhengyang Mao, Hourun Li, Yiyang Gu, Yifang Qin, Nan Yin, Senzhang Wang, et al. A survey of graph neural networks in real world: Imbalance, noise, privacy and ood challenges. *arXiv preprint arXiv:2403.04468*, 2024.

[132] Yuchin Juan, Jianqiang Shen, Shaobo Zhang, Qianqi Shen, Caleb Johnson, Luke Simon, Liangjie Hong, and Wenjing Zhang. Scaling retrieval for web-scale recommenders: Lessons from inverted indexes to embedding search. In *Proceedings of the Nineteenth ACM Conference on Recommender Systems*, pages 1066–1069, 2025.

[133] Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285, 1996.

[134] Vassilis Kalofolias. How to learn a graph from smooth signals. In *Artificial intelligence and statistics*, pages 920–929. PMLR, 2016.

[135] Wang-Cheng Kang and Julian McAuley. Self-attentive sequential recommendation. In *2018 IEEE international conference on data mining (ICDM)*, pages 197–206. IEEE, 2018.

[136] Mozhgan Karimi, Dietmar Jannach, and Michael Jugovac. News recommender systems – survey and roads ahead. *Information Processing and Management*, 54(6):1203–1227, 2018. ISSN 0306-4573. doi: https://doi.org/10.1016/j.ipm.2018.04.008. URL `https://www.sciencedirect.com/science/article/pii/S030645731730153X`.

[137] George Karypis and Vipin Kumar. Metis: A software package for partitioning unstructured graphs, partitioning meshes, and computing fill-reducing orderings of sparse matrices. 1997.

[138] Jinri Kim, Eungi Kim, Kwangeun Yeo, Yujin Jeon, Chanwoo Kim, Sewon Lee, and Joonseok Lee. Content-based graph reconstruction for cold-start item recommendation. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1263–1273, 2024.

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

[140] Scott Kirkpatrick, C Daniel Gelatt Jr, and Mario P Vecchi. Optimization by simulated annealing. *science*, 220(4598):671–680, 1983.

[141] Anton Klenitskiy, Anna Volodkevich, Anton Pembek, and Alexey Vasilev. Does it look sequential? an analysis of datasets for evaluation of sequential recommendations. In *Proceedings of the 18th ACM Conference on Recommender Systems*, pages 1067–1072, 2024.

[142] Hyeyoung Ko, Suyeon Lee, Yoonseo Park, and Anna Choi. A survey of recommendation systems: recommendation models, techniques, and application fields. *Electronics*, 11(1):141, 2022.

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

[144] Bernard F Kubiak and Pawel Weichbroth. Cross-and up-selling techniques in e-commerce activities. *Journal of Internet Banking and Commerce*, 15(3):1–7, 2010.

[145] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. *CoRR*, abs/2006.04779, 2020. URL `https://arxiv.org/abs/2006.04779`.

[146] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 33:1179–1191, 2020.

[147] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. In *NeurIPS*, 2020.

[148] V Kumar, Agata Leszkiewicz, and Angeliki Herbst. Are you back for good or still shopping around? investigating customers' repeat churn behavior. *Journal of Marketing Research*, 55 (2):208–225, 2018.

[149] Stanislav Kuznetsov and Pavel Kordík. Improving recommendation diversity and serendipity with an ontology-based algorithm for cold start environments. *International Journal of Data Science and Analytics*, 20(2):431–443, 2025.

[150] Vijayeta James Lall. Application of analytics to help telecom companies increase revenue in saturated markets. *FIIB Business Review*, 2(3):3–10, 2013.

[151] Anja Lambrecht and Bernd Skiera. Paying too much and being happy about it: Existence, causes, and consequences of tariff-choice biases. *Journal of marketing Research*, 43(2):212–223, 2006.

[152] Danning Lao, Xinyu Yang, Qitian Wu, and Junchi Yan. Variational inference for training graph neural networks in low-data regime through joint structure-label estimation. In *Proceedings of the 28th ACM SIGKDD conference on knowledge discovery and data mining*, pages 824–834, 2022.

[153] Michael Laskin, Aravind Srinivas, and Pieter Abbeel. Curl: Contrastive unsupervised representations for reinforcement learning. In *International conference on machine learning*, pages 5639–5650. PMLR, 2020.

[154] Misha Laskin, Kimin Lee, Adam Stooke, Lerrel Pinto, Pieter Abbeel, and Aravind Srinivas. Reinforcement learning with augmented data. *Advances in neural information processing systems*, 33:19884–19895, 2020.

[155] Hoang Le, Cameron Voloshin, and Yisong Yue. Batch policy learning under constraints. In *International Conference on Machine Learning*, pages 3703–3712. PMLR, 2019.

[156] Junhyun Lee, Inyeop Lee, and Jaewoo Kang. Self-attention graph pooling. In *International conference on machine learning*, pages 3734–3743. pmlr, 2019.

[157] George Lekakos and Petros Caravelas. A hybrid approach for movie recommendation. *Multimedia Tools Appl.*, 36:55–70, 01 2008. doi: 10.1007/s11042-006-0082-7.

[158] Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *CoRR*, abs/2005.01643, 2020. URL `https://arxiv.org/abs/2005.01643`.

[159] Dai Li, Kevin Course, Wei Li, Hongwei Li, Jie Hua, Yiqi Chen, Zhao Zhu, Rui Jian, Xuan Cao, Bi Xue, et al. Realizing scaling laws in recommender systems: A foundation-expert paradigm for hyperscale model deployment. *arXiv preprint arXiv:2508.02929*, 2025.

[160] Kuan Li, Yang Liu, Xiang Ao, Jianfeng Chi, Jinghua Feng, Hao Yang, and Qing He. Reliable representations make a stronger defender: Unsupervised structure refinement for robust gnn. In *Proceedings of the 28th ACM SIGKDD conference on knowledge discovery and data mining*, pages 925–935, 2022.

[161] Pan Li, Yanbang Wang, Hongwei Wang, and Jure Leskovec. Distance encoding: Design provably more powerful neural networks for graph representation learning. *Advances in Neural Information Processing Systems*, 33:4465–4478, 2020.

[162] Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper insights into graph convolutional networks for semi-supervised learning. In *AAAI Conference on Artificial Intelligence*, 2018. URL `https://api.semanticscholar.org/CorpusID:11118105`.

