



Politecnico
di Bari

Repository Istituzionale dei Prodotti della Ricerca del Politecnico di Bari

From structural mitigation to semantic completion: advancing recommender systems under data sparsity

This is a PhD Thesis

Original Citation:

From structural mitigation to semantic completion: advancing recommender systems under data sparsity / Bufi, Salvatore. - ELETTRONICO. - (2026).

Availability:

This version is available at <http://hdl.handle.net/11589/296020> since: 2026-01-21

Published version

DOI:

Publisher: Politecnico di Bari

Terms of use:

(Article begins on next page)



**Politecnico
di Bari**

Department of Electrical and Information Engineering
ELECTRICAL AND INFORMATION ENGINEERING PH.D. PROGRAM
SSD: ING-INF/05 - INFORMATION PROCESSING SYSTEMS

Final Dissertation

From Structural Mitigation to Semantic Completion: Advancing Recommender Systems under Data Sparsity.

by

Salvatore Bui

Supervisor

Prof. Tommaso Di Noia

Coordinator of the Ph.D. Program

Prof. Mario Carpentieri

Course n° 38, 01/11/2022 - 31/10/2025



LIBERATORIA PER L'ARCHIVIAZIONE DELLA TESI DI DOTTORATO

Al Magnifico Rettore
del Politecnico di Bari

Il sottoscritto Salvatore Bufi nato a Molfetta il 18/04/1994

residente a Molfetta in via Antonio Vivaldi n° 22 e-mail bufi.salvatore.94@gmail.com

iscritto al 3° anno di Corso di Dottorato di Ricerca in Ingegneria Elettrica e dell'Informazione ciclo n° 38

ed essendo stato ammesso a sostenere l'esame finale con la prevista discussione della tesi dal titolo:

From Structural Mitigation to Semantic Completion: Advancing Recommender Systems under Data Sparsity.

DICHIARA

- 1) di essere consapevole che, ai sensi del D.P.R. n. 445 del 28.12.2000, le dichiarazioni mendaci, la falsità negli atti e l'uso di atti falsi sono puniti ai sensi del codice penale e delle Leggi speciali in materia, e che nel caso ricorressero dette ipotesi, decade fin dall'inizio e senza necessità di nessuna formalità dai benefici conseguenti al provvedimento emanato sulla base di tali dichiarazioni;
- 2) di essere iscritto al Corso di Dottorato di ricerca in Ingegneria Elettrica e dell'Informazione ciclo n° 38, corso attivato ai sensi del "Regolamento dei Corsi di Dottorato di ricerca del Politecnico di Bari", emanato con D.R. n.286 del 01.07.2013;
- 3) di essere pienamente a conoscenza delle disposizioni contenute nel predetto Regolamento in merito alla procedura di deposito, pubblicazione e autoarchiviazione della tesi di dottorato nell'Archivio Istituzionale ad accesso aperto alla letteratura scientifica;
- 4) di essere consapevole che attraverso l'autoarchiviazione delle tesi nell'Archivio Istituzionale ad accesso aperto alla letteratura scientifica del Politecnico di Bari (IRIS-POLIBA), l'Ateneo archiverà e renderà consultabile in rete (nel rispetto della Policy di Ateneo di cui al D.R. 642 del 13.11.2015) il testo completo della tesi di dottorato, fatta salva la possibilità di sottoscrizione di apposite licenze per le relative condizioni di utilizzo (di cui al sito <http://www.creativecommons.it/Licenze>), e fatte salve, altresì, le eventuali esigenze di "embargo", legate a strette considerazioni sulla tutelabilità e sfruttamento industriale/commerciale dei contenuti della tesi, da rappresentarsi mediante compilazione e sottoscrizione del modulo in calce (Richiesta di embargo);
- 5) che la tesi da depositare in IRIS-POLIBA, in formato digitale (PDF/A) sarà del tutto identica a quelle **consegnate**/inviata/di inviarsi ai componenti della commissione per l'esame finale e a qualsiasi altra copia depositata presso gli Uffici del Politecnico di Bari in forma cartacea o digitale, ovvero a quella da discutere in sede di esame finale, a quella da depositare, a cura dell'Ateneo, presso le Biblioteche Nazionali Centrali di Roma e Firenze e presso tutti gli Uffici competenti per legge al momento del deposito stesso, e che di conseguenza va esclusa qualsiasi responsabilità del Politecnico di Bari per quanto riguarda eventuali errori, imprecisioni o omissioni nei contenuti della tesi;
- 6) che il contenuto e l'organizzazione della tesi è opera originale realizzata dal sottoscritto e non compromette in alcun modo i diritti di terzi, ivi compresi quelli relativi alla sicurezza dei dati personali; che pertanto il Politecnico di Bari ed i suoi funzionari sono in ogni caso esenti da responsabilità di qualsivoglia natura: civile, amministrativa e penale e saranno dal sottoscritto tenuti indenni da qualsiasi richiesta o rivendicazione da parte di terzi;
- 7) che il contenuto della tesi non infrange in alcun modo il diritto d'Autore né gli obblighi connessi alla salvaguardia di diritti morali od economici di altri autori o di altri aventi diritto, sia per testi, immagini, foto, tabelle, o altre parti di cui la tesi è composta.

Luogo e data _____ 19/01/2026

Firma _____ 

Il/La sottoscritto, con l'autoarchiviazione della propria tesi di dottorato nell'Archivio Istituzionale ad accesso aperto del Politecnico di Bari (POLIBA-IRIS), pur mantenendo su di essa tutti i diritti d'autore, morali ed economici, ai sensi della normativa vigente (Legge 633/1941 e ss.mm.ii.),

CONCEDE

- al Politecnico di Bari il permesso di trasferire l'opera su qualsiasi supporto e di convertirla in qualsiasi formato al fine di una corretta conservazione nel tempo. Il Politecnico di Bari garantisce che non verrà effettuata alcuna modifica al contenuto e alla struttura dell'opera.
- al Politecnico di Bari la possibilità di riprodurre l'opera in più di una copia per fini di sicurezza, back-up e conservazione.

Luogo e data _____ 19/01/2026

Firma _____ 

To those who lost themselves,

*yet found the strength
to love the person
in the mirror again.*

Abstract

In the modern digital ecosystem, Recommender Systems (RSs) play a crucial role by learning user preferences from historical interaction data to guide individuals through vast catalogs of content. Despite their widespread adoption, their effectiveness remains fundamentally constrained by the inherent problem of data sparsity. While Knowledge Graphs (KGs) have emerged as a powerful source for enriching sparse collaborative signals, existing approaches often process the entire KG uniformly, learning generic user intents from the whole set of available semantic features. Furthermore, the effectiveness of these models relies on the completeness of the underlying KG, yet real-world KGs are frequently incomplete, creating an upstream challenge of semantic completion. Compounding these issues are emerging challenges where data is intentionally limited, driven by privacy regulations and the push for “Green AI”, and a persistent lack of methodological standardization in the research community.

This dissertation identifies and addresses four research gaps: 1) the tendency of existing Knowledge-aware RSs to process the entire KG rather than distilling personalized semantic signals; 2) the fundamental reliance of these models on the completeness of the underlying Knowledge Graph, which is often an unrealistic assumption; 3) the unexamined consequences of “imposed data sparsity”; and 4) the lack of standardized data management. To address these gaps, this dissertation presents the following contributions:

First, we address structural mitigation by learning user-aligned knowledge. We introduce a graph-based recommender that infers personalized knowledge signals by modeling individual user decision processes, yielding finer personalization and feature-level interpretability. This is complemented by a lightweight simplification that removes explicit intent modules while retaining competitive accuracy.

Second, we focus on the semantic completion of knowledge graphs. We tackle inductive link prediction on KGs when explicit type annotations are coarse or missing by eliciting implicit type cues from Pretrained Language Models (PLMs), improving robustness to type and structural sparsity, and strengthening downstream knowledge-aware recommenders.

Third, we investigate scenarios where data availability is intentionally limited. We systematically audit how various data minimization techniques affect the performance of different model families, evaluating their robustness in these constrained environments.

Finally, to close the reproducibility gap, this work introduces a library for standardized and reproducible data management, designed to unify data handling practices across the research community.

Publications

Certain ideas and figures presented in this thesis have been previously published in other works. Below is a comprehensive list of the Ph.D. candidate's publications. In this list, the Ph.D. candidate's name is highlighted in **boldface** wherever he is the corresponding author or one of the corresponding authors.

- [1] Alessandro De Bellis, **Salvatore Bufi**, Giovanni Servedio, Vito Walter Anelli, Tommaso Di Noia, and Eugenio Di Sciascio. "Type-Less yet Type-Aware Inductive Link Prediction with Pretrained Language Models". In: *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, EMNLP 2025*. Ed. by Christodoulopoulos Christos, Chakraborty Tanmoy, Rose Carolyn, and Peng Violet. Suzhou, China: Association for Computational Linguistics, Nov. 2025, pp. 27181–27197. ISBN: 979-8-89176-332-6.
- [2] Alberto Carlo Maria Mancino, Antonio Ferrara, **Salvatore Bufi**, Daniele Malitesta, Tommaso Di Noia, and Eugenio Di Sciascio. "KGTORe: Tailored Recommendations through Knowledge-aware GNN Models". In: *Proceedings of the 17th ACM Conference on Recommender Systems, RecSys 2023, Singapore, Singapore, September 18–22, 2023*. Ed. by Jie Zhang, Li Chen, Shlomo Berkovsky, Min Zhang, Tommaso Di Noia, Justin Basilico, Luiz Pizzato, and Yang Song. ACM, 2023, pp. 576–587. DOI: 10.1145/3604915.3608804.
- [3] Alberto Carlo Maria Mancino, **Salvatore Bufi**, Angela Di Fazio, Antonio Ferrara, Daniele Malitesta, Claudio Pomo, and Tommaso Di Noia. "DataRec: A Python Library for Standardized and Reproducible Data Management in Recommender Systems". In: *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2025, Padua, Italy, July 13–18, 2025*. Ed. by Nicola Ferro, Maria Maistro, Gabriella Pasi, Omar Alonso, Andrew Trotman, and Suzan Verberne. ACM, 2025, pp. 3478–3487. DOI: 10.1145/3726302.3730320.
- [4] **Salvatore Bufi**, Alberto Carlo Maria Mancino, Antonio Ferrara, Daniele Malitesta, Tommaso Di Noia, and Eugenio Di Sciascio. "KGUF: Simple Knowledge-Aware Graph-Based Recommender with User-Based Semantic Features Filtering". In: *Advances on Graph-Based Approaches in Information Retrieval, IRonGraphs 2024, Glasgow*,

UK, March 24, 2024, *Proceedings*. Ed. by Ludovico Boratto, Daniele Malitesta, Mirko Marras, Giacomo Medda, Cataldo Musto, and Erasmo Purificato. Vol. 2197. Communications in Computer and Information Science. Springer, 2024, pp. 41–59. DOI: 10.1007/978-3-031-71382-8_4.

- [5] **Salvatore Bufi**, Vincenzo Paparella, Vito Walter Anelli, Franco Maria Nardini, Raffaele Perego, and Tommaso Di Noia. “Explaining GNN-based Recommender Systems Performance under Data Minimization: The Role of Data Characteristics”. Manuscript submitted for publication to the journal *Expert Systems with Applications*. 2025.
- [6] **Salvatore Bufi**, Vincenzo Paparella, Vito Walter Anelli, and Tommaso Di Noia. “Legal but Unfair: Auditing the Impact of Data Minimization on Fairness and Accuracy Trade-off in Recommender Systems”. In: *Proceedings of the 33rd ACM Conference on User Modeling, Adaptation and Personalization, UMAP 2025, New York City, NY, USA, June 16-19, 2025. Best Student Paper Award*. ACM, 2025, pp. 114–123. DOI: 10.1145/3699682.3728356.
- [7] **Salvatore Bufi**, Vincenzo Paparella, Vito Walter Anelli, and Tommaso Di Noia. “Less Data, More Questions: Fairness and Accuracy Under Data Minimization in Recommender Systems”. In: *Proceedings of the 15th Italian Information Retrieval Workshop (IIR 2025)*. Ed. by Ludovico Boratto, Allegra De Filippo, Stefano Marchesin, and Mirko Marras. Cagliari, Italy: CEUR Workshop Proceedings, 2025, pp. 129–135.
- [8] **Salvatore Bufi**, Vincenzo Paparella, Alessandro De Bellis, Vito Walter Anelli, Franco Maria Nardini, Raffaele Perego, and Tommaso Di Noia. “Bridging Latent Embeddings and Interpretability with a Linear Factor Model for Recommender Systems”. Manuscript in preparation. The work is presented in Appendix A of this dissertation. 2025.

Contents

1	Introduction	19
1.1	Thesis Statement	21
1.2	Research Contributions	22
1.2.1	Ch. 3: Learning User-Aligned Knowledge Signals for Personalized Graph Recommendation	22
1.2.2	Ch. 4: Lightweight Knowledge-Enhanced Graph Recommendation	22
1.2.3	Ch. 5: Completing Incomplete Knowledge Graphs via Inductive Link Prediction	23
1.2.4	Ch. 6: Auditing Fairness and Accuracy Trade-Off under Imposed Data Sparsity	23
1.2.5	Ch. 7: Assessing the Role of Data Characteristics in Recommender Performance Under Sparsity	24
1.2.6	Ch. 8: Toward Standardized and Reproducible Data Management in Recommender Systems	24
I	Background	27
2	Recommender Systems	29
2.1	Definition of the Recommendation Problem	29
2.2	Recommendation Techniques	31
2.2.1	Collaborative Filtering Approaches	31
2.2.2	Graph-based Approaches	34
2.2.3	Content-based Approaches	37
2.2.4	Hybrid Approaches	38
2.3	Recommendation Pipeline	39
2.3.1	Input Pre-processing	39
2.3.2	Model Optimization	40

2.3.3	Model Evaluation.....	42
-------	-----------------------	----

II Structural Mitigation with Knowledge Graphs under Data Sparsity 51

3	Learning User-Aligned Knowledge Signals for Personalized Graph Recommendation	53
3.1	Introduction.....	54
3.2	Preliminaries	55
3.3	Approach	57
3.3.1	Item Modeling.....	57
3.3.2	User Decision Process Modeling	57
3.3.3	Knowledge-aware Graph-based Recommendation	60
3.3.4	Model Optimization.....	62
3.4	Experiments	63
3.4.1	Datasets	63
3.4.2	Baselines	64
3.4.3	Knowledge Graph and Decision Trees	65
3.4.4	Reproducibility, Evaluation, and Metrics.....	66
3.5	Discussion	67
3.5.1	Performance Comparison with State-of-the-Art Baselines (RQ1)	67
3.5.2	Ablation and Hyperparameter Studies (RQ2).....	68
3.5.3	Interpretability of KGTORe (RQ3).....	71
3.6	Conclusion and Future Work.....	73
4	Lightweight Knowledge-Enhanced Graph Recommendation	75
4.1	Introduction.....	76
4.2	Related work	78
4.2.1	KGs information in recommendation	78
4.2.2	Graph representation learning for recommendation	78
4.2.3	KGs information and graph representation learning for recommendation	79
4.3	Background.....	79
4.3.1	Knowledge graphs for recommendation	79
4.3.2	Decision trees.....	80
4.4	Methodology	80
4.4.1	Modeling items through knowledge graph.....	80
4.4.2	Refining item features with user decision	81
4.4.3	Knowledge-aware aggregation.....	81
4.4.4	Model learning	82

4.5	Experimental settings	83
4.5.1	Datasets and Knowledge Graph.....	83
4.5.2	Baselines	84
4.5.3	Evaluation protocol and reproducibility	84
4.6	Results and discussion	85
4.6.1	RQ1: Overall performance	85
4.6.2	RQ2: Decision Tree Construction	88
4.7	Conclusion and future work.....	89
III	Semantic Completion of Knowledge Graphs	91
5	Completing Incomplete Knowledge Graphs via Inductive Link Prediction	93
5.1	Introduction.....	94
5.2	Background and Motivation.....	95
5.3	Methodology	96
5.4	Experimental Setup	99
5.4.1	Datasets	99
5.4.2	Metrics.....	101
5.4.3	Models	101
5.4.4	Hyperparameter Details.....	102
5.5	Results	102
5.5.1	Type Information in Subgraph-Based Inductive Link Prediction (RQ1)	103
5.5.2	Usefulness of PLM Representations for Implicit Type Signal (RQ2)	105
5.5.3	Effect on Type and Structural Sparsity (RQ3)	105
5.6	Related Work	107
5.7	Conclusion	108
IV	Operating under Imposed Data Sparsity	111
6	Auditing Fairness and Accuracy Trade-Off under Imposed Data Sparsity	113
6.1	Introduction.....	114
6.2	Related Work	115
6.3	Methodology	116
6.3.1	Data minimization strategies	116
6.4	Experimental Setup	117
6.4.1	Dataset	117
6.4.2	Evaluation Protocol	117
6.4.3	Algorithms and Metrics	118
6.4.4	Reproducibility Details.....	119

6.4.5	Data Minimization Strategies.....	120
6.5	Results and Discussion.....	121
6.5.1	Impact of Data Minimization (RQ1).....	121
6.5.2	Data Minimization Strategies Performance (RQ2).....	126
6.5.3	Robustness of Recommenders to Data Minimization (RQ3).....	127
6.6	Conclusion and Future Work.....	128
7	Assessing the Role of Data Characteristics in Recommenders Performance Under Sparsity	131
7.1	Introduction.....	132
7.2	Data Minimization in Recommender Systems.....	133
7.3	Dataset Characteristics.....	134
7.3.1	Structural and distributional characteristics	135
7.3.2	Topological characteristics.....	137
7.3.3	Minimization-Semantic Characteristics.....	140
7.4	Methodology	142
7.4.1	Data collection.....	142
7.4.2	Target Variable.....	143
7.4.3	Explanatory Variables	143
7.4.4	Explanatory Model.....	144
7.5	Experimental Setup	146
7.5.1	Datasets and Preprocessing	146
7.5.2	Models Under Investigations	147
7.5.3	Hyperparameter Optimization	148
7.5.4	Data Minimization Strategies.....	148
7.6	Results	149
7.6.1	Performance of GNN-based Models under Data Minimization (RQ1)	149
7.6.2	Relationship between Dataset Characteristics and Model Performance (RQ2).....	152
7.6.3	Specificity of Influential Characteristics to Data Minimization Strategies (RQ3).....	158
7.6.4	Augmenting the Feature Space with Minimization-Semantic Characteristics (RQ4).....	160
7.7	Conclusion	161

V Closing the Reproducibility Gap in Recommendation163

8	Toward Standardized and Reproducible Data Management in Recommender Systems	165
8.1	Introduction.....	166

8.2	Overview of Data Handling and Processing in Recommender Systems	167
8.2.1	Current Trends in Recommender Research	169
8.2.2	Existing Approaches in Recommendation Frameworks	171
8.3	DATARec	173
8.3.1	The I/O Module	174
8.3.2	The DataRec Class	176
8.3.3	Processing Module	177
8.3.4	Splitting Module	177
8.3.5	Reproducibility	177
8.4	Conclusion and Future Work	178

VI Conclusion 181

9	Closing Remarks	183
A	Appendix: Reconciling Latent Representations with Interpretability	185
A.1	Introduction	186
A.2	Related Work	187
A.2.1	Collaborative Filtering Techniques	187
A.2.2	Interpretability in Collaborative Filtering	189
A.3	Methodology	190
A.3.1	Problem Definition	190
A.3.2	Proposed Approach	190
A.3.3	Training Objectives	192
A.3.4	Interpretability of the Scoring Function	192
A.3.5	Model Analysis	194
A.4	Experimental Setup	195
A.4.1	Datasets	196
A.4.2	Baselines	197
A.4.3	Reproducibility, Evaluation, and Metrics	197
A.5	Results and Discussion	198
A.5.1	Performance Comparison (RQ1)	198
A.5.2	Ablation Studies (RQ2)	199
A.5.3	Hyper-parameter Analysis (RQ3)	201
A.5.4	Interpretability Study (RQ4)	203
A.6	Conclusion and Future Work	204

List of Figures

2.1	Rating prediction under explicit feedback. Left: a fully observed user–item matrix R with each entry $R_{u,i}$ equal to the rating of user u for item i . Right: a realistic sparse case with missing ratings; the task is to estimate the unobserved entries.	30
2.2	Interaction prediction under implicit feedback. Left: a user–item matrix R with $R_{u,i}$ indicating whether user u interacted with item i . Right: sparse observations; recommenders rank unobserved pairs and return the top- k items for each user.....	30
2.3	Exemplification of latent factor models. These models learn latent representations (embeddings) of users and items. Then, the user–item matrix is approximated by computing the dot product between each user–item embedding pair.....	33
2.4	A graphical representation of the long-tail phenomenon. The x-axis represents individual items, sorted in descending order of their interaction counts, which are shown on the y-axis. The "short head" (orange) highlights that a few popular items receive a very high number of interactions, while the "long tail" (teal) shows that the vast majority of items receive very few interactions each. The dashed line marks the transition between these two regions.	46
3.1	Toy example of a user-level decision tree. The user liked two movies and disliked three. Following the characteristics in the item feature sets, the decision tree is able to discriminate between liked and unliked items and understand how the user decides to accept or reject a movie	58

3.2	Representation of a knowledge-aware decision embedding (a) and a content-based item embedding (b) in a 2D feature space. In (a), the example user has chosen <i>Harry Potter</i> because of its director and Johnny Depp <i>not</i> starring: the embedding $p_{u, Harry\ Potter}$ is the sum of these feature embeddings. Starring <i>Daniel Radcliffe</i> does not impact the decision embedding, since it is not in $\mathcal{F}_{u, Harry\ Potter}$. In (b), the embedding is the average of the two features describing it	60
3.3	Illustration of KGTORe. The embeddings of the item features are averaged to build the item content-based representations $e_{i_1}^K$ and $e_{i_2}^K$; the user content-based representation e_u^K is obtained averaging the decision embeddings, in turns computed with positive and negative contributions from the embeddings of the decision paths	64
3.4	Accuracy performance (nDCG) of KGTORe and the baselines on <i>Yahoo! Movies</i> when separately computed on four distinct user groups, corresponding to the quartiles of the number of their interactions.	69
3.5	Impact of the knowledge-aware terms into the user and item representations at different values of α and β on <i>Yahoo! Movies</i>	69
3.6	Performance of KGTORe with and without independence loss. The bars show an accuracy decrease over the three datasets	72
4.1	An example of how KGUF models the item representation. Figure 4.1a shows the entities in the knowledge graph linked to the items. In Figure 4.1b, the most relevant semantic features selected by KGUF are reported	77
4.2	Performance of KGUF regarding nDCG@10 when varying the knowledge contribution. Figure 4.2a reports the impact of the parameter α , which weights the knowledge contribution. Figure 4.2b shows the impact of considering the knowledge or the collaborative signals only	87
4.3	Impact of negative sampling and tree depth on KGUF performance in terms of nDCG@10. Figure 4.3a reports the nDCG@10 achieved for different values of η , i.e., the ratio between sampled negative and positive items, while Figure 4.3b shows the nDCG@10 obtained with varying maximum decision-tree depths (max depth)	89
5.1	Overview of TyleR. The process begins with ① extracting the enclosing subgraph and ② applying a node labeling strategy. Multi-faceted, semantic representations are then derived using a pre-trained language model ③, ④. Finally, a graph neural network ⑤ integrates structural and semantic information to obtain the final prediction.	97
5.2	Link Prediction (Hits@10) evaluation under varying structural sparsity conditions (i.e., the number of edges L in the enclosing subgraph of the target triple, including the target triple) on FB237-V1 (top) and YAGO21K-610 (bottom).	106

5.3	Hits@10 performance across four type sparsity groups for three FB237 variants, computed according to the number of explicit types linked to each entity (details in section 5.5.3). The groups, from left to right, represent scenarios with an increasing number of explicit types associated with the known entity	107
6.1	Comparison of the Euclidean distance (δ) between the point representing the model performance under a minimization strategy with value n and the point representing the model performance under the “full” data strategy. The colored lines represent the different minimization strategy. Smaller δ values indicate better preservation of the model’s overall performance	124
8.1	Overview of the DATARec architecture. The DataRec class provides key dataset metrics and interacts with three modules: I/O for handling different data formats, processing for dataset transformations, and splitting for partitioning into training, validation, and test sets	173
A.1	Performance comparison of ILF-Rec with the full formulation (w/ $\mathcal{L}_{\text{ORTHO}}$) versus its variant excluding the orthogonality term (w/o $\mathcal{L}_{\text{ORTHO}}$). All differences are statistically significant, as determined by paired t-tests ($p < 0.05$)	200
A.2	Performance comparison of ILF-Rec with and without dedicated item embeddings. The full model (w/ E_i) employs a separate item embedding matrix, whereas the ablated variant (w/o E_i) uses the transformation matrix W for both projection and item representation. All differences are statistically significant, as determined by paired t-tests ($p < 0.05$)	200
A.3	nDCG@10 performance of ILF-Rec by varying the number of factors k and the coefficients λ_r and λ_o	202
A.4	Examples of interpretable recommendations of ILF-Rec. The lists correspond to the top five influencing movies to the recommendation of i_{3114} (Toy Story 2) to the users u_{168} and u_{351}	203

List of Tables

3.1	Comparison of accuracy performance between KGTORe and competing methods. All the differences between KGTORe and the baselines are statistically significant based on paired t-tests ($p < 0.05$) except for the values denoted with *	67
3.2	Performance of KGTORe on <i>Yahoo! Movies</i> when compromising the decision tree outcome	71
3.3	Semantic features ordered by affinity with the test user.	72
4.1	Statistics of tested datasets.	84
4.2	Accuracy performance comparison of KGUF with the selected baselines. All the models are grouped into their respective categories.	86
5.1	Statistics of the datasets used in our experiments. The YAGO21K-610 [314] dataset includes ontology triples and entity-type links, while the FB237 dataset variants [235] are further processed to extract ontology triples, type links and textual labels.	100
5.2	Assertion prompts (p_1 - p_6) used in the semantic enrichment step (section 5.3). These templates, with a placeholder for the entity, are fed to the Pre-trained Language Model to elicit representations capturing different semantic aspects (type, geographic context, membership, equivalence, difference, similarity) of the entity.	101
5.3	Link Prediction (LP) evaluation on multiple FB237 variants and YAGO21K-610. Best and second-best scores are in bold and <u>underlined</u> , respectively. Evaluation uses the strictest tie-breaking policy (section 5.4.2), assigning the highest (worst) possible rank to the positive triple in case of ties	102
5.4	Ablation study on the FB237-V1 dataset, evaluating the impact of different Pre-trained Language Models and aggregation functions (Equation 5.4) for semantic embeddings within TyleR. 'TYPE-ONLY' uses only the representation from the p_1 prompt (table 5.2).	103

5.5	Performance of Llama3-8B when directly evaluating the likelihood of verbalized triples on the YAGO21K-610 dataset by scoring them using the (negated) model perplexity.....	103
5.6	Ablation study on the FB237-V1 dataset, evaluating the impact of chosen number of prompts (Equation 5.4) for semantic embeddings within TyleR. For all configurations, the aggregation function was set to SUM.....	104
6.1	Performance results of all models on the ML1M dataset across different minimization strategies and values of n . The percentage difference between each value and the corresponding value obtained with the full strategy is indicated below each result. For each scenario, bold and underline stand for best and second-to-best values, respectively. All results are statistically significant based on paired t-tests ($p < 0.05$), except for the values denoted with †	122
6.2	Performance results of all models on the Ambar dataset across different minimization strategies and values of n . The percentage difference between each value and the corresponding value obtained with the full strategy is indicated below each result. For each scenario, bold and underline stand for best and second-to-best values, respectively. All results are statistically significant based on paired t-tests ($p < 0.05$), except for the values denoted with †	123
6.3	Mean (μ) and standard deviation (σ) of the distances among the points represented by each model performance under different strategies and the point corresponding to the model performance under the “full” data strategy, across different n . For each scenario, bold and underline stand for best and second-to-best values, respectively	126
7.1	Summary of the dataset characteristics analyzed in this work. Each metric is associated with its notation and a concise description of what it quantifies	144
7.2	Key statistics of datasets	146
7.3	Performance results in terms of Recall of all models on the Yelp dataset across different minimization strategies and values of β . Bold values indicate that a GNN-based model outperforms all non-GNN models for a given strategy and β . Underlined values indicate that a GNN-based model outperforms all but one non-GNN model	150
7.4	Performance results in terms of Recall of all models on the Amazon-Book dataset across different minimization strategies and values of β . Bold values indicate that a GNN-based model outperforms all non-GNN models for a given strategy and β . Underlined values indicate that a GNN-based model outperforms all but one non-GNN model.....	151

7.5	Variance Inflation Factor (VIF) values before and after removing highly collinear features for each dataset and model. Features with VIFs > 5 were excluded to ensure regression stability. After correction, all remaining characteristics show acceptable VIFs, confirming the absence of harmful multicollinearity.	153
7.6	Comparison between regression models fitted on subsampled datasets and under data minimization conditions for LightGCN, UltraGCN, and GFCF, across the Amazon-Book and Yelp datasets. We consider structural, distributional, and topological characteristics to train the regression model. The table reports coefficients, significance levels (* for $p < 0.05$, ** for $p < 0.01$, and *** for $p < 0.001$), and goodness-of-fit metrics (R^2 , R^2_{adj}). The symbol “ X ” indicates that the corresponding feature is removed after the sanity check due to multicollinearity.	155
7.7	Comparison between regression models fitted on subsampled datasets and under data minimization conditions for LightGCN, UltraGCN, and GFCF, across the Amazon-Book and Yelp datasets. Beyond structural, distributional, and topological characteristics, we consider also minimization-semantic characteristics to train the regression model. The table reports coefficients, significance levels (* for $p < 0.05$, ** for $p < 0.01$, and *** for $p < 0.001$), and goodness-of-fit metrics (R^2 , R^2_{adj}). The symbol “ X ” indicates that the corresponding feature is removed after the sanity check due to multicollinearity.	162
8.1	Datasets, pre-processing and splitting strategies adopted in 55 research papers published between 2020 and 2024. Only datasets and strategies appearing in at least 3 research papers are shown.	168
8.2	Dataset referencing in surveyed papers. The table reports the different ways datasets are cited, including references to the original data source, copies of the dataset, citations of the dataset’s original paper or other scholarly papers, missing references, and broken links.	169
8.3	Comparison of DATAREC with existing recommendation frameworks based on data input/output, pre-filtering and dataset splitting strategies. .	172
8.4	Overview of the datasets implemented in DATAREC. The table lists the dataset name, file format, available versions, and source links.	175
A.1	Comparison of CF baselines in terms of scoring function and space complexity ($S.C.$). We refer the reader to the original papers for additional details.	195
A.2	Performance comparison of ILF-Rec with the baselines. Boldface and underline refer to best and second-to-best values, respectively. All the differences between ILF-Rec and the baselines are statistically significant based on paired t-tests ($p < 0.05$) except for the values denoted with $\dagger$...	196

A.3	nDCG@10 performance comparison of ILF-Rec under different scaling schemes. Results are shown for no scaling (ILF-Rec w/ $\beta_{u_i} = 1$), symmetric scaling (ILF-Rec w/ $\beta_{u_i} = 1/\sqrt{ \mathcal{N}_u \mathcal{N}_i }$), user-centric scaling (ILF-Rec w/ $\beta_{u_i} = 1/\sqrt{ \mathcal{N}_u }$), and item-centric scaling (ILF-Rec w/ $\beta_{u_i} = 1/\sqrt{ \mathcal{N}_i }$). Differences relative to the no-scaling variant are statistically significant ($p < 0.05$, paired t-test), except for values marked with †. Boldface denotes the best nDCG values	200
-----	--	-----

Chapter 1

Introduction

In today’s digital landscape, individuals spend a substantial portion of their daily lives navigating an ever-growing ecosystem of platforms from e-commerce and food delivery to on-demand media and social feeds. This abundance of content has created a significant challenge known as *information overload*. For users, the costs of finding relevant content become prohibitively high. Simultaneously, this dynamic penalizes providers, as it results in reduced visibility for many of their products and services, making it harder to connect with a relevant audience.

Recommender systems (RS) [199] have become the canonical response to this challenge: by modeling user preferences and item characteristics, they deliver personalized and ranked suggestions from massive item catalogs, drastically simplifying content discovery and improving the quality of user choices. A wide array of recommendation strategies has been explored, ranging from heuristic and similarity-based approaches [182, 198, 212] to advanced machine learning and deep learning techniques [116, 140, 165]. Among the most widely adopted paradigms is Collaborative Filtering (CF) [83]. The core principle of CF is that users who have agreed in the past are likely to agree in the future; in other words, users who have interacted with similar items are likely to share preferences. However, this paradigm grapples with an intrinsic and persistent challenge: *data sparsity*. In any realistic setting, the interaction matrix is extremely sparse, as each user interacts with only a minimal fraction of the total available items. This lack of data severely degrades the quality of recommendations, collaborative signal alone is often too weak to reliably infer user preferences. To exploit more of the structure latent in interaction data, recent work recasts recommendation as learning on graphs [39, 115, 167]. By representing interactions as a bipartite user–item graph, *Graph Neural Networks* (GNNs) can mine high-order relations, empirically improving ranking quality across benchmarks. However, simply applying more powerful models is often not enough and does not remove the root causes of sparsity.

To this end, a complementary line of research moves beyond pure interaction data by incorporating side information such as reviews [178], social connections [22],

or images [11]. Among these, Knowledge Graphs (KGs) [121] have proven particularly effective, encoding semantic information as a graph of typed entities and relations. Linking catalog items to these KG entities provides a structured, semantically rich context that bridges sparse areas of the catalog and offers explanatory factors, ultimately leading to more accurate user recommendations. Since this process transforms the original user-item graph into a more complex, heterogeneous structure, GNNs have become the natural modeling choice. Consequently, a new class of *Knowledge-Aware Graph Collaborative Filtering*-based models (KGCF) [127, 174, 261] has emerged, demonstrating a remarkable ability to capture the nuanced relationships between users, items, and external entities. However, a critical limitation persists in their monolithic approach: they often process the entire KG and model user intents as a generic combination of all available semantic features. This "one-size-fits-all" strategy neglects the personal rationale behind user choices, leading to models that are computationally heavy and lack true personalization, pointing to the need for a more *lightweight and user-aligned paradigm*.

Furthermore, the effectiveness of KGCFs fundamentally relies on the quality and completeness of the underlying Knowledge Graph. Yet, real-world KGs are often *incomplete* or *coarse* [269], precisely where sparsity bites hardest. Naively injecting incomplete semantic information can degrade, rather than improve, recommendation performance [316, 322]. While inductive link prediction offers a path forward, many state-of-the-art methods rely heavily on explicit type annotations to guide their reasoning. This reliance becomes a critical bottleneck when such type information is itself sparse, noisy, or overly generic, shifting the core challenge toward *semantic completion in environments characterized by both structural and type sparsity*. To make KGCFs truly robust, it is not enough to select features from an existing KG; it becomes necessary to refine the KG itself, inferring missing relations even when explicit type annotations are absent, ensuring downstream models operate on a dense, more reliable semantic space.

Beyond purely technical considerations, a shifting regulatory landscape further complicates the picture: sparsity is not only a natural feature of data but also an imposed constraint. Regulatory frameworks, such as the GDPR [192], mandate strict data minimization principles, which require platforms to collect and retain only what is necessary for a specific purpose. For recommendation, this effectively enforces operation under systematically reduced interaction budgets, creating a form of "*imposed sparsity*" [31]. The reduction of data, in turn, is not without consequences. It can alter the distribution of users and items, introducing unintended shifts in recommendation outcomes whose effects remain a critical yet largely unexplored area of investigation. The literature lacks a systematic evaluation of how different data minimization strategies influence recommendation quality, and, crucially, an explanation of the underlying dataset characteristics that mediate these effects. Crucially, this perspective shifts the research focus from simple mitigation tactics to *understanding RS behavior when data becomes a constrained resource, not just a missing one*.

Finally, addressing these complex challenges is hindered by a foundational methodological friction: the lack of standardized and reproducible data management practices [63]. Heterogeneous preprocessing, splitting pipelines, and limited reporting transparency make it difficult to distinguish genuine modeling advances from experimental artifacts, underscoring an overarching *need for standardization*.

The remainder of this dissertation builds on this framing to formalize and address these challenges.

1.1 Thesis Statement

This dissertation advances the field of Recommender Systems by addressing the multifaceted challenge of data sparsity, moving from structural mitigation to semantic completion. This dissertation is organized into six parts, each consisting of chapters that build upon and elaborate on the core contributions of the author’s corresponding scientific papers.

The first part provides the necessary background, with Chapter 2 establishing the fundamental notions and definitions regarding Recommender Systems, crucial for understanding the discussions in subsequent chapters.¹

The second part, *Structural Mitigation with Knowledge Graphs under Data Sparsity*, focuses on leveraging external knowledge to enhance recommendations. Chapter 3 introduces a novel framework that learns user-aligned knowledge signals by modeling the user decision process to achieve fine-grained personalization. Chapter 4 builds on this by proposing a lightweight simplification that efficiently filters semantic features, demonstrating that simpler strategies can yield comparable or superior performance.

The third part shifts the focus upstream to the *Semantic Completion of Knowledge Graphs*. Chapter 5 presents an inductive link prediction framework that enriches incomplete knowledge graphs using Pre-trained Language Models, building a more robust semantic foundation for downstream recommendation tasks in sparse environments.

The fourth part addresses the emerging challenge of *Operating under Imposed Data Sparsity*. Chapter 6 audits the complex trade-offs between accuracy and fairness that arise from data minimization principles. Chapter 7 further deepens this analysis by developing an explanatory framework to identify the specific data characteristics that drive model performance in these resource-constrained settings.

Finally, the fifth part, *Closing the Reproducibility Gap in Recommendation*, tackles methodological challenges in the field. Chapter 8 introduces a novel, open-source library for standardized and reproducible data management, designed to foster more reliable and transparent experimental practices.