[163] Xinyi Li, Yifan Chen, Benjamin Pettit, and Maarten De Rijke. Personalised reranking of paper recommendations using paper content and user behavior. *ACM Transactions on Information Systems (TOIS)*, 37(3):1–23, 2019.

[164] Yujia Li, Chenjie Gu, Thomas Dullien, Oriol Vinyals, and Pushmeet Kohli. Graph matching networks for learning the similarity of graph structured objects. In *International conference on machine learning*, pages 3835–3845. PMLR, 2019.

[165] Zhou Lian-Ying, Daniel M Amoh, Louis K Boateng, and Andrews A Okine. Combined appetency and upselling prediction scheme in telecommunication sector using support vector machines. *International Journal of Modern Education and Computer Science*, 14(6):1, 2019.

[166] Xurong Liang, Vu Nguyen, Vuong Le, Paul Albert, and Julien Monteil. In-context learning for addressing user cold-start in sequential movie recommenders. In *Proceedings of the Nineteenth ACM Conference on Recommender Systems*, pages 979–982, 2025.

[167] David Liben-Nowell and Jon Kleinberg. The link prediction problem for social networks. In *Proceedings of the Twelfth International Conference on Information and Knowledge Management*, CIKM '03, page 556–559, New York, NY, USA, 2003. ACM. ISBN 1581137230. doi: 10.1145/956863.956972. URL https://doi.org/10.1145/956863.956972.

[168] Jan Malte Lichtenberg, Giuseppe Di Benedetto, and Matteo Ruffini. Ranking across different content types: The robust beauty of multinomial blending. In *Proceedings of the 18th ACM Conference on Recommender Systems*, pages 823–825, 2024.

[169] TP Lillicrap. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[170] Dandan Lin, Shijie Sun, Jingtao Ding, Xuehan Ke, Hao Gu, Xing Huang, Chonggang Song, Xuri Zhang, Lingling Yi, Jie Wen, et al. Platogl: Effective and scalable deep graph learning system for graph-enhanced real-time recommendation. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, pages 3302–3311, 2022.

[171] Bin Liu, Niannan Xue, Huifeng Guo, Ruiming Tang, Stefanos Zafeiriou, Xiuqiang He, and Zhenguo Li. Autogroup: Automatic feature grouping for modelling explicit high-order feature interactions in ctr prediction. In *Proceedings of the 43rd international ACM SIGIR conference on research and development in information retrieval*, pages 199–208, 2020.

[172] Nian Liu, Xiao Wang, Lingfei Wu, Yu Chen, Xiaojie Guo, and Chuan Shi. Compact graph structure learning via mutual information compression. In *Proceedings of the ACM web conference 2022*, pages 1601–1610, 2022.

[173] Ying Chieh Liu and Min Qi Huang. Examining the matthew effect on youtube recommendation system. In *2021 International Conference on Technologies and Applications of Artificial Intelligence (TAAI)*, pages 146–148, New York, NY, USA, 2021. IEEE. doi: 10.1109/TAAI54685.2021.00035.

[174] Yixin Liu, Yu Zheng, Daokun Zhang, Hongxu Chen, Hao Peng, and Shirui Pan. Towards unsupervised deep graph structure learning. In *Proceedings of the ACM Web Conference 2022*, pages 1392–1403, 2022.

[175] Oren E Livne and Achi Brandt. Lean algebraic multigrid (lamg): Fast graph laplacian linear solver. *SIAM Journal on Scientific Computing*, 34(4):B499–B522, 2012.

[176] Andreas Loukas and Pierre Vandergheynst. Spectrally approximating large graphs with smaller graphs. In *International conference on machine learning*, pages 3237–3246. PMLR, 2018.

[177] Yuxun Lu, Kosuke Nakamura, and Ryutaro Ichise. Hyperrs: hypernetwork-based recommender system for the user cold-start problem. *IEEE Access*, 11:5453–5463, 2023.

[178] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30, New

York, NY, USA, 2017. Curran Associates, Inc. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/8a20a8621978632d76c43dfd28b67767-Paper.pdf.

[179] Yunze Luo, Yuezihan Jiang, Yinjie Jiang, Gaode Chen, Jingchi Wang, Kaigui Bian, Peiyi Li, and Qi Zhang. Online item cold-start recommendation with popularity-aware meta-learning. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 1*, pages 927–937, 2025.

[180] Clare Lyle, Mark Rowland, Georg Ostrovski, and Will Dabney. On the effect of auxiliary tasks on representation dynamics. In *International Conference on Artificial Intelligence and Statistics*, pages 1–9. PMLR, 2021.

[181] Lucie Charlotte Magister, Pietro Barbiero, Dmitry Kazhdan, Federico Siciliano, Gabriele Ciravegna, Fabrizio Silvestri, Mateja Jamnik, and Pietro Liò. Concept distillation in graph neural networks. In Luca Longo, editor, *Explainable Artificial Intelligence*, pages 233–255, Cham, 2023. Springer Nature Switzerland. ISBN 978-3-031-44070-0.

[182] Grzegorz Malewicz, Matthew H Austern, Aart JC Bik, James C Dehnert, Ilan Horn, Naty Leiser, and Grzegorz Czajkowski. Pregel: a system for large-scale graph processing. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*, pages 135–146, 2010.

[183] Yu A Malkov and Dmitry A Yashunin. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE transactions on pattern analysis and machine intelligence*, 42(4):824–836, 2018.

[184] Timothy A Mann, Sven Gowal, Ray Jiang, Huiyi Hu, Balaji Lakshminarayanan, and Andras Gyorgy. Learning from delayed outcomes with intermediate observations. *arXiv preprint arXiv:1807.09387*, 2018.

[185] Yixiu Mao, Qi Wang, Chen Chen, Yun Qu, and Xiangyang Ji. Offline reinforcement learning with ood state correction and ood action suppression. *Advances in Neural Information Processing Systems*, 37:93568–93601, 2024.

[186] Fernando B Pérez Maurera, Maurizio Ferrari Dacrema, Pablo Castells, and Paolo Cremonesi. Impression-aware recommender systems. *arXiv preprint arXiv:2308.07857*, 2023.

[187] Rachana Mehta. Integrating matrix factorization with graph based models. In *Proceedings of the 18th ACM Conference on Recommender Systems*, pages 1314–1317, 2024.

[188] Nicholas Meisburger, Vihan Lakshman, Benito Geordie, Joshua Engels, David Torres Ramos, Pratik Pranav, Benjamin Coleman, Benjamin Meisburger, Shubh Gupta, Yashwanth Adunukota, et al. Bolt: An automated deep learning framework for training and deploying large-scale search and recommendation models on commodity cpu hardware. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, pages 4738–4744, 2023.

[189] Abhirup Mondal, Anirban Majumder, and Vineet Chaoji. Aspire: Air shipping recommendation for e-commerce products via causal inference framework. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 3584–3592, 2022.

[190] Mohammadmehdi Naghiaei, Hossein A Rahmani, and Yashar Deldjoo. Cpfair: Personalized consumer and producer fairness re-ranking for recommender systems. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 770–779, 2022.

[191] Mitsuhiko Nakamoto, Simon Zhai, Anikait Singh, Max Sobol Mark, Yi Ma, Chelsea Finn, Aviral Kumar, and Sergey Levine. Cal-ql: Calibrated offline rl pre-training for efficient online fine-tuning. *Advances in Neural Information Processing Systems*, 36:62244–62269, 2023.

[192] Maxim Naumov, Dheevatsa Mudigere, Hao-Jun Michael Shi, Jianyu Huang, Narayanan Sundaraman, Jongsoo Park, Xiaodong Wang, Udit Gupta, Carole-Jean Wu, Alisson G Azzolini, et al. Deep learning recommendation model for personalization and recommendation systems. *arXiv preprint arXiv:1906.00091*, 2019.

[193] Zahra Nazari, Christophe Charbuillet, Johan Pages, Martin Laurent, Denis Charrier, Briana Vecchione, and Ben Carterette. Recommending podcasts for cold-start users based on music listening and taste. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1041–1050, 2020.

[194] Sabina-Cristiana Necula and Vasile-Daniel Păvăloaia. Ai-driven recommendations: A systematic review of the state of the art in e-commerce. *Applied Sciences*, 13(9):5531, 2023.

[195] Mark EJ Newman. The structure and function of complex networks. *SIAM review*, 45(2): 167–256, 2003.

[196] Tien T. Nguyen, Pik-Mai Hui, F. Maxwell Harper, Loren Terveen, and Joseph A. Konstan. Exploring the filter bubble: the effect of using recommender systems on content diversity. In *Proceedings of the 23rd International Conference on World Wide Web*, WWW '14, page 677–686, New York, NY, USA, 2014. ACM. ISBN 9781450327442. doi: 10.1145/2566486.2568012. URL `https://doi.org/10.1145/2566486.2568012`.

[197] Jianmo Ni, Jiacheng Li, and Julian McAuley. 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)*, pages 188–197, 2019.

[198] Zhongyu Ouyang, Mingxuan Ju, Soroush Vosoughi, and Yanfang Ye. Non-parametric graph convolution for re-ranking in recommendation systems. In *Proceedings of the Nineteenth ACM Conference on Recommender Systems*, pages 330–339, 2025.

[199] Charul Paliwal, Anirban Majumder, and Sivaramakrishnan Kaveri. Predictive relevance uncertainty for recommendation systems. In *Proceedings of the ACM Web Conference 2024*, pages 3900–3909, 2024.

[200] Mehal Pandya and Abhigna Dholakia. Upselling strategy: A review. *Towards Excellence*, 13 (1), 2021.

[201] Yitong Pang, Lingfei Wu, Qi Shen, Yiming Zhang, Zhihua Wei, Fangli Xu, Ethan Chang, Bo Long, and Jian Pei. Heterogeneous global graph neural networks for personalized session-based recommendation. In *Proceedings of the fifteenth ACM international conference on web search and data mining*, pages 775–783, 2022.

[202] Dimitris Paraschakis, Bengt J Nilsson, and John Holländer. Comparative evaluation of top-n recommenders in e-commerce: An industrial perspective. In *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*, pages 1024–1031. IEEE, 2015.

[203] Rankyung Park, Amit Pande, David Relyea, Pushkar Chennu, and Prathyusha Kanmanth Reddy. Slh-bia: Short-long hawkes process for buy it again recommendations at scale. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2965–2969, 2024.

[204] Roma Patel, Marta Garnelo, Ian Gemp, Chris Dyer, and Yoram Bachrach. Game-theoretic vocabulary selection via the shapley value and banzhaf index. In Kristina Toutanova, Anna Rumshisky, Luke Zettlemoyer, Dilek Hakkani-Tur, Iz Beltagy, Steven Bethard, Ryan Cotterell, Tanmoy Chakraborty, and Yichao Zhou, editors, *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2789–2798, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.223. URL `https://aclanthology.org/2021.naacl-main.223`.

[205] Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *International conference on machine learning*, pages 2778–2787. PMLR, 2017.

[206] Koen Pauwels and Allen Weiss. Moving from free to fee: How online firms market to change their business model successfully. *Journal of Marketing*, 72(3):14–31, 2008.

[207] Karl Pearson. Principal components analysis. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 6(2):559, 1901.

[208] Zhiyong Peng, Changlin Han, Yadong Liu, and Zongtan Zhou. Weighted policy constraints for offline reinforcement learning. In *AAAI Conference on Artificial Intelligence*, 2023. URL `https://api.semanticscholar.org/CorpusID:259757185`.

[209] Fernando Benjamín Pérez Maurera, Maurizio Ferrari Dacrema, Pablo Castells, and Paolo Cremonesi. Impression-aware recommender systems. *ACM Transactions on Recommender Systems*, 3(4):1–46, 2025.

[210] Cong Dan Pham, Tuan Anh Chu, Huy Hung Pham, Manh Linh Dao, Thanh Son Pham, Duc Hai Nguyen, et al. A recommendation system for offers in telecommunications. In *2020 IEEE Eighth International Conference on Communications and Electronics (ICCE)*, pages 302–306. IEEE, 2021.

[211] Damien Poirier, Françoise Fessant, and Isabelle Tellier. Reducing the cold-start problem in content recommendation through opinion classification. In *2010 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*, volume 1, pages 204–207. IEEE, 2010.