1. To improve readability without requiring constant cross-referencing, certain definitions are intentionally revisited across chapters. In some cases, they have been slightly reformulated to better align with the specific focus and narrative of each chapter.

To end, the final chapter concludes this thesis by synthesizing the insights and advancements discussed within this dissertation.

1.2 Research Contributions

This section details the primary scientific contributions of this thesis, presented thematically to align with each chapter. For every contribution, a summary outlines its core concepts, followed by a reference to the associated scientific publication and a description of the personal role of the Ph.D. candidate, Salvatore Bufi, in the work.

1.2.1 *Ch. 3: Learning User-Aligned Knowledge Signals for Personalized Graph Recommendation*

Contributions This chapter proposes a fundamental shift in how knowledge-aware GNNs interpret user-item interactions. It addresses a critical gap: existing models either leave the rationale of user choices unexplored or rely on flawed assumptions of static, shared intents. The proposed framework enriches each edge of the user-item graph with an explicit, personalized semantic meaning. It leverages user-level decision trees to model the specific decision process behind each interaction, identifying the presence or absence of item features that led to a choice. Crucially, this user-centric distillation frees the model from the need to process the entire knowledge graph at training time. This allows the GNN to be trained on targeted, decision-aware signals rather than simple connections, resulting not only in a more lightweight and scalable architecture but also a more nuanced understanding of user preferences and significant gains in recommendation accuracy.

Publications The work in this chapter is based on “KGTORe: Tailored Recommendations through Knowledge-aware GNN Models”, published in the Proceedings of the 17th ACM Conference on Recommender Systems (RecSys 2023) [162].

Ph.D. Candidate’s role The Ph.D. candidate is the corresponding author for this work [162].

1.2.2 *Ch. 4: Lightweight Knowledge-Enhanced Graph Recommendation*

Contributions Building on the previous chapter’s approach, this chapter investigates the necessity of simplification within KGCF models. It proposes a framework that achieves competitive performance through a more streamlined formalization, questioning whether the explicit modeling of user intents is essential. The method repurposes decision trees as a filtering mechanism to retain only the most historically relevant knowledge graph features. These distilled semantic signals are then integrated directly into the item representation during the graph learning phase. This results in a more lightweight and efficient architecture, demonstrating that a care-

fully designed simplification can achieve comparable or even superior performance to more complex intent-aware models.

Publications The work in this chapter is based on “KGUF: Simple Knowledge-Aware Graph-Based Recommender with User-Based Semantic Features Filtering”, published in the International Workshop on Graph-Based Approaches in Information Retrieval (IRonGraphs 2024) [206].

Ph.D. Candidate’s role The Ph.D. candidate is the corresponding author for this work [206].

1.2.3 Ch. 5: Completing Incomplete Knowledge Graphs via Inductive Link Prediction

Contributions Shifting the focus upstream, this chapter addresses the problem of incomplete knowledge graphs, which limits the effectiveness of knowledge-aware models. It introduces an inductive link prediction framework that does not rely on explicit type annotations, which are often sparse or unavailable. The framework leverages pre-trained language models (PLMs) to enrich node representations with implicit type signals enabling robust semantic completion even in environments characterized by high structural and type sparsity.

Publications The work in this chapter is based on “Type-Less yet Type-Aware Inductive Link Prediction with Pretrained Language Models”, published in the Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing (EMNLP 2025) [25].

Ph.D. Candidate’s role The Ph.D. candidate is the corresponding author for this work [25].

1.2.4 Ch. 6: Auditing Fairness and Accuracy Trade-Off under Imposed Data Sparsity

Contributions This chapter investigates scenarios where data availability is intentionally limited. A systematic, multi-objective audit is conducted to dissect the complex interplay between various data minimization strategies and their impact on the competing objectives of accuracy, consumer fairness, and provider fairness. The findings demonstrate that data minimization can enhance group-level consumer fairness at the expense of accuracy, while simultaneously undermining the fair exposure of items. Furthermore, the analysis provides preliminary evidence that model architectures vary in their resilience to data reduction, with graph-based models in particular demonstrating notable robustness.

Publications The work in this chapter is based on “Legal but Unfair: Auditing the Impact of Data Minimization on Fairness and Accuracy Trade-off in Recommender Systems”, published in the Proceedings of the 33rd ACM Conference on User Modeling, Adaptation and Personalization (UMAP 2025) and honored with the Best Student Paper Award [208]. A condensed version of the work has been published at

the 15th edition of the Italian Information Retrieval Workshop (IIR) 2025 in the paper titled “Less Data, More Questions: Fairness and Accuracy Under Data Minimization in Recommender System” [209].

Ph.D. Candidate’s role The Ph.D. candidate is the corresponding author for these works [208, 209].

1.2.5 *Ch. 7: Assessing the Role of Data Characteristics in Recommender Performance Under Sparsity*

Contributions Building on the preliminary evidence from the previous chapter, this work investigates the underlying factors that determine the performance of GNN-based recommender systems under data minimization. To this end, the chapter introduces a comprehensive explanatory framework that links model effectiveness to a wide range of dataset characteristics—structural, topological, and semantic. The analysis reveals that a small set of global structural characteristics, such as dataset scale and density, are the primary drivers of GNN performance, dominating other factors. Crucially, the chapter demonstrates that the significance of these characteristics is not universal but is a direct consequence of the semantic criteria guiding the minimization process. Building on this insight, the chapter proposes and validates a novel family of “minimization-semantic” characteristics designed to explicitly encode the logic of the retention criteria. The inclusion of these features is shown to substantially enhance the model’s explanatory power, providing actionable guidelines for designing more effective minimization strategies.

Publications The work is presented in the paper “Explaining GNN-based Recommender Systems Performance under Data Minimization: The Role of Data Characteristics,” submitted for publication to the Expert Systems with Applications journal (ESWA).

Ph.D. Candidate’s role The Ph.D. candidate is the corresponding author for this work [207].

1.2.6 *Ch. 8: Toward Standardized and Reproducible Data Management in Recommender Systems*

Contributions Recognizing that a lack of standardized data management fundamentally limits research reproducibility, this chapter sets out to build a common methodological foundation for the recommender systems community. The work begins by systematically diagnosing this issue through a comprehensive review of the contemporary research landscape, which reveals widespread inconsistencies in data handling pipelines—from preprocessing and filtering to splitting strategies. Informed by the critical gaps identified in this analysis, this chapter then introduces an open-source library designed to unify and streamline these workflows. By promoting methodological standardization and interoperability across different experimental setups, this library enable fairer benchmarking across the research

community.

Publications The work in this chapter is based on “DataRec: A Python Library for Standardized and Reproducible Data Management in Recommender Systems”, published in the Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2025) [163].

Ph.D. Candidate’s role The Ph.D. candidate is the corresponding author for this work [163].

Part I

Background

Chapter 2

Recommender Systems

In an era of vast digital catalogs, recommender systems (RSs) play a pivotal role: they guide individual *users* toward relevant *items* by generating personalized suggestions [199]. This capability arises from the system’s ability to infer user preferences learned from their historical interactions. Any recommendation task is fundamentally built upon three core components:

- **Users:** The set of individuals or entities that receive and consume the recommendations.
- **Items:** The objects or services available for recommendation. An item can represent products, movies, songs, or any entity relevant to the application domain.
- **Interactions:** The observable feedback between users and items, which serves as the primary signal of user preference. These interactions, also known as *transactions*, manifest in two forms. *Explicit* feedback captures a direct expression of preference, typically as a discrete rating on a scale (e.g., $\{1, \dots, 5\}$). In contrast, *implicit* feedback is inferred from user behavior—such as clicks, purchases, or views. This type of signal is often unary, indicating positive interest but providing no information about dislike.

2.1 Definition of the Recommendation Problem

The recommendation problem is defined over a set of users $\mathcal{U}$ and a set of items $\mathcal{I}$. Their interactions are captured in a user-item preference matrix $R \in \mathbb{R}^{|\mathcal{U}| \times |\mathcal{I}|}$, where each entry $r_{u,i} \in \mathbb{R}$ encodes either explicit or implicit evidence provided by user u about item i . Figures 2.1a and 2.2a sketch the explicit and implicit settings, respectively. In any realistic setting, this matrix is highly sparse, as users only ever interact with a small subset of the total item catalog [119]. The central challenge of recommendation, therefore, is to infer user preferences from this sparse data. To this end, the primary objective of a recommendation algorithm is to learn a utility function capable of predicting which unobserved items a user might prefer.

$$\begin{bmatrix} 2 & 4 & \dots & 3 & 5 \\ 3 & 3 & \dots & 5 & 2 \\ \dots & \dots & \dots & \dots & \dots \\ 3 & 4 & \dots & 5 & 2 \\ 2 & 3 & \dots & 2 & 5 \end{bmatrix}$$

(a) Full user-item matrix.

$$\begin{bmatrix} 2 & ? & \dots & 3 & 5 \\ 3 & ? & \dots & ? & 2 \\ \dots & \dots & \dots & \dots & \dots \\ 3 & ? & \dots & 5 & 2 \\ 2 & 3 & \dots & ? & ? \end{bmatrix}$$

(b) Missing ratings.

Figure 2.1. Rating prediction under explicit feedback. Left: a fully observed user-item matrix R with each entry $R_{u,i}$ equal to the rating of user u for item i . Right: a realistic sparse case with missing ratings; the task is to estimate the unobserved entries..

$$\begin{bmatrix} 1 & 0 & \dots & 1 & 1 \\ 1 & 0 & \dots & 0 & 1 \\ \dots & \dots & \dots & \dots & \dots \\ 1 & 0 & \dots & 1 & 1 \\ 1 & 1 & \dots & 0 & 0 \end{bmatrix}$$

(a) Full user-item matrix.

$$\begin{bmatrix} 1 & ? & \dots & 1 & 1 \\ 1 & ? & \dots & ? & 1 \\ \dots & \dots & \dots & \dots & \dots \\ 1 & ? & \dots & 1 & 1 \\ 1 & 1 & \dots & ? & ? \end{bmatrix}$$

(b) Missing interactions.

Figure 2.2. Interaction prediction under implicit feedback. Left: a user-item matrix R with $R_{u,i}$ indicating whether user u interacted with item i . Right: sparse observations; recommenders rank unobserved pairs and return the top- k items for each user..

Definition 2.1 (Recommendation Problem). *Given a utility function $f : \mathcal{U} \times \mathcal{I} \rightarrow \mathbb{R}$, the recommendation problem is defined as:*

$$\forall u \in \mathcal{U}, i' = \arg \max_{i \in \mathcal{I}} f(u, i), \quad (2.1)$$

where $i' \in \mathcal{I} / \mathcal{I}_u^+$ is not in the list of (positive) already consumed items $\mathcal{I}_u^+$ by the user u .

The nature of the feedback data dictates how this utility function is operationalized. In explicit feedback settings, the problem is framed as a *rating prediction* task, where the system's goal is to accurately estimate the numeric rating a user would assign to an unobserved item, thereby reconstructing the matrix (see Figure 2.1b).

In contrast, with **implicit feedback**, where only positive interactions are observed (Figure 2.2a), predicting a numerical rating is not meaningful. Here, the utility function generates relevance scores; accordingly, the problem is framed as *top- k prediction*, where items are ranked by these scores and the k most promising unobserved items, see Figure 2.2b, are recommended. While these two tasks are conceptually distinct, they often overlap in practice, as a top- k list can be generated by ranking items based on the scores from a rating prediction model.

In its most general form, therefore, a recommender system is a tool that generates a personalized *ranked list* of items for each user. It achieves this by modeling prefer-

ences from explicit or implicit data, often enhanced with other available information. More formally, this can be expressed as:

$$u \rightarrow \begin{cases} i_1, \\ i_2, \\ \dots, \\ i_k \end{cases}, \quad (2.2)$$

where $f(u, i_1) > f(u, i_2) > \dots > f(u, i_k)$.

Historically, the field was dominated by the rating prediction task. However, practical experience has since revealed the limitations of this paradigm. Collecting explicit ratings at scale is often impractical and can degrade the user experience. Furthermore, studies have shown that explicit ratings are not always a reliable proxy for a user's true preferences [171]. Consequently, the research community has largely converged on the implicit feedback setting and the top-k prediction task, which has now become the predominant objective in recommender systems [62].

2.2 Recommendation Techniques

The field of Recommender Systems (RSs) has produced a wide spectrum of state-of-the-art methodologies. Each approach addresses the recommendation problem by operating on a distinct assumption about user behavior. For instance, some methods are based on the premise that users prefer items similar to those they have previously liked, while others infer preferences based on the behavior of similar users. These diverse techniques are generally classified into three primary paradigms: (i) collaborative filtering, (ii) content-based filtering, and (iii) hybrid methods [199]. The following sections provide a concise yet comprehensive overview of these recommendation approaches and their classification.

2.2.1 Collaborative Filtering Approaches

Collaborative Filtering (CF) is one of the most pervasive paradigms and rests on the intuition that “users who agreed in the past are likely to agree in the future” [198]. In practice, a user's preference for an unseen item can be inferred from the behavior of other users with comparable interaction patterns. A key strength of CF is that it requires no auxiliary side information (e.g., item attributes), relying solely on user–item feedback. Conversely, its effectiveness depends heavily on data availability, making CF sensitive to *data sparsity* and *cold-start* conditions.

CF techniques are typically categorized into two main families: memory-based and model-based approaches [199]. Memory-based algorithms compute recommendations directly from observed interactions, whereas model-based algorithms learn latent user and item representations that generalize to unseen pairs.

Memory-based Collaborative Filtering

Memory-based CF algorithms work directly with the user-item interaction matrix, employing similarity measures to quantify relationships between users or items.

This family of algorithms is best represented by the k -Nearest Neighbors (k NN) method, which builds a similarity matrix and then scores candidates by pooling signals from the k closest entities. The specific implementation of the k NN algorithm hinges on the dimension of similarity computation. **User k NN** [198], operationalizes the intuitive idea of seeking recommendations from like-minded peers. To estimate a user u 's preference for a target item i , the algorithm first identifies a neighborhood of users who have also previously interacted with that item. The core assumption is that the opinions of these users, weighted by their similarity to u , are a strong predictor of u 's own taste. The interaction score is therefore computed by aggregating the similarity signals from this user neighborhood:

$$f^{\text{User}k\text{NN}}(u, i) = \sum_{v \in \mathcal{U}_i^+} \text{sim}(u, v), \quad (2.3)$$

where $\text{sim}(\cdot)$ is a similarity function (e.g., Pearson correlation or cosine similarity) and $\mathcal{U}_i^+$ is the set of users who have interacted with item i . While this method is highly intuitive and effective at capturing community tastes, its primary drawback is scalability, as its computational cost, which scales with the number of user, quickly becomes prohibitive in large-scale systems. To address these scalability limitations, **Item k NN** [212], inverts the logic by focusing on item-item similarities. Instead of looking for similar users, this technique predicts a user's preference for a target item i based on its similarity to other items that the user has previously consumed. The algorithm's utility function aggregates the similarity scores between the target item i and all items in the user's historical profile:

$$f^{\text{Item}k\text{NN}}(u, i) = \sum_{j \in \mathcal{I}_u^+} \text{sim}(i, j), \quad (2.4)$$

where $\text{sim}(\cdot)$ is a similarity function and $\mathcal{I}_u^+$ is the set of items interacted by the user u . By shifting the computational focus from user-user similarity matrix to the item-item similarity matrix, Item k NN provides a significantly more scalable solution.

Although memory-based methods are valued for their simplicity and interpretability, their direct reliance on overlapping interaction histories makes them brittle in sparse data environments and limits their ability to generalize. These inherent limitations motivated the development of more sophisticated, model-based approaches designed to capture deeper, latent patterns within the data.

Model-based Collaborative Filtering

The inherent limitations of memory-based methods, motivated a fundamental shift in the field toward model-based Collaborative Filtering (CF). Instead of relying on

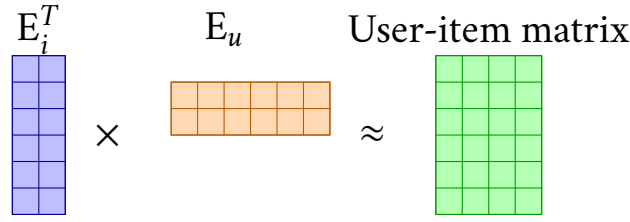


Figure 2.3. Exemplification of latent factor models. These models learn latent representations (embeddings) of users and items. Then, the user–item matrix is approximated by computing the dot product between each user–item embedding pair.

direct, explicit similarities between entities, this paradigm seeks to learn a compact model of the data that demonstrated superior accuracy compared to memory-based counterparts[138].

Within this line of work, **Matrix Factorization** (MF) [140]—often described as a latent factor model—occupies a central role. MF approach decomposes the sparse user–item interaction matrix into two latent factor matrices for users and items by projecting users and items into a shared low-dimensional space. The latent dimensions can be interpreted as abstract preference attributes on the user side and attribute intensities on the item side, providing a compact explanation of high-dimensional, sparse feedback. By learning these latent representations, MF captures complex user–item relationships that are not directly observable. As illustrated in Figure 2.3, this process maps both users and items into a shared latent space of dimensionality d , where d is a hyperparameter much smaller than the number of users or items, i.e., $d \ll |\mathcal{U}|$ and $d \ll |\mathcal{I}|$. In this space, a user u is represented by an embedding vector $e_u \in \mathbb{R}^d$, and an item i by $e_i \in \mathbb{R}^d$. The full set of these embeddings constitutes the model’s learnable parameters, Θ . The interaction score for a given user–item pair (u, i) is then estimated by combining their latent representations. A standard utility function for MF is:

$$f^{\text{MF}}(u, i) = \mu + b_u + b_i + e_i^T e_u, \quad (2.5)$$

where μ represents the global average rating, b_u and b_i are learnable bias terms associated with user u and item i , respectively, and the dot product $e_i^T e_u$ models the core latent interaction between the user and the item.

The success of MF inspired a wave of more expressive, non-linear extensions, particularly with the rise of deep learning. Neural Collaborative Filtering (NCF) [116], for example, generalizes MF by replacing the simple dot product with a multi-layer perceptron (MLP). This allows the model to capture more complex and nuanced interaction patterns that a linear model cannot:

$$f^{\text{NCF}}(u, i) = \sigma \left(W \cdot \left(\text{MLP}([e_u, e_i]) + e_u \odot e_i \right) \right) \quad (2.6)$$

where $\sigma(\cdot)$ is an activation function, W is a weight matrix, $\text{MLP}(\cdot)$ is a multilayer perceptron and $\odot$ denotes the element-wise product.

Alternatively, SimpleX [165] introduces a simplified framework aiming for the explicit disentanglement of user preferences and item characteristics. To achieve this, it defines a scoring function that incorporates the importance of the items with which a user has previously interacted:

$$f^{\text{SimpleX}}(u, i) = \cos(e_i, (\alpha e_u + (1 - \alpha) W \cdot e'_u)) \quad (2.7)$$

where α is a hyperparameter, W is a learnable weight matrix, e'_u is computed as a weighted aggregation of the embeddings of items interacted with by u , and $\cos(., .)$ is the cosine similarity.

Although these extensions share the core principle of learning latent representations, they showcase the architectural diversity in how the final utility function f is modeled.

While dominant in practice, model-based CF is not without constraints. Scalability remains a central concern, as training and inference costs grow with the number of users and items. Moreover, these methods struggle to generate meaningful recommendations for new entities with insufficient interaction data, a limitation known as the cold-start problem, and are prone to overfitting, which calls for robust regularization. A final key challenge is their limited interpretability; the coupling of latent vectors often obscures the rationale behind individual recommendations, hindering diagnosis and user trust. While interpretability is not the primary focus of this dissertation, we address this specific limitation with a transparent linear-factor baseline in Appendix A.

2.2.2 Graph-based Approaches

A key limitation of the latent factor models previously discussed, is that they primarily learn user and item embeddings from individual interactions, largely overlooking the rich topological structure inherent in the user-item feedback data. The complete set of interactions naturally forms a *bipartite graph*, where users and items are two disjoint sets of nodes, and an edge exists between a user and an item if an interaction has occurred. This graph-centric perspective reveals that the core principle of collaborative filtering, the notion that "users who agreed in the past will agree in the future", can be reinterpreted as a function of high-order relationships and structural similarities within this graph.

This realization has motivated a paradigm shift toward architectures explicitly designed to learn from graph-structured data: **Graph Neural Networks (GNNs)** [259]. GNNs are a class of deep learning models that learn node representations by iteratively aggregating information from their local neighborhoods, a process known as *message passing*. By stacking multiple message-passing layers, GNNs can capture high-order connectivities, allowing a node's final representation to be influenced by nodes that are multiple "hops" away in the graph.

For recommendation, this process directly models the flow of collaborative information. A user's embedding is refined not just by the items they have interacted with

(1-hop neighbors), but also by other users who have interacted with those same items (2-hop neighbors), with each subsequent layer of the GNN extending this influence to an ever-wider neighborhood of nodes. This ability to leverage the graph topology is particularly powerful for mitigating data sparsity, as it allows the model to learn richer embeddings for users and items with few interactions by propagating signals from denser regions of the graph.

Basic Notions about Graphs

Formally, a graph is defined by a set of nodes $\mathcal{V}$ and a set of edges $\mathcal{E}$ connecting them, denoted as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. In recommendation, the user-item interaction data forms a bipartite graph, where the node set is partitioned into users $\mathcal{U}$ and items $\mathcal{I}$, such that $\mathcal{V} = \mathcal{U} \cup \mathcal{I}$, and edges only connect nodes from different partitions.

This structure is represented computationally by an adjacency matrix $A \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}|}$, where an entry $A_{vw} = 1$ if an edge exists between node v and node w , and 0 otherwise.

In recommender systems, the adjacency matrix is constructed from the interaction matrix $R \in \mathbb{R}^{|\mathcal{U}| \times |\mathcal{I}|}$ as follows:

$$A = \begin{bmatrix} 0_{|\mathcal{U}| \times |\mathcal{U}|} & R \\ R^\top & 0_{|\mathcal{I}| \times |\mathcal{I}|} \end{bmatrix},$$

The set of nodes directly connected to a node v is its *neighborhood*, formally defined as:

$$\mathcal{N}(v) = \{w \in \mathcal{V} \mid A_{vw} = 1\}.$$

The degree of a node v , denoted $d(v)$, is the number of edges incident to v , i.e., the size of its neighborhood:

$$d(v) = |\mathcal{N}(v)| = \sum_{w \in \mathcal{V}} A_{vw}.$$

Each node is initialized with a feature vector or embedding, $e_v^{(0)}$, which serves as its initial representation. To stabilize the learning process, the adjacency matrix is often symmetrically normalized: $\tilde{A} = D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$, where D is the diagonal degree matrix.

The Message Passing Algorithm

The core mechanism of any GNN is the message passing algorithm, an iterative procedure that refines the latent representation (embedding) of each node (the *ego node*) by aggregating information from its neighbors. This process is repeated for a total of L layers, where L is a hyper-parameter defining the maximum depth of neighborhood exploration. The process at each layer l is composed of three fundamental steps: message aggregation, node embedding update, and layer combination.

Message Aggregation.. At each layer l , a node v receives *messages* from its neighbors. A message is typically the embedding of a neighboring node from the previous layer, $e_w^{(l-1)}$. The aggregation step then combines these incoming messages into a single representation. A crucial property of any aggregation function is that it must be permutation invariant. This is because nodes in a neighborhood form an unordered set; the aggregation's output must therefore be independent of the order in which neighbor embeddings are processed. The aggregated message $m_v^{(l)}$ for node v is thus formally computed as:

$$m_v^{(l)} = \text{AGGREGATE}^{(l)}(\{e_w^{(l-1)} : w \in \mathcal{N}(v)\}). \quad (2.8)$$

where $\text{AGGREGATE}^{(l)}$ is a differentiable function, such as a sum, mean, or max pooling operator. A common choice is the normalized sum, which accounts for node degrees to stabilize the learning process.

Node Embedding Update.. Once the neighborhood information has been aggregated into a single message vector $m_v^{(l)}$, the ego node's embedding is updated. This step merges the aggregated message with the node's own representation from the previous layer, $e_v^{(l-1)}$, allowing the model to combine external information with its internal state. Formally:

$$e_v^{(l)} = \text{UPDATE}^{(l)}(e_v^{(l-1)}, m_v^{(l)}). \quad (2.9)$$

A common implementation of this update function is a linear transformation followed by a non-linear activation function:

$$e_v^{(l)} = \sigma(W_1^{(l)} e_v^{(l-1)} + W_2^{(l)} m_v^{(l)}),$$

where $W_1^{(l)}$ and $W_2^{(l)}$ are learnable weight matrices for the l -th layer and σ is the activation function of the neural network (e.g., ReLU or sigmoid).

Layer Combination. After executing message passing for L layers, the model generates a set of embeddings $\{e_v^{(0)}, e_v^{(1)}, \dots, e_v^{(L)}\}$ for each node v , where $e_v^{(0)}$ is the initial feature vector. Each embedding $e_v^{(l)}$ is a rich representation that captures the structural information from the node's l -hop neighborhood. The final, node representation e_v is then produced by combining these layer-specific representation:

$$e_v = \text{COMBINE}(\{e_v^{(0)}, e_v^{(1)}, \dots, e_v^{(L)}\}). \quad (2.10)$$

Common strategies include:

- Concatenation, which preserves all information from each layer at the cost of increasing the final embedding dimensionality:

$$e_v = e_v^{(0)} \parallel e_v^{(1)} \parallel \dots \parallel e_v^{(L)}. \quad (2.11)$$

- **Summation or Mean Pooling**, which produces a fixed-size embedding by summing or averaging the layer-specific representations:

$$\mathbf{e}_v = \sum_{l=0}^L \mathbf{e}_v^{(l)} \quad \text{or} \quad \mathbf{e}_v = \frac{1}{L+1} \sum_{l=0}^L \mathbf{e}_v^{(l)}. \quad (2.12)$$

- **Last Layer**, a simple approach that uses only the representation from the final layer, which contains the most distant neighborhood information:

$$\mathbf{e}_v = \mathbf{e}_v^{(L)}. \quad (2.13)$$

2.2.3 Content-based Approaches

Content-Based Filtering (CBF) addresses recommendation by exploiting descriptive signals about items (and, when available, about users) to align item characteristics with individual preferences [183]. The underlying assumption is straightforward: if a user has shown interest in items exhibiting certain attributes, then future items sharing those attributes are likely to be relevant. The effectiveness of CBF, therefore, depends on the availability and richness of descriptive item attributes rather than on a dense history of user interactions.

A typical CBF pipeline is organized into three modules [199]:

- **Content Analyzer**: ingests heterogeneous sources (e.g., textual descriptions, tags, reviews, images) and encodes items into a shared description space, ranging from classical vector-space models (e.g., TF-IDF) to modern semantic embeddings and knowledge-driven entity features.
- **Profile Learner**: constructs user profiles from explicit ratings and/or implicit feedback (clicks, views, dwell), using probabilistic estimators, relevance-feedback updates, or similarity-based rules to generalize a user's interests within the same feature space as items.
- **Filtering Component**: matches the learned user profile against item representations—via similarity functions or learned scoring models—to produce ranked recommendation lists.

Early approaches of CBF relied on simple *keyword-based* representations, where items were described by metadata tags or terms extracted from textual descriptions. A significant evolution of this paradigm came with *concept-based* approaches, which characterize items through richer semantic representations. In this context, Knowledge Graphs (KGs) [121] have emerged as the most powerful source of structured information. By linking items from the recommendation catalog to entities within a KG, it becomes possible to enrich their profiles with a structured, and semantically meaningful context. This process has given rise to a specialized subfield known as **Knowledge-Aware Recommender Systems (KaRS)** [14, 20, 130]. These systems leverage the inherent graph structure of KGs to move beyond simple attribute matching, enabling more nuanced forms of reasoning. Thanks to their design,

content-based and knowledge-aware systems offer several compelling advantages. They can provide inherent explainability by highlighting the specific item attributes that align with a user's profile, thereby fostering trust. Additionally, their reliance on item features rather than a dense interaction history makes them particularly effective at addressing the cold-start problem for new items. However, these methods are not without their own limitations. A primary drawback is their tendency toward overspecialization where users are only recommended items that are too similar to those they have already consumed. This inherently limits the diversity of the recommendations, hindering the discovery of novel content outside a user's immediate sphere of interest. Furthermore, their performance is dependent on the quality and completeness of the underlying knowledge base. In scenarios where semantic information is sparse or the KG itself is incomplete, the effectiveness of these models can be significantly compromised, merely trading one form of data sparsity for another. The explicitly graphical nature of KGs has recently motivated the application of models designed to operate on graph-structured data. This has led to a new generation of hybrid models that synergistically combine collaborative signals with semantic knowledge, a topic that will be explored later in this thesis.

2.2.4 *Hybrid Approaches*

While Collaborative Filtering (CF) and Content-Based Filtering (CBF) represent two distinct philosophical approaches to recommendation, their respective limitations are strikingly complementary. Where CF struggles with data sparsity and new items, CBF excels; where CBF is prone to overspecialization, CF introduces diversity through collaborative signals. This natural synergy has motivated the development of hybrid architectures that combine both paradigms to create more robust, accurate, and diverse recommender systems.

The fusion of collaborative and content data can be engineered through several established strategies [3]:

- **Ensemble and Fusion Methods:** Often described as parallel integration, this approach treats CF and CBF as independent experts. Each model generates its own recommendations, and a fusion mechanism—such as a weighted sum of scores or a rank aggregation method—combines their outputs to produce a single, consolidated list.
- **Cascading and Pipelined Models:** In a sequential integration, one paradigm serves to augment or refine another. For example, content-based features can be used to enrich a CF model's input, or a CBF model can act as a pre-filtering step to generate a candidate set that is subsequently re-ranked by a collaborative algorithm.
- **Intrinsically Hybrid Models:** This unified approach moves beyond combining separate models and instead designs a single architecture that is inherently capable of processing both collaborative and content signals simultaneously. Most mod-

ern deep learning-based recommenders, particularly those leveraging knowledge graphs or other forms of side information, belong to this family.

By effectively merging these information sources, hybrid approaches directly confront the core challenges of pure CF and CBF and have consequently become the field’s dominant paradigm.

2.3 Recommendation Pipeline

The diversity of recommender-system architectures makes a universally detailed procedure difficult to define. Still, the increasing complexity of modern approaches calls for a disciplined, standardized workflow to support reproducibility and fair benchmarking [64, 232].

Since the contributions in this thesis implement model-based recommenders, we adopt a three-phase pipeline: data preprocessing, model optimization (training and hyperparameter search), and evaluation. The next sections elaborate on these phases.

2.3.1 Input Pre-processing

The pre-processing stage is foundational for any recommender system, as model performance is directly dependent on the quality and structure of the input data. This phase transforms raw user-item interactions into a format suitable for the target algorithm. While user-item interactions are the primary input, models often leverage supplementary metadata, such as item attributes, user profiles, or Knowledge Graphs (KGs). This section details two common pre-processing operations applied to interaction data.

Converting Explicit Feedback into Implicit Signals

As highlighted in section 2.1, the interactions can be categorized into two types: *explicit* feedback (e.g., numerical ratings) and *implicit* feedback (e.g., clicks, purchases, or views). When the utility f is optimized for a top- k ranking objective, a common pre-processing step is to map explicit ratings to implicit positives. Visually, this turns a matrix like Figure 2.1b into the implicit form in Figure 2.2b. This binarization is typically performed using a predefined threshold τ :

$$y_{u,i} = \begin{cases} 1 & \text{if } r_{u,i} \geq \tau, \\ 0 & \text{otherwise,} \end{cases}$$

where $r_{u,i}$ is the explicit rating provided by user u for item i . Ratings at or above τ are treated as positive interactions; lower values are treated as non-interactions.

P-core Filtering

p -core filtering is procedure used to ensure a minimum level of data density for experimental evaluation. This procedure is designed to enhance the quality of the training data by creating a denser, more reliable subset of interactions. The technique works by retaining only those users and/or items that meet a minimum interaction threshold, p . Formally, for a given interaction matrix, the resulting dataset satisfies:

$$\deg(u) \geq p \quad \text{and/or} \quad \deg(i) \geq p, \quad \forall u \in \mathcal{U}, i \in \mathcal{I},$$

where $\deg(u)$ and $\deg(i)$ represent the number of interactions (degree) of user u and item i , respectively, and p is the minimum interaction threshold.

The rationale behind this procedure is that entities with very few interactions provide a weak and often noisy signal, making it difficult for models to learn generalizable preference patterns. By removing these sparse entities, p -core filtering allows algorithms to train on a dataset where behaviors are better established, leading to more stable model convergence and robust performance. As a consequence of this procedure, the resulting user and item populations can be categorized as either *warm* (those with more than p interactions) or *cold* (those with fewer), effectively isolating the analysis from the specific challenges of the extreme cold-start scenario when it falls outside the scope of the research.

2.3.2 Model Optimization

Following data preparation, the model optimization phase focuses on learning the parameters of the recommendation model. This stage is exclusive to model-based RSs, where latent representations of users and items, collectively denoted by Θ , are learned by optimizing a loss function that reflects the specific recommendation task.

For explicit feedback scenarios, the goal is to accurately predict user ratings. A standard choice for this task is the *Mean Squared Error* (MSE) loss [248], which directly minimizes the difference between predicted and true ratings:

$$\mathcal{L}_{\text{MSE}} = \sum_{(u,i) \in \mathcal{D}} (r_{u,i} - f(u,i))^2, \quad (2.14)$$

where $r_{u,i}$ denotes the actual rating for user u and item i , $\mathcal{D}$ is the set of observed interactions, and $f(u,i)$ is the predicted rating for the pair (u,i) .

For implicit feedback scenarios, the objective shifts from rating prediction to item ranking. A common strategy is to learn the ranking directly through pairwise comparisons. To operationalize this, a training set $\mathcal{T}$ of triplets is constructed from the observed interactions $\mathcal{D}$ as follows:

$$\mathcal{T} = \{(u, i^+, i^-) \mid (u, i^+) \in \mathcal{D}, (u, i^-) \notin \mathcal{D}, i^- \in \mathcal{I}\} \quad (2.15)$$

The Bayesian Personalized Ranking (BPR) loss function [195] is a seminal example of this paradigm. It is based on the assumption that a user prefers an item with

which they have interacted (a positive item) over an item with which they have not (a negative item). BPR optimizes the model to rank an observed (positive) item i^+ higher than an unobserved (negative) item i^- for a given user u :

$$\mathcal{L}_{\text{BPR}} = \sum_{(u, i^+, i^-) \in \mathcal{T}} -\ln \sigma(f(u, i^+) - f(u, i^-)). \quad (2.16)$$

In this expression, $\sigma(\cdot)$ is the sigmoid; $f(u, i^+)$ and $f(u, i^-)$ indicate the predicted utilities for user u paired with the positive item i^+ and the negative item i^- , respectively. Following a similar line of reasoning, the Pairwise Hinge Loss also relies on pairwise comparisons but does so more stringently by enforcing that the positive item's score exceeds the negative one's by a predefined margin m [157]:

$$\mathcal{L}_{\text{PH}} = \sum_{(u, i^+, i^-) \in \mathcal{T}} \max(0, m - (f(u, i^+) - f(u, i^-))). \quad (2.17)$$

A complementary perspective is to cast the implicit setting as a binary classification problem. In this setting, the model learns to distinguish positive interactions (u, i^+) from negative (unobserved) ones (u, i^-) , treating them as individual data points with binary labels. The training set, denoted by $\hat{\mathcal{D}}$, is composed of both the observed interactions (positive examples) and a set of sampled unobserved interactions (negative examples). For this purpose, the Binary Cross-Entropy (BCE) loss is widely used [37]. It is defined as:

$$\mathcal{L}_{\text{BCE}} = - \sum_{(u, i) \in \hat{\mathcal{D}}} r_{u, i} \ln f(u, i) + (1 - r_{u, i}) \ln(1 - f(u, i)), \quad (2.18)$$

where $r_{u, i}$ is the ground-truth label, set to 1 for positive interactions and 0 for negative ones, respectively. The goal is to push the predicted probabilities for positive and negative interactions towards 1 and 0, respectively.

A critical component for all these implicit feedback losses is *negative sampling*, i.e., the strategy to select unobserved items to serve as negative examples. While the original BPR paper suggested sampling only one negative item for each positive interaction for efficiency [195], subsequent research has shown that using a larger number of negative samples, and particularly focusing on "hard" negatives that the model finds difficult to distinguish from positives, can significantly improve performance [74, 219].

Extending this principle, recent approaches have moved towards contrastive learning objectives that formalize the comparison against a large batch of negatives. A prime example is the InfoNCE loss which reframes the ranking task as a multi-class classification problem [203]. For each positive pair $(u, i^+) \in \mathcal{D}$, the model's task is to identify this single positive item from a set of K sampled unobserved items $\{i_k^-\}_{k=1}^K$. For a positive pair (u, i^+) and a set of K negative items $\{i_k^-\}_{k=1}^K$, it is defined as:

$$\mathcal{L}_{\text{InfoNCE}} = \sum_{(u, i^+) \in \mathcal{D}} -\log \frac{\exp(f(u, i^+)/\tau)}{\exp(f(u, i^+)/\tau) + \sum_{k=1}^K \exp(f(u, i_k^-)/\tau)}, \quad (2.19)$$

where the temperature τ is a hyperparameter that controls how strongly the loss reacts to differences between the positive and the negatives. A smaller τ amplifies score differences and yields larger gradients, often speeding learning but with a higher risk of instability or overfitting.

The design of loss functions for recommender systems is a continuously evolving field. As such, the objectives detailed here represent only a subset of the many criteria available, as researchers continue to propose novel formulations to advance the state of the art [165, 249, 294, 312].

2.3.3 *Model Evaluation*

Following the training phase, a recommender system’s performance is quantified through a structured evaluation process. This process begins with inference, where the model’s generated scores for user-item pairs are used to produce personalized ranked lists. These lists are then evaluated for their quality, either against a previously unseen test set or through live user interactions. In practice, these two approaches correspond to two main paradigms: online evaluation, prevalent in industry, and offline evaluation, which is the standard in academic research. As the works in this dissertation are validated through offline methods, this section details the three key components of this approach: (i) dataset splitting, (ii) hyper-parameter tuning, and (iii) evaluation metrics.