[212] Antonio Purificato, Giulia Cassarà, Federico Siciliano, Pietro Liò, and Fabrizio Silvestri. Sheaf4rec: Sheaf neural networks for graph-based recommender systems. *ACM Transactions on Recommender Systems*, 2023.

[213] Tao Qin and Tie-Yan Liu. Introducing letor 4.0 datasets. *arXiv preprint arXiv:1306.2597*, 2013.

[214] Lara Quijano-Sanchez, Juan A Recio-Garcia, Belen Diaz-Agudo, and Guillermo Jimenez-Diaz. Social factors in group recommender systems. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 4(1):1–30, 2013.

[215] Usha Nandini Raghavan, Réka Albert, and Soundar Kumara. Near linear time algorithm to detect community structures in large-scale networks. *Physical Review E—Statistical, Nonlinear, and Soft Matter Physics*, 76(3):036106, 2007.

[216] Yuhang Ran, Yi-Chen Li, Fuxiang Zhang, Zongzhang Zhang, and Yang Yu. Policy regularization with dataset constraint for offline reinforcement learning. In *ICML*, 2023.

[217] Jérémie Rappaz, Julian McAuley, and Karl Aberer. Recommendation on live-streaming platforms: Dynamic availability and repeat consumption. In *Proceedings of the 15th ACM Conference on Recommender Systems*, pages 390–399, 2021.

[218] Davina Rauhaus, Jochen Gönsch, and Claudius Steinhardt. On the value of booking data for upsell decision-making in revenue management. *Flexible Services and Manufacturing Journal*, pages 1–34, 2024.

[219] Shaina Raza, Mizanur Rahman, Safiullah Kamawal, Armin Toroghi, Ananya Raval, Farshad Navah, and Amirmohammad Kazemeini. A comprehensive review of recommender systems: Transitioning from theory to practice. *arXiv preprint arXiv:2407.13699*, 2024.

[220] Yankun Ren, Xinxing Yang, Xingyu Lu, Longfei Li, Jun Zhou, Jinjie Gu, and Guannan Zhang. Greenseq: automatic design of green networks for sequential recommendation systems. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 3364–3368, 2023.

[221] Yuxin Ren, Qiya Yang, Yichun Wu, Wei Xu, Yalong Wang, and Zhiqiang Zhang. Non-autoregressive generative models for reranking recommendation. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 5625–5634, 2024.

[222] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "why should i trust you?": Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, page 1135–1144, New York, NY, USA, 2016. ACM. ISBN 9781450342322. doi: 10.1145/2939672.2939778. URL https://doi.org/10.1145/2939672.2939778.

[223] Peter J Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65, 1987.

[224] Benedek Rozemberczki, Lauren Watson, Péter Bayer, Hao-Tsung Yang, Olivér Kiss, Sebastian Nilsson, and Rik Sarkar. The shapley value in machine learning, 2022.

[225] Siddhartha Sahu, Amine Mhedhbi, Semih Salihoglu, Jimmy Lin, and M Tamer Özsu. The ubiquity of large graphs and surprising challenges of graph processing. *Proceedings of the VLDB Endowment*, 11(4):420–431, 2017.

[226] Darnbi Sakong, Thanh Trung Huynh, and Jun Jo. Hypergraph diffusion for high-order recommender systems. *arXiv preprint arXiv:2501.16722*, 2025.

[227] Scott Sanner, Krisztian Balog, Filip Radlinski, Ben Wedin, and Lucas Dixon. Large language models are competitive near cold-start recommenders for language-and item-based preferences. In *Proceedings of the 17th ACM conference on recommender systems*, pages 890–896, 2023.

[228] Marlesson R. O. Santana, Luckeciano C. Melo, Fernando H. F. Camargo, Bruno Brandão, Anderson Soares, Renan M. Oliveira, and Sandor Caetano. Mars-gym: A gym framework to model, train, and evaluate recommender systems for marketplaces. *CoRR*, abs/2010.07035, 2020. URL `https://arxiv.org/abs/2010.07035`.

[229] Marlesson RO Santana, Luckeciano C Melo, Fernando HF Camargo, Bruno Brandão, Anderson Soares, Renan M Oliveira, and Sandor Caetano. Mars-gym: A gym framework to model, train, and evaluate recommender systems for marketplaces. In *2020 International Conference on Data Mining Workshops (ICDMW)*, pages 189–197. IEEE, 2020.

[230] Alessandro Sbandi, Federico Siciliano, and Fabrizio Silvestri. Mitigating extreme cold start in graph-based recsys through re-ranking. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, pages 4844–4851, 2024.

[231] Alessandro Sbandi, Federico Siciliano, and Fabrizio Silvestri. Tim-rec: Explicit sparse feedback on multi-item upselling recommendations in an industrial dataset of telco calls. In *Proceedings of the Nineteenth ACM Conference on Recommender Systems*, pages 865–873, 2025.

[232] Ben Schafer, Joseph Konstan, and John Riedl. Recommender systems in e-commerce. *1st ACM Conference on Electronic Commerce, Denver, Colorado, United States*, 10 1999. doi: 10.1145/336992.337035.

[233] J Ben Schafer, Joseph Konstan, and John Riedl. Recommender systems in e-commerce. In *Proceedings of the 1st ACM conference on Electronic commerce*, pages 158–166, 1999.

[234] Takuma Seno and Michita Imai. d3rlpy: An offline deep reinforcement learning library. *CoRR*, abs/2111.03788, 2021. URL `https://arxiv.org/abs/2111.03788`.

[235] Marco Serafini and Hui Guan. Scalable graph neural network training: The case for sampling. *ACM SIGOPS Operating Systems Review*, 55(1):68–76, 2021.

[236] Jianzhun Shao, Yun Qu, Chen Chen, Hongchang Zhang, and Xiangyang Ji. Counterfactual conservative q learning for offline multi-agent reinforcement learning. *Advances in Neural Information Processing Systems*, 36:77290–77312, 2023.

[237] Kartik Sharma, Yeon-Chang Lee, Sivagami Nambi, Aditya Salian, Shlok Shah, Sang-Wook Kim, and Srijan Kumar. A survey of graph neural networks for social recommender systems. *ACM Computing Surveys*, 56(10):1–34, 2024.

[238] RR Sharma and S Rajan. Evaluating prediction of customer churn behavior based on artificial bee colony algorithm. *International Journal of Engineering and Computer Science*, 6(1): 20017–20021, 2017.

[239] Yutaka Shimizu, Joey Hong, Sergey Levine, and Masayoshi Tomizuka. Strategically conservative q-learning. In *ICML*, 2024.

[240] Federico Siciliano, Maria Sofia Bucarelli, Gabriele Tolomei, and Fabrizio Silvestri. Newron: A new generalization of the artificial neuron to enhance the interpretability of neural networks. In *2022 International Joint Conference on Neural Networks (IJCNN)*, pages 01–17, New York, NY, USA, 2022. IEEE. doi: 10.1109/IJCNN55064.2022.9892367.

[241] Brent Smith and Greg Linden. Two decades of recommender systems at amazon. com. *Ieee internet computing*, 21(3):12–18, 2017.

[242] Klaus Solberg Söilen. *Cross-Selling and Upselling*, pages 291–299. Springer Nature Switzerland, Cham, 2024. ISBN 978-3-031-69518-6. doi: 10.1007/978-3-031-69518-6_28. URL `https://doi.org/10.1007/978-3-031-69518-6_28`.

[243] Angelika Steger and Nicholas C Wormald. Generating random regular graphs quickly. *Combinatorics, Probability and Computing*, 8(4):377–396, 1999.

[244] Erik Štrumbelj and Igor Kononenko. Explaining prediction models and individual predictions with feature contributions. *Knowledge and information systems*, 41:647–665, 2014.

[245] Xiaoyuan Su and Taghi M. Khoshgoftaar. A survey of collaborative filtering techniques. *Advances in Artificial Intelligence*, 2009(1):421425, 2009. doi: https://doi.org/10.1155/2009/421425. URL `https://onlinelibrary.wiley.com/doi/abs/10.1155/2009/421425`.

[246] Jiaxi Tang and Ke Wang. Personalized top-n sequential recommendation via convolutional sequence embedding. In *Proceedings of the eleventh ACM international conference on web search and data mining*, pages 565–573, 2018.

[247] Jiliang Tang, Jie Tang, and Huan Liu. Recommendation in social media: recent advances and new frontiers. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1977–1977, 2014.

[248] Georgios Theocharous, Philip S Thomas, and Mohammad Ghavamzadeh. Ad recommendation systems for life-time value optimization. In *Proceedings of the 24th international conference on world wide web*, pages 1305–1310, 2015.

[249] Antonela Tommasel and Filippo Menczer. Do recommender systems make social media more susceptible to misinformation spreaders? In *Proceedings of the 16th ACM Conference on Recommender Systems*, pages 550–555, 2022.

[250] Floriano Tori, Vincent Holst, and Vincent Ginis. The effectiveness of curvature-based rewiring and the role of hyperparameters in gnns revisited. *arXiv preprint arXiv:2407.09381*, 2024.

[251] Vincent A Traag and Lovro Šubelj. Large network community detection by fast label propagation. *Scientific Reports*, 13(1):2701, 2023.

[252] Vincent A Traag, Ludo Waltman, and Nees Jan Van Eck. From louvain to leiden: guaranteeing well-connected communities. *Scientific reports*, 9(1):1–12, 2019.

[253] Peter JM Van Laarhoven, Emile HL Aarts, Peter JM van Laarhoven, and Emile HL Aarts. *Simulated annealing*. Springer, 1987.

[254] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks, 2017.

[255] Quan Vuong, Aviral Kumar, Sergey Levine, and Yevgen Chebotar. Dasco: Dual-generator adversarial support constrained offline reinforcement learning. *Advances in Neural Information Processing Systems*, 35:38937–38949, 2022.

[256] Dingrong Wang, Krishna P. Neupane, and Qi Yu. Evidential conservative q-learning for dynamic recommendations. In *ICLR*, 2024.

[257] Dong Wang, Junyi Jiao, Arnab Bhadury, Yaping Zhang, Mingyan Gao, and Onkar Dalal. Item-centric exploration for cold start problem. In *Proceedings of the Nineteenth ACM Conference on Recommender Systems*, pages 987–990, 2025.

[258] Hang Wang, Sen Lin, and Junshan Zhang. Adaptive ensemble q-learning: Minimizing estimation bias via error feedback. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021. URL `https://openreview.net/forum?id=YL6e9oSeInj`.

[259] Hao Wang, Jiaxin Yang, and Jianrong Wang. Leverage large-scale biological networks to decipher the genetic basis of human diseases using machine learning. In *Artificial Neural Networks*, pages 229–248. Springer, 2020.

[260] Jianian Wang, Sheng Zhang, Yanghua Xiao, and Rui Song. A review on graph neural network methods in financial applications. *arXiv preprint arXiv:2111.15367*, 2021.

[261] Jizhe Wang, Pipei Huang, Huan Zhao, Zhibo Zhang, Binqiang Zhao, and Dik Lun Lee. Billion-scale commodity embedding for e-commerce recommendation in alibaba. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 839–848, 2018.

[262] Menghan Wang, Yujie Lin, Guli Lin, Keping Yang, and Xiao-ming Wu. M2grl: A multi-task multi-view graph representation learning framework for web-scale recommender systems. In

Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining, pages 2349–2358, 2020.

[263] Ruijia Wang, Shuai Mou, Xiao Wang, Wanpeng Xiao, Qi Ju, Chuan Shi, and Xing Xie. Graph structure estimation neural networks. In *Proceedings of the web conference 2021*, pages 342–353, 2021.

[264] Tao Wang, Shanshan Chen, Xiaoxia Wang, and Jinfang Wang. Label propagation algorithm based on node importance. *Physica A: Statistical Mechanics and its Applications*, 551:124137, 2020.

[265] Wei Wang, Zhenzhen Quan, Siwen Zhao, Guoqiang Sun, Yujun Li, Xianye Ben, and Jianli Zhao. User-context collaboration and tensor factorization for gnn-based social recommendation. *IEEE Transactions on Network Science and Engineering*, 10(6):3320–3330, 2023.

[266] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. Neural graph collaborative filtering. In *Proceedings of the 42nd international ACM SIGIR conference on Research and development in Information Retrieval*, pages 165–174, 2019.