Data Splitting

To assess a model’s ability to generalize to new, unseen data, the offline evaluation process requires partitioning the user-item interaction data into separate subsets for training, validation, and testing. Each of these subsets has a distinct purpose. The training set is used to fit the model, the validation set provides an unbiased benchmark for hyper-parameter tuning, and the test set is used only for the final performance report. Adherence to this separation is a prerequisite for a methodologically sound evaluation. As demonstrated in recent studies, failing to isolate the test set during the tuning phase can result in overly optimistic performance estimates and findings that are not reproducible [63, 230]. In practice, this data partitioning is accomplished through one of two main approaches.

The most common strategy is **random splitting**, where user-item interactions are distributed across the train, validation, and test sets in a randomized manner [131, 231]. This can be performed globally on the entire set of interactions, or, at a *user-level*, where a fixed percentage of each user’s interaction history is randomly allocated to each partition (e.g., 80% for training, 10% for validation, and 10% for testing). While computationally efficient, this method overlooks the temporal order of events, which can produce unrealistic scenarios where future interactions inform the prediction of past behavior, an assumption that rarely holds in real-world deployments.

To address this temporal inconsistency, **time-aware splitting** organizes the

data chronologically. Under this paradigm, interactions occurring before a specific timestamp are used for training and validation, while all subsequent interactions are reserved for testing. This approach ensures that the evaluation realistically simulates the real-world task of predicting future interactions based on past behavior. However, this method's practical application is subject to two significant limitations. The first is a strict data dependency: its implementation requires that time information is recorded alongside each user-item interaction. Unfortunately, this is not the case for several datasets in the current literature, making this evaluation strategy inapplicable. Secondly, this method may result in small or even empty test samples for specific users. This issue is particularly pronounced for users who engage with a platform for only a short period, as in datasets spanning long periods, their entire limited activity may fall within the training partition, leaving no subsequent data points for the test set.

Regardless of the splitting strategy, it is important to note that most recommender systems operate under a *transductive* learning setting. This means models are typically evaluated only on users and items that were also present in the training set. As such, any interactions in the validation or test sets involving entities not present in the training set must be filtered out before evaluation.

Hyper-parameter Tuning

The performance of any recommendation algorithm is highly sensitive to its hyper-parameters, such as learning rate or embedding dimensionality, which are set prior to the training phase. The process of identifying the optimal set of these values is known as *hyper-parameter tuning*. This is achieved by systematically testing different hyper-parameter configurations, which are selected according to a specific search strategy. Regardless of the search strategy employed, the model selection protocol remains consistent: for each candidate set of hyper-parameters, the model is trained on the training data, and its performance is subsequently evaluated on the validation set. After exploring all candidate configurations, the one that achieves the highest score on a pre-defined target metric, such as $nDCG@10$ or $Recall@10$, is identified as optimal. To ensure an unbiased assessment of its generalization ability, this single, selected model is then evaluated just once on the unseen test set to generate the final reported results.

However, this process is also a frequent source of methodological flaws. A common pitfall in academic literature is the asymmetric application of tuning, where novel algorithms are meticulously optimized while established baselines are not, leading to biased comparisons and questionable claims of superiority [61, 63, 217]. The frequent omission of details on the tuning process further hinders reproducibility [310].

To navigate the hyper-parameter search space, this thesis employs two distinct and complementary strategies. The first is **grid search**, a traditional method that ensures a comprehensive exploration by exhaustively testing every possible configuration within a predefined discrete grid [88]. While its primary advantage is thoroughness,

its computational cost grows exponentially with the number of parameters, making it impractical for larger search spaces.

As a more efficient alternative, we therefore also leverage **Bayesian optimization**, specifically through the use of **Tree-structured Parzen Estimators (TPE)** [29]. Unlike the brute-force nature of grid search, TPE performs a guided search through the hyper-parameter space. At each step, it uses the results from previous trials to inform its next choice, prioritizing configurations that have a high probability of outperforming the best results found so far. This allows it to efficiently balance the exploration of new parameter settings with the exploitation of well-performing regions.

It is important to acknowledge, however, that offline hyper-parameter tuning has its limitations. There is no guarantee that a configuration optimized in an offline setting will yield the best performance in a dynamic online environment [300]. Therefore, model robustness to small changes in hyper-parameters remains a significant practical consideration.

Assessing Model Effectiveness

The effectiveness of a recommender system is determined by quantifying the quality of its output. In an offline setting, this is achieved by evaluating the ranked recommendation lists generated by a trained model against an unseen test set. While historical evaluations have prioritized accuracy, a truly effective system must also account for a wider range of objectives, including the ability to promote diversity and mitigate algorithmic bias [241]. To provide a holistic view of performance, this section details a range of evaluation metrics from the literature, organized by the dimensions they measure: accuracy, novelty, diversity, and fairness.

Accuracy Metrics. A primary dimension of evaluation concerns the correctness of the generated recommendations, which is quantified using accuracy metrics.

For the task of *rating prediction*, performance is typically quantified using error-based metrics that measure the discrepancy between the predicted ratings and the actual ratings provided by users. Given a set of observed interactions $\mathcal{D}$, the most common error metrics are defined as follows:

Definition 2.2 (Mean Absolute Error). *The Mean Absolute Error (MAE) quantifies the average magnitude of the errors between predicted and actual ratings, without considering their direction. It is defined as:*

$$MAE = \frac{1}{|\mathcal{D}|} \sum_{(u,i) \in \mathcal{D}} |r_{u,i} - f(u, i)| \quad (2.20)$$

Definition 2.3 (Mean Squared Error). *The Mean Squared Error (MSE) measures the average of the squares of the errors. By squaring the differences, it places a significantly*

higher penalty on larger prediction errors compared to smaller ones. It is defined as:

$$MSE = \frac{1}{|\mathcal{D}|} \sum_{(u,i) \in \mathcal{D}} (r_{u,i} - f(u,i))^2 \quad (2.21)$$

Definition 2.4 (Root Mean Squared Error). *The Root Mean Squared Error (RMSE) is the square root of the MSE. It is often preferred in practice as it translates the error back into the same units as the original rating scale, making it more directly interpretable. It is calculated as:*

$$RMSE = \sqrt{\frac{1}{|\mathcal{D}|} \sum_{(u,i) \in \mathcal{D}} (r_{u,i} - f(u,i))^2} \quad (2.22)$$

However, as the predominant objective in modern RSs is *top-k ranking*, the focus of evaluation has shifted to metrics that assess the quality of the ranked lists presented to users. These metrics evaluate the presence and position of relevant items within the top- k entries of a recommendation, where k is typically chosen from $\{1, 5, 10, 20, 50\}$. Given the set of users $\mathcal{U}$, the recommendation list $\mathcal{L}_u$ for a user u , its top- k items $\mathcal{L}_u^{(1,\dots,k)}$, and the relevant items in the test set $\mathcal{I}_u^{+,test}$, the most common ranking-based accuracy metrics are defined as follows:

Definition 2.5 (Precision@ k). *Precision@ k is the mean, across users, of the fraction of relevant items contained within the top- k recommendations:*

$$Precision@k = \frac{1}{|\mathcal{U}|} \sum_{u \in \mathcal{U}} \frac{|\mathcal{L}_u^{(1,\dots,k)} \cap \mathcal{I}_u^{+,test}|}{k}. \quad (2.23)$$

Definition 2.6 (Recall@ k). *Recall@ k is the mean, across users, of the fraction of each user's relevant items that appear in the top- k :*

$$Recall@k = \frac{1}{|\mathcal{U}|} \sum_{u \in \mathcal{U}} \frac{|\mathcal{L}_u^{(1,\dots,k)} \cap \mathcal{I}_u^{+,test}|}{|\mathcal{I}_u^{+,test}|}. \quad (2.24)$$

Definition 2.7 (normalized Discounted Cumulative Gain@ k). *nDCG@ k is a rank-aware effectiveness measure that rewards relevant items appearing higher in the list:*

$$nDCG@k = \frac{1}{|\mathcal{U}|} \sum_{u \in \mathcal{U}} \frac{1}{IDCG@k} \sum_{i \in \mathcal{L}_u^{(1,\dots,k)}} \frac{2^{rel_i} - 1}{\log_2(i + 1)}, \quad (2.25)$$

where $rel_i = 1$ if the item at rank $i \in \{1, \dots, k\}$ is relevant for the user and 0 otherwise. The normalizer $IDCG@k = \sum_{i=1}^k \frac{1}{\log_2(i+1)}$ corresponds to the DCG of an ideal top- k where all k items are relevant.

Definition 2.8 (Hit Ratio). *Hit Ratio (HR@ k) is the proportion of users whose top- k list contains at least one relevant item:*

$$HR@k = \frac{|\mathcal{U}_{hits@k}|}{|\mathcal{U}|}, \quad (2.26)$$

where $|\mathcal{U}_{hits@k}|$ is the number of users for whom $\mathcal{L}_u^{(1,\dots,k)}$ includes at least one relevant item. Larger values indicate a higher frequency of per-user hits.

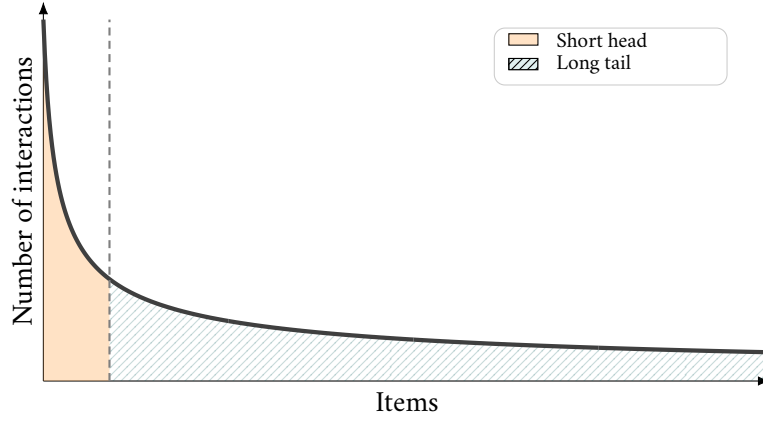


Figure 2.4. A graphical representation of the long-tail phenomenon. The x-axis represents individual items, sorted in descending order of their interaction counts, which are shown on the y-axis. The "short head" (orange) highlights that a few popular items receive a very high number of interactions, while the "long tail" (teal) shows that the vast majority of items receive very few interactions each. The dashed line marks the transition between these two regions.

Novelty and Diversity. Beyond accuracy, it is essential to evaluate a system's ability to provide suggestions that are both interesting and novel. **Novelty** measures how surprising or unknown a recommended item is to a user [119, 171]. A common proxy for novelty is the inverse of an item's popularity, with less popular items from the *long-tail* of the distribution (Figure 2.4) considered more novel [136]. Metrics such as Expected Popularity Complement (EPC) and Expected Free Discovery (EFD) are specifically designed to quantify a system's ability to recommend relevant long-tail items [44].

Definition 2.9 (Expected Popularity Complement). *Expected Popularity Complement (EPC) quantifies a recommender system's propensity to suggest relevant items that are also novel, i.e., part of the long-tail:*

$$EPC@k = \frac{c}{|\mathcal{U}|} \sum_{u \in \mathcal{U}} \sum_{i_k \in \mathcal{L}_u^{(1, \dots, k)}} p(\text{seen} \mid k, u, \mathcal{L}_u^{(1, \dots, k)}) p(\text{rel} \mid i_k, u) (1 - p(\text{seen} \mid i_k)), \quad (2.27)$$

where c is a normalizing constant, $p(\text{seen} \mid \cdot)$ is the probability of an item being seen, and $p(\text{rel} \mid \cdot)$ is its probability of being relevant. The term $(1 - p(\text{seen} \mid i_k))$ explicitly models the "complement" aspect, rewarding items that are less likely to have been seen before. A higher EPC score indicates a stronger focus on relevant long-tail content.

Definition 2.10 (Expected Free Discovery). *Expected Free Discovery (EFD) evaluates the expected novelty of relevant recommendations using an inverse-collection-frequency weighting, thus emphasizing exposure to rare items in the catalog. The Expected Free Discovery (EFD) assesses the novelty of recommendations by weighting each relevant item by the inverse of its popularity. It operates on the principle that recommending a*

rare yet relevant item provides more new information to the user than recommending a popular one:

$$EFD@k = \frac{c}{|\mathcal{U}|} \sum_{u \in \mathcal{U}} \sum_{i_k \in \mathcal{L}_u^{(1, \dots, k)}} p(\text{seen} \mid k, u, \mathcal{L}_u^{(1, \dots, k)}) p(\text{rel} \mid i_k, u) (-\log_2(p(i \mid \text{seen}))), \quad (2.28)$$

where c is a normalizing constant. The weight of each item, $-\log_2(p(i \mid \text{seen}))$, is derived from its probability of being seen, assigning a high value to long-tail items and a low value to short-head items. Consequently, higher EFD values indicates that the system is effective at providing recommendations that are both relevant and genuinely novel.

In contrast to novelty, **diversity** assesses the dissimilarity among the items within a recommendation list. High diversity helps prevent monotonous suggestions and broadens user discovery. Aggregate diversity across all users can be measured using the Gini Index, which quantifies the inequality of item exposure, and Item Coverage (IC), which measures the total number of unique items recommended by the system [44].

Definition 2.11 (Gini Index). *Gini Index ($\hat{\text{Gini}}@k$) evaluates how evenly a recommendation algorithm distributes its suggestions across the entire item catalog:*

$$\hat{\text{Gini}}@k = \frac{1}{|\mathcal{U}|} \sum_{u \in \mathcal{U}} \frac{1}{k-1} \frac{\sum_{i=1}^{|\mathcal{L}_u^{(1, \dots, k)}|} (2i - |\mathcal{L}_u^{(1, \dots, k)}| \text{times}(i))}{\sum_{i=1}^{|\mathcal{L}_u^{(1, \dots, k)}|} \text{times}(i)}, \quad (2.29)$$

where $\text{times}(i)$ counts the frequency of each item across all recommendations. A score of 0 represents a perfectly uniform distribution where all items are recommended equally, whereas a score of 1 indicates extreme concentration on a single item.

Definition 2.12 (Item Coverage). *Item Coverage ($\text{IC}@k$) is an aggregate diversity metric that quantifies the proportion of the item catalog actively utilized by the recommender system. It is defined as the total number of unique items that appear across all users' top- k recommendation lists:*

$$\text{IC}@k = \left| \bigcup_{u \in \mathcal{U}} \mathcal{L}_u^{(1, \dots, k)} \right|. \quad (2.30)$$

Larger Item Coverage values reflects a broader use of the catalog and, consequently, higher diversity in the recommended items.

Fairness and Algorithmic Bias. Beyond accuracy and diversity, a critical dimension of evaluation concerns the *fairness* of a recommender system. The impact of recommendations extends to a multi-stakeholder ecosystem, and fairness addresses the equitable distribution of the system's benefits and harms across its various participants [1]. This concept can be broadly categorized according to the stakeholder it concerns [82], i.e., *Consumer Fairness* or *Provider Fairness*.

Consumer Fairness focuses on ensuring that all users receive a comparable quality of service. The underlying principle is that recommendation utility should not systematically favor certain user groups over others. This can be assessed at an individual level by measuring the consistency of performance across the entire user base, for which Variance is a common metric [277]. At a group level, fairness is often evaluated by measuring disparities between predefined user groups (e.g., by gender), using metrics like the Mean Absolute Deviation (MAD) [67].

Definition 2.13 (Variance). *The variance of recommendation quality, $\sigma^2(Q)$ quantifies how consistently a recommender system performs for its entire user base $\mathcal{U}$. Given Q_u as the quality score for a user u (measured, for instance, by nDCG), and Q as the set of all such scores, the variance is defined by the following formula:*

$$\sigma^2(Q) = \frac{1}{|\mathcal{U}|} \sum (Q_u - \bar{Q})^2, \quad (2.31)$$

with $\bar{Q}$ representing the mean of the set Q . A low variance signifies equitable performance across users, suggesting fairness at the individual level. In contrast, a high variance indicates significant performance gaps, implying that the system may be serving some users less effectively than others.

Definition 2.14 (Mean Absolute Deviation). *The Mean Absolute Deviation (MAD) assesses group-level fairness by directly comparing the average quality of recommendations received by two user groups, $\mathcal{U}_x$ and $\mathcal{U}_y$. Given Q_u as the quality score for a user u (e.g., nDCG), MAD is calculated by taking the absolute difference of the average recommendation quality experienced by each group:*

$$MAD(Q^{(\mathcal{U}_x)}, Q^{(\mathcal{U}_y)}) = \left| \frac{\sum_{x \in \mathcal{U}_x} Q_x}{|\mathcal{U}_x|} - \frac{\sum_{y \in \mathcal{U}_y} Q_y}{|\mathcal{U}_y|} \right|. \quad (2.32)$$

A low MAD value indicates performance parity between the groups, suggesting equitable treatment. Conversely, high MAD values reveals that one group is systematically receiving better (or worse) recommendations than the other. When the users are partitioned in more than two groups, MAD values are calculated for all possible pairwise group combinations. The final metric is derived as the mean of these computed MAD values.

Provider Fairness addresses the equitable exposure of items, and, consequently, their associated providers. Since an item's position in a ranked list heavily influences its visibility and potential for interaction [30, 221], disparities in exposure can lead to unequal opportunities. Metrics for this dimension assess how fairly the system distributed exposure across different item categories. Two prominent examples are Ranking-based Statistical Parity (RSP) and Ranking-based Equal Opportunity (REO) [319].

Definition 2.15 (Ranking-based Equal Opportunity). *Ranking-based Equal Opportunity (REO) adapts the principle of equal opportunity to ranking tasks, focusing on the*

fair treatment of relevant items. Specifically, it posits that all relevant items, those a user is known to like ($y = 1$), should have an equal chance of appearing in a top- k recommendation, regardless of their category ($c_1, \dots, c_n$). This condition of equal true positive rates across categories is expressed as $P(R@k | c_1, y = 1) = \dots = P(R@k | c_n, y = 1)$. The REO@ k metric then quantifies the disparity from this ideal as follows:

$$REO@k = \frac{\text{std}(P(R@k | c_1, y = 1), \dots, P(R@k | c_n, y = 1))}{\text{mean}(P(R@k | c_1, y = 1), \dots, P(R@k | c_n, y = 1))}, \quad (2.33)$$

where $\text{std}(\cdot)$ and $\text{mean}(\cdot)$ are the standard deviation and mean operators. Lower REO values indicate that the system is equally effective at recommending relevant items from all categories, thus suggesting greater fairness.

The conditional probability for a category C_A is computed as:

$$P(R@k | C = C_A, y = 1) = \frac{\sum_{u \in \mathcal{U}} \sum_{i=1}^k \eta_{C_A}(R_{u,i}) Y(u, R_{u,i})}{\sum_{u \in \mathcal{U}} \sum_{i \in I \setminus I_u^+} \eta_{C_A}(i) Y(u, i)}, \quad (2.34)$$

where $Y(u, i)$ is an indicator function that returns 1 if user u likes the item $R_{u,i}$ ranked at position i , 0 otherwise. The numerator counts, for each user, the number of relevant items from category C_A correctly placed in the top- k recommendations (true positives). The denominator counts, for each user, the total number of relevant items from category C_A in the test set.

Definition 2.16 (Ranking-based Statistical Parity). *Ranking-based Statistical Parity (RSP) quantifies whether items from different categories receive comparable visibility in the top- k . Let $P(R@k | c_i)$ be the probability that an item belonging to category c_i occurs among the first k positions of users' recommendation lists. Statistical parity requires $P(R@k | c_1) = \dots = P(R@k | c_n)$. RSP summarizes dispersion around this requirement via the coefficient of variation of these probabilities:*

$$RSP@k = \frac{\text{std}(P(R@k | c_1), \dots, P(R@k | c_n))}{\text{mean}(P(R@k | c_1), \dots, P(R@k | c_n))}, \quad (2.35)$$

where $\text{std}(\cdot)$ and $\text{mean}(\cdot)$ are the standard deviation and mean operator, respectively; lower values indicate fairer exposure. The probability for a given category C_A is determined by the ratio of its items in the top- k lists to its total available items, aggregated across all users:

$$P(R@k | C = C_A) = \frac{\sum_{u \in \mathcal{U}} \sum_{i=1}^k \eta_{C_A}(R_{u,i})}{\sum_{u \in \mathcal{U}} \sum_{i \in I \setminus I_u^+} \eta_{C_A}(i)}.$$

where the numerator aggregates, across all users, the count of items from category C_A that appear within the top- k recommended lists. The denominator, in turn, represents the total number of items from category C_A in the test set.

Closely related to fairness is the issue of **algorithmic bias**, which refer systematic patterns in a model's behavior that disadvantage certain items, groups, or individuals [164]. A dominant form in in RSs scenario is **popularity bias**, which describes

the tendency of many algorithms to over-recommend popular items, leading to the under-representation of content from the long-tail [27]. This bias is typically measured via Average Percentage of items in the Long-Tail (APLT) [2] and Average Recommendation Popularity (ARP) [292]. To formalize these metrics, let $\mathcal{U}$ be the set of all users, and for each user $u \in \mathcal{U}$, let $\mathcal{L}_u^{(1,\dots,k)}$ be the list of their top- k recommended items.

Definition 2.17 (Average Percentage of Long-Tail Items). *The Average Percentage of Long-Tail Items (APLT@ k) evaluates the extent to which a recommender system promotes content from the long tail. It computes the average proportion of long-tail items within the top- k recommendations across all users.*

$$APLT@k = \frac{1}{|\mathcal{U}|} \sum_{u \in \mathcal{U}} \frac{|\{i \mid i \in \mathcal{L}_u^{(1,\dots,k)} \cap \Gamma\}|}{|\mathcal{L}_u^{(1,\dots,k)}|} \quad (2.36)$$

In the formula, Γ denotes the set of items classified as long-tail. A higher APLT@ k score signifies a greater representation of niche items in the recommendations.

Definition 2.18 (Average Recommendation Popularity). *The Average Recommendation Popularity (ARP@ k) provides a direct measure of the overall popularity of items featured in top- k recommendations. It is calculated by averaging the mean popularity of the items in each user's recommendation list across the entire user population.*

$$ARP@k = \frac{1}{|\mathcal{U}|} \sum_{u \in \mathcal{U}} \frac{\sum_{i \in \mathcal{L}_u^{(1,\dots,k)}} \varphi(i)}{|\mathcal{L}_u^{(1,\dots,k)}|} \quad (2.37)$$

Here, $\varphi(i)$ is a function returning the popularity of item i , typically its total interaction count. Consequently, lower ARP@ k values suggest the system tends to recommend less popular, long-tail items.

Part II

**Structural Mitigation with
Knowledge Graphs under Data
Sparsity**

Chapter 3

Learning User-Aligned Knowledge Signals for Personalized Graph Recommendation

Knowledge graphs (KG) have been proven to be a powerful source of side information to enhance the performance of recommendation algorithms. Their graph-based structure paves the way for the adoption of graph-aware learning models such as Graph Neural Networks (GNNs). In this respect, state-of-the-art models achieve good performance and interpretability via user-level combinations of intents leading users to their choices. Unfortunately, such results often come from an end-to-end learning that considers a combination of the whole set of features contained in the KG without any analysis of the user decisions. This chapter introduces KGTORe, a GNN-based model that exploits KG to learn latent representations for the semantic features, and consequently, interpret the user decisions as a personal distillation of the item feature representations. Differently from previous models, KGTORe does not need to process the whole KG at training time but relies on a selection of the most discriminative features for the users, thus resulting in improved performance and personalization. Experimental results on three well-known datasets show that KGTORe achieves remarkable accuracy performance and several ablation studies demonstrate the effectiveness of its components. The implementation of KGTORe is available at: <https://github.com/sisinflab/KGTORe>.¹

1. This chapter is based on the work published in the *Proceedings of the 17th ACM conference on recommender systems (RecSys 2023)* “Kgtore: tailored recommendations through knowledge-aware GNN models”.

3.1 Introduction

Nowadays, recommender systems are more and more employed in a wide range of domains thanks to their capability of providing users with a personalized list of items [8, 60, 293, 299].

As a common factor, recommender systems mostly leverage past user-item interactions to estimate new preferences and produce personalized recommendations [78, 116]. Collaborative filtering methods accurately mine general similarity patterns in the user-item interactions but neglect some peculiarities of user preferences about the item characteristics, thus missing the opportunity to further improve user personalization. On the contrary, content-based recommendation strategies analyze the characteristics of the products users have previously liked [38], suggesting unseen items matching their personal tastes. Although these strategies may be successful in terms of explainability, the lack of data from peer users frequently results in overspecialization and a dearth of serendipity [59, 128, 317]. To minimize the limitations of the two recommendation approaches and benefit at the same time from their advantages, hybrid recommender systems integrate into collaborative filtering the content information utilized in content-based approaches.

Knowledge graphs (KG) are a large source of structured information used both in content-based and hybrid recommender systems, thanks to their ability to characterize items by means of relations with other entities (e.g., *Harry Potter* $\xrightarrow{\text{genre}}$ *Fantasy*). Knowledge graphs have given rise to the promising research trend of knowledge-aware recommendation [7], including Graph Neural Networks (GNN) [252, 254, 255, 309], which have shown to provide competitive accuracy by aggregating information from multi-hop neighbors with long-term connectivity.

However, most of the knowledge-aware GNN-based methods do not investigate the hidden meaning of the edges between users and items, leaving the rationale of the user choices unexplored. Even state-of-the-art approaches aiming to explore hidden intentions assume that a user always has the same intent when choosing any available item and that possible intents are shared among all the users and can be learned in an end-to-end fashion as a combination of a large set of semantic relations [261]. Instead, we argue that each user-item interaction relates to a particular user choice determined by the presence or the absence of some characteristics in the item profile. Moreover, this choice should be used as an explicit interpretation of the edge between the user and the item, which in turn should be accounted for to drive the training of a GNN-based recommender system.

Motivated by these hypotheses, this chapter introduces a novel approach that we name *Knowledge-Graph- and Tree-Oriented Recommendation* (KGTORe), able to leverage the explicit rationale behind the user-item interactions for improving the quality of the personalized recommendations. Knowledge graphs are used to feed rigorous and transparent processes aiming to identify only the most relevant semantic motivations that led users to their historical interactions with the items of

the catalog.

Concretely, the main contributions of KGTORe consists of three elements.

1. **Item Modeling.** Knowledge graphs provide a plethora of semantic characteristics which are used in this component to describe the items in the catalog (i.e. *genre*, *director*, *cast*) and that are later used to disentangle the semantic characteristics driving the user decisions.
2. **User Decision Process Modeling.** KGTORe aims to decompose each user-item interaction into an accurate selection of features describing the motivations behind such interaction. For instance, a user may choose *Harry Potter* because it is directed by *Chris Columbus* and falls into the *fantasy* genre. Moreover, the same user may decide to reject *The Hobbit* because directed by *Peter Jackson*, even if it is a *fantasy* movie. To model these behaviors, KGTORe relies on the widely accepted notion of decision trees [98] for retrieving fine-grained paths of semantic features that more likely led the user to choose or reject specific items. This modeling has the two-fold merit of shedding light on the existing user-item interactions with an explicit semantics and serving as a lightweight and small set of useful features for providing the next recommendation.
3. **Knowledge-aware Graph-based Recommendation.** KGTORe integrates the item and the user decision process in a graph neural network where the edges between users and items are decision-aware combinations of influential features modeling the user choices. The network is able to learn latent representations for the semantic features (e.g., $\langle \text{director}, \text{Chris Columbus} \rangle$) in an end-to-end fashion, and consequently the representations for users and items, through an aggregation scheme over multi-hop neighbors.

We conduct extensive experiments on three well-known datasets to show the performance of KGTORe outperforming the state-of-the-art baselines and to demonstrate the effectiveness of its components through in-depth hyperparameter and ablation studies.

3.2 Preliminaries

This section introduces the problem to be addressed, the available data, the notion of knowledge graphs used for content modeling, and the basics of decision trees, used to model the user decisions.

Problem Formulation. Let $\mathcal{I}$ be a collection of items and $\mathcal{U}$ the set of users who interacted with the system with implicit feedback (e.g., view, click, like, purchase). In detail, the element r_{ui} of the user-item matrix $R \in \{0, 1\}^{|\mathcal{U}| \times |\mathcal{I}|}$ contains either 1 denoting a positive/implicit feedback from user u to item i or 0 for denoting absence of feedback [129].

The objective of KGTORe is to predict for the users possible rankings of unrated items reflecting their actual interests. KGTORe solves this task by modeling the

user decision processes to catch the motivations behind the known interactions and injecting this semantic knowledge into a GNN. As a possible choice, KGTORe models the user decisions by exploiting the well-known decision tree classifiers, resulting in an effective solution that leverages their simplicity and interpretability.

Knowledge Graphs for Content Modeling. Content-based solutions, and so hybrid models, provide users with items that are likely to be appreciated based on their attributes and characteristics. Knowledge graphs can be leveraged for this aim since they store structured information about real-world facts in the form of a heterogeneous graph. Formally, given a set of entities $\mathcal{V}$ and a set of relations $\mathcal{R}$, a knowledge graph $\mathcal{KG}$ is a set of triples in the form $\sigma \xrightarrow{\rho} \omega$. A triple represents the connection between two entities $\sigma, \omega \in \mathcal{V}$ (aka the subject and the object of the triple, respectively) through an edge $\rho \in \mathcal{R}$ (aka the predicate of the triple):

$$\mathcal{KG} = \{\sigma \xrightarrow{\rho} \omega \mid \sigma, \omega \in \mathcal{V}, \rho \in \mathcal{R}\}. \quad (3.1)$$

Having a mapping between items in $\mathcal{I}$ and the entities of the knowledge graph, i.e., $\mathcal{I} \subseteq \mathcal{V}$, the knowledge graph can characterize items and provide information for interpreting user-item edges.

Decision Trees and Information Gain. As we will discuss later, we deem that decision trees are a solution for catching the reasons why users choose to consume or reject an item, thanks to their ability to explain a classification based on a path of possible features. Practically, decision trees look for the variables whose values reduce the most the entropy, i.e., the uncertainty, of another random variable.

Mathematically, the entropy $H(V)$ of a random variable V with k possible values in $\{v_1, \dots, v_k\}$ is defined as:

$$H(V) = - \sum_{i=1}^k \Pr(V = v_i) \log_2 \Pr(V = v_i). \quad (3.2)$$

If the observation of the value of another variable X with n possible values in $\{x_1, \dots, x_n\}$ can reduce the entropy of V , then X is said to have an information gain over V :

$$\begin{aligned} IG(V, X) &= H(V) - H(V|X), \\ H(V|X) &= \sum_{i=1}^n P(X = x_i) H(V|X = x_i). \end{aligned} \quad (3.3)$$

One of the practical reasons behind the unquestionable success of the notion of information gain is the possibility of using it both with binary and non-binary values. When binary features $X_1, X_2, \dots, X_l$ are used for binary classification, decision trees can find paths of positive/negative features leading to determining whether V is a positive or negative class.

3.3 Approach

This section presents the proposed KGTORe as expressed by its three building blocks: the item modeling, the user decision process modeling, and the knowledge-aware graph-based recommendation.

3.3.1 Item Modeling

Assuming each item in $\mathcal{I}$ is linked to an entity in $\mathcal{V}$, i.e., it is one of the entities of $\mathcal{KG}$, we can model the items of the catalog employing the knowledge encoded in $\mathcal{KG}$. The rationale of this modeling is that each item can be substantially framed by a different collection of characteristics, which will differently influence the user decisions about that item. Under this assumption, $\mathcal{KG}$ can be explored to identify, for each item, connections with other entities that can be used as its attributes in the form $\langle \rho, \omega \rangle$, where ρ and ω are the predicate and the object of the $\mathcal{KG}$ triple. For instance, if correctly encoded in the reference knowledge graph, the movie *Harry Potter* could be semantically described by the predicate-object pairs $\langle \text{director}, \text{Chris Columbus} \rangle$ and $\langle \text{genre}, \text{Fantasy} \rangle$. Formally, we use the set $\mathcal{F}_i$ of semantic features to describe $i \in \mathcal{I}$ by means of its first-order connectivity extracted from $\mathcal{KG}$:

$$\mathcal{F}_i = \{ \langle \rho, \omega \rangle \mid i \xrightarrow{\rho} \omega \in \mathcal{KG} \}. \quad (3.4)$$

3.3.2 User Decision Process Modeling

KGTORe captures the behavior of users in deciding to accept or discard a new item. We deem the training data encodes this information, even when only positive feedback is available. Facing a new item, users may positively or negatively consider the presence or the absence of some characteristics with a certain priority and to a certain extent.

To model such behavior, KGTORe performs a **decision analysis** on the user ratings via user-level classifiers based on decision trees. Then, the retrieved decision is represented as a **knowledge-aware decision embedding** in a latent space.

User-Level Decision Analysis

Suppose a user liked the movies *Harry Potter* and *Mrs. Doubtfire*, both directed by *Chris Columbus*, while explicitly disliked movies starring *Johnny Depp*. Given these facts, KGTORe should consider *Chris Columbus* to have a high positive impact on the recommendations for this user if zero or very few disliked movies have been directed by *Chris Columbus*. At the same time, if a feature is common to (almost) any positive and negative item (e.g., $\langle \text{type}, \text{movie} \rangle$), it should be not considered detrimental in the user decision process. Analogously, *Johnny Depp* should be considered a good attribute to identify negative items. In fact, the fewer liked movies star *Johnny Depp*

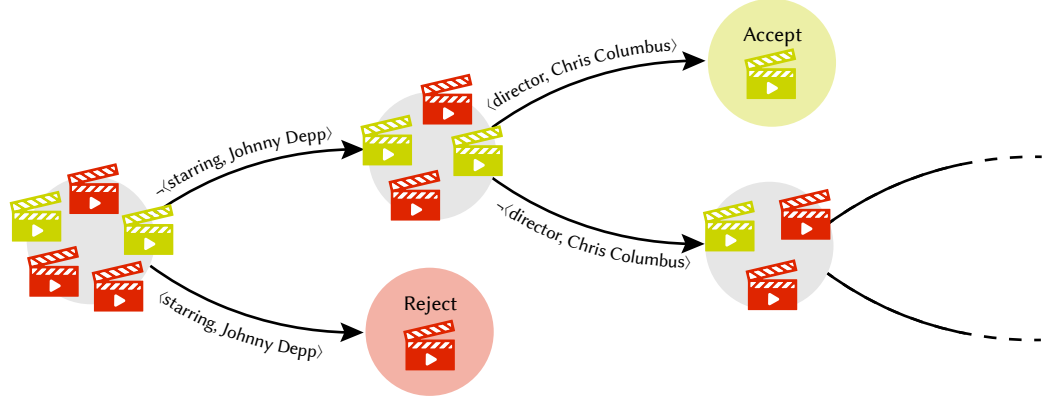


Figure 3.1. Toy example of a user-level decision tree. The user liked two movies and disliked three. Following the characteristics in the item feature sets, the decision tree is able to discriminate between liked and unliked items and understand how the user decides to accept or reject a movie .

the more likely is to identify negative items by means of this characteristic. Figure 3.1 shows a possible user decision tree accounting for the preferences of the example user and aiming to model her decision process.

However, when only positive feedback is available, negative decisions should be inferred from the unrated items. To do so, the decision tree classifier has to be fed with liked items and with a certain amount of randomly picked unrated items. Nevertheless, the number of unrated items is huge with respect to the number of rated items, and picking just some of them may not be sufficient to understand which features users use to accept or discard items. As an example, concluding that the presence of a certain feature in all the positive items is perfectly discriminant should be supported by the absence of this feature in any negative item. In addition, picking a small number of unrated items may actually lead to this absence just due to chance. To limit the effect of randomly picking negative items, which usually may not be effective for discovering patterns of negative choices, KGTORe compares the user's positive items with an ample set of negative items.

Let $\mathcal{I}_u^+ \subseteq \mathcal{I}$ be the set of items rated by the user u , and suppose $\mathcal{I}_u^- \subset \mathcal{I} \setminus \mathcal{I}_u^+$ contains a certain number of items not enjoyed by u . Supposing the items in $\mathcal{I}_u^T = \mathcal{I}_u^+ \cup \mathcal{I}_u^-$ have been semantically characterized, their features can be used to train a decision tree T_u , able to distinguish whether an item comes from $\mathcal{I}_u^+$ or $\mathcal{I}_u^-$ based on the presence or the absence of some features in the item feature sets. Specifically, the decision tree aims to iteratively find the features $\langle \rho, \omega \rangle$ such that the knowledge about $\langle \rho, \omega \rangle \in \mathcal{F}_i$ or $\langle \rho, \omega \rangle \notin \mathcal{F}_i$ reduces the most the entropy of $\mathcal{I}_u^T$, i.e., helps the most in distinguishing rated from unrated items. As a result, KGTORe is able to retrieve the semantic decision path that conducted the user to a positive decision or to a negative decision about an item.

Knowledge-Aware Feature and Decision Embeddings

Narrowing the analysis to the enjoyed items and analyzing the user decisions, it is possible to describe each user-item interaction of the system through a set of features $\mathcal{F}_{ui}$ composing the decision path $\mathcal{P}_{ui}$ that, as resulting from T_u , led user u to choose item i :

$$\mathcal{P}_{ui} = \{(s_{uif}, f) \mid f \in \mathcal{F}_{ui}\}, \quad (3.5)$$

where $s_{uif} \in \{-1, 1\}$ indicates whether $f \notin \mathcal{F}_i$ or $f \in \mathcal{F}_i$ respectively, i.e., the **feature absence** or the **feature presence** in the item content.