[267] Xiao Wang, Peng Cui, Jing Wang, Jian Pei, Wenwu Zhu, and Shiqiang Yang. Community preserving network embedding. *Proceedings of the AAAI Conference on Artificial Intelligence*, 31(1), Feb. 2017. doi: 10.1609/aaai.v31i1.10488. URL `https://ojs.aaai.org/index.php/AAAI/article/view/10488`.

[268] Yuyan Wang, Jing Zhong, Yuxin Cui, Zhaohui Guo, Chuanqi Wei, Yanchen Wang, and Zellux Wang. Not all impressions are created equal: Psychology-informed retention optimization for short-form video recommendation. In *Proceedings of the Nineteenth ACM Conference on Recommender Systems*, pages 1022–1025, 2025.

[269] Zongyi Wang, Yanyan Zou, Anyu Dai, Linfang Hou, Nan Qiao, Luobao Zou, Mian Ma, Zhuoye Ding, and Sulong Xu. An industrial framework for personalized serendipitous recommendation in e-commerce. In *Proceedings of the 17th ACM Conference on Recommender Systems*, pages 1015–1018, 2023.

[270] Duncan J Watts and Steven H Strogatz. Collective dynamics of 'small-world'networks. *nature*, 393(6684):440–442, 1998.

[271] Kangning Wei, Jinghua Huang, and Shaohong Fu. A survey of e-commerce recommender systems. In *2007 international conference on service systems and service management*, pages 1–5. IEEE, 2007.

[272] Wei Wei, Xubin Ren, Jiabin Tang, Qinyong Wang, Lixin Su, Suqi Cheng, Junfeng Wang, Dawei Yin, and Chao Huang. Llmrec: Large language models with graph augmentation for recommendation. In *Proceedings of the 17th ACM international conference on web search and data mining*, pages 806–815, 2024.

[273] Yinwei Wei, Xiang Wang, Qi Li, Liqiang Nie, Yan Li, Xuanping Li, and Tat-Seng Chua. Contrastive learning for cold-start recommendation. In *Proceedings of the 29th ACM International Conference on Multimedia*, pages 5382–5390, 2021.

[274] Cheng Wu, Chaokun Wang, Jingcao Xu, Ziwei Fang, Tiankai Gu, Changping Wang, Yang Song, Kai Zheng, Xiaowei Wang, and Guorui Zhou. Instant representation learning for recommendation over large dynamic graphs. In *2023 IEEE 39th International Conference on Data Engineering (ICDE)*, pages 82–95. IEEE, 2023.

[275] Fang Wu and Bernardo A Huberman. Finding communities in linear time: a physics approach. *The European Physical Journal B*, 38:331–338, 2004.

[276] Fangzhao Wu, Ying Qiao, Jiun-Hung Chen, Chuhan Wu, Tao Qi, Jianxun Lian, Danyang Liu, Xing Xie, Jianfeng Gao, Winnie Wu, et al. Mind: A large-scale dataset for news recommendation. In *Proceedings of the 58th annual meeting of the association for computational linguistics*, pages 3597–3606, 2020.

[277] Jiancan Wu, Xiang Wang, Fuli Feng, Xiangnan He, Liang Chen, Jianxun Lian, and Xing Xie. Self-supervised graph learning for recommendation. In *Proceedings of the 44th international ACM SIGIR conference on research and development in information retrieval*, pages 726–735, 2021.

[278] Qitian Wu, Wentao Zhao, Zenan Li, David P Wipf, and Junchi Yan. Nodeformer: A scalable graph structure learning transformer for node classification. *Advances in Neural Information Processing Systems*, 35:27387–27401, 2022.

[279] Shiwen Wu, Fei Sun, Wentao Zhang, Xu Xie, and Bin Cui. Graph neural networks in recommender systems: A survey. *ACM Comput. Surv.*, 55(5), December 2022. ISSN 0360-0300. doi: 10.1145/3535101. URL `https://doi.org/10.1145/3535101`.

[280] Shiwen Wu, Fei Sun, Wentao Zhang, Xu Xie, and Bin Cui. Graph neural networks in recommender systems: a survey. *ACM Computing Surveys*, 55(5):1–37, 2022.

[281] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S Yu. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32(1):4–24, 2020.

[282] Fen Xia, Tie-Yan Liu, Jue Wang, Wensheng Zhang, and Hang Li. Listwise approach to learning to rank: theory and algorithm. In *Proceedings of the 25th international conference on Machine learning*, pages 1192–1199, 2008.

[283] Teng Xiao and Donglin Wang. A general offline reinforcement learning framework for interactive recommendation. In *AAAI Conference on Artificial Intelligence*, 2021.

[284] Yan Xing, Fanrong Meng, Yong Zhou, Mu Zhu, Mengyu Shi, and Guibin Sun. A node influence based label propagation algorithm for community detection in networks. *The Scientific World Journal*, 2014(1):627581, 2014.

[285] Chen Xu, Jujia Zhao, Wenjie Wang, Liang Pang, Jun Xu, Tat-Seng Chua, and Maarten de Rijke. Understanding accuracy-fairness trade-offs in re-ranking through elasticity in economics. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 539–548, 2025.

[286] Da Xu, Chuanwei Ruan, Evren Korpeoglu, Sushant Kumar, and Kannan Achan. Product knowledge graph embedding for e-commerce. In *Proceedings of the 13th international conference on web search and data mining*, pages 672–680, 2020.

[287] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.

[288] Yuanbo Xu, En Wang, Yongjian Yang, and Hui Xiong. Gs-rs: A generative approach for alleviating cold start and filter bubbles in recommender systems. *IEEE Transactions on Knowledge and Data Engineering*, 2023.

[289] Yue Xu, Hao Chen, Zefan Wang, Jianwen Yin, Qijie Shen, Dimin Wang, Feiran Huang, Lixiang Lai, Tao Zhuang, Junfeng Ge, et al. Multi-factor sequential re-ranking with perception-aware diversification. In *Proceedings of the 29th ACM SIGKDD conference on knowledge discovery and data mining*, pages 5327–5337, 2023.

[290] Jia-Qi Yang, Xiang Li, Shuguang Han, Tao Zhuang, De chuan Zhan, Xiaoyi Zeng, and Bin Tong. Capturing delayed feedback in conversion rate prediction via elapsed-time sampling. In *AAAI Conference on Artificial Intelligence*, 2020. URL `https://api.semanticscholar.org/CorpusID:227333826`.