We argue that this path captures the rationale of the particular user-item interaction and that the overall intent is a combination of the feature presences and absences in the decision path, e.g., the user may choose to watch the movie *Harry Potter* because it is directed by *Chris Columbus* and it does *not* star *Johnny Depp*. With this intuition in mind, we model a **knowledge-aware decision embedding** as:

$$\mathbf{p}_{ui} = \frac{1}{|\mathcal{P}_{ui}|} \sum_{(s_{uif}, f) \in \mathcal{P}_{ui}} s_{uif} \mathbf{e}_f, \quad (3.6)$$

where $\mathbf{e}_f \in \mathbb{R}^d$ is the embedding describing the feature f . While the set of features involved in the summation comes from the knowledge-aware content analysis, the decision embeddings are learned in an end-to-end and collaborative manner by means of the feature embeddings they are made up of. It is noteworthy that each $\mathbf{p}_{ui}$ contains both positive and negative representations of some features, according to the user explicitly willing its presence or absence in the decision process. Considering the example, the decision of the user about *Harry Potter* is represented in the feature space summing up the embedding of $\langle \text{director}, \text{Chris Columbus} \rangle$ and the opposite of the embedding of $\langle \text{starring}, \text{Johnny Depp} \rangle$. This summation suggests that the user decision is built up of these two features and at the same time avoids the second feature, i.e., it goes in the opposite direction, as shown in Figure 3.2a.

As an additional constraint, we encourage the elements in the user decision path to describe different concepts and different motivations the user uses to discriminate items in the catalog. In order to favor each feature in the path to represent an independent concept, we model this **feature independence** and remove as much as possible information redundancy, KGTORe aims to minimize the correlation between the features of the same decision path. Technically, we introduce a term in the training loss, which will be described in Section 3.3.4.

Finally, to better describe the items based on the user choices, we hereinafter characterize each item i with its restricted feature set $\mathcal{F}_i^*$, where features not used by the users in their decision processes are removed:

$$\mathcal{F}_i^* = \mathcal{F}_i \cap \bigcup_{u \in \mathcal{U}} \mathcal{F}_{ui}. \quad (3.7)$$

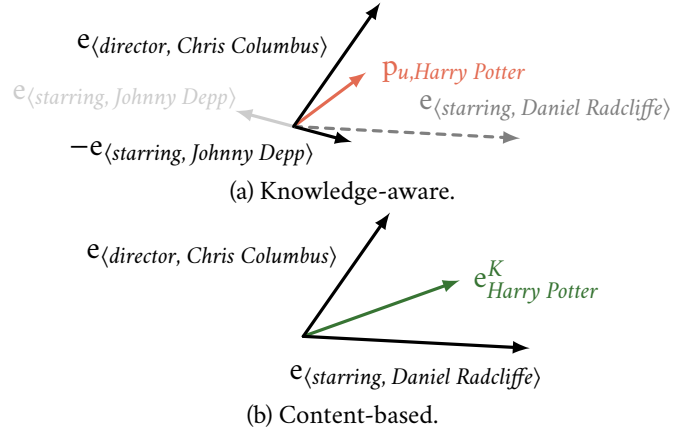


Figure 3.2. Representation of a knowledge-aware decision embedding (a) and a content-based item embedding (b) in a 2D feature space. In (a), the example user has chosen *Harry Potter* because of its director and Johnny Depp *not* starring: the embedding $p_{u, \text{Harry Potter}}$ is the sum of these feature embeddings. Starring *Daniel Radcliffe* does not impact the decision embedding, since it is not in $\mathcal{F}_{u, \text{Harry Potter}}$. In (b), the embedding is the average of the two features describing it.

3.3.3 Knowledge-aware Graph-based Recommendation

With the knowledge-aware modeling of items and user decision processes we have done so far, we move to the representation of users and items. In KGTORe each user and item embeddings, $e_u \in \mathbb{R}^d$ and $e_i \in \mathbb{R}^d$ respectively, are obtained by recursively aggregating the representations of their multi-hop neighbors via a message propagation through L convolutional layers. Different representations at different levels ($e_u^{(0)}, \dots, e_u^{(L)}$) and ($e_i^{(0)}, \dots, e_i^{(L)}$) are the result of mixing content information with the collaborative signal coming from the first-hop neighbor nodes and propagated from the previous layer.

Content Representation Learning of Items

Expressing features of the knowledge graph with latent vectors, KGTORe can represent items based on their explicit characteristics. Such modeling gives a precise content-based representation of each item, positioning it in the latent feature space as an aggregation of the vectors describing its characteristics. Concretely, KGTORe formulates the content-based embedding e_i^K of the item i as:

$$e_i^K = \frac{1}{|\mathcal{F}_i^*|} \sum_{f \in \mathcal{F}_i^*} e_f, \quad (3.8)$$

where e_f is the feature embedding of the feature $f \in \mathcal{F}_i^*$. Figure 3.2b shows a practical example of item content representation.

Content Representation Learning of Users

Analogously to the item content-based representation, users can be described by the knowledge-aware decisions they adopted in their personal history.

In detail, KGTORe frames each user as the combination of the features used for deciding whether to interact with an item, in terms of the knowledge-aware decision embeddings p_{ui} involved within her interactions:

$$e_u^K = \frac{1}{|\mathcal{N}_u|} \sum_{i \in \mathcal{N}(u)} p_{ui}. \quad (3.9)$$

Since p_{ui} is located in the same latent space as the items and explicitly relates to the item features representing the user motivation for the choice of item i , it contributes to the realization of the content-based knowledge-aware user profile. The left side of Figure 3.3 shows how content representations are built from feature embeddings.

Collaborative Representation Learning of Users and Items

We build upon the graph neural network design to recursively propagate embeddings via high-order connectivity. In KGTORe, we get inspired by LightGCN [115] to define the user and item collaborative representations. Here, the aggregation of the neighbors is easily performed by weighting the user and item embeddings at the previous layer based on the topology of the graph with the symmetric normalization term $\frac{1}{\sqrt{|\mathcal{N}(u)|}\sqrt{|\mathcal{N}(i)|}}$, sticking to the standard design of GCN [137]:

$$\begin{aligned} e_i^{C(l)} &= \sum_{u \in \mathcal{N}(i)} \frac{1}{\sqrt{|\mathcal{N}(u)|}\sqrt{|\mathcal{N}(i)|}} e_u^{(l-1)}, \\ e_u^{C(l)} &= \sum_{i \in \mathcal{N}(u)} \frac{1}{\sqrt{|\mathcal{N}(u)|}\sqrt{|\mathcal{N}(i)|}} e_i^{(l-1)}, \end{aligned} \quad (3.10)$$

where $e_u^{(l-1)}$ and $e_i^{(l-1)}$ refer to the holistic representations of user u and item i , respectively, as described in the following. The right side of figure 3.3 represents an overview of how KGTORe builds the collaborative representations.

Holistic Representation of Users and Items

The overall representation of users and items is obtained by combining both their content and collaborative embeddings. In KGTORe, both the items and the users are not solely modeled through their respective neighbors, as in pure collaborative approaches, but also keeping track of the semantic characteristics describing the items and the personal user preferences. The collaborative and the content representations are integrated in a weighted linear combination that counterbalances their

impact. Formally:

$$\begin{aligned} \mathbf{e}_i^{(l)} &= \alpha \mathbf{e}_i^K + (1 - \alpha) \mathbf{e}_i^{C(l)}, \\ \mathbf{e}_u^{(l)} &= \beta \mathbf{e}_u^K + (1 - \beta) \mathbf{e}_u^{C(l)}, \end{aligned} \quad (3.11)$$

where $\alpha, \beta \in \mathbb{R}$ are employed to weigh content and collaborative contributions in items and user representations, respectively. It is noteworthy that, compared to LightGCN, KGTORe adds a knowledge-aware correction term in the item and user embeddings. The joint contribution of collaborative and content signals guarantees capturing long-range contributions and taking advantage of the rich information coming from knowledge graphs, leading to a high personalization capability. Since the content information is mediated by decision trees, only selected relevant features to participate in the embedding modeling, granting a concise and meaningful representation of users and items.

Since $(\mathbf{e}_u^{(1)}, \dots, \mathbf{e}_u^{(L)})$ and $(\mathbf{e}_i^{(1)}, \dots, \mathbf{e}_i^{(L)})$ are recursively calculated, the only trainable embeddings are $\mathbf{e}_u^{(0)}$ and $\mathbf{e}_i^{(0)}$, as well as the feature embeddings that compose the edge embeddings. After L propagation hops, the final user and item embedding representations are obtained as:

$$\mathbf{e}_u = \sum_{l=0}^L \frac{1}{1+l} \mathbf{e}_u^{(l)}, \quad \mathbf{e}_i = \sum_{l=0}^L \frac{1}{1+l} \mathbf{e}_i^{(l)}. \quad (3.12)$$

The scaling factor $1/(1+l)$ is applied to further alleviate the oversmoothing problem when increasing the number of explored hops [115]. Given the embeddings $\mathbf{e}_u$ and $\mathbf{e}_i$, KGTORe predicts the interaction between a user u and an item i as:

$$\hat{r}_{ui} = \mathbf{e}_u^\top \mathbf{e}_i. \quad (3.13)$$

In KGTORe, the only trainable parameters are the user, item, and feature embeddings, so that the **space complexity** is $O(|\mathcal{U}| + |\mathcal{I}| + |\mathcal{F}|)$, where $\mathcal{F}$ is the set of all the item features involved in the systems. Moreover, Eq. 3.13 shows that the **time complexity** of KGTORe for a single prediction is $O(d)$, i.e., as in matrix factorization, it only depends on the embedding size.

3.3.4 Model Optimization

The training phase is responsible for updating the values of the user, item, and feature embeddings. The updates ensure that user-item estimated interactions reflect the real preferences of the users. We use the pair-wise optimization algorithm **Bayesian Personalized Ranking** (BPR), which assumes that a user u prefers a consumed item i^+ over a non-consumed item i^- , and optimizes the model by maximizing, for each pair of i^+ and i^- , a function of the difference $r_{ui^+} - r_{ui^-}$. Concretely, built a training set $\mathcal{T} = \{(u, i^+, i^-) \mid (u, i^+) \in \mathcal{R}, (u, i^-) \notin \mathcal{R}, i^- \in \mathcal{I}\}$, BPR aims to optimize

the following loss, where σ is the sigmoid function:

$$\mathcal{L}_{\text{BPR}} = \sum_{(u,i^+,i^-) \in \mathcal{T}} -\ln \sigma(\hat{r}_{ui^+} - \hat{r}_{ui^-}), \quad (3.14)$$

To model the independence of the features discussed in Section 3.3.2, we minimize the correlation between the feature embeddings within the same decision path, and we take the average over all the user-item interactions:

$$\mathcal{L}_{\text{IND}} = \frac{1}{|\mathcal{R}|} \sum_{(u,i) \in \mathcal{R}} \sum_{f, f' \in \mathcal{F}_{ui}, f \neq f'} |P(e_f, e_{f'})|, \quad (3.15)$$

where P represents the Pearson correlation function returning the correlation coefficient between e_f and $e_{f'}$.

Finally, the objective function minimized with gradient descent is built with the above-mentioned losses and a specific L2-regularization loss:

$$\mathcal{L} = \mathcal{L}_{\text{BPR}} + \lambda_{\text{IND}} \mathcal{L}_{\text{IND}} + \lambda_{\text{REG}} \|\Theta\|_2^2, \quad (3.16)$$

where $\Theta = \{e_u, e_i, e_f \mid u \in \mathcal{U}, i \in \mathcal{I}, f \in \bigcup_{i \in \mathcal{I}} \mathcal{F}_i^*\}$, while λ_{IND} and λ_{REG} are two hyperparameters controlling the independence loss and the regularization term, respectively.

3.4 Experiments

This section presents the experimental setup for assessing the recommendation performance of KGTORe and demonstrating the effectiveness of its components. In particular, we aim to answer the following Research Questions:

RQ1 How does KGTORe perform compared to the state-of-the-art baselines and how does it address the cold-start issue?

RQ2 How effective are the KGTORe components for improving recommendation performance? Which is the impact of their hyperparameters?

RQ3 Can we interpret the results of KGTORe through its semantic embeddings?

3.4.1 Datasets

We evaluated KGTORe and the other baselines on three well-known recommendation datasets, namely *MovieLens 1M*², *Yahoo! Movies*³, and *Facebook Books*⁴. The items within the datasets are mapped through a URI with their corresponding semantic

2. MovieLens 1M Dataset: <https://grouplens.org/datasets/movielens/1m/>

3. Yahoo! Movies User Ratings and Descriptive Content Information, v.1.0: <https://webscope.sandbox.yahoo.com/catalog.php?datatype=r>

4. Facebook Books (Linked Open Data challenge co-located with ESWC 2015): <https://2015.eswc-conferences.org/important-dates/call-RecSys.html>

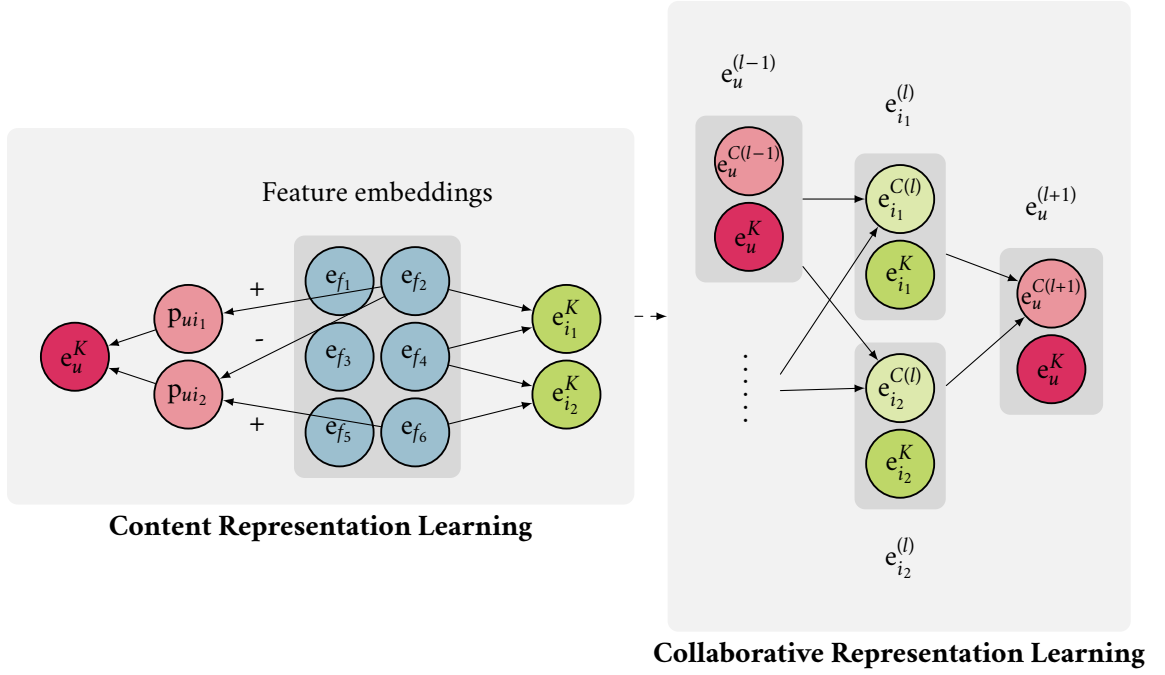


Figure 3.3. Illustration of KGTORe. The embeddings of the item features are averaged to build the item content-based representations $e_{i_1}^K$ and $e_{i_2}^K$; the user content-based representation e_u^K is obtained averaging the decision embeddings, in turns computed with positive and negative contributions from the embeddings of the decision paths.

entity in the public knowledge graph *DBpedia*. *MovieLens 1M* is a dataset in the movie domain that collects 1,000,209 explicit ratings (from 1 to 5) of 6,040 users over a catalog of 3,706 items. *Yahoo! Movies* is a collection of 69,846 movie explicit ratings on a [1-5] scale collected on the *Yahoo! Movies* website up to November 2003. It contains 4,000 user implicit feedback with respect to 2,626 items. *Facebook Books* is a more sparse recommendation dataset in the book domain. It contains 18,978 implicit feedback collected from 1,398 users about 2,933 items. We transform explicit feedback into implicit by applying a threshold of 3. Items with a rating greater or equal to the threshold are marked as positive feedback, or as negative otherwise. Finally, with the aim of improving data quality, we apply an iterative-5-core strategy on the users and items within the datasets: we keep only users and items with at least five occurrences.

3.4.2 Baselines

To measure the effectiveness of KGTORe we experimentally compare its performance with state-of-the-art baselines from different categories: *non-personalized*, *collaborative filtering*, *knowledge-aware*, and *graph-based*. To grant a fair comparison, we train the baselines and look for the hyperparameter configuration that best fits the training data. In the following, we detail the competing baselines, grouped according

to the recommendation category.

Non-personalized Random and Most Popular return random and the most popular recommendations, respectively. They are used as references.

Collaborative Filtering BPR-MF [195] is a state-of-the-art factorization model that shares with KGTORe a pair-wise optimization approach. Item-kNN [139] is an implementation of the k-nearest neighbors algorithm based on item similarities.

Knowledge-Aware KaHFM [18] exploits the information encoded in knowledge graphs for initializing the latent factors in factorization machines [194]. KGFlex [17, 86] follows a sparse factorization approach where the items are modeled through semantic features. Users are viewed only from the most relevant characteristics they have experienced in the past. Both of them are based on a factorization approach.

Graph-Based LightGCN [115] lightens the GCN aggregation phase by removing the feature transformation phase and non-linear activation functions. Each node linearly aggregates the neighbor’s messages by equally weighting all the edges. DGFC [262] looks at the user-item interaction from the perspective of different latent user intents. They introduce a graph disentangling layer to compute intent-weighted representations. Like DGFC, KGIN [261] assumes that users and items interact through a limited set of intents. These intents are shared between all users and built as a combination of knowledge graph relations.

3.4.3 Knowledge Graph and Decision Trees

For each dataset, we download all the DBpedia triples whose subjects are linked to the items of the experimental datasets. In order to be fair with the baselines, we filter the knowledge graph following the procedures described in the original papers of each approach. The entire preprocessing phase is reproducible and accessible from our public repository.

Knowledge Graph Exploration

In KGTORe, similarly to other approaches such as kaHFM and KGFlex, the items are described by a set of semantic features derived from knowledge graph triples. Given an item i and a knowledge graph triple $i \xrightarrow{\rho} \omega$, the pair $\langle \rho, \omega \rangle$ is a feature describing the item i . To remove noisy triples from the knowledge graph, we apply the filtering strategy described in the following.

Filtering Strategy

Knowledge graphs could contain a large amount of information, including noisy triples that poorly contribute to the item description but heavily impact the computational costs of the model. Moreover, these features could negatively affect the

effectiveness of the decision trees built by KGTORe. In particular, some features have been removed as they offer supplementary details that are considered irrelevant for describing the item’s content. Specifically, the following predicates have been filtered out: *dbo:wikiPageWikiLink*, *owl:sameAs*, *schema:sameAs*, *gold:hypernym*, *prov:wasDerivedFrom*, *dbp:wikiPageUsesTemplate*, *owl:differentFrom*, *dbo:wikiPageExternalLink*, *rdfs:seeAlso*, *dbp:wordnet_type*, *rdf:type*, *dbo:thumbnail*. Then, to further clean the graph from the remaining noisy triples, we remove the predicates linking to a very rare object (i.e., any entity with less than 3 occurrences in the graph) and not linking to at least one frequent object (i.e., any entity with more than 10 occurrences in the graph).

Decision Trees

As described in Section 3.3.2, for modeling the user decisions we build the user-level decision trees using the information gain as the splitting strategy. Given the lack of explicitly disliked items, we populate the first node of the user tree with all her rated items and an appropriate amount of randomly sampled negative items, in order to mitigate the effect of random sampling (cf. Section 3.3.2).

Therefore, we proportionally weigh the importance of the positive items to keep the maximum entropy in the first node. More specifically, we select from 5 up to 40 negative items for each positive feedback.

3.4.4 Reproducibility, Evaluation, and Metrics

In the interest of granting reproducibility, we publicly provide our code, data, and instructions for replicating all the phases of our experiments⁵.

For all the models, we conduct the *all unrated items* protocol [227], where for each user the recommendable items are the items not already experienced by her. Applying an 80-20 hold-out strategy, we split the datasets retaining the 80% of the user ratings in the training set, and using the remaining 20% for building the test set. Then, we retain 10% of the obtained training set as a validation set [102]. The embedding size is fixed to 64 for all models. Specifically, in KGTORe, the embedding parameters are randomly initialized with the Xavier initialization [97]. The models have been trained on a set of 20 different hyperparameter configurations, following the Bayesian hyperparameter optimization algorithm. The search space has been configured in accordance with the best parameters cited by the authors in their papers.

Furthermore, we adopt nDCG, which strongly emphasizes top-rated items [142], as the validation metric, and we set the patience value to 5 epochs for the early stopping. Finally, we evaluate the models with the same metric on the test set and report the results in Table 3.1. In addition, we choose that the training batch size depends only on the number of interactions in the three datasets: 64 for *Facebook*

5. <https://github.com/sisinflab/KGTORe>.

Table 3.1. Comparison of accuracy performance between KGTORe and competing methods. All the differences between KGTORe and the baselines are statistically significant based on paired t-tests ($p < 0.05$) except for the values denoted with *.

	Facebook Books				Yahoo! Movies				MovieLens 1M			
	nDCG	HR	Prec.	Recall	nDCG	HR	Prec.	Recall	nDCG	HR	Prec.	Recall
Random	0.0074	0.0256	0.0027	0.0110	0.0065	0.0342	0.0036	0.0090	0.0096	0.0836	0.0091	0.0036
MostPop	0.0787	0.2535	0.0272	0.1117	0.1185	0.3238	0.0571	0.1501	0.1730	0.6537	0.1539	0.0773
BPR-MF	0.0945	0.2973	0.0333	0.1358	0.2303	0.5228	0.0965	0.2843	0.3100	0.8811	0.2686	0.1627
Item-kNN	0.1039	0.3353	0.0378	0.1556	0.2258	0.5033	0.0929	0.2658	0.2932	0.8504	0.2564	0.1490
KGFlex	0.0712	0.2235	0.0246	0.1001	0.1758	0.4360	0.0680	0.2022	0.0111	0.1058	0.0122	0.0053
kaHFM	0.0843	0.2776	0.0300	0.1221	0.2085	0.5064	0.0902	0.2621	0.2992	0.8654	0.2581	0.1526
LightGCN	0.0969	0.3170	0.0355	0.1435	0.2230	0.5085	0.0937	0.2712	0.3081	0.8813	0.2683	0.1614
DGCF	0.1001	0.3192	0.0354	0.1444	0.2356	0.5308	0.0981	0.2848	0.3124	0.8819	0.2714	0.1637
KGIN	0.1023	0.3112	0.0356	0.1448	0.2368	0.5500*	0.1006*	0.2971	0.3082	0.8854	0.2695	0.1638
KGTORe	0.1188	0.3601	0.0414	0.1698	0.2471	0.5566	0.1018	0.2944	0.3212	0.8957	0.2772	0.1710

Books, 256 for *Yahoo! Movies*, and 2048 for *MovieLens 1M*. For evaluating the models' accuracy we compute nDCG, HR, Precision, and Recall on the test set. All the metrics are measured with a cutoff at 10, i.e., considering the top-10 recommended items to each user.

3.5 Discussion

In the following, to answer the Research Questions, we discuss the performance of KGTORe compared with the state-of-the-art baselines. Then, we report some hyperparameter and ablation studies performed to assess the effectiveness of the KGTORe components.

3.5.1 Performance Comparison with State-of-the-Art Baselines (RQ1)

Table 3.1 shows the overall comparison of the accuracy performance between KGTORe and the competing baselines, along with the results obtained with some unpersonalized and classic recommendation models, i.e., *Random*, *Most Popular*, *BPR-MF*, and *Item-kNN*. For each model, the results refer to the performance on the test set of the hyperparameter configuration that showed the best nDCG value on the validation set. We highlight the best and the second-best result, by **boldfacing** the former, and underlying the latter. All the results have been statistically validated with a Student's paired t-test with p -value 0.05 (all the significance values can be found in our public repository). Since the majority of the results are statistically significant, we denote with a * the non-significant results only.

The reader can notice that KGTORe consistently outperforms all the other baselines when considering nDCG, which is our target metric. This proves the effec-

tiveness of the approach proposed by KGTORe. We believe this is due to two major factors:

- the content information coming from the knowledge graph helps to better describe the items in the catalog and supports the collaborative signal propagated through the GNN;
- the explicit interpretation of the edges between users and items drives the training of KGTORe, aiming to explain the user-item connections by means of a fine-grained set of features relevant to the users and precisely relating to the specific interaction;
- the features selected for describing the interactions frame restricted and lightweight sets of embeddings, which are trained in a focused fashion by the users and items involving them in their interactions and descriptions.

Regarding Hit Ratio, Precision, and Recall, it surpasses all the baselines on the three datasets, except for Recall when measured on *Yahoo! Movies*. It is notable to point up that models based on graph architectures, i.e. LightGCN, DGCF, and KGIN, regularly achieve competing performance on the three datasets. Among the classic models, ItemKNN performs appreciably on *Facebook Books*, while BPR-MF achieves a competing $nDCG$ value *MovieLens 1M*. By looking at factorization-based knowledge-aware models, i.e. KGFlex and kaHFM, it is clear that they are penalized by their incapability of mining long-distance connections like graph-based models do. In fact, LightGCN, DGCF, and in particular KGIN and KGTORe, which also make use of knowledge, always outperform them.

Finally, we carry out a comparative analysis to investigate the ability of KGTORe to provide cold users with accurate recommendations, as compared to the other baseline methods. Undoubtedly, the cold-start issue represents a serious problem in recommender systems which may result in the loss of new potential users [32]. To do so, we divide the users of *Yahoo! Movies* into quartiles based on the number of their interactions and separately compute $nDCG$ on these groups. The results reported in Figure 3.4 show that KGTORe outperforms the baselines not only in addressing this issue for the coldest users but also in leveraging the greater availability of information becoming more and more accurate than the other baselines as users become warmer.

3.5.2 Ablation and Hyperparameter Studies (RQ2)

In KGTORe different components cooperate to achieve effective performance. In the following, we investigate how each component impacts the model behavior by conducting hyperparameters and ablation studies. In particular, we examine the trade-off between the content and collaborative signal, how the model benefits from adequate user decision modeling, and the independence of characteristics among the same path.

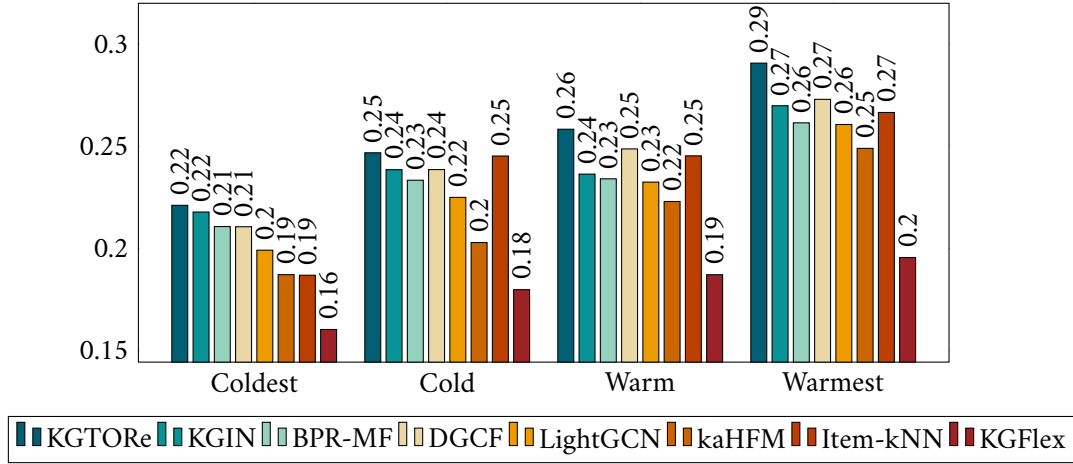


Figure 3.4. Accuracy performance (nDCG) of KGTORe and the baselines on *Yahoo! Movies* when separately computed on four distinct user groups, corresponding to the quartiles of the number of their interactions.

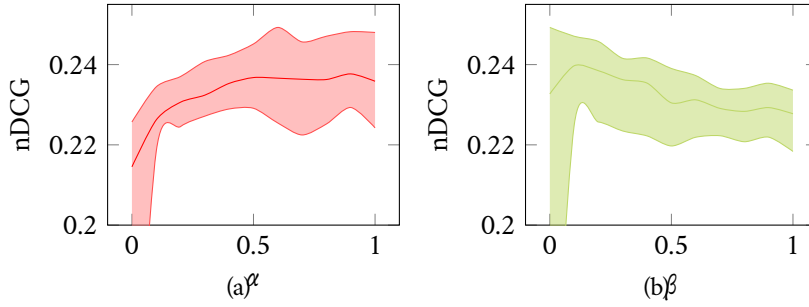


Figure 3.5. Impact of the knowledge-aware terms into the user and item representations at different values of α and β on *Yahoo! Movies*.

Impact of Knowledge-Aware Embeddings

To analyze the influence of the knowledge-aware content representations of items and users (*cf.* Section 3.3.3), we plot in Figure 3.5 the performance of KGTORe on *Yahoo! Movies* when separately changing α and β in $[0.0, 1.0]$. These values control the impact of our content-based correction terms in the item and user embeddings that are propagated through the GNN. The figure shows the average, the worst, and the best performance obtained when fixing α and varying β , and vice versa.

Looking at the trade-off between the content and collaborative terms in the item representations (Figure 3.5a), we find that increasing α , i.e., raising the contribution of the content term at the expense of the collaborative effect, generally gives a positive trend in terms of ranking quality. This behavior suggests that a carefully built content representation, i.e., describing items with features that are relevant to the users, not only compensates but also outperforms the propagation of the collaborative signal

towards the item embeddings.

At a first glance, a higher contribution of the content term in the user representations (Figure 3.5b) seems to harm the recommendation performance. However, it must be observed that increasing β makes the content information compete with the collaborative one and turns off the propagation of the collaborative signal towards the user embeddings. In fact, while $\beta = 0$ seems to provide the maximum accuracy performance, it should be noted that this value also allows the worst nDCG performance (≈ 0.13) when α is not properly set. Instead, increasing β , i.e., giving the user profile its knowledge-aware representation along with the collaborative one, makes KGTORe more stable in terms of performance, especially when β is in the range $[0.1, 0.2]$, when KGTORe reaches its best trade-off between accuracy and stability around its average performance.

In short, we find that the knowledge-aware content information extracted from the decision trees helps describe the users as long as it does not dominate the collaborative signal. However, the same information is extremely valuable in the item descriptions, which are built up of the users' most relevant features, suggesting the pivotal role of the knowledge-aware user decision process modeling.

Role of the User Decision Process Modeling

In order to assess the importance of the user decision process modeling, we perform two ablation studies on *Yahoo! Movies*. In fact, this component of KGTORe has two main roles that we separately analyze:

- decomposing each user-item interaction as features describing the motivations behind such interaction, i.e. $\mathcal{P}_{ui}$,
- choosing which characteristics should be used to describe the items, based on the users' choices, i.e. $\mathcal{F}_i^*$.

In the first study, we remove the ability of this component to meaningfully describe the user-item interactions through sets of features. To do this, we change at random the features in the set $\mathcal{F}_{ui}$, thus the path $\mathcal{P}_{ui}$, of each user-item interaction, substituting *a posteriori* each feature with one randomly picked from $\bigcup_{i \in \mathcal{I}} \mathcal{F}_i^*$, i.e., the set of all the features in the system. We call this version KGTORe_{SDT}, where SDT states for Shuffled Decision Trees. While the picked features are still the ones obtained from the user-level decision trees, this ablation is extremely detrimental for the first role of the user decision process modeling, since it affects the semantics of the user-item decision paths resulting in meaningless decision embeddings.

As stated above and as also pointed out by the analysis in Section 3.5.2, the user decision process modeling is pivotal for injecting the set of features essential for the item descriptions in the system. To assess this finding, the second ablation study wrecks the feature selection role of the user decision process modeling by building the $\mathcal{F}_{ui}$'s with features randomly picked from $\mathcal{F}_i$. According to Eq. 3.7, modifying $\mathcal{F}_{ui}$ will in turn change the set of $\mathcal{F}_i^*$ hence totally removing the role and effects of the

Table 3.2. Performance of KGTORe on *Yahoo! Movies* when compromising the decision tree outcome.

	nDCG	Precision
KGTORe_{SDT}	0.2201	0.0925
KGTORe_{RDT}	0.2113	0.0905
KGTORe	0.2363	0.0963

user-level decision trees *a priori*. We called this version KGTORe_{RDT}, where RDT states for Random Decision Trees. This ablation differs from the previous one, where the $\mathcal{F}_i^*$'s are instead the ones obtained from the true analysis based on decision trees. To eliminate any random effect of this experiment, we repeat this ablation 10 times and report the average performance on the three datasets in Table 3.2. Moreover, to fairly balance the contribution of the content and collaborative signals, we fix α and β to 0.5.

The obtained results highlight the expected behaviour.

In the first study, it can be seen a drop in performance in both nDCG and precision compared to vanilla KGTORe thus suggesting how the explicit interpretation of user-item interaction can effectively drive the GNN-based RSs. The same trend is also reflected in the second study, where performance is even lower than in the previous ablation, emphasizing the effectiveness of retrieving the features behind the user intents to describe the items in the catalog.

Effectiveness of the Independence Loss

As discussed in Section 3.3.2, we encourage the feature embeddings to be as much as possible independent of each other, to let them describe different concepts. To assess the positive effect of this component, we have removed its term from the training loss and compared the recommendation performance on the three datasets with the complete KGTORe model. Figure 3.6 shows the decrease of nDCG and Precision caused by the lack of independence between the semantic features. It can be observed how, while the results without independence loss are still acceptable, this term plays a significant role in making KGTORe deliver outstanding performance.

3.5.3 Interpretability of KGTORe (RQ3)

Through the training process, KGTORe places users and items in a shared latent space in form of embeddings. To qualitatively show the meaning of the interactions between the embeddings and how KGTORe can learn the information encoded in training data, we analyze which are the feature embeddings that better fit a real user from the *Facebook Books* dataset. To accomplish this objective, we compute the dot product between the embedding of the target user and the ones of the features in the system. This product gives us an interpretable measure of the affinity between the

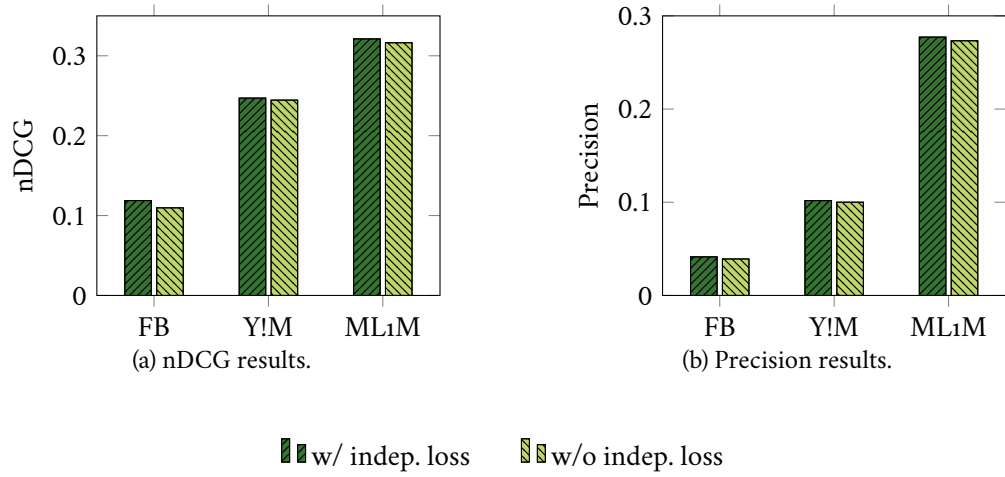


Figure 3.6. Performance of KGTORe with and without independence loss. The bars show an accuracy decrease over the three datasets.

Table 3.3. Semantic features ordered by affinity with the test user.

Feature	Affinity
<i><author, Anne Rice></i>	0.1808
<i><illustrator, The Vampire Chronicles></i>	0.1769
<i><subject, The Vampire Chronicles novels></i>	0.1735
<i><subject, Innovation Publishing titles></i>	0.1504
<i><subsequent work, The Tale of the Body Thief></i>	0.1405

user and the features that KGTORe inherently uses to provide the recommendations.

Table 3.3 shows the most affine features with our target user and points out how, according to the KGTORe embeddings, this user may be mostly interested in books written by Anne Rice, as well as horror and vampire-themed books. In fact, looking at the training set of the target user, we notice she has interacted with books like *Paradise Lost*, *The Vampire Lestat*, *Interview with Vampire* (written by Anne Rice), and *The Witching Hour*, which entirely reflect the affinities highlighted by the interactions between the embeddings. Therefore, this analysis empirically shows how KGTORe can learn the relations between users, items, and features when they are immersed in the latent space, thus being able not only to provide accurate recommendations but also to provide an interpretation of the interactions between the semantic object placed within it.

3.6 Conclusion and Future Work

This chapter has presented KGTORe, a GNN-based knowledge-aware recommendation model able to leverage explicit motivations behind each user-item interaction, which lead the model to highly personalized recommendations. KGTORe consists of three main components: **Item Modeling**, which describes items in the catalog through the semantic features extracted from a knowledge graph; **User Decision Process Modeling**, which decomposes each user-item interaction in terms of the semantic features representing the rationale of the choice, unveiled by means of a decision analysis performed with user-level decision trees; **Knowledge-aware Graph-based Recommendation**, which integrates the semantic information into a graph neural network able to learn latent representations for the semantic features and for the users and the items related with them. In contrast with other solutions proposed in the literature, KGTORe does not share intents between users but is able to characterize each graph edge with a different selection of semantic features. Through the use of decision trees, we efficiently deal with the extensive information coming from knowledge graphs by describing user decisions with the most discriminative characteristic. Despite its simplicity, when compared with state-of-the-art solutions, KGTORe demonstrates outstanding performance in terms of accuracy, suggesting that its approach effectively mines the user-item relations and the motivations behind them, as shown in an exhaustive set of ablation studies. We believe that future works could deeply investigate other approaches for selecting user-relevant features with the same simplicity and interpretability of decision trees.

Chapter 4

Lightweight Knowledge-Enhanced Graph Recommendation

The recent integration of Graph Neural Networks (GNNs) into recommendation has led to a novel family of Collaborative Filtering (CF) approaches, namely Graph Collaborative Filtering (GCF). Following the same GNNs wave, recommender systems exploiting Knowledge Graphs (KGs) have also been successfully empowered by the GCF rationale to combine the representational power of GNNs with the semantics conveyed by KGs, giving rise to Knowledge-aware Graph Collaborative Filtering (KGCF), which use KGs to mine hidden user intents. Nevertheless, empirical evidence suggests that computing and combining user-level intent might not always be necessary, as simpler approaches can yield comparable or superior results while keeping explicit semantic features. Under this perspective, user historical preferences become essential to refine the KG and retain the most discriminating features, thus leading to concise item representation. Driven by the assumptions above, we propose KGUF, a KGCF model that learns latent representations of semantic features in the KG to better define the item profile. By leveraging user profiles through decision trees, KGUF effectively retains only those features relevant to users. Results on three datasets justify KGUF’s rationale, as our approach is able to reach performance comparable or superior to SOTA methods while maintaining a simpler formalization.¹

1. This chapter is based on the work published in the *International Workshop on Graph-Based Approaches in Information Retrieval* (IRonGraphs 2024) “KGUF: simple knowledge-aware graph-based recommender with user-based semantic features filtering”.

4.1 Introduction

Recommender Systems find widespread application across diverse domains, owing to their proficiency in furnishing users with personalized items [60, 293, 299]. Typically, these systems rely on past user-item interactions to extrapolate new preferences and generate personalized suggestions [78, 116]. Collaborative Filtering [140] (CF) methods adeptly extract similarity patterns from user-item interactions, constituting (to date) the most diffused and winning recommendation paradigm [116, 165, 195]. Nevertheless, by design, the learned user and item profiles in traditional CF recommender systems cannot embed the information conveyed by the underlying multi-hop user-item relations, thus disregarding a rich source of potentially-useful information [167, 259]. Following the popular graph representation learning [106] wave, Graph Neural Networks (GNNs) have recently been applied to CF approaches, giving rise to a fresh family of recommender systems [115, 167, 275, 294], better known as Graph Collaborative Filtering (GCF). Despite the success of traditional CF and GCF recommendation systems, such approaches often overlook the nuances in user preferences related to specific item characteristics; indeed, this represents a missed opportunity for enhancing user personalization. As opposed to CF systems, content-based recommendation [96] leverages the attributes of historically-favored products, proposing unseen items that align with user distinct tastes [116]. In these systems, Knowledge Graphs (KGs) have been shown to provide a meaningful source of structured information [17, 76, 320, 321] used to craft features that enrich the item description, thanks to the mapping between items and semantic entities, and helps to tackle open challenges in recommendation, such as the graph sparsity.

As a recent trend, an increasing number of recommender system algorithms currently propose to combine the representational capabilities of GNNs with the rich information conveyed by KGs, resulting in a novel hybrid family of recommendation techniques which might be suitably named as Knowledge-aware Graph Collaborative Filtering [16, 127, 261, 289, 320] (KGCF). Most of KGCF methods model the user intent to enhance recommendation quality and interpretability, thus outperforming similar baselines techniques [162, 258, 261]. However, such architectures often require convoluted aggregations of several ad-hoc methodologies and strategies to learn meaningful user and item profiles, making it hard to apply KGCF approaches in real-world scenarios. In this regard, we believe that these architectures can be dramatically simplified while maintaining comparable or even superior performance. In particular, we consider to utilize the content information from KGs solely for describing item representations, refraining from any attempt to directly incorporate it into building user profiles, which can be instead indirectly learned by means of collaborative filtering techniques.

Driven by these assumptions, this chapter presents a novel KGCF recommendation algorithm named **KGUF** (short for **Knowledge Graph User-based Filtering**) designed to efficiently harness side information from KGs. From a technical per-

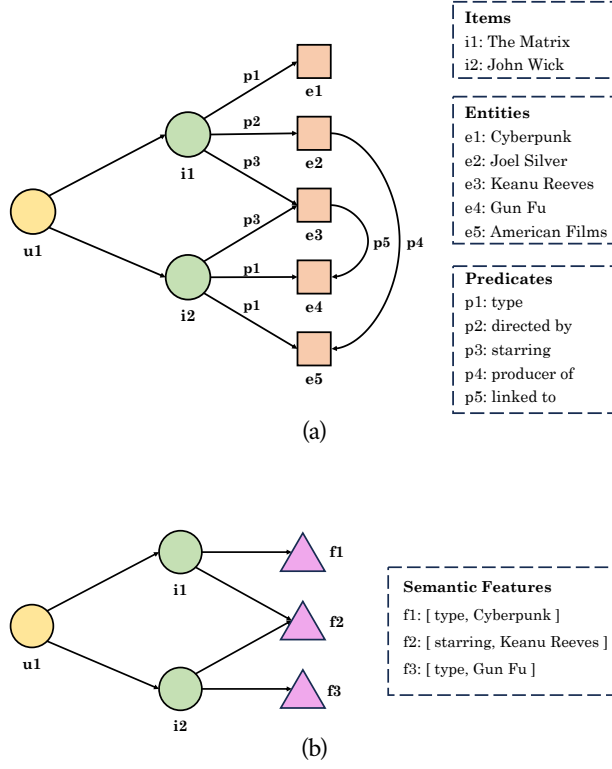


Figure 4.1. An example of how KGUF models the item representation. Figure 4.1a shows the entities in the knowledge graph linked to the items. In Figure 4.1b, the most relevant semantic features selected by KGUF are reported.

spective, KGUF learns user and item embeddings through linear propagation on the collaborative graph. In contrast to other approaches, our solution, sketched in Figure 4.3, is able to integrate latent representations for semantic features right into the item representation. Moreover, sharing the same principle but replacing it with a simple mechanism, we propose substituting the intent modeling with a decision tree mechanism that selects the most meaningful semantic features in the KGs and filters out noisy features that poorly contribute to the item description but heavily impact the computational costs. In our work, we take inspiration from the KGTORE method [162] and employ a similar technique to identify the most influential user features, thereby eliminating the need to process the entire knowledge graph. However, our approach differs from KGTORE in that we do not explicitly incorporate the reasoning behind user-item interactions into our model. This results in a more lightweight and efficient methodology.

All in all, the key contributions of this chapter are summarized as follows:

1. We propose a novel recommendation model, namely KGUF, which effectively selects and integrates user-relevant semantic features during the graph learning phase to improve item representation.

2. We conduct extensive experiments over three well-known recommendation datasets to assess the efficacy of our framework and evaluate the contribution of collaborative and content signals on KGUF.
3. We delve into using decision trees for selecting semantic features relevant to users, investigating how the effects of negative sampling and the presence of constraints regarding the tree depth affect the recommendation performance.

To foster the replicability of the experiments and results, we release the code for KGUF and the baselines, along with the datasets and configurations ².

4.2 Related work

This section aims to survey and compare other approaches in recommendation literature leveraging KGs and graph representation learning strategies.

4.2.1 *KGs information in recommendation*

As a general trend, the incorporation of information coming from KGs into recommendation systems has consistently demonstrated improvements in model performance and interpretability across diverse domains [17, 76, 320, 321]. Indeed, the utilization of KGs has been effective in boosting different recommendation tasks, including hybrid collaborative/content-based recommendation [17, 144], where the KG is of support to the lack of collaborative information, eventually enhancing performance; explainable recommendation [19, 239], where the KG serves as a robust information source for delivering human-like explanations; and user modeling [18, 253], with the semantic description of items contributing to defining user profiles. KGs also offer a solution to the cold-start problem by augmenting observed interactions with KG-reachable unobserved items [240]. However, it is crucial to note that in some cases, improper extraction of information may potentially weaken the model capabilities [51].

4.2.2 *Graph representation learning for recommendation*

Recommender systems based on graph representation learning strategies have demonstrated their superior performance by mining high-order user-item relationships embedded within the graph structure [218, 220, 237, 275]. Inspired by the graph convolutional network layer (GCN) [137], early approaches like PinSage [293] and NGCF [259] have adapted GCN to the user-item bipartite graph. Later research has explored simplifying the GCN layer, as seen in methods like LightGCN [115] and LR-GCCF [48], which streamline model complexity by eliminating non-linearities and feature transformations in node embeddings. Additionally, UltraGCN [167] proposes a mathematical proxy to the infinite message-passing layer, thus limiting the

2. <https://github.com/sisinflab/KGUF>.

negative effects of over-smoothing. Another avenue of research focuses on pruning noisy user-item interactions to discern meaningful preference patterns. Inspired by the graph attention network [245] (GAT), methodologies like DGCF [262] aim to identify the underlying intents of individual user-item interactions on a fine-grained level. Furthermore, works such as [13] prove how user-item edges can incorporate multimodal content, such as user reviews, to enhance recommendation and address the challenge of over-smoothing.

4.2.3 *KGs information and graph representation learning for recommendation*

Recent literature [127, 289, 320] has ventured beyond traditional collaborative filtering graph-based methods by incorporating information from KGs to enhance item descriptions. One example is KGAT [258], which explicitly models the KG in an end-to-end manner. It refines node embeddings using an attention mechanism to discern the importance of neighbors. In a different approach, the authors of KGIN [261] utilize the KG to define users' intents as weights, combining these weights with their relations. CKAN [266] proposes a method that integrates collaborative signals with knowledge data. More recently, KGs have been employed to formulate first-order logical rules, combined with graph embeddings, enabling recommenders based on neuro-symbolic reasoning [225]. Adding to this evolving landscape, in KGTORe [162] the authors propose an explicit interpretation of the edge between the user and the item, which is accounted for to drive the training of a GNN-based recommender system. Unlike previous models, KGTORe does not need to process the entire KG at training time but selects the most discriminative features for users, enhancing performance and personalization. Drawing inspiration from KGTORe, we employ a similar mechanism to pinpoint the most discriminative features for users, thereby sidestepping the necessity of processing the entire knowledge graph. Nonetheless, our methodology diverges from KGTORe by avoiding explicit integration of the rationale behind user-item interaction in our model, leading to a lighter and more efficient approach.

4.3 Background

In this section, we introduce the notion of KGs and the basics of decision trees used for content modelling, along with the problem we aim to address.

4.3.1 *Knowledge graphs for recommendation*

A knowledge graph stores structured information of real-world facts in the form of a heterogeneous graph which can be used to retrieve item attributes [42, 93]. Formally, it is presented as a set of triples in the form $\sigma \xrightarrow{\rho} \omega$. Each triple indicates that the head entity σ is connected to the tail entity ω through the relation ρ . Given a set of

entities $\mathcal{V}$ and a set of relations $\mathcal{R}$, a knowledge graph $\mathcal{KG}$ can be represented as follows:

$$\mathcal{KG} = \{\sigma \xrightarrow{\rho} \omega \mid \sigma, \omega \in \mathcal{V}, \rho \in \mathcal{R}\}. \quad (4.1)$$

Mapping items in the catalog using knowledge graphs, i.e., $\mathcal{I} \subseteq \mathcal{V}$, enables the descriptions of items based on semantic features.

4.3.2 Decision trees

A decision tree is a hierarchical tree-like structure used in machine learning for classification tasks. It iteratively partitions the dataset into subsets depending on the values of input features, resulting in a tree structure in which each internal node represents a feature-based decision. The decision-making process is guided by criteria such as Gini impurity or information gain.

In practical terms, decision trees aim to identify variables whose values contribute the most to reducing another random variable's entropy, or uncertainty. Mathematically, the entropy $H(V)$ of a random variable V with k possible values in $v_1, \dots, v_k$ is formally defined as:

$$H(V) = - \sum_{i=1}^k \Pr(V = v_i) \log_2 \Pr(V = v_i). \quad (4.2)$$

If observing the value of another variable X with n possible values in $x_1, \dots, x_n$ results in a reduction of the entropy of V , then X is said to exhibit information gain over V :

$$\begin{aligned} IG(V, X) &= H(V) - H(V|X), \\ H(V|X) &= \sum_{i=1}^n P(X = x_i) H(V|X = x_i). \end{aligned} \quad (4.3)$$

Decision trees are one possible lightweight solution to filter out noisy semantic features in the KG and select the most relevant ones from the user perspective.

4.4 Methodology

In this section, we present KGUF and its components. In particular, we delve into the semantic item modeling and its refinements based on user decisions. Then, we introduce our simple but effective knowledge-aware aggregation schema.

4.4.1 Modeling items through knowledge graph

Assuming that each item in the item set $\mathcal{I}$ of our recommendation problem is associated with an entity in $\mathcal{V}$, essentially being one of the entities within the knowledge graph ($\mathcal{KG}$), we can model the items in the catalog by leveraging the knowledge embedded in the KG. The rationale behind this modeling lies in the

notion that each item can be comprehensively characterized by a distinct set of features, some of them influencing user decisions in varying ways.

Given this premise, the KG can be explored to identify connections with other entities for each item, serving as its attributes in the form $\sigma \xrightarrow{\rho} \omega$, where ρ and ω represent the predicate and object of the KG triple, respectively. As an example, if properly encoded in the reference knowledge graph, the movie *Harry Potter* could be semantically described by the predicate-object pairs $\langle \text{director}, \text{Chris Columbus} \rangle$ and $\langle \text{genre}, \text{Fantasy} \rangle$. Formally, we utilize the set of semantic features to depict the item $i \in \mathcal{I}$ based on its first-order connectivity extracted from the $\mathcal{KG}$:

$$\mathcal{F}_i = \{ \langle \rho, \omega \rangle \mid i \xrightarrow{\rho} \omega \in \mathcal{KG} \}. \quad (4.4)$$

4.4.2 Refining item features with user decision

When confronted with a novel item, users focus on a subset of its numerous features. To capture this user behavior, KGUF uses decision trees to conduct a behavioral analysis at the user level, leveraging her past ratings. The objective is to discern the most pertinent features within the knowledge graph that effectively describe items in the catalog while filtering out noisy ones. This tailored approach ensures that the model accentuates features that hold significant influence across all users in the system.

Formally, let $\mathcal{I}_u^+ \subseteq \mathcal{I}$ denote the set of items positively rated by each user u in the set $\mathcal{U}$ of our recommendation problem, and let $\mathcal{I}_u^- = \mathcal{I} \setminus \mathcal{I}_u^+$ be the set of items not enjoyed by user u . Let $\tilde{\mathcal{I}}_u^- \subseteq \mathcal{I}_u^-$ be a selection of items randomly sampled from $\mathcal{I}_u^-$, such that $|\tilde{\mathcal{I}}_u^-| = |\mathcal{I}_u^+|$. Finally, $\mathcal{I}_u = \mathcal{I}_u^+ \cup \tilde{\mathcal{I}}_u^-$ is the set of items selected to depict the past behavior of the user.

For each user u , the set of semantic features $\mathcal{F}_u = \bigcup_{i \in \mathcal{I}_u} \mathcal{F}_i$ is used to compute the user decision tree T_u that, accordingly to the definition of Information Gain [17], is able to find the minimum subset of semantic features $\mathcal{F}_u^{T_u} \subseteq \mathcal{F}_u$ that can distinguish whether an item belongs to $\mathcal{I}_u^+$ or $\tilde{\mathcal{I}}_u^-$. Finally, each item i is formally described by the subset of its semantic features that appear in at least one user decision tree:

$$\mathcal{F}_i^* = \mathcal{F}_i \cap \bigcup_{u \in \mathcal{U}} \mathcal{F}_u^{T_u}. \quad (4.5)$$

4.4.3 Knowledge-aware aggregation

Inspired by LightGCN [115], the embeddings of each user and item, denoted as $\mathbf{e}_u \in \mathbb{R}^d$ and $\mathbf{e}_i \in \mathbb{R}^d$ respectively, are acquired through the recursive aggregation of their multi-hop neighbors' representations via message propagation across L convolutional layers. The resulting representations at various levels, $(\mathbf{e}_u^{(0)}, \dots, \mathbf{e}_u^{(L)})$ and $(\mathbf{e}_i^{(0)}, \dots, \mathbf{e}_i^{(L)})$, emerge from the integration of content information with collaborative signals originating from first-hop neighbor nodes and propagated from the preceding layer.

By expressing knowledge graph features through latent vectors e_f , KGUF adeptly represents items based on their explicit characteristics. This modeling approach furnishes an explicit content-based representation for each item, situating it within the latent feature space as an amalgamation of vectors delineating its distinct characteristics. Formally:

$$\begin{aligned} e_u^{(l)} &= \sum_{i \in \mathcal{N}(u)} \frac{1}{\sqrt{|\mathcal{N}(u)|} \sqrt{|\mathcal{N}(i)|}} e_i^{(l-1)}, \\ e_i^{(l)} &= \frac{\alpha}{|\mathcal{F}_i^*|} \sum_{f \in \mathcal{F}_i^*} e_f + (1 - \alpha) \sum_{u \in \mathcal{N}(i)} \frac{1}{\sqrt{|\mathcal{N}(u)|} \sqrt{|\mathcal{N}(i)|}} e_u^{(l-1)}, \end{aligned} \quad (4.6)$$

where α serves as a weight factor for balancing content and collaborative contributions in item representations.

Noticeably, in contrast to LightGCN, KGUF introduces a knowledge-aware correction term in the item embeddings, which will affect also user representation through message passing schema. The collaborative and content signals collaboratively contribute to capturing long-range influences and leveraging the rich information derived from knowledge graphs. The use of decision trees to mediate content information guarantees that only relevant features engage in embedding modeling, ensuring a succinct and meaningful representation of users and items.

After completing L propagation hops, the user and item embeddings are derived as follows:

$$\begin{aligned} e_u^* &= \sum_{l=0}^L \frac{1}{1+l} e_u^{(l)}, \\ e_i^* &= \sum_{l=0}^L \frac{1}{1+l} e_i^{(l)}. \end{aligned} \quad (4.7)$$

The introduction of the scaling factor $1/(1+l)$ serves to mitigate the oversmoothing problem that may arise with an increased number of explored hops [115].

4.4.4 Model learning

The training stage involves learning the values of user, item, and feature embeddings to align user-item estimated interactions with the actual preferences of users. Given the embeddings e_u^* and e_i^* , we employ the dot product of user and item representations as the prediction, expressed as:

$$\hat{r}_{ui} = e_u^{*\top} e_i^*. \quad (4.8)$$

To optimize the model parameters, we utilize the pairwise optimization algorithm Bayesian Personalized Ranking (BPR). This algorithm operates on the assumption that a user u favors a consumed item i^+ over a non-consumed item i^- . It optimizes the

model by maximizing, for each pair of i^+ and i^- , a function based on the difference $r_{ui^+} - r_{ui^-}$. Specifically, by constructing a training set $\mathcal{T} = \{(u, i^+, i^-) \mid (u, i^+) \in \mathcal{R}, (u, i^-) \notin \mathcal{R}, i^- \in \mathcal{I}\}$, BPR aims to optimize the following loss, where σ represents the sigmoid function:

$$\mathcal{L}_{\text{BPR}} = \sum_{(u, i^+, i^-) \in \mathcal{T}} -\ln \sigma(\hat{r}_{ui^+} - \hat{r}_{ui^-}), \quad (4.9)$$

Finally, we use a joint learning approach to include the L2-regularization loss as follows:

$$\mathcal{L} = \mathcal{L}_{\text{BPR}} + \lambda \|\Theta\|_2^2, \quad (4.10)$$

here Θ encompass all the learnable parameter, specifically $\Theta = \{e_u, e_i, e_f \mid u \in \mathcal{U}, i \in \mathcal{I}, f \in \cup_{i \in \mathcal{I}} \mathcal{F}_i^*\}$, while λ controls the regularization term.

4.5 Experimental settings

This section delineates the experimental configuration employed to evaluate the recommendation performance of KGUF.

4.5.1 Datasets and Knowledge Graph

We compare the performance of KGUF with state-of-the-art solutions on three well-known recommendation datasets: *MovieLens 1M*³, *Yahoo! Movies*⁴, and *Facebook Books*⁵. *MovieLens 1M* and *Yahoo! Movies* are two datasets in the movie recommendation domain that collect explicit ratings (ranging from 1 to 5) from 6,040 and 4,000 users, respectively, over catalogs of 3,706 and 2,626 items. Overall, they report 1,000,209 and 69,846 ratings, respectively. *Facebook Books* is a dataset that collects 18,978 implicit user feedback in the book recommendation domain. Since its ratings refer to 1,398 users and 2,933 items, it is characterized by greater sparsity. All the datasets are evaluated in an implicit feedback scenario: explicit ratings are binarized by applying a threshold value of 3. Only ratings greater or equal to the threshold value are recorded as implicit positive feedback. In order to reduce the noisy contribution of users and items with too few occurrences, we apply an iterative 5-core strategy, dropping users and items with less than five appearances. Through explicit linkings, the items in the dataset catalogs are directly linked to their semantic representation in *DBpedia*, a well-recognized public knowledge graph. For a fair comparison, we filter the noisy item-linked entities following, for each model, the filtering approach reported by the original authors. The overall statistics of the tested datasets are reported in Table 4.1.

3. MovieLens 1M Dataset: <https://grouplens.org/datasets/movielens/1m/>

4. Yahoo! Movies User Ratings and Descriptive Content Information, v.1.0: <https://webscope.sandbox.yahoo.com/catalog.php?datatype=r>

5. Facebook Books (Linked Open Data challenge co-located with ESWC 2015): <https://2015.eswc-conferences.org/important-dates/call-RecSys.html>

Dataset	# Users	# Items	# Interactions	Sparsity
Facebook Books	1,398	2,933	18,978	99.54%
Yahoo! Movies	4,000	2,626	69,846	99.34%
Movielens 1M	6,040	3,706	1,000,209	95.53%

Table 4.1. Statistics of tested datasets.

4.5.2 Baselines

We empirically validate the effectiveness of KGUF by comparing its performance in terms of recommendation accuracy with different baselines from five related categories described in the following.

- **Unpersonalized.** We test Random and Most Popular recommendations as a basis for discerning significance and behaviors influenced by popularity.
- **Collaborative Filtering.** BPR-MF [195] and Item-kNN [139] are two widely recognized collaborative approaches. The former follows a pair-wise learning approach as KGUF, and the latter exploits similarities between items by computing the nearest neighbors.
- **Knowledge-Aware.** Among these models, we selected KaHFM [18] and KGFlex [17, 86], since they share with KGUF the feature representation but differ in the way they exploit it. KaHFM uses knowledge for initializing latent factors in factorization machines. KGFlex, instead, models the user-item interaction by means of a sparse collection of the most relevant semantic features.
- **Graph-Based.** LightGCN [115] rethinks the classic GCN aggregation schema by discarding feature transformation and non-linear activation functions. It results in linear aggregations with improved performance. DGFC [262] introduces a graph disentangling strategy for intercepting hidden user intents to model intent-dependent user-item interactions.
- **Graph-based with Knowledge.** KGIN [261] exploits the KG for computing users' intents as a weighted combination of semantic relations. User-item connections are interpreted by means of a limited set of relevant intents. KG-TORe [162] mines the user motivations by looking at the discriminant semantic features in her history. These explicit features are then used to enrich the user and the item node representation.

4.5.3 Evaluation protocol and reproducibility

For the sake of reproducibility, we provide our code, data, and comprehensive instructions to enable the replication of all stages in our experimental procedures⁶.

6. <https://github.com/sisinflab/KGUF>.

This work follows the same evaluation protocol and experimental setting of [162]. Moreover, in order to compare KGUF with the same results, we adopt the reproducibility framework Elliot [9]. Each dataset is split into train, test, and validation sets by retaining 72%, 20%, and 8% of the original user ratings, respectively [102], and adopting the *all unrated protocol* [227].

Coherently with [162], the embedding size is fixed to 64, and each vector is randomly initialized using a Xavier initializer [97]. Then, we train KGUF on 20 different combinations of hyperparameters explored with a Bayesian optimization algorithm and apply early-stopping with a 5-epoch patience. We validate the our model on nDCG@10, since this metric enhances the capability of predicting top-ranking items. Moreover, the batch size is adapted proportionally to the dataset size: 64, 256, and 2048, respectively, for *Facebook Books*, *Yahoo! Movies*, and *Movielens 1M*.

4.6 Results and discussion

This section examines the performance of KGUF when compared to state-of-the-art baselines. Additionally, we present hyperparameter and ablation studies conducted to evaluate the effectiveness of the proposed approach. Specifically, this section delves into experimental analyses aimed at answering the following research questions:

- **RQ1 Overall performance:** How does KGUF performance compare to that of SOTA recommenders? Additionally, what is the impact of the proposed knowledge-aware aggregation schema on the overall performance?
- **RQ2 Decision Tree Construction:** To what extent does the construction of trees impact the performance of the proposed framework?

4.6.1 RQ1: Overall performance

Firstly, we compare the recommendation accuracy of all the methods on three different datasets and report the results in Table 4.2. We denote the highest-performing result by boldfacing it and underscore the second-best result.

We observe that KGUF shows performance comparable or superior to all baselines over the three datasets. This improvement can be attributed to two key factors. Firstly, by employing user-based filtering, KGUF selectively learns the most pertinent characteristics for describing items in the catalog. In other words, it effectively filters knowledge information that proves genuinely valuable for the recommendation task.

Secondly, the linear aggregation schema adeptly combines collaborative and knowledge signals, thus improving both item and user representations. Another observation is that GNN-based techniques (LightGCN, DGCF, KGIN, KGTORe) consistently outperform non-graph-based ones, underscoring the efficacy of these architectures in leveraging long-term user-item relationships. The superior performance

Dataset	Model	nDCG	HR	Recall
Facebook Books	Random	0.0074	0.0256	0.0110
	MostPop	0.0787	0.2535	0.1117
	BPR-MF	0.0945	0.2973	0.1358
	Item-kNN	0.1039	0.3353	0.1556
	KGFlex	0.0712	0.2235	0.1001
	kaHFM	0.0843	0.2776	0.1221
	LightGCN	0.0969	0.3170	0.1435
	DGCF	0.1001	0.3192	0.1444
	KGIN	0.1023	0.3112	0.1448
	KGTORe	0.1188	0.3601	0.1698
	KGUF	<u>0.1156</u>	<u>0.3594</u>	<u>0.1697</u>
Yahoo! Movies	Random	0.0065	0.0342	0.0090
	MostPop	0.1185	0.3238	0.1501
	BPR-MF	0.2303	0.5228	0.2843
	Item-kNN	0.2258	0.5033	0.2658
	KGFlex	0.1758	0.4360	0.2022
	kaHFM	0.2085	0.5064	0.2621
	LightGCN	0.2230	0.5085	0.2712
	DGCF	0.2356	0.5308	0.2848
	KGIN	0.2368	0.5500	<u>0.2971</u>
	KGTORe	<u>0.2471</u>	<u>0.5566</u>	0.2944
	KGUF	0.2561	0.5685	0.3058
MovieLens 1M	Random	0.0096	0.0836	0.0036
	MostPop	0.1730	0.6537	0.0773
	BPR-MF	0.3100	0.8811	0.1627
	Item-kNN	0.2932	0.8504	0.1490
	KGFlex	0.0111	0.1058	0.0053
	kaHFM	0.2992	0.8654	0.1526
	LightGCN	0.3081	0.8813	0.1614
	DGCF	0.3124	0.8819	0.1637
	KGIN	0.3082	0.8854	0.1638
	KGTORe	<u>0.3212</u>	<u>0.8957</u>	<u>0.1710</u>
	KGUF	0.3277	0.8998	0.1755

Table 4.2. Accuracy performance comparison of KGUF with the selected baselines. All the models are grouped into their respective categories.

of knowledge-aware GNN-based methods is particularly noteworthy, highlighting how graph models effectively incorporate and exploit knowledge graphs.

Finally, we observe that not all knowledge-aware models (KGFlex, kaHFM) outper-

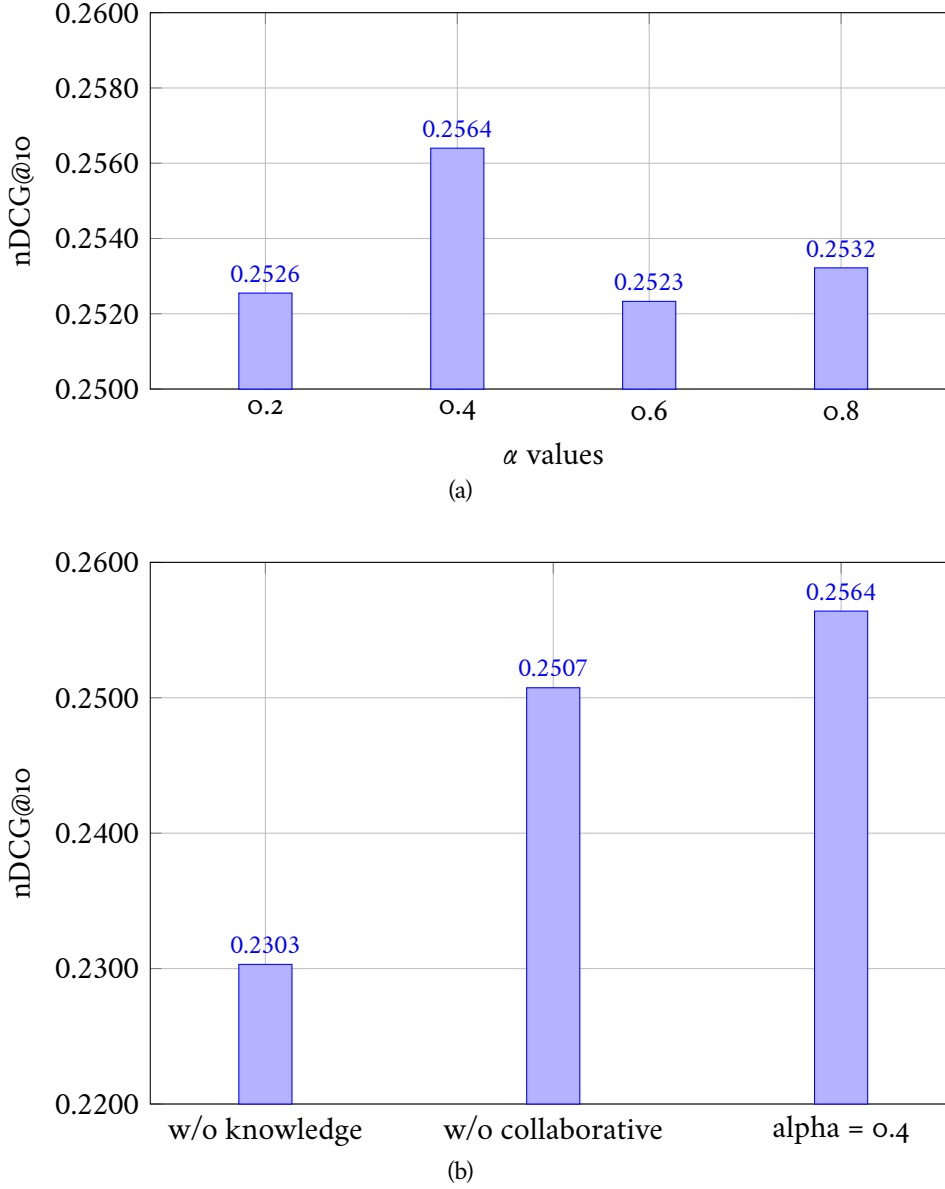


Figure 4.2. Performance of KGUF regarding nDCG@10 when varying the knowledge contribution. Figure 4.2a reports the impact of the parameter α , which weights the knowledge contribution. Figure 4.2b shows the impact of considering the knowledge or the collaborative signals only.

form pure collaborative approaches, emphasizing how merely utilizing knowledge-aware information may not necessarily lead to improved performance.

To delve deeper into assessing the efficacy of the proposed aggregation schema, we analyze how varying the weight assigned to the knowledge-aware component within the KGUF impacts the outcome on the dataset *Yahoo! Movies*. The results in Figure 4.2a involve varying the weight of α in Eq. 4.6 across a spectrum from 0.2 to 0.8, i.e.,

increasing the contribution of knowledge on the collaborative information. The results show that searching for the best balance between content and collaborative contributions is necessary, which in this case is 40% content and 60% collaborative. However, it is even more interesting to question the collaborative and the knowledge signals' separate impact when switched on/off. Performance in Figure 4.2b makes it clear that removing the knowledge information affects the model capabilities harder than removing the collaborative one. This shows how KGUF benefits from semantic information and how combining the two signals gives superior performance to considering only one.

4.6.2 RQ2: Decision Tree Construction

In this study, we explore how the tree construction strategies can impact the overall performance of KGUF, assessing how the negative sampling augmentation for tree building and the maximum depth of the trees contribute to the overall recommendation quality.

Effect of negative sampling

We investigate how KGUF behaves when we enhance the negative sampling employed in constructing decision trees by introducing a number of negative items equal to η times the number of positive items, such that $|\tilde{\mathcal{I}}_u^-| = \eta |\mathcal{I}_u^+|$. The results depicted in Figure 4.3a indicate an improvement in accuracy performance with respect to the target metric as η increases. We attribute this trend to the higher values of η mitigating the influence of randomly selecting negative items, consequently reducing noise impact. However, it is important to note that increasing the value of η does not guarantee a constant improvement in performance. Therefore, there is no ideal value for η , and this parameter must be explored individually for each dataset.

Effect of tree depth

The maximum depth of a decision tree, representing the length of the longest path from the tree root to a leaf, directly influences the selection of features. We investigate how imposing constraints on the tree depth may impact the recommendation performance. To address this, we trained the model with diverse maximum tree depths—1, 2, 5, 10, 15, 20, and without a limit (∞). We evaluated accuracy performance repeating the experiment five times and presenting the average, minimum, and maximum values of the target metric in Figure 4.3b. While unlimited depth yields satisfactory results, it is noteworthy that exploring the entire tree incurs higher computational and memory costs. Excessive features can lead to an overly generalized model, and comparable or even superior performance can be achieved without traversing the entire tree, as it is evident at depth 10, where we witness the least variance around the average performance.

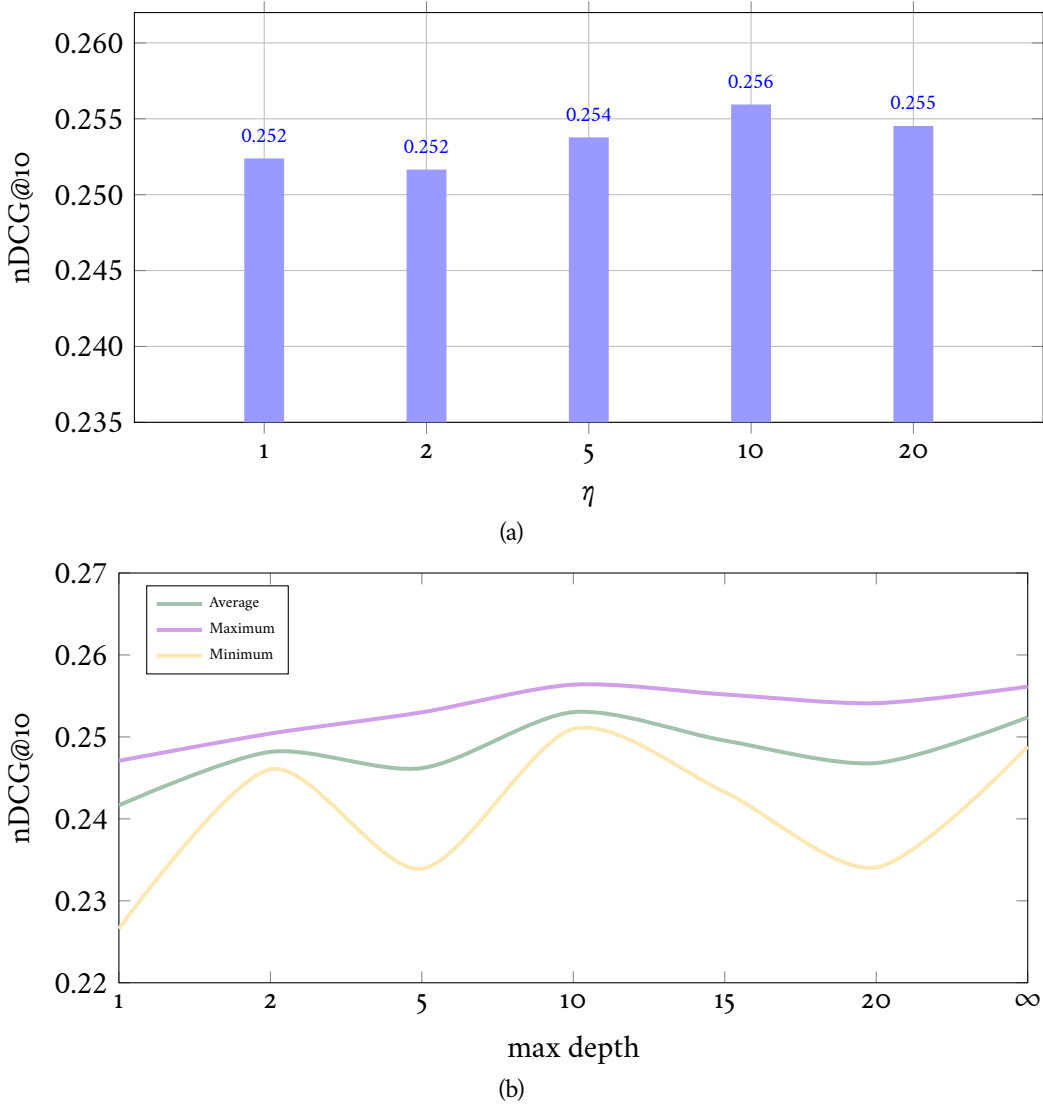


Figure 4.3. Impact of negative sampling and tree depth on KGUF performance in terms of nDCG@10. Figure 4.3a reports the nDCG@10 achieved for different values of η , i.e., the ratio between sampled negative and positive items, while Figure 4.3b shows the nDCG@10 obtained with varying maximum decision-tree depths (max depth).