[291] Renchi Yang, Jieming Shi, Xiaokui Xiao, Yin Yang, Sourav S Bhowmick, and Juncheng Liu. Pane: scalable and effective attributed network embedding. *The VLDB journal*, 32(6):1237–1262, 2023.

[292] Zhilin Yang, William Cohen, and Ruslan Salakhudinov. Revisiting semi-supervised learning with graph embeddings. In *International conference on machine learning*, pages 40–48. PMLR, 2016.

[293] Yelp. Yelp open dataset, 2021. URL `https://www.yelp.com/dataset`. Accessed: 2025-05-01.

[294] Junghyuk Yeom, Yonghyeon Jo, Jeongmo Kim, Sanghyeon Lee, and Seungyul Han. Exclusively penalized q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 37:113405–113435, 2024.

[295] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 974–983, 2018.

[296] Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, Will Hamilton, and Jure Leskovec. Hierarchical graph representation learning with differentiable pooling. *Advances in neural information processing systems*, 31, 2018.

[297] Chenghui Yu, Haoze Wu, Jian Ding, Bingfeng Deng, and Hongyu Xiong. Unified survey modeling to limit negative user experiences in recommendation systems. In *Proceedings of the Nineteenth ACM Conference on Recommender Systems*, pages 1104–1107, 2025.

[298] Donghan Yu, Ruohong Zhang, Zhengbao Jiang, Yuexin Wu, and Yiming Yang. Graph-revised convolutional network. In *Joint European conference on machine learning and knowledge discovery in databases*, pages 378–393. Springer, 2020.

[299] Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Y Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma. Mopo: Model-based offline policy optimization. *Advances in Neural Information Processing Systems*, 33:14129–14142, 2020.

[300] Hengrui Zhang, Zhongming Yu, Guohao Dai, Guyue Huang, Yufei Ding, Yuan Xie, and Yu Wang. Understanding gnn computational graph: A coordinated computation, io, and memory perspective. *Proceedings of Machine Learning and Systems*, 4:467–484, 2022.

[301] Hengyu Zhang, Chunxu Shen, Xiangguo Sun, Jie Tan, Yanchao Tan, Yu Rong, Hong Cheng, and Lingling Yi. Hierarchical graph information bottleneck for multi-behavior recommendation. In *Proceedings of the Nineteenth ACM Conference on Recommender Systems*, pages 155–164, 2025.

[302] Jiahao Zhang, Rui Xue, Wenqi Fan, Xin Xu, Qing Li, Jian Pei, and Xiaorui Liu. Linear-time graph neural networks for scalable recommendations. In *Proceedings of the ACM on Web Conference 2024*, pages 3533–3544, 2024.

[303] Jing Zhang, Linjiajie Fang, Kexin Shi, Wenjia Wang, and Bing-Yi Jing. Q-distribution guided q-learning: Uncertainty-penalized q-value via consistency model. In *NeurIPS*, 2024.

[304] Muhan Zhang and Yixin Chen. Link prediction based on graph neural networks. *Advances in neural information processing systems*, 31, 2018.

[305] Qianru Zhang, Lianghao Xia, Xuheng Cai, Siu-Ming Yiu, Chao Huang, and Christian S Jensen. Graph augmentation for recommendation. In *2024 IEEE 40th International Conference on Data Engineering (ICDE)*, pages 557–569. IEEE, 2024.

[306] Wei Zhang, Pengye Zhang, Bo Zhang, Xingxing Wang, and Dong Wang. A collaborative transfer learning framework for cross-domain recommendation. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '23, page 5576–5585, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9798400701030. doi: 10.1145/3580305.3599758. URL `https://doi.org/10.1145/3580305.3599758`.

[307] Xiao Zhang, Hao Jia, Hanjing Su, Wenhan Wang, Jun Xu, and Ji-Rong Wen. Counterfactual reward modification for streaming recommendation with delayed feedback. *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2021. URL `https://api.semanticscholar.org/CorpusID:235792333`.

[308] Xiao Zhang, Haonan Jia, Hanjing Su, Wenhan Wang, Jun Xu, and Ji-Rong Wen. Counterfactual reward modification for streaming recommendation with delayed feedback. In *Proceedings of the 44th international ACM SIGIR conference on research and development in information retrieval*, pages 41–50, 2021.

[309] Yi Zhang, Yichen Zhao, and Jie Tang. Roler: Effective reward shaping in offline reinforcement learning for recommender systems. In *Proceedings of the 48th International ACM SIGIR Conference*, 2024.

[310] Yifei Zhang, Hao Zhu, Zixing Song, Piotr Koniusz, Irwin King, et al. Mitigating the popularity bias of graph collaborative filtering: A dimensional collapse perspective. *Advances in Neural Information Processing Systems*, 36:67533–67550, 2023.

[311] Zeyu Zhang, Lu Li, Shuyan Wan, Wang Wang, Zhiyi Wang, Zhiyuan Lu, Dong Hao, and Wanli Li. Dropedge not foolproof: effective augmentation method for signed graph neural networks. *Advances in Neural Information Processing Systems*, 37:117041–117069, 2024.

[312] Qian Zhao, Hao Qian, Ziqi Liu, Gong-Duo Zhang, and Lihong Gu. Breaking the barrier: utilizing large language models for industrial recommendation systems through an inferential knowledge graph. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, pages 5086–5093, 2024.

[313] Tong Zhao, Yozen Liu, Leonardo Neves, Oliver J. Woodford, Meng Jiang, and Neil Shah. Data augmentation for graph neural networks. In *AAAI Conference on Artificial Intelligence*, 2020. URL `https://api.semanticscholar.org/CorpusID:219635816`.

[314] Guanjie Zheng, Fuzheng Zhang, Zihan Zheng, Yang Xiang, Nicholas Jing Yuan, Xing Xie, and Zhenhui Li. Drn: A deep reinforcement learning framework for news recommendation. In *Proceedings of the 2018 world wide web conference*, pages 167–176, 2018.