4.7 Conclusion and future work

This chapter addresses the need for simplification in existing KGCF recommender systems, proposing a novel algorithm called **KGUF** (Knowledge Graph User-based Filtering). We advocate for a streamlined approach, asserting that simpler strategies have the potential to yield comparable or even superior accuracy.

In KGUF, we focus on efficiently utilizing KG side information by learning user and item embeddings through linear propagation on the user-item interaction graph. Unlike other approaches, our solution integrates latent representations for semantic

features directly into item representations. Additionally, we exploit the simple but effective decision tree mechanism to select meaningful semantic features from KGs and filter out noise triples.

We conduct extensive experiments on well-known recommendation datasets, evaluating the effectiveness of our framework. Furthermore, we explore improving the decision tree capability in selecting relevant semantic features. We investigate the impact of negative sampling and tree depth constraints and evaluate their impact on the overall recommendation performance.

Part III

Semantic Completion of Knowledge Graphs

Chapter 5

Completing Incomplete Knowledge Graphs via Inductive Link Prediction

Knowledge-aware recommender systems leverage knowledge graphs (KGs) to enrich sparse user-item signals with semantic context. In practice, however, real-world KGs are often incomplete, which limits the availability of information needed to better refine user and item profiles. This chapter addresses KG incompleteness through inductive link prediction.

Inductive link prediction is emerging as a key paradigm for real-world knowledge graphs (KGs), where new entities frequently appear and models must generalize to them without retraining. Predicting links in a KG faces the challenge of guessing previously unseen entities by leveraging generalizable node features such as subgraph structure, type annotations, and ontological constraints. However, explicit type information is often lacking or incomplete. Even when available, type information in most KGs is often coarse-grained, sparse, and prone to errors due to human annotation. In this work, we explore the potential of pre-trained language models (PLMs) to enrich node representations with *implicit* type signals. We introduce *TyleR*, a Type-less yet type-awaRe approach for subgraph-based inductive link prediction that leverages PLMs for semantic enrichment. Experiments on standard benchmarks demonstrate that TyleR outperforms state-of-the-art baselines in scenarios with scarce type annotations and sparse graph connectivity. Although evaluated on KG benchmarks, the method targets an upstream completion step that can support knowledge-aware RS under sparsity.

To ensure reproducibility, we share our code at <https://github.com/sisinflab/tyler>.¹

1. This chapter is based on the work published in the *Proceedings of the 2025 conference on Empirical Methods in Natural Language Processing* (EMNLP 2025), “Type-Less yet Type-Aware Inductive Link Prediction with Pretrained Language Models”.

5.1 Introduction

Knowledge graphs (KGs) represent complex relationships between entities in a structured, graph-based format [122]. Their ability to encode semantic information and support reasoning makes them valuable in a variety of applications, such as natural language processing [185], recommendation systems [256], and biomedical research [95]. However, KGs are notoriously incomplete: many valid relations are absent, reducing their effectiveness in downstream tasks [202].

Link prediction aims to infer these missing relationships by analyzing the existing graph’s structure and patterns. Traditional link prediction methods aim to predict links among entities observed during training. Although effective in static settings, they are limited in dynamic environments where new entities are incrementally introduced. Inductive link prediction (ILP) addresses this challenge by aiming to generalize to previously unseen entities, leveraging transferable features such as structural information and type information.

Prior work has demonstrated that incorporating entity type information can enhance generalization capabilities of ILP models. For instance, Zhou et al. [314] explicitly integrate type annotations and ontological constraints into the learning process. Yet, these methods face a critical bottleneck: available explicit type information in real-world KGs is often coarse-grained, incomplete, or even erroneous. This limitation is particularly acute when facing structural sparsity. Consider, for instance, the triple $\langle \text{Lionel Messi}, \text{playedFor}, \text{Barcelona FC} \rangle$. A model might assign similar plausibility to $\langle \text{Cristiano Ronaldo}, \text{playedFor}, \text{Barcelona FC} \rangle$ if both subject entities (i.e., Lionel Messi and Cristiano Ronaldo) lack distinct neighborhood information and are categorized only under the broad type "Footballer". This highlights a fundamental inadequacy of type-informed ILP approaches when explicit type signals are weak and local graph structure is uninformative.

To address this gap, our idea is to leverage the rich semantic knowledge captured by pre-trained language models (PLMs). We hypothesize that the semantic understanding these models acquire during their extensive pre-training on vast textual corpora [111, 186] offers a pathway to a more fine-grained representation of entities. This "inner knowledge," encoded within the PLM’s parameters, offers a dense representation of diverse semantic facets. For example, prompting a PLM like BERT [73] with "Paris is located in __," generates a hidden representation for the missing token that (ideally) enables it to correctly predict "France," reflecting the model’s "understanding" of Paris’s geographical location. We aim to utilize the implicit semantic insights of PLMs to derive fine-grained entity representations, overcoming limitations in explicit type information. We start from these two observations: (i) an entity can be described by a set of assertions defining its properties; (ii) the same assertions, when used as prompts for a PLM, can elicit dense, multifaceted representations that implicitly capture a "type-aware" understanding of the entity. This potential led us to ask: *Can PLM-derived entity representations compensate for*

structural and type sparsity in inductive knowledge graph completion? To investigate this question, we introduce **TyleR–Type-less yet type-awaRe**—a novel inductive link prediction framework that leverages PLMs to embed implicit type-aware signals within node representations, thus eliminating reliance on explicit type annotations. Our contributions are:

1. We introduce a novel methodology for harnessing PLMs to derive and embed implicit type semantics within an ILP model, thereby enabling nuanced entity representations without relying on explicit type data.
2. We demonstrate TyleR’s effectiveness on multiple benchmark datasets, showing its capability to perform competitively, especially in settings with limited or coarse-grained type information and sparse graph structures.
3. We conduct an empirical analysis investigating the interplay between PLM-derived semantic features and varying levels of type and structural sparsity, thereby characterizing the resilience of our approach.

The remainder of this chapter is organized as follows: section 5.2 introduces the idea behind subgraph-based relational inference; section 5.3 details the methodology; section 5.4 describes the experimental setup and evaluation; section 5.5 presents the results; section 5.6 reviews related work; and section 5.7 concludes with future directions.

5.2 Background and Motivation

Inductive link prediction aims to predict the likelihood of triples (h, r, t) , where h and t are unseen entities. In practice, this is done by means of a scoring function $f(h, r, t)$. At training time, f is optimized on the triples in a training graph $\mathcal{G}_{train}$. At test time, the same scoring function is used to predict the plausibility of triples (h', r, t') belonging to a test graph $\mathcal{G}_{test}$, based on the triples in an inference graph $\mathcal{G}_{inf}$. Unlike traditional embedding-based approaches, subgraph-based relation prediction methods such as GraIL [235] can be viewed as learning logical rules that capture entity-independent relational semantics. For example, one can derive the simple rule:

$$spouse_of(X, Y) \wedge lives_in(Y, Z) \rightarrow lives_in(X, Z).$$

As demonstrated by Zhou et al. [314], the reasoning capabilities of GraIL can be enhanced by incorporating explicit *type information* about entities. This additional semantic context enables the model to induce more precise and type-aware rules:

$$\begin{aligned} &Employee(X) \wedge Department(Y) \wedge Office(Z) \wedge \\ &\wedge part_of(X, Y) \wedge located_in(Y, Z) \rightarrow works_in(X, Z). \end{aligned}$$

Type-constrained rules enhance both accuracy and interpretability in relational inference by reducing spurious predictions and enforcing semantic validity. However,

explicit type information is often incomplete or missing in real-world knowledge graphs. To address this, we propose learning a function τ_{PLM} , parameterized by a pre-trained language model, that maps entities to implicit type representations capturing their latent semantics. These PLM-derived embeddings enable type-aware reasoning without explicit type labels and can be integrated into the logical rule induction process. For example, a type-aware rule may take the form:

$$\tau_{PLM}(X) \wedge \tau_{PLM}(Y) \wedge \tau_{PLM}(Z) \wedge \text{part_of}(X, Y) \wedge \\ \wedge \text{located_in}(Y, Z) \rightarrow \text{works_in}(X, Z),$$

with $\tau_{PLM}(X)$, $\tau_{PLM}(Y)$, and $\tau_{PLM}(Z)$ such that

$$\begin{aligned} \tau_{PLM}(X) &\approx \text{Employee}(X), \\ \tau_{PLM}(Y) &\approx \text{Department}(Y), \\ \tau_{PLM}(Z) &\approx \text{Office}(Z), \end{aligned}$$

where $\tau_{PLM}(\cdot)$ for X is an approximation of the logical statement $\text{Employee}(\cdot)$ while, for Y and Z , $\tau_{PLM}(\cdot)$ is an approximation of their types, Department and Office , respectively (more details in Section 5.3). This guides the rule induction process towards more meaningful and generalizable patterns, allowing us to infuse latent type semantics into subgraph-based link prediction models, even when explicit type information is absent.

5.3 Methodology

In this section, we introduce **TyleR** (**T**ype-**l**ess yet type-**a**wa**R**e inductive link prediction with pre-trained language models). Building on Graph Inductive Learning [235], which infers relations from local subgraph patterns, TyleR leverages PLM-derived semantics to enrich node representations. However, integrating PLMs into full-graph models is computationally expensive due to high-dimensional embeddings and large graph size. TyleR adopts a subgraph-reasoning approach, restricting triple scoring to compact and informative subgraphs, making PLM integration tractable.

As illustrated in Figure 5.1, TyleR’s pipeline consists of four stages: ① extracting an enclosing subgraph, ② structurally labeling nodes (following GraIL [235]), ③ enriching nodes with PLM-based semantic embeddings, and ④ feeding the enhanced subgraph into a GNN architecture from Zhou et al. [314]. The following sections provide further details on each step.

Subgraph Extraction ①. Given a target triple (u, r_t, v) , we define $\mathcal{N}_k(u)$ and $\mathcal{N}_k(v)$ as the sets of k -hop neighboring nodes of u and v , respectively. We also define a specific distance metric $d(i, u)$ as the shortest path from a node i to u that does not pass through v , and $d(i, v)$ is similarly the shortest path distance from i to v that does not pass through u . The *enclosing subgraph* of triple (u, r_t, v) is computed by (i) forming an initial set of candidate nodes by taking the intersection $\mathcal{N}_k(u) \cap \mathcal{N}_k(v)$

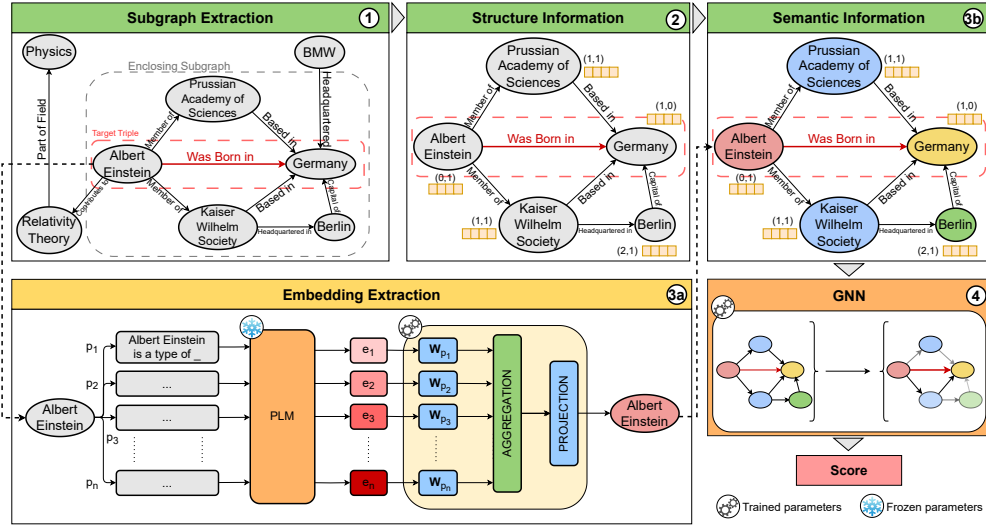


Figure 5.1. Overview of Tyler. The process begins with ① extracting the enclosing subgraph and ② applying a node labeling strategy. Multi-faceted, semantic representations are then derived using a pre-trained language model ③a, ③b. Finally, a graph neural network ④ integrates structural and semantic information to obtain the final prediction..

and (ii) pruning nodes that are either isolated (i.e., have no edges connecting it to other nodes within the subgraph after this pruning step) or for which $d(i, u) > k$ or $d(i, v) > k$. The remaining nodes and their edges form the enclosing subgraph.

Subgraph Labeling ②. Each node i in the extracted subgraph is labeled with a pair of shortest path distances $(d(i, u), d(i, v))$ to the target nodes u and v , respectively, within the subgraph. This pair captures the relative *position* of node i with respect to the target nodes u and v . The final positional embedding h_i^{pos} is:

$$h_i^{\text{pos}} = \text{one-hot}(d(i, u)) \oplus \text{one-hot}(d(i, v)), \quad (5.1)$$

where $\oplus$ denotes the concatenation operator and $\text{one-hot}(\cdot)$ is the one-hot encoding function. All nodes in the enclosing subgraph are within k hops of u or v , so $h_i^{\text{pos}} \in \mathbb{R}^{2k+2}$.

Semantic Enrichment ③a, ③b. Semantic enrichment leverages a pre-trained language model (PLM), supporting either masked token prediction or next-token generation. A straightforward approach for encoding entity type semantics involves prompting the PLM with an explicit query to elicit the most plausible type for a given entity. For masked language models (MLMs) (e.g., RoBERTa), this operation results in a prompt such as “The type of Paris is [MASK]”, with the type semantics encoded in the last hidden layer representation of the [MASK] token; for causal language models (CLMs) (e.g., Llama), this representation corresponds to the last hidden representation of the final sequence token. However, relying solely on representations derived

from such direct type queries can be suboptimal. Prior research has shown that transformer-based representations tend to be highly anisotropic, often concentrated in narrow cones [84], which can limit their discriminative utility. To address this, we propose to refine type semantics through multiple prompts, designed to extract different semantic aspects. Given an entity i with textual label l_i , we define a set of **assertion prompts** $P = \{p_1, p_2, \dots, p_n\}$, where each p_k targets a semantic facet of the entity (e.g., type, location, membership). Each prompt $p_k(l_i)$ is processed by the PLM (5) to yield a latent representation:

$$z_{p_k,i} = \text{Extract}(\text{PLM}(p_k(l_i))). \quad (5.2)$$

Here, $\text{PLM}(\cdot)$ denotes the forward pass of the language model given an input prompt, and $\text{Extract}(\cdot)$ selects the relevant hidden state (i.e., the [MASK] token's final hidden layer for MLMs or the last token's representation for CLMs). These representations $z_{p_k,i}$ are refined and projected into a unified space using an assertion-specific projection block:

$$z_{p_k,i}^h = W_{p_k} \text{LN}(z_{p_k,i}) + b_{p_k}, \quad (5.3)$$

where W_{p_k} and b_{p_k} are specific learnable parameters for each assertion prompt p_k , and LN denotes layer normalization [21]. We aggregate ($\text{AGG}_p(\cdot)$) the prompt representations with different strategies such as *sum*, *mean* or *concatenation*:

$$z_i^{\text{agg}} = \text{AGG}_p(\{z_{p_k,i}^h\}_{k=1}^n). \quad (5.4)$$

The semantic embedding h_i^{sem} is obtained as:

$$h_i^{\text{sem}} \equiv \tau_{\text{PLM}}(i) = \sigma(W_o \text{ReLU}(z_i^{\text{agg}})). \quad (5.5)$$

where $\tau_{\text{PLM}}(i)$ is a function capturing the semantics of i by aggregating multiple prompt-based representations via a PLM (as introduced in Section 5.2), and $\sigma(\cdot)$ is the sigmoid function. Given a node i , we then construct the embedding h_i^0 as (5):

$$h_i^0 = [h_i^{\text{pos}} \oplus h_i^{\text{sem}}]. \quad (5.6)$$

GNN Scoring ④. As suggested by Zhou et al. [314], our base GNN follows the R-GCN [213] architecture. At layer l , the embedding for a node i is computed as:

$$h_i^{(l)} = \text{ReLU}(W_0^{(l)} h_i^{(l-1)} + a_i^{(l)}), \quad (5.7)$$

where $W_0^{(l)}$ is a self-loop learnable matrix and $a_i^{(l)}$ is the AGGREGATE function, based on edge attention [235] and entity-relation composition [244]:

$$a_i^{(l)} = \sum_{r \in R} \sum_{j \in \mathcal{N}^r(i)} \alpha_{rr_i j i}^{(l)} W_r^{(l)} (h_j^{(l-1)} - e_r^{(l-1)}), \quad (5.8)$$

where $W_r^{(l)}$ is a relation-specific transformation matrix at layer l , $\mathcal{N}^r(i)$ is the set of outgoing neighboring nodes of node i under relation r . We adopt basis sharing [213]

as regularization for the $W_r^{(l)}$ transformation matrices, whereas $e_r^{(l)}$ is the relation embedding at layer l :

$$e_r^{(l)} = W_{rel}^{(l)} e_r^{(l-1)}. \quad (5.9)$$

The edge attention weight $\alpha_{rr,ji}^{(l)}$ quantifies the importance of an edge (j, r, i) when inferring relation r_t at layer l .

$$\alpha_{rr,ji}^{(l)} = \sigma(W_a^{(l)} s_{rr,ji}^{(l)} + b_a^{(l)}), \quad (5.10)$$

$$s_{rr,ji}^{(l)} = \text{ReLU}(W_s^{(l)} [h_j^{(l-1)} \oplus h_i^{(l-1)} \oplus e_r^{(l-1)} \oplus e_{r_t}^{(l-1)}] + b_s^{(l)}). \quad (5.11)$$

To obtain the final representation of a node, Teru et al. [235] suggest adopting JK-Connections [286], i.e., by concatenating all the intermediate-layer representations. After the aggregation, the final score is computed as

$$f(u, r_t, v) = W_f^\top \bigoplus_{l=1}^L \left[h_{\mathcal{G}}^{(l)}(u, r_t, v) \oplus h_u^{(l)} \oplus h_v^{(l)} \oplus e_{r_t}^L \right]. \quad (5.12)$$

where $h_{\mathcal{G}}^{(l)}(u, r_t, v)$ is the subgraph representation, obtained via average pooling over all node representations at level l in the subgraph.

Loss Function. We adopt a margin-based pairwise loss function, which aims at maximizing the score on positive triples and minimizing the score on randomly sampled negative triples:

$$\mathcal{L} = \sum_{(u, r_t, v) \in G} \max(0, f_e(u', r_t, v') - f_e(u, r_t, v) + \gamma), \quad (5.13)$$

where γ is a margin hyperparameter, (u, r_t, v) is a positive triple and (u', r_t, v') is a negative triple.

5.4 Experimental Setup

In this section we detail our experimental setup, including datasets, baselines, training and evaluation details. Experiments were conducted with Python 3.8.19 and PyTorch 2.3.0, using an NVIDIA Ampere A100 GPU (64GB VRAM) and CUDA 12.1.

5.4.1 Datasets

We conduct experiments on YAGO21K-610 [314] and three FB15K-237 (FB237 in short) variants (v1–v3) from Teru et al. [235]. Dataset statistics are summarized in Table Table 5.1. Dataset density, defined as $2|T|/|E|$ [187], is the lowest for YAGO21K-610 train (3.67) and increases across FB237 variants (i.e., from 5.33 to 9.80). This

Dataset	Split	Entities	Relations	Triples	Types	Meta Rel.	Onto. Triples	Type Links	Text Labels
fb237_v1	train	1594	180	4245	458	29	680	2163	1516
	valid	567	103	489	124	15	86	756	539
	test	550	102	492	113	13	85	764	517
fb237_v1_ind	train	1093	142	1993	458	29	680	1525	1041
	valid	287	66	206	124	15	86	406	275
	test	301	68	205	113	13	85	434	289
fb237_v2	train	2608	200	9739	575	33	865	3586	2489
	valid	1139	143	1166	160	15	109	1511	1083
	test	1142	140	1180	153	16	108	1515	1094
fb237_v2_ind	train	1660	172	4145	575	33	865	2257	1561
	valid	548	92	469	160	15	109	757	516
	test	562	107	478	153	16	108	745	524
fb237_v3	train	3668	215	17986	732	31	1060	5114	3484
	valid	1882	183	2194	196	17	133	2575	1787
	test	1871	179	2214	192	16	133	2520	1773
fb237_v3_ind	train	2501	183	7406	732	31	1060	3426	2379
	valid	973	120	866	196	17	133	1275	920
	test	981	128	865	192	16	133	1290	924
YAGO21K-610	train	16357	30	30000	610	24	1983	4861	16357
	valid	4388	21	3000	166	14	248	1783	4388
	test	3938	25	6970	159	13	248	1898	3938

Table 5.1. Statistics of the datasets used in our experiments. The YAGO21K-610 [314] dataset includes ontology triples and entity-type links, while the FB237 dataset variants [235] are further processed to extract ontology triples, type links and textual labels.

pattern also holds for the inference graphs (density ranging from 3.54 for YAGO21K-610 to 5.92 for FB237-V3), allowing us to analyze the impact of type information under varying graph sparsity. For YAGO21K-610, we use the original splits with the provided ontology graph and type links; test entities are unseen during training, while relations are shared. Each FB237 variant contains disjoint train and inductive test graphs with distinct entities but shared relations. For each FB237 variant, we train on its designated training set and evaluate using its corresponding "ind" (inductive) set as the inference graph, with testing performed on its test set. For YAGO21K-610, when evaluating a specific target triple, the inference graph includes all other test triples (excluding the target itself), following Zhou et al. [314].

Since FB237 lacks concept annotations, we build ontology graphs and type links for all variants. To construct the ontology graph we mapped all the freebase entities appearing in the dataset to their Wikidata identifier, using the publicly available Freebase-Wikidata mappings ². Using the public Wikidata API ³, we then retrieved for every mapped entity its respective textual label and the values associated with its "instance of" property, which indicates the type(s) an entity is associated to. With the set of relevant concepts established, we constructed the schema-level ontology. For each concept identified in the previous step, its full set of concepts was fetched from Wikidata. A schema-level triple $\text{Concept}_1 \text{ PropertyLabel } \text{Concept}_2$ was generated

2. <https://developers.google.com/freebase>

3. <https://www.wikidata.org/w/api.php>

ID	Aspect	Template
p_1	type	Paris is a type of __
p_2	geographic	Paris is located in __
p_3	membership	Paris is member of __
p_4	equivalence	Paris is equivalent to __
p_5	difference	Paris is different from __
p_6	similarity	Paris is similar to __

Table 5.2. Assertion prompts (p_1 - p_6) used in the semantic enrichment step (section 5.3). These templates, with a placeholder for the entity, are fed to the Pre-trained Language Model to elicit representations capturing different semantic aspects (type, geographic context, membership, equivalence, difference, similarity) of the entity.

and added to our ontology graph if, and only if, the target value of a Concept ($Concept_2$) was itself one of the recognized concepts. In the entity triples, the entities in the test set do not appear in the train set and valid set, while the relations in both the test set and valid set are included in the train set. We train on the train graph and test on the test graph. In addition, to achieve ontology training, we randomly divide the ontology triples into a train set, a valid set, and a test set using hold-out splitting in the ratio of 80%, 10%, 10%, respectively.

5.4.2 Metrics

We evaluate models using Mean Reciprocal Rank (MRR) and Hits@ K for $K \in \{1, 10\}$, averaging over 5 evaluation runs. Following standard protocol [235], each positive test triple is ranked against 50 negative triples generated by randomly corrupting either its head or tail entity.

Tie resolution markedly affects these metrics. While methods like random tie-breaking [202]—which randomly assign ranks among tied entities—are prevalent, they can lead to an overestimation of true model performance. This issue is particularly evident in sparse settings where limited structural or type information leads to frequent ties, an issue amplified by the candidate pool of 50. To address these concerns and provide a more stringent and reliable evaluation, we adopt a **strict tie-breaking strategy**. This approach assigns the positive triple the highest (i.e., worst-case/pessimistic) rank when its score is identical to one or more negative triples.

5.4.3 Models

To isolate the contribution of our semantic-enrichment module, we focus the comparison on methods that share a similar subgraph-reasoning backbone as Tyler. We evaluate TyleR against GraIL [235], a type-agnostic baseline that relies solely on subgraph structure, and the ontology-enhanced method of Zhou et al. [314], which

Inductive LP Model	FB237-V1			FB237-V2			FB237-V3			YAGO21K-610		
	MRR	Hits@1	Hits@10	MRR	Hits@1	Hits@10	MRR	Hits@1	Hits@10	MRR	Hits@1	Hits@10
GraIL [235]	.456	34.97	64.44	.618	50.46	82.70	.609	<u>49.94</u>	82.26	<u>.661</u>	62.76	68.68
Zhou et al. [314]	.398	27.85	64.55	.576	44.69	82.45	.554	41.85	81.42	.673	<u>60.36</u>	<u>76.56</u>
TyleR-RoBERTa-L (2025)	.470	35.66	<u>69.95</u>	.602	47.51	83.28	.630	50.60	86.72	.660	58.26	79.68
TyleR-Llama3-8B (2025)	.481	36.88	70.63	<u>.610</u>	<u>49.37</u>	82.01	<u>.620</u>	<u>49.94</u>	<u>84.46</u>	.651	59.70	69.30

Table 5.3. Link Prediction (LP) evaluation on multiple FB237 variants and YAGO21K-610. Best and second-best scores are in **bold** and underlined, respectively. Evaluation uses the strictest tie-breaking policy (section 5.4.2), assigning the highest (worst) possible rank to the positive triple in case of ties.

explicitly incorporates type information via learnable embeddings and ontological constraints, even though its effectiveness is tied to the availability and quality of type annotations and ontology triples. We chose these baselines to enable a focused and meaningful comparison: (1) GraIL serves as the foundational type-agnostic framework upon which many subsequent methods build [47, 160]; (2) the method of Zhou et al. [314] exemplifies a type-enhanced approach, with few comparable models in literature. In contrast, Tyler is designed for scenarios where explicit type information is scarce as it is able to infer implicit type semantics from PLMs. Our semantic enrichment strategy, detailed in Section 5.3, is model-agnostic and compatible with any PLM supporting masked or causal language modeling. We use RoBERTa-Large [156] and Llama3-8B [77] without fine-tuning, aggregating representations from six manually crafted assertion prompts (Table 5.2). To further evaluate the robustness of TyleR with respect to prompt selection, we report the results on FB15K-237-V1 for both RoBERTa and Llama3-8B across varying numbers of templates (Table 5.6). The prompt aggregation function $AGG_p(\cdot)$ was set to SUM, as it yielded the most consistent results in our experiments (Table 5.4). Moreover, SUM offers better scalability by producing fixed-size outputs regardless of the number of prompts and may also help regularize prompt-specific noise.

5.4.4 Hyperparameter Details

Baselines are trained using the hyperparameter settings reported in their original papers. For our model, we adopt the configuration from Zhou et al. [314] to ensure fair comparison, tuning only the learning rate, which we set empirically to $1e-3$. All models are trained for 50 epochs with early stopping (patience of 100 iterations) and a batch size of 16. We adopt the Adam optimizer. For all models, the number of hops in the enclosing subgraph is 3. We set the semantic embedding dimension to 24, the layer-o embedding dimension to 32, and the margin γ in the loss function to 10.

5.5 Results

Experiments aim to answer three core questions:

PLM	Aggregation	MRR	Hits@1	Hits@10
TyleR-RoBERTa-L	TYPE-ONLY	.442	32.93	66.49
TyleR-RoBERTa-L	SUM	.470	35.66	69.95
TyleR-RoBERTa-L	MEAN	<u>.468</u>	<u>35.22</u>	<u>68.05</u>
TyleR-RoBERTa-L	CONCAT	.455	34.10	67.76
TyleR-Llama3-8B	TYPE-ONLY	<u>.477</u>	37.12	68.29
TyleR-Llama3-8B	SUM	.481	<u>36.88</u>	<u>70.63</u>
TyleR-Llama3-8B	MEAN	.465	35.51	68.73
TyleR-Llama3-8B	CONCAT	.474	35.95	71.12

Table 5.4. Ablation study on the FB237-V1 dataset, evaluating the impact of different Pre-trained Language Models and aggregation functions (Equation 5.4) for semantic embeddings within TyleR. 'TYPE-ONLY' uses only the representation from the p_1 prompt (table 5.2).

- RQ1.** Does explicit type information improve subgraph-based inductive link prediction?
- RQ2.** Can PLMs enhance node representations for subgraph-based inductive link prediction?
- RQ3.** Can PLMs mitigate type and structural sparsity challenges in inductive link prediction?

5.5.1 Type Information in Subgraph-Based Inductive Link Prediction (RQ1)

PLM	MRR	Hits@1	Hits@10
Llama3-8B (No GNN)	.264	13.21	55.06

Table 5.5. Performance of Llama3-8B when directly evaluating the likelihood of verbalized triples on the YAGO21K-610 dataset by scoring them using the (negated) model perplexity.

Table 5.3 presents link prediction results across various models and datasets, emphasizing the role of type information in inductive link prediction. GraIL, which operates without type information, performs competitively overall. It achieves strong results in both MRR and Hits@10, and obtains the highest Hits@1 on the sparse YAGO21K-610 dataset. This suggests its ability to rank correct entities precisely in low-density settings without relying on type cues. When explicit type information is incorporated, as in Zhou et al. [314], performance patterns shift: while Hits@10 often

PLM	Template Choice (p_i)	MRR	Hits@1	Hits@10
TyleR-RoBERTa-L	1 (TYPE-ONLY)	.442	32.93	66.49
TyleR-RoBERTa-L	1-2	.459	34.97	68.00
TyleR-RoBERTa-L	1-3	.451	34.87	65.32
TyleR-RoBERTa-L	1-4	.472	36.00	68.58
TyleR-RoBERTa-L	1-5	.468	35.32	69.46
TyleR-RoBERTa-L	1-6 (ALL)	<u>.470</u>	<u>35.66</u>	69.95
TyleR-Llama3-8B	1 (TYPE-ONLY)	<u>.477</u>	<u>37.12</u>	68.29
TyleR-Llama3-8B	1-2	.458	35.07	69.36
TyleR-Llama3-8B	1-3	.470	35.95	69.46
TyleR-Llama3-8B	1-4	.467	35.22	<u>70.19</u>
TyleR-Llama3-8B	1-5	.471	37.36	65.36
TyleR-Llama3-8B	1-6 (ALL)	.481	36.88	70.63

Table 5.6. Ablation study on the FB237-V1 dataset, evaluating the impact of chosen number of prompts (Equation 5.4) for semantic embeddings within TyleR. For all configurations, the aggregation function was set to SUM.

remain competitive—or even surpass GraIL on sparse datasets like YAGO21K-610—Hits@1 consistently decline. This indicates that explicit types may mitigate sparsity by providing useful semantic signals, but also introduce complexity that reduces precision in top-ranked predictions. As dataset density increases, the performance gap between GraIL and type-informed models narrows, and in some cases, GraIL even outperforms the latter. This trend suggests that **explicit type information becomes less helpful—and potentially detrimental—in denser graphs**, where structural cues are already sufficient. In contrast, implicit type information, as leveraged by TyleR, generally leads to more robust and consistent improvements. While not always achieving the best Hits@1, models using implicit types (TyleR variants) rank first or second across most datasets for both Hits@10 and Hits@1. These models show particular strength on sparse datasets, such as YAGO21K-610, where the gap in Hits@10 is most pronounced. This suggests that **implicit typing is more robust against topological variations in the data**, exhibiting a higher generalization potential.

Addressing RQ1, the benefit of explicit type information is dataset-dependent. It aids relational inference in sparse graphs lacking rich topology, but can add detrimental complexity in denser graphs. Implicit type signals, however, consistently enhance inference, mitigating structural sparsity.

5.5.2 Usefulness of PLM Representations for Implicit Type Signal (RQ2)

The results in Table 5.3 provide compelling evidence that PLMs can significantly enhance node representations in subgraph-based link prediction. However, **the impact of PLMs is not uniform across all datasets**. For example, on FB237-V1 and FB237-V3, RoBERTa-L and Llama3-8B models exhibit competitive performance, especially in terms of Hits@1 and MRR, suggesting that PLMs provide a strong inductive bias for relational reasoning. In contrast, models without PLMs, like GraIL, show lower performance on these datasets, particularly regarding the Hits@10 metric. This highlights the ability of PLMs to generalize and make more accurate predictions in larger, more complex graphs, where non-PLM models may struggle. Table 5.4 shows the impact of different aggregation strategies on the FB237-V1 dataset, with SUM showing the most consistent results. For example, Llama3-8B with SUM outperforms GraIL across all metrics. In addition, we compare the results of different aggregation strategies with the scenario where only the "type" prompt is considered (i.e., p_1 in 5.2), showing consistent improvements.

To comprehensively evaluate the utility of PLMs in link prediction, we further explore the potential of using LLMs directly. Specifically, we verbalize all evaluation triples (including both positive and negative candidates) in the form $label(h) \oplus label(r) \oplus label(t)$. We then employ Llama3-8B to score these verbalized triples based on their (negated) sentence perplexity and report the resulting rankings in Table 5.5. We exclude the FB15K-237 variants from this evaluation due to the lack of clear and consistent relational labels, which makes verbalization unreliable. The performance on YAGO21K-610, which is substantially lower than that achieved by GNN-based approaches, highlights the critical importance of incorporating neighborhood structural information for this task.

Regarding RQ2, PLMs effectively enhance node representations for subgraph-based inductive link prediction. By providing richer semantic features, models like RoBERTa-L and Llama3-8B improve relational inference. Aggregating diverse PLM-derived semantic embeddings (i.e., from different prompts) boosts representation expressiveness.

5.5.3 Effect on Type and Structural Sparsity (RQ3)

To investigate the effect of our approach on *type sparsity*, we categorize entities into four groups based on the number of explicit type annotations they possess. Group 0 consists of entities with *no explicit type*. Group 1 includes entities with *exactly one* explicit type. The remaining entities (those with *more than one* type) are split into two additional groups based on the median number of types in this subset: the lower 50% form the group labeled ">1 (1st)", and the upper 50% form ">1 (2nd)". For each test triple, we determine the group membership of the *known* entity (irrespective of the type information available for the candidate entities) and report the model performance in Figure 5.3. We report the average Hits@10 across the three

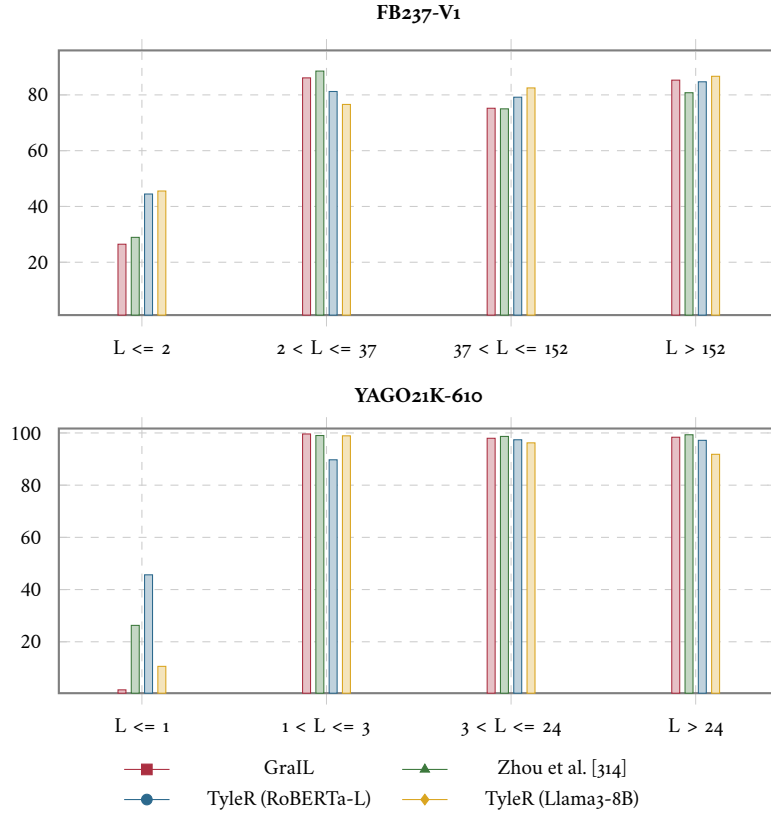


Figure 5.2. Link Prediction (Hits@10) evaluation under **varying structural sparsity conditions** (i.e., the number of edges L in the enclosing subgraph of the target triple, including the target triple) on FB237-V1 (top) and YAGO21K-610 (bottom). .

FB237 variants for each group. Group 0 represents the most type-sparse setting with entities lacking any explicit type; in this scenario, both variants of TyleR consistently outperform the typeless baseline GraIL across all dataset variants. This reinforces the intuition that **type signals derived from PLMs can enhance inference capabilities in sparse settings**. For instance, on FB237-V1, TyleR (RoBERTa-L) yields a 6.15% relative improvement over the non-PLM baselines, while on FB237-V2, it achieves an even greater gain of 10.75% over GraIL. Interestingly, in the case of entities with multiple types, TyleR continues to outperform the explicit-type-based method of Zhou et al. [314]. This suggests that while explicit type information is useful, its effectiveness may diminish when type annotations are noisy or overly numerous, highlighting the need for better strategies to aggregate multiple type signals.

We further analyze the role of *implicit type information* in addressing *structural sparsity*. For this analysis, we consider the two datasets with the lowest graph density: YAGO21K-610 and FB237-V1. Evaluation triples are grouped into four bins based on the number of edges in their enclosing subgraphs, using percentiles to capture

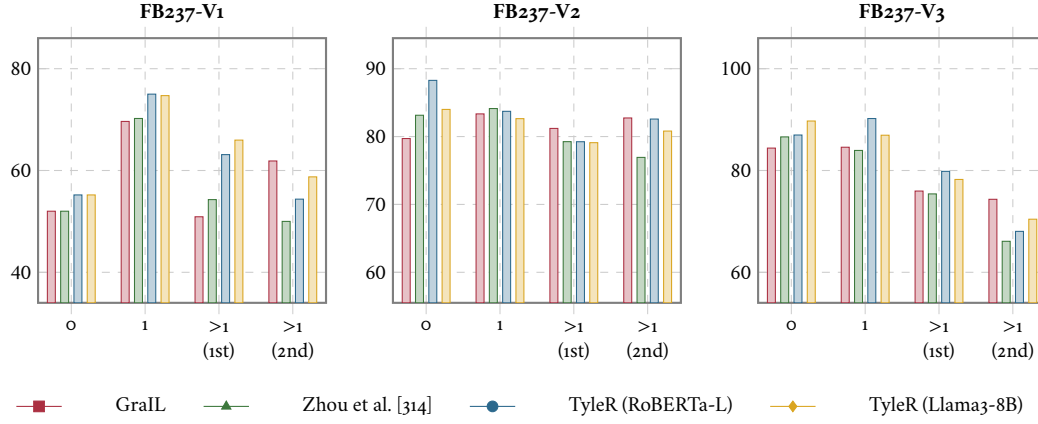


Figure 5.3. Hits@10 performance across four **type sparsity** groups for three FB237 variants, computed according to the number of explicit types linked to each entity (details in section 5.5.3). The groups, from left to right, represent scenarios with an increasing number of explicit types associated with the known entity .

varying levels of sparsity. Figure 5.2 presents Hits@10 across these structural sparsity conditions. The results indicate that PLM-based approaches, particularly those using RoBERTa-L, demonstrate strong performance in extremely sparse subgraphs. For example, RoBERTa-L performs best in scenarios with one or fewer edges (for YAGO21K-610) and more than 152 edges (for FB237-V1), demonstrating its robustness at both ends of the sparsity spectrum. However, in *moderate sparsity* settings (e.g., $2 < L \leq 37$ in FB237-V1), models such as GraIL and Zhou et al. [314] perform comparably or better, due to their reliance on structural patterns that are still informative in such contexts.

Answering RQ3, PLM-based approaches such as TyleR address both type and structural sparsity. They consistently outperform baselines in scenarios with minimal explicit type information or sparse subgraph structures by inferring meaningful semantics from PLMs. Although challenges persist with moderate sparsity and noisy types, PLMs show significant potential.

5.6 Related Work

Inductive Link Prediction.. Inductive Link Prediction (ILP) in Knowledge Graphs (KGs) aims to infer missing links that involve entities unseen during training, thereby enabling models to generalize to evolving KGs. Unlike traditional embedding-based models [35, 153, 267], inductive methods explicitly handle unseen entities. Early approaches relied on rule-based reasoning [172, 287], but graph neural networks (GNNs) soon became dominant [107], with GraIL [235] leveraging enclosing subgraph structures for relational inference. Extensions include CoMPILE [160], emphasizing relational directionality, and TACT [47], introducing relation-level reasoning. Zhou et al. [314] incorporate ontological data, but assume complete type information, an

assumption that seldom holds in real-world KGs.

Entity Representation with Language Models.. Pre-trained language models (PLMs) capture factual and relational knowledge from large corpora [36, 186], encoding rich entity semantics [318] and retrieving factual information via prompting [271]. This makes PLMs well-suited for link prediction because they can enrich entity representations. For example, KGBERT [290] verbalizes triples as text and fine-tunes BERT to classify their plausibility. Subsequent methods [66, 263, 308] integrate entity descriptions into KG completion to induce embeddings for new entities via PLMs. BERTRL [301] exemplifies this trend by injecting GNN-discovered reasoning paths into a BERT-based model. A promising direction involves integrating LLMs with subgraph-based methods to reduce model queries while preserving structural reasoning. Li et al. [146] propose CATS, a hybrid model that leverages latent type cues and neighbor facts to fine-tune an LLM for triple scoring, combining semantic understanding with explicit subgraph evidence. Unlike prior approaches that fine-tune PLMs, our method extracts semantic knowledge from a frozen PLM, and we investigate how effectively such pre-trained models enable a subgraph-reasoning module to capture the type semantics underlying each relation.

5.7 Conclusion

We present Tyler, a novel inductive link-prediction approach designed to handle incomplete or noisy type information. By leveraging pre-trained language models (PLMs), Tyler enriches node representations with implicit type signals, overcoming the limitations of methods reliant on explicit annotations. Experiments show that Tyler exhibits competitive performance, particularly when type data are sparse or unreliable. The results underscore the potential of PLMs for semantic enrichment, enabling robust link prediction without complete type supervision. Future work will examine domain-specific PLMs, more embedding-aggregation strategies, and broader applications to graph-based tasks.

Limitations

Our study employs a set of predefined prompts, which, while effective for the scope of our experiments, may not represent the most informative or optimal configurations. More sophisticated strategies for adaptive prompt selection or prompt tuning could potentially enhance model performance. Exploring these approaches is left as a direction for future research. Additionally, the hyperparameters for our models were selected empirically, based on extensive experimentation and informed judgment. While this approach yielded strong results, it may not guarantee optimal configurations. A more systematic or exhaustive hyperparameter search could lead to improved outcomes. Nonetheless, the computational cost and complexity associated with such procedures, particularly given the scale and resource demands of our

training setup, render them infeasible within the constraints of this study.

Part IV
Operating under Imposed Data
Sparsity

Chapter 6

Auditing Fairness and Accuracy Trade-Off under Imposed Data Sparsity

Data minimization, required by recent data privacy regulations, is crucial for user privacy; nevertheless, it further amplifies the inherent sparsity of RS datasets, leaving its impact on recommender systems largely unclear. The core problem lies in the fact that reducing or altering the training data of these systems can drastically affect their performance. While previous research has explored how data minimization affects recommendation accuracy, a critical gap remains: How does data minimization impact consumers' and providers' fairness? This chapter addresses this gap by systematically examining how data minimization influences multiple objectives in recommender systems, i.e., the trade-offs between accuracy, user fairness, and provider fairness. Our investigation includes (i) an analysis of how the data minimization strategies affect RS performance across these objectives, (ii) an assessment of data minimization techniques to determine those that can balance better the trade-off among the considered objectives, and (iii) an evaluation of the robustness of different recommendation models under diverse minimization strategies to identify those that best maintain performance. The findings reveal that data minimization can sometimes undermine provider fairness, albeit enhancing group-based consumer fairness to the detriment of accuracy. Additionally, different strategies can offer diverse trade-offs for the assessed objectives. The source code supporting this study is available at <https://github.com/salvatore-bufi/DataMinimizationFairness>.¹

1. This chapter is based on the work published in the *Proceedings of the 33rd ACM Conference on User Modeling, Adaptation and Personalization* (UMAP 2025), honored with the **Best Student Paper Award**: “Legal but Unfair: Auditing the Impact of Data Minimization on Fairness and Accuracy Trade-off in Recommender Systems”.

6.1 Introduction

Recommender Systems (RSs) have become a cornerstone for modern digital platforms, powering personalized content delivery across e-commerce, media streaming, and social networks [147, 155, 162, 210]. These systems rely on extensive user data to accurately infer preferences and produce highly tailored recommendations. However, growing concerns about user privacy, data security, and system scalability create a significant tension between the need to maintain high-quality recommendations and the increasingly important requirement to minimize data collection [79, 89]. This tension is further intensified by rigid regulatory frameworks like the GDPR [192], CCPA [40], CPRA [236], and PIPL [242], which command principles such as data minimization. The legal principle of data minimization underscores the need to collect only the strictly necessary data for a specific purpose. Given these regulations, many works have started to study the effects and risks of data minimization on machine learning-driven systems [58, 92, 226, 272]. In the realm of recommendation, Biega et al. [31] performed pioneering work to understand the performance-based effects of data minimization on system accuracy. Hence, other works have analyzed the influence of data minimization on privacy [80], algorithmic bias [58], and individual users' accuracy [222] performance of recommendations. Nonetheless, data minimization reshapes available user-item interactions by discarding or selectively retaining interactions, potentially altering the distribution of users and items. These alterations can amplify existing biases and undermine system fairness from consumer and producer perspectives. For this reason, it is pertinent to ask: *To what extent does data minimization risk to impact fair exposure of items? Can data minimization favor privileged user groups with higher-quality recommendations?* For example, a popular item may appear underrepresented in a minimized dataset if many interactions are removed, potentially pushing it into the long-tail category and reducing its visibility. Similarly, specific user groups may receive less accurate or less diverse recommendations, further widening existing disparities. While these risks are evident, a comprehensive understanding of the broader ramifications of data minimization on accuracy and fairness remains lacking, highlighting a critical gap in current research, given the increasing emphasis on fairness [12, 70, 188, 190, 264] in RSs.

This study bridges the gap by systematically investigating the interplay between data minimization, accuracy, and fairness in recommendation. To this end, we investigate how data minimization singularly impacts the accuracy and fairness of recommendations by identifying observable trade-offs. Then, we adopt a multi-objective evaluation approach to audit to what extent different data minimization strategies can preserve RS performance on multiple dimensions. Finally, we analyze how different recommendation models are robust to data minimization, maintaining a balance between accuracy and fairness. In summary, our key contributions are the following:

- We experimentally evaluate the impact of data minimization on recommendation accuracy, consumer fairness, and provider fairness across various data minimization strategies and real-world datasets. The evaluation demonstrates that the effectiveness of data minimization cannot be evaluated by accuracy alone but must consider how it affects different stakeholders in the system.
- We adopt a multi-objective evaluation approach to study which data minimization strategies can simultaneously preserve recommendation performance when considering multiple aspects. Our findings reveal that strategies that can shape better user profiles are crucial to avoid alterations in recommendation performance.
- We investigate the effects of data reduction on various recommendation algorithms, including neighborhood-based, matrix factorization, autoencoder-based, and graph-based methods. We understand how different model architectures react to data minimization when balancing the competing objectives of accuracy and fairness. We show that graph-based and factorization-based models can preserve recommendation performance under strong minimization conditions.

6.2 Related Work

User privacy [65, 71, 150] and data privacy [120, 298] have become a critical concern in recommender systems research, driven by regulations such as the GDPR, CCPA, CPRA, and PIPL. These legal frameworks mandate that only data necessary for a specific purpose be collected, creating tension between the extensive datasets typically required by recommender systems and the principle of data minimization.

Several works have begun to explore how to reconcile the need for personalized recommendations with the constraints imposed by these regulations. Biega et al. [31] introduced operational definitions of data minimization aimed at maintaining system quality while satisfying legal requirements. However, they also pointed out that minimization can worsen outcomes for specific user groups if not carefully managed. A family of works focuses on designing data minimization techniques to be integrated during model training to retain acceptable accuracy performance. Shanmugam et al. [216] proposed FIDO, a strategy that adaptively restricts data collection based on performance scaling rules, illustrating how data minimization can be integrated directly into training workflows. Sonboli et al. [222] have explored the trade-off between reducing data and maintaining accuracy at the user level by using active learning approaches that select only highly informative user-item interactions. While these methods can improve efficiency, they can also worsen outcomes for groups with sparse interaction histories. Other works [224, 246] study the effects of data reduction on environmental factors, showing that smaller datasets obtained through data reduction lead to reduced energy consumption but may compromise recommendation accuracy. Regarding the interplay between data minimization and fairness, Clavell et al. [58] highlight the complexities of simultaneously addressing privacy, performance, and fairness, emphasizing the need for more comprehensive frameworks to evaluate these trade-offs.

The existing works concentrate on a single objective—such as privacy, bias, or accuracy—rather than investigating how these factors interact in a unified manner. Our work extends the current state-of-the-art by systematically examining how data minimization strategies simultaneously affect both accuracy and fairness in recommender systems.

6.3 Methodology

6.3.1 Data minimization strategies

In our study, we evaluate several approaches to minimize user data. These strategies are designed to select subsets of user-item interactions that serve as input to the system. Inspired by existing methodologies, we focus on practical techniques that avoid the computational complexity of greedy algorithms, which, although theoretically promising, are often infeasible due to their runtime demands. In the following, we outline the data minimization strategies explored [31].

Full data. We begin by defining a baseline that utilizes the complete set of user interactions without any minimization. This serves as a reference point against which the effectiveness of all other strategies is measured. Formally, the system observes all interactions R_{ui} in the user history.

Random selection. This approach randomly selects n interactions from each user's history. Although randomness may introduce variability, the average profile characteristics of users are often preserved due to the sampling process.

Most recent. This strategy prioritizes the most recent n interactions of a user. It assumes that recent behaviors are more relevant to the recommendation task. However, the effectiveness of this method depends on the temporal distribution of data; long observation periods may lead to performance variation when compared to random selection.

Most and least favorite. These techniques select the n items with the highest or lowest scores for a users, respectively. While the "most favorite" approach emphasizes user preferences, the "least favorite" strategy explores how negative interactions influence recommendations.

Most frequently rated. For each user, this method selects n items that have been rated most frequently across the entire dataset. This global popularity measure identifies items that are statistically significant within the system.

Most characteristic. It selects items based on their proximity to the system-wide average item profile. Each item is represented as a binary vector indicating which users have interacted with it. The system's average item profile is computed by averaging these vectors. For each user, the n items closest to this average profile (measured by Euclidean distance) are selected. This method rewards items frequently rated by highly active users, capturing the overall trends of the system while avoiding niche or rarely rated items.

Highest variance. This strategy prioritizes items with the highest variability in their ratings across all users. Items with high variance are considered more informative as they capture diverse or polarizing opinions. For each user, the nn items with the highest rating variance are selected.

6.4 Experimental Setup

This research carries out a comprehensive investigation into the effects of data minimization on recommender systems, moving beyond a simple accuracy analysis to consider its broader implications for both users and item providers. In this section, we detail the experimental setup employed to address the following research questions:

- RQ1:** How does data minimization impact the accuracy, provider fairness, and consumer fairness of recommendations? What are the observable trade-offs at the strategy level?
- RQ2:** Which data minimization strategies most effectively preserve recommender system performance compared to training on the full dataset?
- RQ3:** How robust are different recommendation models to data minimization, particularly concerning maintaining a balance between accuracy and fairness?

6.4.1 Dataset

Our experiments were conducted using two datasets to ensure a comprehensive evaluation: *MovieLens1M* (ML1M) [113] and *Ambar* [99]. Setting a minimum interaction threshold is essential when studying data minimization since it ensures each user profile is represented well enough for reliable model training and evaluation [31]. For this reason, we preprocessed both datasets to guarantee sufficient representation for each user and item, aligning with the methodology of Biega et al. [31]. In ML1M, we selected users who have at least 45 ratings in their profile. Furthermore, we subsampled the dataset to 2500 users, covering 3605 items and totaling 522152 ratings. The resulting dataset has a density of 0.0579, with a median of 137 interactions per user and an average of 208.86. In Ambar, we applied an iterative 45-core filtering, ensuring that each user rated at least 45 items and that at least 45 users rated each item. We then subsampled the dataset to 2500 users, 4996 items, and 214678 total ratings, resulting in a density of 0.0172. The median number of interactions per user is 74, with an average of 85.87.

6.4.2 Evaluation Protocol

Our experimental evaluation aims to emulate a real-world scenario in which users can grant permission for their data storage. Hence, solely data related to users

who allowed their data storage explicitly are included in the analysis, resulting in a reduced dataset called *Minimization Data* (D_M).

To create this scenario, the users of the entire dataset are randomly divided into two groups. The first group, representing 70% of all users, simulates those who do not consent to data storage and are excluded from the experiments. The second group consists of the remaining 30% of users who have consented to their data being stored. The interaction data related to these users constitute D_M . Within D_M , each user's ratings are further partitioned into three sets: a candidate set (70%), a validation set (10%), and a test set (20%). The candidate set serves as a pool of user data from which different minimization strategies can select a subset of ratings to train the system. Data is selected from this candidate set using a specific data minimization strategy and a minimization parameter n , which indicates the number of items to choose for each user. The study uses the following values for $n = \{1, 3, 7, 15, 100\}$. This approach draws inspiration from Biega et al. [31] but adapts it to a practical scenario, where individual user choices determine the availability of their data to the system.

6.4.3 Algorithms and Metrics

To provide a comprehensive view of how data minimization affects different types of recommender systems, we consider five representative algorithms spanning different methodological families:

- **LightGCN** [115] is a graph-based model that simplifies the aggregation process of Graph Convolutional Networks by eliminating the feature transformation step and non-linear activation functions. It aggregates messages from neighboring nodes linearly, assigning equal weights to all edges.
- **User-kNN** [198] is a neighborhood-based model that computes recommendations by identifying the nearest neighbors of a target user and aggregating their preferences. Similarity between users is typically defined based on the overlap of items they have co-rated, and final predictions are derived from a weighted combination of neighbor ratings.
- **BPRMF** [196] is a pairwise learning-to-rank method based on matrix factorization.
- **EASE^R** [228] is a linear model conceptually resembling an autoencoder by computing a closed-form item-item similarity matrix.
- **MultiVAE** [148] harnesses a deep variational autoencoder to learn non-linear latent representations from entire user interactions.

We evaluate the quality of recommendations of the selected algorithms through nDCG [142]. Furthermore, we select two fairness-related metrics to assess the impact of data minimization on recommendation performance on consumer and provider fairness:

- **Mean Absolute Deviation (MAD)** [67]: it measures the disparity in recommendation quality across different user groups. Formally, let Q_u denote the quality of

the recommendation for the user u and let $\mathcal{U}_x$ and $\mathcal{U}_y$ be two distinct groups of users. The mean absolute deviation (MAD) between these groups is defined as:

$$MAD(Q^{(\mathcal{U}_x)}, Q^{(\mathcal{U}_y)}) = \left| \frac{\sum_{x \in \mathcal{U}_x} Q_x}{|\mathcal{U}_x|} - \frac{\sum_{y \in \mathcal{U}_y} Q_y}{|\mathcal{U}_y|} \right|. \quad (6.1)$$

When the users are partitioned in more than two groups, MAD values are calculated for all possible pairwise group combinations. The final metric is derived as the mean of these computed MAD values. A smaller MAD value implies more equitable treatment across the groups, whereas a larger MAD reveals greater discrepancies and, thus, a higher degree of group-level unfairness. In our experiments, we use nDCG as the quality metric. In the ML1M dataset, users are partitioned by gender, while users are grouped by continent for the Ambar dataset.

- **Ranking-based Statistical Parity (RSP)** [319]: it measures how evenly different item categories are exposed to users in the top- k positions. Let us denote c_i as the item categories, and $P(R@k \mid c_i)$ the probability of a user encountering items in category c_i among the top- k recommendations ($R@k$). The RSP@ k is defined as:

$$RSP@k = \frac{\text{std}(P(R@k \mid c_1), \dots, P(R@k \mid c_n))}{\text{mean}(P(R@k \mid c_1), \dots, P(R@k \mid c_n))} \quad (6.2)$$

where $\text{std}(\cdot)$ and $\text{mean}(\cdot)$ denote the standard deviation and mean, respectively. A lower RSP indicates that the ranking probabilities are more evenly distributed, signifying less bias in how items are exposed to users. For ML1M, we categorize items into four release-year ranges. In the Ambar dataset, we group items by their continent of origin.

We also employ the **Hypervolume (HV)** [90] as a multi-objective metric to assess the model performance on accuracy, consumer, and provider fairness simultaneously. We choose the reference point $r = (1, 1.5, 1)$ for the three objectives: nDCG, RSP, and MAD.

6.4.4 Reproducibility Details

We adopt the all-unrated-items protocol [227], where the recommendable items are those not previously interacted with for each user. In our experiments, the embedding size is fixed at 64, and the batch size and the maximum number of epochs are 1024 and 300, respectively. Furthermore, we tune the model's hyperparameters through grid search. For BPRMF and LightGCN, the learning rate and the regularization term are tuned in $\{0.01, 0.005, 0.001\}$. The number of layers in LightGCN is set to 2. EASE^R is tuned by varying its regularization factor in $\{500, 1000, 5000\}$. Moreover, for UserKNN, the neighborhood size is searched in $\{20, 30, 50\}$, while we use Cosine and Jaccard as similarity measures. Finally, for

MultiVAE, the learning rate is tuned in $\{0.01, 0.001\}$ while the dimension of both the latent representation and the hidden layer is searched in $\{100, 200\}$. Combining all hyperparameter explorations for every model and strategy resulted in 2345 total trained models. We select the best models by adopting nDCG@10 as the validation metric, employing a patience value of 5 epochs for early stopping. Then, model performance results are computed on the test set. All the metrics are measured with a cutoff at 10, i.e., considering the top 10 suggested items to each user. To foster reproducibility, the models are trained using the reproducibility framework Elliot [9].

6.4.5 Data Minimization Strategies

In our study, we evaluate several approaches to minimize user data. These strategies are designed to select subsets of user-item interactions that serve as input to the system. In this chapter, we study the data minimization strategies explored by Biega et al. [31], aiming at broadening their assessment to fairness issues. We briefly describe such strategies in the following.

Full data. This baseline utilizes the complete set of user interactions without minimization. This strategy serves as a reference point against which the effectiveness of all other strategies is measured. Formally, the system observes all interactions in the user history.

Random selection. This approach randomly selects n interactions from each user's history. Although randomness may introduce variability, the average profile characteristics of users are often preserved due to the sampling process.

Most recent. This strategy selects the n most recent user interactions by assuming that recent behaviors are more relevant to the recommendation task. The effectiveness of this method depends on the temporal data distribution. Long observation periods may lead to performance variation when compared to random selection. We do not apply this strategy to the Ambar dataset due to lack of timestamp information.

Most and least favorite. These techniques select the n items with the highest or lowest scores for users, respectively. While the former emphasizes user preferences, the latter explores how negative interactions influence recommendations.

Most rated. This method selects n items rated most frequently in the dataset for each user. This global popularity measure identifies items that are statistically significant within the system.

Most characteristic. It selects items based on their proximity to the average item profile. Each item is represented as a binary vector indicating which users have interacted with it. The average item profile is computed by averaging all these binary vectors in the system. We calculate the Euclidean distance for each item between its binary representation and this average vector. Items with lower distances are deemed more representative of the overall trends and are therefore considered "most characteristic." For each user, we select the n items they have interacted with closest

to the system-wide profile, ensuring that the retained items best capture standard user behavior while avoiding niche or rarely rated items.

Highest variance. This strategy prioritizes items with the highest variability in their ratings across all users. Items with high variance are considered more informative as they capture diverse or polarizing opinions. The n items with the highest rating variance are selected for each user.

6.5 Results and Discussion

This section presents the empirical findings that address our three research questions.

6.5.1 Impact of Data Minimization (RQ1)

We answer RQ1 by discussing the impact of data minimization strategies on the accuracy, provider fairness, and consumer fairness of recommendation algorithms. Tables 6.1 and 6.2 report the model's performance measured with nDCG, RSP, MAD, and HV (see Section 6.4.3) for various minimization strategies and parameter n values for the ML1M and Ambar datasets, respectively. For each metric, we report the percentage differences in the performance between the considered strategy and the “full” strategy in parenthesis. Following Biega et al. [31], we say that minimization has a low impact on accuracy if the difference in nDCG is statistically insignificant or minimal. From the fairness perspective, minimization positively (negatively) impacts fairness if it strongly reduces (increases) the values of RSP and MAD, for which the lower, the better.

Impact on accuracy. Data minimization seriously alters the accuracy of recommendation algorithms. Reducing the parameter n affects recommendation accuracy, measured by nDCG, across datasets, models, and minimization strategies. This analysis corroborates the insights from Biega et al. [31], revealing that smaller n leads to a consistent drop in accuracy. Specifically, extreme minimization ($n = 1$) results in accuracy reductions that generally exceed 90% for many configurations, rendering recommendations impractical, as users cannot receive relevant suggestions. The accuracy performance loss is smaller but still noteworthy for moderate minimization ($n = 3$ and $n = 7$). At $n = 15$, dataset-specific differences emerge. In ML1M, the drop in accuracy remains pronounced, while the nDCG values become more acceptable for specific strategies in Ambar. Interestingly, the “Most Favorite” strategy even improves accuracy compared to the “full” strategy in some cases. At $n = 100$, accuracy differences become minimal or statistically insignificant in Ambar, demonstrating the feasibility of data minimization in this dataset under moderate conditions. In ML1M, performance differences remain measurable but less severe than smaller n values.

Impact on provider fairness. Data minimization introduces variability in provider

Table 6.1. Performance results of all models on the MLiM dataset across different minimization strategies and values of n . The percentage difference between each value and the corresponding value obtained with the full strategy is indicated below each result. For each scenario, bold and underline stand for best and second-to-best values, respectively. All results are statistically significant based on paired t-tests ($p < 0.05$), except for the values denoted with † .

Strategy	n = 1				n = 3				n = 7				n = 15				n = 100			
	nDCG↑	RSP↓	MAD↓	HVT↑	nDCG↑	RSP↓	MAD↓	HVT↑	nDCG↑	RSP↓	MAD↓	HVT↑	nDCG↑	RSP↓	MAD↓	HVT↑	nDCG↑	RSP↓	MAD↓	HVT↑
UserKNN																				
Random	0.0405	0.7432	0.0028	0.7852	0.0497	0.2577	0.009	1.2922	0.0711	0.379	0.0155	1.182	0.0977	0.5374	0.0306	1.0243	0.2351	0.4241	0.023	1.2982
	(-87.52)	(73.69)	(-93.62)	(-21.48)	(-84.7)	(-39.77)	(-79.76)	(29.22)	(-78.09)	(-11.43)	(-65.0)	(18.2)	(-69.89)	(25.59)	(-31.09)	(2.43)	(-27.55)	(-0.88)	(-48.17)	(29.82)
Most Recent	0.027	0.9834	0.0052	0.5277	0.0443	0.573	0.0061	0.9622	0.0736	0.4608	0.0046	1.1106	0.1165	0.4273	0.0002	1.1975	0.2556	0.4617	0.0216	1.2756
	(-91.68)	(129.83)	(-88.26)	(-47.23)	(-86.34)	(33.91)	(-86.32)	(-3.78)	(-77.3)	(7.68)	(-89.57)	(11.06)	(-64.11)	(-0.15)	(-99.64)	(19.75)	(-21.21)	(7.89)	(-51.36)	(27.56)
Most Favorite	0.0509	0.6137	0.0098	0.9223	0.0713	0.4552	0.0105	1.1075	0.0984	0.4927	0.0165	1.0882	0.1395	0.5473	0.0192	1.0647	0.2838	0.4362	0.0508	1.2964
	(-84.31)	(43.43)	(-78.0)	(-7.77)	(-78.02)	(6.38)	(-76.33)	(10.75)	(-69.67)	(15.14)	(-62.94)	(8.82)	(-57.01)	(27.9)	(-56.69)	(6.47)	(-12.53)	(1.93)	(14.34)	(29.64)
Least Favorite	0.0442	1.0051	0.0054	0.514	0.0336	0.6857	0.0041	0.8382	0.053	0.3819	0.0022	1.1748	0.0782	0.6994	0.0047	0.8591	0.2248	0.4354	0.0248	1.2716
	(-86.38)	(134.9)	(-87.91)	(-48.6)	(-89.65)	(60.25)	(-90.73)	(-16.18)	(-83.66)	(-10.75)	(-95.05)	(17.48)	(-75.9)	(63.45)	(-89.33)	(-14.09)	(-30.73)	(1.74)	(-44.09)	(27.16)
Most Rated	0.0921	1.0067	0.0159	0.5302	0.1447	1.0039	0.0157	0.559	0.2069	0.9652	0.0297	0.6262	0.2518	0.7458	0.0491	0.8978	0.3105	0.3669	0.0403	1.425
	(-71.61)	(135.26)	(-64.2)	(-46.98)	(-55.4)	(134.61)	(-64.54)	(-44.1)	(-36.23)	(125.57)	(-33.02)	(-37.38)	(-22.38)	(74.3)	(10.58)	(-10.22)	(-4.31)	(-14.25)	(-9.15)	(42.5)
Most Characteristic	0.0025	0.9607	0.0008	0.5402	0.0049	0.4391	0.0005	1.0656	0.011	0.1799	0.0024	1.3313	0.033	0.286	0.0048	1.2481	0.2122	0.3446	0.0178	1.3757
	(-99.22)	(124.51)	(-98.12)	(-45.98)	(-98.48)	(2.61)	(-98.87)	(6.56)	(-96.62)	(-57.95)	(-94.54)	(33.13)	(-89.82)	(-33.16)	(-89.23)	(24.81)	(-34.59)	(-19.47)	(-59.9)	(37.57)
Highest Variance	0.0152	0.8777	0.0004	0.6315	0.0381	0.1019	0.004	1.4455	0.0687	0.3048	0.0102	1.2643	0.1198	0.4355	0.01	1.1801	0.2756	0.4153	0.0251	1.3489
	(-95.32)	(105.12)	(-99.03)	(-36.85)	(-88.27)	(-76.19)	(-90.91)	(44.55)	(-78.82)	(-28.77)	(-77.08)	(26.43)	(-63.07)	(1.78)	(-77.5)	(18.01)	(-15.06)	(-2.94)	(-43.52)	(34.89)
BPRMF																				
Random	0.0358	0.3937	0.0043	1.1409	0.0453	0.4242	0.009	1.1144	0.0538	0.4735	0.0108	1.0701	0.0923	0.6841	0.013	0.8796	0.2329	0.4274	0.0161	1.3011
	(-88.82)	(-30.35)	(-82.55)	(14.09)	(-85.85)	(-24.95)	(-63.94)	(11.44)	(-83.18)	(-16.23)	(-56.69)	(7.01)	(-71.16)	(21.02)	(-47.48)	(-12.04)	(-27.25)	(-24.38)	(-35.35)	(30.11)
Most Recent	0.0275	0.456	0.0003	1.0723	0.0388	0.5041	0.0122	1.0219	0.0471	0.448	0.0071	1.0937	0.0924	0.5417	0.0115	1.0348	0.2396	0.4838	0.0012	1.2582
	(-91.4)	(-19.32)	(-98.72)	(7.23)	(-87.87)	(-10.82)	(-50.97)	(2.19)	(-85.3)	(-20.75)	(-71.31)	(9.37)	(-71.15)	(-4.17)	(-53.72)	(3.48)	(-25.14)	(-14.41)	(-95.35)	(25.82)
Most Favorite	0.0348	0.2236	0.0023	1.3178	0.0588	0.3714	0.011	1.1818	0.0889	0.5047	0.0288	1.0526	0.1215	0.6164	0.0226	0.9685	0.2823	0.4733	0.0307	1.2762
	(-89.13)	(-60.45)	(-90.81)	(31.78)	(-81.63)	(-34.29)	(-55.82)	(18.18)	(-72.24)	(-10.71)	(15.71)	(5.26)	(-62.04)	(9.05)	(-8.95)	(-3.15)	(-11.79)	(-16.27)	(23.54)	(27.62)
Least Favorite	0.025	0.5616	0.0041	0.958	0.021	0.4908	0.0004	1.03	0.0392	0.7105	0.0047	0.8165	0.0587	0.9444	0.0144	0.5798	0.225	0.6161	0.0053	1.077
	(-92.18)	(-0.64)	(-83.66)	(-4.2)	(-93.43)	(-13.18)	(-98.19)	(3.0)	(-87.76)	(25.7)	(-80.91)	(-18.35)	(-81.66)	(67.07)	(-42.24)	(-42.02)	(-29.69)	0.9104	(-78.6)	(7.7)
Most Rated	0.116	0.7281	0.0247	0.8402	0.1504	1.0669	0.0434	0.4796	0.1923	1.0969	0.0443	0.4632	0.2443	0.865	0.0496	0.751	0.299	0.5114	0.0421	1.2302
	(-63.76)	(28.8)	(-0.47)	(-15.98)	(-53.01)	(88.75)	(50.58)	(-52.04)	(-39.92)	(93.45)	(78.26)	(-53.68)	(-23.66)	(53.02)	(99.48)	(-24.9)	(-6.58)	(-9.53)	(69.27)	(23.02)
Most Characteristic	0.0021	0.1067	0.0001	1.396	0.0033	0.1248	0.0001	1.3796	0.0067	0.0427	0.0019	1.4642	0.0171	0.0989	0.0038	1.4197	0.1957	0.3798	0.0001	1.3393
	(-99.36)	(-81.12)	(-99.67)	(39.6)	(-98.98)	(-77.93)	(-99.54)	(37.96)	(-97.9)	(-92.44)	(-92.26)	(46.42)	(-94.65)	(-82.51)	(-84.58)	(41.97)	(-38.84)	(-32.81)	(-99.53)	(33.93)
Highest Variance	0.011	0.1258	0.0023	1.3861	0.025	0.1452	0.0037	1.3835	0.0628	0.7276	0.0097	0.8129	0.1004	0.6421	0.0165	0.9284	0.2648	0.5155	0.0066	1.237
	(-96.56)	(-77.74)	(-90.68)	(38.61)	(-92.19)	(-74.31)	(-85.28)	(38.35)	(-80.38)	(28.72)	(-61.0)	(-18.71)	(-68.65)	(13.59)	(-33.52)	(-7.16)	(-17.27)	(-8.8)	(-73.51)	(23.7)
EASE ^R																				
Random	0.0377	0.77	0.0011	0.7567	0.0495	0.2089	0.01	1.3415	0.0742	0.3458	0.0129	1.2339	0.115	0.7393	0.014	0.8363	0.2539	0.5541	0.0324	1.1477
	(-88.79)	(31.47)	(-97.93)	(-24.33)	(-85.28)	(-64.33)	(-80.69)	(34.15)	(-77.96)	(-40.96)	(-75.15)	(22.39)	(-65.83)	(26.22)	(-72.95)	(-16.37)	(-24.55)	(-5.41)	(-37.36)	(14.77)
Most Recent	0.0268	1.0037	0.0049	0.5072	0.0453	0.4955	0.005	1.0447	0.0718	0.4212	0.008	1.147	0.1246	0.4998	0.001	1.1237	0.2719	0.5382	0.0276	1.1894
	(-92.03)	(71.36)	(-90.61)	(-48.54)	(-86.55)	(-15.4)	(-90.27)	(4.47)	(-78.67)	(-28.09)	(-84.54)	(14.7)	(-62.98)	(-14.67)	(-98.07)	(12.37)	(-19.19)	(-8.11)	(-46.57)	(18.94)
Most Favorite	0.0491	0.6675	0.0091	0.8654	0.0676	0.425	0.0045	1.1424	0.115	0.6754	0.0178	0.903	0.1482	0.7227	0.0303	0.8655	0.2969	0.5889	0.0491	1.1236
	(-85.41)	(13.96)	(-82.37)	(-13.46)	(-79.91)	(-27.43)	(-91.23)	(14.24)	(-65.82)	(15.32)	(-65.53)	(-9.7)	(-55.96)	(23.39)	(-41.51)	(-13.45)	(-11.75)	(0.55)	(-5.06)	(12.36)
Least Favorite	0.0454	1.0169	0.0044	0.5028	0.034	0.6346	0.0005	0.8944	0.0498	0.2519	0.0069	1.3013	0.0913	0.7488	0.0093	0.8122	0.2418	0.5392	0.0244	1.164
	(-86.51)	(73.61)	(-91.47)	(-49.72)	(-89.88)	(8.35)	(-99.01)	(-10.56)	(-85.19)	(-57.0)	(-86.7)	(30.13)	(-72.86)	(27.84)	(-82.06)	(-18.78)	(-28.13)	(-7.94)	(-52.87)	(16.4)
Most Rated	0.0898	1.0172	0.0162	0.5177	0.16	0.9405	0.0264	0.6318	0.209	1.1198	0.0405	0.441	0.257	0.9422	0.0494	0.6666	0.325	0.6228	0.0654	1.0862
	(-73.3)	(73.67)	(-68.76)	(-48.23)	(-52.43)	(60.58)	(-48.89)	(-36.82)	(-37.88)	(91.19)	(-21.68)	(-55.9)	(-23.62)	(60.86)	(-4.55)	(-33.34)	(-3.4)	(6.33)	(26.49)	(8.62)
Most Characteristic	0.0026	1.0692	0.0009	0.4315	0.0044	0.1381	0.0007	1.3669	0.015	0.2157	0.0017	1.2968	0.0315	0.321	0.4998	1.2138	0.2253	0.3865	0.0204	1.3365
	(-99.22)	(82.55)	(-98.19)	(-56.85)	(-98.71)	(-76.42)	(-98.7)	(36.69)	(-96.59)	(-63.17)	(-96.72)	(29.68)	(-90.65)	(-45.2)	(-96.36)	(21.38)	(-33.04)	(-34.01)	(-60.51)	(33.65)
Highest Variance	0.0138	1.0179	0.0003	0.4886	0.0414	0.1476	0.001	1.4069	0.0792	0.5182	0.0083	1.0508	0.1174	0.7187	0.0078	0.8663	0.2932	0.5941	0.0321	1.1339
	(-95.9)	(73.79)	(-99.45)	(-51.14)	(-87.7)	(-74.8)	(-98.0)	(40.69)	(-76.46)	(-11.53)	(-83.91)	(5.08)	(-65.11)	(22.7)	(-85.0)	(-13.37)	(-12.84)	(1.43)	(-37.88)	(13.39)
LightGNN																				
Random	0.0437	0.306	0.0034	1.242	0.0436	0.2697	0.0038	1.2791	0.0687	0.4756	0.0127	1.0809	0.1183	0.8726	0.0161	0.6903	0.2439	0.3928	0.0226	1.3461
	(-87.0)	(-73.47)	(-90.65)	(24.2)	(-87.01)	(33.61)	(-89.37)	(27.91)	(-79.56)	(16.98)	(-64.61)	(8.09)	(-61.89)	(114.66)	(-54.96)	(-30.9)	(-27.38)	(-3.16)	(-36.99)	(34.61)
Most Recent	0.032	0.5049	0.0077	1.019	0.0427	0.4607	0.0093	1.0736	0.0641	0.4211	0.0031	1.1444	0.1183	0.4871	0.0035	1.1288	0.2406	0.461	0.0003	1.3461
	(-90.48)	(2.42)	(-78.41)	(9.1)	(-87.28)	(33.34)	(-74.02)	(7.36)	(-80.93)	(3.59)	(-91.22)	(14.44)	(-64.79)	(19.83)	(-90.37)	(12.88)	(-28.39)	(2.41)	(-99.13)	(34.4)
Most Favorite	0.0536	0.411	0.009	1.1371	0.0617	0.2915	0.0108	1.2692	0.112	0.598	0.0137	0.9982	0.1625	0.5028	0.0285	1.1297	0.2905	0.4596	0.0365	1.2935
	(-84.05)	(1.1)	(-74.95)	(13.71)	(-81.63)	(-28.29)	(-69.97)	(26.92)	(-66.67)	(66.41)	(-61.73)	(-0.18)	(-61.63)	(23.68)	(-0.22)	(12.97)	(-13.53)	(13.01)	(1.9)	(29.35)
Least Favorite	0.0318	0.6495	0.0155	0.8762	0.0292	0.6542	0.0028	0.868	0.0406	0.6807	0.0033	0.8498	0.084	0.8337	0.0149	0.7115	0.2463	0.4197	0.0701	1.3402
	(-90.53)	(59.78)	(-95.76)	(-12.38)	(-91.31)	(60.94)	(-92.12)	(1.13)	(-87.92)	(67.45)	(-90.81)	(-15.02)	(-75.07)	(105.08)	(-58.38)	(-28.85)	(-2.40)	(1.29)	(-80.06)	(34.02)
Most Rated	0.1094	0.5996	0.0146	0.9844	0.1524	1.0152	0.0296	0.5422	0.184											

Table 6.2. Performance results of all models on the Ambar dataset across different minimization strategies and values of n . The percentage difference between each value and the corresponding value obtained with the full strategy is indicated below each result. For each scenario, bold and underline stand for best and second-to-best values, respectively. All results are statistically significant based on paired t-tests ($p < 0.05$), except for the values denoted with † .