[315] Jiaqi Zheng, Cheng Guo, Yi Cao, Chaoqun Hou, Tong Liu, and Bo Zheng. Usd: A user-intent-driven sampling and dual-debiasing framework for large-scale homepage recommendations. In *Proceedings of the Nineteenth ACM Conference on Recommender Systems*, pages 1108–1111, 2025.

[316] Chang Zhou, Jianxin Ma, Jianwei Zhang, Jingren Zhou, and Hongxia Yang. Contrastive learning for debiased candidate generation in large-scale recommender systems. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, KDD '21, page 3985–3995, New York, NY, USA, 2021. ACM. ISBN 9781450383325. doi: 10.1145/3447548.3467102. URL `https://doi.org/10.1145/3447548.3467102`.

[317] Guorui Zhou, Na Mou, Ying Fan, Qi Pi, Weijie Bian, Chang Zhou, Xiaoqiang Zhu, and Kun Gai. Deep interest evolution network for click-through rate prediction. *ArXiv*, abs/1809.03672, 2018. URL `https://api.semanticscholar.org/CorpusID:52188056`.

[318] Yunhong Zhou, Dennis Wilkinson, Robert Schreiber, and Rong Pan. Large-scale parallel collaborative filtering for the netflix prize. In *Algorithmic Aspects in Information and Management: 4th International Conference, AAIM 2008, Shanghai, China, June 23-25, 2008. Proceedings 4*, pages 337–348. Springer, 2008.

[319] Zhiyao Zhou, Sheng Zhou, Bochao Mao, Xuanyi Zhou, Jiawei Chen, Qiaoyu Tan, Daochen Zha, Yan Feng, Chun Chen, and Can Wang. Opengsl: A comprehensive benchmark for graph structure learning. *Advances in Neural Information Processing Systems*, 36:17904–17928, 2023.

[320] Zhiyuan Zhou, Andy Peng, Qiyang Li, Sergey Levine, and Aviral Kumar. Efficient online reinforcement learning fine-tuning need not retain offline data. *arXiv preprint arXiv:2412.07762*, 2024.

[321] Yanqiao Zhu, Weizhi Xu, Jinghao Zhang, Yuanqi Du, Jieyu Zhang, Qiang Liu, Carl Yang, and Shu Wu. A survey on graph structure learning: Progress and opportunities. *arXiv preprint arXiv:2103.03036*, 2021.

[322] Yanqiao Zhu, Weizhi Xu, Jinghao Zhang, Qiang Liu, Shu Wu, and Liang Wang. Deep graph structure learning for robust representations: A survey. *arXiv preprint arXiv:2103.03036*, 14: 1–1, 2021.

[323] Kevin Zielnicki and Ko-Jen Hsiao. Orthogonal low rank embedding stabilization. In *Proceedings of the Nineteenth ACM Conference on Recommender Systems*, pages 1030–1033, 2025.

[324] Oded Zinman, Nazmul Chowdhury, Leandro Fiaschetti, Yuri M Brovman, Guy Feigenblat, and Yotam Eshel. Personalized interest graphs for theme-driven user behavior. In *Proceedings of the Nineteenth ACM Conference on Recommender Systems*, pages 1038–1041, 2025.

[325] Dongcheng Zou, Hao Peng, Xiang Huang, Renyu Yang, Jianxin Li, Jia Wu, Chunyang Liu, and Philip S Yu. Se-gsl: A general and effective graph structure learning framework through structural entropy optimization. In *Proceedings of the ACM Web Conference 2023*, pages 499–510, 2023.

[326] Kuan Zou and Aixin Sun. A survey of real-world recommender systems: Challenges, constraints, and industrial perspectives. *arXiv preprint arXiv:2509.06002*, 2025.

Acknowledgements

The completion of this thesis marks the end of a long and rewarding journey into the field of graph recommender systems, an experience that has profoundly shaped me as a researcher and as a person. This journey was never a solitary one, and I owe my deepest gratitude to the many people who walked alongside me, offered guidance, and provided support. This work is a testament not only to my own efforts but to the collective encouragement, patience, and wisdom of a network of mentors, colleagues, and friends. Without them, this manuscript would simply not exist.

This entire journey was first made possible by the foundational partnership between TIM and Sapienza University of Rome. I am especially grateful to Paolo and Clara, who saw a spark of passion in me and worked to transform it into a tangible research opportunity. They believed in this project from its beginning and, in doing so, believed in me. To Daniele, for his gentle guidance and for allowing this academic pursuit to grow alongside my professional responsibilities. His trust gave me the serenity and balance needed to dedicate myself to both worlds, and for that, I am deeply thankful.

My intellectual journey was shaped by my mentors, above all my supervisor, Professor Fabrizio Silvestri. When I first joined the group, I was quite isolated. He was the one who consistently looked out for me, not just as a scientist, but as a person. That foundation of support was crucial, and I learned so much from him, not only because of his deep knowledge, but because his enthusiasm made me feel the work truly mattered. Alongside him, Federico served as a constant scientific tutor, and confidant, a role he embraced when we began collaborating in my second year. He became my essential bridge between the worlds of industry and academia, a vital link without which this industrial PhD would have been a far more solitary and challenging path. Much of my growth as a researcher is a direct result of his patient mentorship and continuing support, and for his teachings, I am immensely grateful.

This path began long before my formal PhD, alongside friends and colleagues. A special thank you to Matteo, a colleague and a friend. Together, we conceived the ideas that would later become the CRAB project. He has never stopped believing in me as a researcher, and his friendship has been a cornerstone of this experience. My thanks also go to all my university friends. Though our academic paths diverged after our Master's, the bond we formed through shared curiosity and countless hours of study has remained. Their friendship has been a source of support and has made this entire journey a pleasure.

None of this would have been possible without the bedrock of my family. To my parents, who have always trusted my choices, even from my earliest university days when our paths seemed to diverge. Your willingness to let me follow my own instincts has been the greatest gift, and I owe everything to the lessons of trust and rationality you taught me. To my brother, Luigi, for always being there when I needed support.

Finally, to Nina. For three years, you have been my constant through all the ups and downs. Your faith in me was unwavering, and your support unconditional. Our debates and discussions were always a positive influence. Thank you for being there through it all.

This work is truly the work of many minds and many hearts. It is my hope that this thesis serves as a testament to our collective efforts in advancing industrial recommender systems. Thank you all.

Alessandro Sbandi, Rome, October 2025