Strategy	n = 1				n = 3				n = 7				n = 15				n = 100			
	nDCG↑	RSP↓	MAD↓	HV↑	nDCG↑	RSP↓	MAD↓	HV↑	nDCG↑	RSP↓	MAD↓	HV↑	nDCG↑	RSP↓	MAD↓	HV↑	nDCG↑	RSP↓	MAD↓	HV↑
UserKNN																				
Random	0.0036	<u>1.4422</u>	0.0049	<u>0.0577</u>	0.0243	0.9005	0.0756	<u>0.5676</u>	0.0572	0.7075	0.0324	0.8107	<u>0.1147</u>	0.8074	0.0463	0.7363	0.1373 [†]	0.8199	0.062	0.7255
	(-97.26)	(75.43)	(-91.79)	(-94.23)	(-81.53)	(9.54)	(25.48)	(-43.24)	(-56.55)	(-13.95)	(-46.17)	(-18.93)	(-12.88)	(-1.8)	(-23.21)	(-26.37)	(4.29)	(-0.27)	(2.93)	(-27.45)
Most Favorite	<u>0.0101</u>	1.4621	0.0044	0.0381	<u>0.049</u>	<u>0.8691</u>	0.0241	<u>0.6459</u>	0.1092	0.7307	0.0361	0.8225	0.1498	0.8174	<u>0.0483</u>	<u>0.7469</u>	<u>0.1404</u> [†]	0.8755	0.06	0.6604
	(-92.35)	(77.84)	(-92.72)	(-96.19)	(-62.78)	(5.71)	(-60.03)	(-35.41)	(-17.03)	(-11.12)	(-40.13)	(-17.75)	(13.77)	(-0.57)	(-19.8)	(-25.31)	(6.64)	(6.5)	(-0.42)	(-33.06)
Least Favorite	0.0055	1.4863	0.0354	0.0133	0.0115	0.9036	<u>0.0053</u>	0.6	0.0345	<u>0.5747</u>	<u>0.0153</u>	<u>0.9426</u>	0.0806	0.9176	0.1341	0.545	0.1362 [†]	0.8072	0.0683	0.7335
	(-95.84)	(80.79)	(-41.26)	(-98.67)	(-91.3)	(9.91)	(-91.19)	(-40.0)	(-73.81)	(-30.1)	(-74.6)	(-5.74)	(-38.75)	(11.61)	(122.41)	(-45.5)	(3.48)	(-1.82)	(13.31)	(-26.65)
Most Rated	0.0164	1.447	0.0101	0.0534	0.0769	1.1329	0.0644	0.3699	0.0938	1.0878	0.0525	0.4272	0.1103	0.9898	0.0524	0.5369	0.1311 [†]	0.8177	0.0594	0.7259
	(-87.55)	(76.0)	(-83.21)	(-94.66)	(-41.56)	(37.8)	(6.79)	(-63.01)	(-28.74)	(32.31)	(-12.91)	(-57.28)	(-16.2)	(20.39)	(-13.05)	(-46.31)	(-0.4)	(-0.53)	(-1.54)	(-27.41)
Most Characteristic	0.0016	1.5222	0.0005	0.0	0.01	1.1757	0.0041	0.3262	0.0285	0.491	0.0077	1.0297	0.0407	0.6132	0.0504	0.8763	0.1437 [†]	0.8548	0.0707	0.6858
	(-98.75)	(85.16)	(-99.18)	(-100.0)	(-92.44)	(43.01)	(-93.14)	(-67.38)	(-78.37)	(-40.28)	(-87.17)	(2.97)	(-69.12)	(-25.41)	(-16.35)	(-12.37)	(9.13)	(3.97)	(17.29)	(-31.42)
Highest Variance	0.0015	0.9225	<u>0.0007</u>	0.578	0.0237	0.6099	0.0121	0.9001	0.062	0.6658	0.0396	0.8508	0.0976	<u>0.7875</u>	0.0553	0.7388	0.1318 [†]	0.8138	0.0598	0.7301
	(-98.83)	(12.21)	(-98.84)	(-42.2)	(-82.02)	(-25.81)	(-79.92)	(-9.99)	(-52.9)	(-19.01)	(-34.26)	(-14.92)	(-25.83)	(-4.21)	(-8.25)	(-26.12)	(0.09)	(-1.01)	(-0.72)	(-26.99)
BPRMF																				
Random	0.0064	0.7976	0.0091	0.7005	0.004	0.6976	<u>0.0022</u>	0.8038	0.0329	0.7796	0.0168	0.7316	0.0884	0.7978	0.0484	0.7272	0.1169 [†]	0.8874	0.0476	0.6516
	(-94.64)	(-1.26)	(-78.83)	(-29.95)	(-96.61)	(-13.63)	(-94.82)	(-19.62)	(-72.3)	(-3.49)	(-61.05)	(-26.84)	(-25.52)	(-1.22)	(12.26)	(-27.28)	(-1.52)	(9.87)	(10.4)	(-34.84)
Most Favorite	0.0089	<u>0.5577</u>	0.0099	<u>0.9413</u>	<u>0.0324</u>	0.6987	0.0179	0.8125	0.0798	0.6438	0.0266	0.8999	0.1227	0.7991	0.0451	<u>0.7514</u>	0.1235 [†]	0.9042	0.0442	0.6398
	(-92.53)	(-30.96)	(-77.09)	(-5.87)	(-72.69)	(-13.49)	(-58.62)	(-18.75)	(-32.8)	(-20.29)	(-38.33)	(-10.01)	(3.34)	(-1.07)	(4.57)	(-24.86)	(3.99)	(11.94)	(2.56)	(-36.02)
Least Favorite	0.0062	0.5746	0.0034	0.9279	0.0098	0.7216	0.0041	0.7828	0.0157	0.7214	0.0051	0.7869	0.0657	<u>0.7916</u>	0.125	0.6605	0.118 [†]	0.8898	0.0485	0.6491
	(-94.78)	(-28.86)	(-92.04)	(-7.21)	(-91.73)	(-10.66)	(-90.49)	(-21.72)	(-86.74)	(-10.69)	(-88.22)	(-21.31)	(-44.66)	(-1.99)	(189.85)	(-33.95)	(-0.62)	(10.17)	(12.51)	(-35.09)
Most Rated	0.0245	1.2525	0.01	0.2511	0.0744	1.1952	0.0634	0.3067	0.0917	1.1311	0.0459	0.3842	0.0957	1.0444	<u>0.0374</u>	0.4805	0.1064	<u>0.8685</u>	0.0398	0.6709
	(-79.38)	(55.06)	(-76.84)	(-74.89)	(-37.33)	(47.97)	(46.92)	(-69.33)	(-22.79)	(40.04)	(6.35)	(-61.58)	(-19.41)	(29.31)	(-13.31)	(-51.95)	(-10.36)	(7.53)	(-7.73)	(-32.91)
Most Characteristic	0.0008	0.3023	0.0003	1.1983	0.0039	0.5627	0.0017	0.9393	0.0087	0.413	<u>0.008</u>	1.0877	0.034	0.5905	0.012	0.9292	0.1192	0.8914	0.0439	0.6512
	(-99.35)	(-62.58)	(-99.29)	(19.83)	(-96.74)	(-30.34)	(-96.03)	(-6.07)	(-92.71)	(-48.87)	(-81.56)	(8.77)	(-71.34)	(-26.9)	(-72.24)	(-7.08)	(0.35)	(10.37)	(1.7)	(-34.88)
Highest Variance	0.0025	0.5701	<u>0.0012</u>	0.9311	0.0046	0.6246	0.0017	0.878	0.0437	<u>0.6134</u>	0.0377	0.8904	0.0809	0.8204	0.0451	0.7015	0.1177 [†]	0.8217	0.0441	0.7247
	(-97.89)	(-29.42)	(-97.13)	(-6.89)	(-96.11)	(-22.67)	(-96.09)	(-12.2)	(-63.22)	(-24.06)	(-12.58)	(-10.96)	(-31.85)	(1.58)	(4.44)	(-29.83)	(-0.91)	(1.73)	(2.34)	(-27.53)
EASE ^R																				
Random	0.0033	<u>1.4402</u>	0.0049	<u>0.0597</u>	0.0255	0.8246	0.077	<u>0.6393</u>	0.0583	0.7536	0.0469	0.7529	<u>0.1119</u>	0.8115	0.0449	0.7312	0.149	0.8715	0.0646	0.6755
	(-97.75)	(64.81)	(-92.29)	(-94.03)	(-82.42)	(-5.63)	(20.7)	(-36.07)	(-59.83)	(-13.76)	(-26.4)	(-24.71)	(-22.89)	(-7.13)	(-29.64)	(-26.88)	(2.71)	(-0.26)	(1.3)	(-32.45)
Most Favorite	<u>0.0101</u>	1.4621	0.0044	0.0381	<u>0.0543</u>	0.8143	0.0349	0.6977	0.1058	0.7562	<u>0.0338</u>	0.7947	0.1508	0.8305	<u>0.0477</u>	0.7338	<u>0.1507</u> [†]	0.8769	0.0646	0.6706
	(-93.06)	(67.32)	(-93.12)	(-96.19)	(-62.61)	(-6.82)	(-45.32)	(-30.23)	(-27.06)	(-13.46)	(-47.0)	(-20.53)	(3.95)	(-4.96)	(-25.27)	(-26.62)	(3.85)	(0.35)	(1.33)	(-32.94)
Least Favorite	0.0049	1.5398	<u>0.0018</u>	0.0	0.0128	0.7903	0.0426	0.6882	0.0363	<u>0.6153</u>	0.0787	<u>0.8447</u>	0.0709	0.9006	0.1432	0.55	0.15 [†]	<u>0.8504</u>	0.0717	<u>0.6935</u>
	(-96.6)	(76.21)	(-97.12)	(-100.0)	(-91.17)	(-9.56)	(-33.24)	(-31.18)	(-74.95)	(-29.59)	(23.34)	(-15.53)	(-51.15)	(3.06)	(124.6)	(-45.0)	(3.36)	(-2.68)	(12.37)	(-30.65)
Most Rated	0.0164	1.447	0.0101	0.0534	0.0741	1.1241	0.055	0.3816	0.0937	1.0882	0.0513	0.4273	0.1088	1.0503	0.0561	0.4706	0.1436	0.8698	0.063	0.6753
	(-88.7)	(65.59)	(-84.13)	(-94.66)	(-48.93)	(28.64)	(-13.73)	(-61.84)	(-35.41)	(24.53)	(-19.52)	(-57.27)	(-25.03)	(20.2)	(-12.05)	(-52.94)	(-1.06)	(-0.47)	(-1.15)	(-32.47)
Most Characteristic	0.0016	1.5222	0.0005	0.0	0.01	0.6672	0.004	<u>0.8377</u>	0.0265	0.4624	0.0072	1.0574	0.0387	0.5933	0.0597	0.8856	0.152 [†]	0.8489	0.0733	0.6952
	(-98.87)	(74.2)	(-99.22)	(-100.0)	(-93.14)	(-23.64)	(-93.71)	(-16.23)	(-81.77)	(-47.08)	(-88.75)	(5.74)	(-73.35)	(-32.11)	(-6.39)	(-11.44)	(4.77)	(-2.86)	(14.93)	(-30.48)
Highest Variance	0.0008	0.9525	0.0005	0.5476	0.0226	0.5795	<u>0.0111</u>	0.9309	0.0602	0.7098	0.0386	0.8054	0.0989	<u>0.7882</u>	0.0567	<u>0.7378</u>	0.1445	0.8624	0.0633	0.6836
	(-99.44)	(9.01)	(-99.22)	(-45.24)	(-84.45)	(-33.69)	(-82.56)	(-6.91)	(-58.52)	(-18.77)	(-39.5)	(-19.46)	(-31.81)	(-9.8)	(-11.05)	(-26.22)	(-0.38)	(-1.31)	(-0.76)	(-31.64)
MultiVAE																				
Random	0.0116	0.8987	0.0094	0.6026	0.0192	0.8336	<u>0.0079</u>	0.6739	0.0284	0.9992	<u>0.0175</u>	0.506	0.0676	0.8202	0.0471	0.6915	0.101 [†]	0.8094	0.044	0.7269
	(-88.45)	(3.88)	(-78.09)	(-39.74)	(-80.87)	(-3.64)	(-81.63)	(-32.61)	(-71.79)	(15.5)	(-59.08)	(-49.4)	(-32.78)	(-5.19)	(10.01)	(-30.85)	(0.48)	(-6.44)	(2.88)	(-27.31)
Most Favorite	<u>0.0198</u>	0.8718	0.0262	0.6239	0.0419	0.6107	0.0192	0.9088	<u>0.073</u>	0.7678	0.0244	0.7666	0.105	0.7589	<u>0.032</u>	0.7902	0.1056 [†]	0.7931	0.0455	<u>0.746</u>
	(-80.31)	(0.77)	(-38.81)	(-37.61)	(-58.31)	(-29.41)	(-55.24)	(-9.12)	(-27.36)	(-11.25)	(-43.08)	(-23.34)	(0.92)	(-12.28)	(-25.2)	(-20.98)	(5.07)	(-8.32)	(6.4)	(-25.4)
Least Favorite	0.0085	0.5952	0.0033	0.9095	0.0096	0.9784	0.0523	0.4991	0.0167	0.9492	0.0263	0.5453	0.042	0.8337	0.0614	0.6321	<u>0.1031</u> [†]	<u>0.789</u>	0.0517	0.7438
	(-91.52)	(-31.21)	(-92.28)	(-9.05)	(-90.42)	(13.09)	(22.25)	(-50.09)	(-83.43)	(9.72)	(-38.67)	(-45.47)	(-58.21)	(-1.32)	(43.4)	(-36.79)	(2.54)	(-8.8)	(20.68)	(-25.62)
Most Rated	0.0212	1.2257	0.0158	0.2757	0.0723	1.1437	0.0713	0.3548	0.087	1.2147	0.0462	0.2958	<u>0.1009</u>	0.9789	0.0403	0.5506	<u>0.1016</u> [†]	0.8352	0.0436	0.7005
	(-78.94)	(41.68)	(-63.14)	(-72.43)	(-28.09)	(32.2)	(66.69)	(-64.52)	(-13.43)	(40.4)	(8.03)	(-70.42)	(0.38)	(13.15)	(-5.9)	(-44.94)	(1.1)	(-3.46)	(1.78)	(-29.95)
Most Characteristic	0.0018	0.4404	0.0006	1.0608	0.0006	0.5933	0.0006	0.9067	0.014	0.389	0.013	1.1119	0.0242	0.5258	0.0128	0.9851	0.1023 [†]	0.813	0.0331	0.7323
	(-98.24)	(-49.1)	(-98.58)	(6.08)	(-99.36)	(-31.42)	(-98.49)	(-9.33)	(-86.11)	(-55.03)	(-69.63)	(11.19)	(-75.89)	(-39.22)	(-70.08)	(-1.81)	(1.78)	(-6.03)	(-22.59)	(-26.77)
Highest Variance	0.0029	<u>0.5896</u>	<u>0.0016</u>	<u>0.9115</u>	0.0241	0.6378	0.0114	0.873	0.0492	<u>0.6604</u>	0.0271	<u>0.8571</u>	0.0787	<u>0.637</u>	0.0388	<u>0.8948</u>	0.101 [†]	0.7737	0.0478	0.7614
	(-97.16)	(-31.85)	(-96.31)	(-8.85)	(-76.0)	(-26.27)	(-73.47)	(-12.7)	(-51.02)	(-23.67)	(-36.64)	(-14.29)	(-21.69)	(-26.37)	(-9.39)	(-10.52)	(0.43)	(-10.57)	(11.57)	(-23.86)
LightGCN																				
Random	0.0083	0.7742	0.0302	0.709																

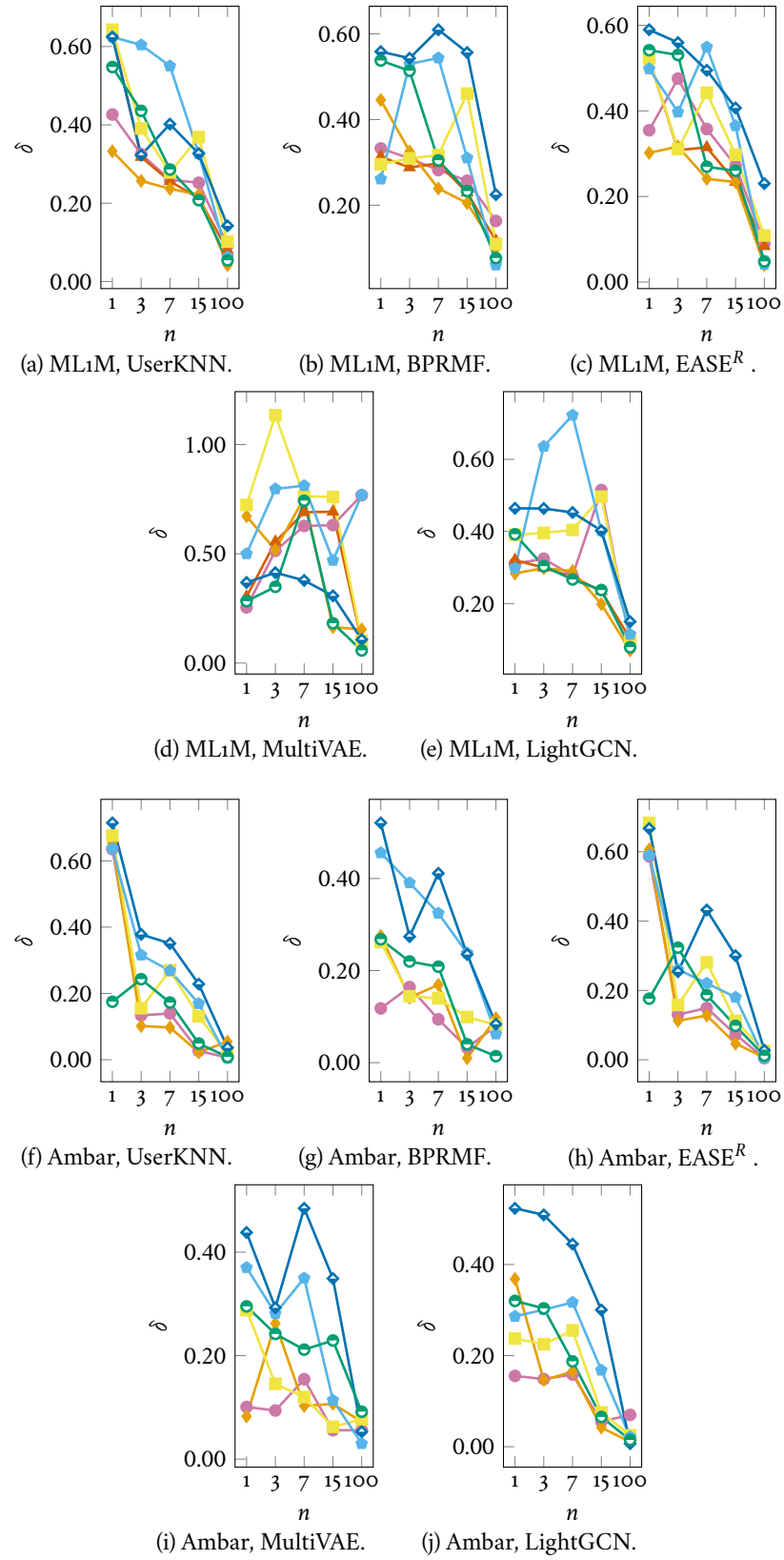


Figure 6.1. Comparison of the Euclidean distance (δ) between the point representing the model performance under a minimization strategy with value n and the point representing the model performance under the “full” data strategy. The colored lines represent the different minimization strategy. Smaller δ values indicate better preservation of the model’s overall performance.

fairness, as measured by RSP. Lower n values lead to inconsistent changes in RSP, with both improvements (negative changes compared to the “full” strategy) and deteriorations (positive changes) observed. This variability indicates a loss of control over provider fairness, particularly at smaller n , where fairness outcomes become unpredictable. Regarding the Ambar dataset, the relationship between accuracy and provider fairness is evident: strategies achieving higher nDCG often induce worse provider fairness. For example, the “Most Rated” strategy, which excels in accuracy, consistently increases RSP compared to the “full” strategy, indicating fairness degradation. In contrast, the “Most Characteristic” strategy sacrifices accuracy but consistently improves provider fairness, highlighting a trade-off between these two objectives.

Impact on consumer fairness. Data minimization generally enhances consumer fairness, as lower MAD values indicate. This improvement mainly stems from a reduction in overall nDCG, which levels the quality of recommendations across user groups to the detriment of overall recommendation relevance.

Impact of strategies. The interplay between data minimization and the objectives of accuracy, provider fairness, and consumer fairness is intricate, with varying impacts on recommendation performance. To unravel these complexities, we analyze the minimization strategies to identify patterns in their performance when $n \leq 15$. The “Most Rated” strategy, which emphasizes the presence of popular items in the training dataset, generally achieves the best or second-best nDCG values. However, this focus on popularity comes at a cost: provider fairness, as measured by RSP, deteriorates significantly, especially at smaller n . As n increases, the items are exposed more fairly due to the inclusion of a broader range of items in the training set. Nevertheless, the low values of HV indicate that this strategy predominantly optimizes for accuracy, with limited alignment to beyond-accuracy objectives. The “Most Favorite” strategy prioritizes the items most liked by individual users, resulting in competitive nDCG values (often the second-best) while mitigating the fairness degradation seen in “Most Rated.” This strategy strikes a better balance between accuracy and fairness since the items most liked by users are not necessarily the most popular. Indeed, the acceptable HV values suggest a more harmonious fulfillment of accuracy and fairness objectives. In contrast, the “Most Characteristic” strategy consistently achieves the worst performance in terms of accuracy but demonstrates improvements in provider fairness. This strategy favors the representation of items most interacted with by the most active users, effectively tailoring user profiles for this subgroup. However, on most platforms, most users are less active, resulting in a decline in overall recommendation quality when adopting this strategy. The best HV values associated with this strategy indicate that it excels in achieving beyond-accuracy objectives, albeit with poor accuracy performance. The remaining strategies introduce higher variability in user profiles and tend to achieve a better balance between accuracy and fairness. This balance is reflected in their HV values, which suggest moderate fulfillment of all objectives.

Table 6.3. Mean (μ) and standard deviation (σ) of the distances among the points represented by each model performance under different strategies and the point corresponding to the model performance under the “full” data strategy, across different n . For each scenario, bold and underline stand for best and second-to-best values, respectively.

Model	$n = 1$		$n = 3$		$n = 7$		$n = 15$		$n = 100$		Global	
	$\mu \downarrow$	$\sigma \downarrow$	$\mu \downarrow$	$\sigma \downarrow$	$\mu \downarrow$	$\sigma \downarrow$	$\mu \downarrow$	$\sigma \downarrow$	$\mu \downarrow$	$\sigma \downarrow$	$\mu \downarrow$	$\sigma \downarrow$
Movielens1M												
UserKNN	0.547	0.1216	0.379	<u>0.1146</u>	0.3241	0.1134	0.2736	0.0656	0.0821	<u>0.0339</u>	<u>0.3212</u>	0.1777
BPRMF	<u>0.3921</u>	0.1211	0.4027	0.1188	<u>0.3705</u>	0.1443	0.3217	0.1346	0.1216	0.056	0.3217	0.1537
EASER	0.4758	<u>0.1058</u>	0.4141	0.1082	0.3814	<u>0.1169</u>	<u>0.2957</u>	<u>0.0666</u>	<u>0.0918</u>	0.0667	0.3317	0.1622
MultiVAE	0.4435	0.1916	0.6115	0.2701	0.6814	0.1461	0.4583	0.2455	0.2935	0.3259	0.4977	0.2676
LightGCN	0.3515	0.0657	<u>0.3887</u>	0.1253	0.384	0.1662	0.3551	0.1309	0.1014	0.0261	0.3161	<u>0.1538</u>
Ambar												
UserKNN	0.5821	0.2017	0.2212	0.1098	0.2163	0.0956	0.1043	0.085	<u>0.0212</u>	<u>0.02</u>	0.229	0.2236
BPRMF	0.3166	0.1467	0.2223	0.0969	<u>0.2245</u>	0.1203	<u>0.109</u>	0.1031	0.0698	0.0294	<u>0.1884</u>	<u>0.1339</u>
EASER	0.5516	0.1884	0.2063	<u>0.0847</u>	0.2328	0.1116	0.135	<u>0.0926</u>	0.0133	0.0105	0.2278	0.21
MultiVAE	0.2625	<u>0.1433</u>	<u>0.2196</u>	0.0809	0.2372	0.1506	0.1529	0.1145	0.0632	0.0216	0.1871	0.1275
LightGCN	<u>0.3151</u>	0.1254	0.2722	0.1351	0.2546	<u>0.1113</u>	0.1177	0.1	0.0249	0.0231	0.1969	0.1483

To answer RQ1, we conclude that data minimization can lead to a decrease in the accuracy performance of recommendation models. While this outcome directly offers fairer recommendations across different user groups, data minimization can be responsible for unfair exposure of items. Hence, these objectives potentially show a critical trade-off, underscoring the need for nuanced strategies to balance these competing objectives in recommendation systems.

6.5.2 Data Minimization Strategies Performance (RQ2)

To answer RQ2, we evaluate how effectively data minimization strategies preserve the performance of recommender systems compared to the “full” strategy. We adopt a multi-objective evaluation approach [180, 181] that simultaneously considers multiple dimensions of analysis. Fixing a value of n and a recommendation model, we measure the model’s performance regarding nDCG, RSP, and MAD for each minimization strategy applied. These metrics define a three-dimensional objective function space, where each model trained under a specific minimization strategy is represented as a point identified by the corresponding metric values. The model trained using the “full” strategy serves as a reference point in this space. To quantify the similarity of performance between the “full” strategy and each minimization strategy, we compute the Euclidean distance δ of every point (i.e., the model trained under a minimization strategy) from the reference point. A smaller Euclidean distance indicates that the minimization strategy better preserves the model’s overall performance compared

to the “full” strategy. Figure 6.1 presents the Euclidean distance results for each model and dataset as n varies.

Consistent with the trends observed in Tables 6.1 and 6.2, setting $n = 100$ enables most strategies to effectively preserve model performance, as evidenced by the lowest δ values. No single strategy emerges as a definitive winner at this level, reflecting similar performance across strategies. Conversely, as n decreases, a more evident hierarchy among the strategies emerges.

The “Most Rated” and “Most Characteristic” strategies generally exhibit higher δ values, indicating poorer performance preservation—except for MultiVAE trained on ML1M (Figure 6.1d). This behavior aligns with the observations in Section 6.5.1: the “Most Rated” strategy degrades model performance primarily regarding fairness, while the “Most Characteristic” strategy compromises accuracy.

In contrast, the “Most Favorite” strategy consistently preserves model performance, frequently achieving the lowest distance values. As discussed in Section 6.5.1, this strategy effectively delineates user profiles without adversely affecting fairness. Similar performance trends can be observed for the “Random,” “Most Recent,” and “Highest Variance” strategies. These strategies introduce variability into user profiles, yet this variability enables the retention of average user profile characteristics. Consequently, these strategies maintain acceptable accuracy levels while avoiding a disproportionate emphasis on specific items in the training dataset. The resulting datasets closely resemble those generated by the “full” strategy, which explains their superior performance compared to the “Most Rated” and “Most Characteristic” strategies. Among these, the “Random” strategy is the most effective. As noted in Section 6.4.5, it best maintains the average profile of each user. Finally, the “Least Favorite” strategy generally underperforms compared to the “Random” strategy. This behavior is likely because models find learning effectively from negative feedback more challenging, resulting in more significant performance degradation.

To answer RQ2, we say that data minimization strategies that can shape better user profiles and introduce variability strike a better balance among the considered objectives, being able to retain the model’s performance compared to the “full” strategy.

6.5.3 Robustness of Recommenders to Data Minimization (RQ3)

To address RQ3, we evaluate the robustness of recommendation models under data minimization while considering both accuracy and fairness. Here, robustness is defined as a model’s ability to preserve its performance under data minimization compared to the full-data setting. We adopt the multi-objective approach outlined in Section 6.5.2 to quantify this. We compute the performance achieved under each data minimization strategy for each model in the three-dimensional objective function space—defined by the nDCG, RSP, and MAD metrics. Given a model and fixed n , we measure the Euclidean distances of these performance points from the reference point, i.e., the model’s performance under the “full” strategy. The mean μ and standard deviation σ of these distances capture the model’s robustness: a lower μ indicates

better overall robustness, while a lower σ suggests more consistent performance across strategies. Table 6.3 reports μ and σ for each model across different values of n and their global averages across all settings (i.e., without fixing n).

In Ambar and ML1M, UserKNN demonstrates strong robustness for $n \geq 3$, consistently achieving the lowest (or second-lowest) values of μ and σ . This finding aligns with previous works [31, 56], which showed that neighborhood-based models can maintain performance even after removing random interactions. Similarly, EASE^R is robust only when data minimization is moderate. Its reliance on item co-occurrence statistics makes it highly sensitive to severe sparsity. If the goal is to ensure robustness even with minimal data ($n = 1$), LightGCN emerges as the most reliable model across both datasets. This finding aligns with its graph-based structure, which propagates information from neighboring users and items, mitigating the impact of data sparsity. Moreover, LightGCN maintains low μ and σ across all scenarios, making it the most robust model overall. BPRMF follows closely behind LightGCN, demonstrating strong generalization under data minimization. Its factorization-based approach enables it to retain performance even with limited interactions, though it lacks the graph-based smoothing effect of LightGCN, leading to slightly higher variability. Finally, MultiVAE exhibits the most dataset-dependent behavior, showing weaker robustness in ML1M but achieving the best performance in Ambar at the global level. This observation suggests that its variational latent space representations are highly sensitive to dataset characteristics, which may impact its stability under data minimization.

To answer RQ3, we conclude that graph-based modeling is crucial for robustness under extreme data sparsity when considering accuracy and fairness objectives, enhancing the classic factorization-based representation of users and items. Additionally, traditional neighborhood-based and non-linear methods show limitations in such scenarios, suffering from high data reduction.

6.6 Conclusion and Future Work

In this study, we systematically investigated the impact of data minimization on recommender systems, with particular attention to the trade-offs among accuracy, consumer fairness, and provider fairness. Our contributions extend the current understanding of data minimization in three key ways. First, we empirically evaluate how different data minimization strategies across real-world datasets impact the accuracy, provider, and consumer fairness of recommendations. Second, we employ a multi-objective evaluation to investigate which data minimization strategies can simultaneously preserve recommendation performance across multiple objectives. Third, we explore how different model classes respond to varying degrees of data reduction, shedding light on the relationship between architectural design and data scarcity.

Our findings show that although data minimization can diminish accuracy per-

formance, it levels the recommendation quality of different user groups, enhancing consumer fairness. However, reducing data can inadvertently lead to the unfair exposure of specific items, highlighting a critical trade-off between data minimization, accuracy, and fairness. Further analysis highlights specific behaviors of data minimization strategies. Particularly, approaches that introduce variability when shaping the user profiles can strike a better balance among data minimization, accuracy, and fairness of recommendations. Finally, our insights indicate that graph-based recommender systems remain more resilient under extreme data minimization when fairness and accuracy are jointly considered, in contrast to traditional methods that exhibit lower robustness.

Future works may seek to devise more refined data minimization strategies that promote equitable recommendations while minimizing adverse effects on accuracy performance. In addition, our insight can guide the development of recommender systems specifically devised to be robust under severe data minimization conditions while accomplishing fairness objectives.

Chapter 7

Assessing the Role of Data Characteristics in Recommenders Performance Under Sparsity

The growing emphasis on regulatory compliance and data efficiency has placed the principle of data minimization, learning from the smallest feasible subset of data, a key challenge for Recommender Systems (RSs). While Graph Neural Networks (GNNs) have become the dominant paradigm in collaborative filtering, their behaviour under deliberate data reduction remains largely unexplored. It is unclear which underlying data characteristics determine their robustness and how these factors are influenced by the minimization strategy itself. This chapter conducts an extensive empirical and explanatory analysis of GNN-based RSs under realistic data minimization scenarios. We first benchmark GNNs models against non-graph baselines across a wide spectrum of minimization strategies, finding that GNNs maintain superior performance even under severe data reduction. Beyond performance comparison, we introduce an explanatory framework to link model effectiveness to dataset characteristics, revealing that the significance of these characteristics are a direct consequence of the criteria guiding the minimization process. Building on this insight, we propose a novel family of minimization-semantic characteristics designed to encode the logic of the retention criteria, demonstrating that their inclusion substantially enhances our model's explanatory power and provides actionable guidelines for designing more effective minimization strategies.¹

1. This chapter is based on the work submitted for publication to the journal *Expert Systems with Applications* (ESWA) "Explaining GNN-based Recommender Systems Performance under Data Minimization: The Role of Data Characteristics" .

7.1 Introduction

Recommender systems (RSs) have emerged as pivotal tools in the digital ecosystem, enabling users to navigate vast repositories of information and discover items aligned with their preferences. From e-commerce platforms [154] to streaming services [24] and social networks [109], RSs leverage historical user interactions to generate personalized suggestions, thereby enhancing user engagement and satisfaction [199]. Traditional approaches, such as collaborative filtering (CF) via matrix factorization, have been effective in modeling user-item relationships but often fall short in capturing complex, higher-order dependencies inherent in interaction data [140]. This limitation has spurred the adoption of graph-based paradigms, where user-item interactions are represented as bipartite graphs, allowing graph neural networks (GNNs) to propagate information across connected nodes and learn more nuanced embeddings [107]. Consequently, GNN-based recommender systems have gained significant momentum, with numerous solutions proposed to enhance existing models [115, 259, 276].

Despite these advancements, the escalating volume of user data collected by RSs raises significant concerns regarding privacy, security, and the ethical use of data. Regulations such as the GDPR [192], CCPA [40], CPRA [236], and PIPL [242] mandate the principle of data minimization, requiring that only the minimal amount of personal data necessary for a given purpose be collected and processed [79]. In RSs, this translates to training models on reduced subsets of interactions—selected via specific minimization strategies that encode semantic criteria (e.g., recency, popularity, diversity)—while striving to preserve recommendation quality [31].

However, applying data minimization introduces significant challenges: it alters the structural, distributional, and topological properties of the dataset, potentially degrading model performance if the retained interactions fail to capture essential patterns [208]. This issue is particularly critical for GNN-based RSs, which explicitly model interactions as bipartite graphs and rely on message-passing mechanisms to exploit higher-order connectivity. Since data minimization directly prunes edges and reshapes node degrees, clustering, and connectivity, the effectiveness of GNNs becomes inherently tied to the quality and semantics of the retained subgraph—making the investigation of their performance under minimization not only relevant but fundamentally necessary.

Prior research has explored dataset characteristics that influence RS performance through explanatory frameworks [4, 68, 72]. Notably, [161] proposed an evaluation pipeline to uncover statistical correspondences between graph topological properties and GNN-based RS performance. However, no prior work has examined how data minimization—through its specific semantic criteria—modifies these characteristics and, in turn, impacts GNN efficacy. The importance of this investigation is twofold: (i) GNNs may exhibit resilience or even superior performance under minimization due to their ability to infer from sparse, interconnected data—a hypothesis requiring

empirical validation; (ii) understanding the role of dataset characteristics could enable semantically informed data reduction strategies.

This chapter addresses these gaps by investigating GNN-based RSs under data minimization scenarios. We first empirically demonstrate that GNNs consistently outperform traditional RSs in such settings. We then adopt an explanatory framework to analyze how structural, distributional, and topological characteristics of minimized datasets influence GNN performance, identifying the most predictive features. Furthermore, we show that the importance of these characteristics is specific to minimization strategies and does not generalize to random pruning. Finally, we introduce minimization-semantic characteristics to capture the criteria used in interaction retention, augmenting the explanatory power of the framework.

Specifically, we address the following research questions:

- **RQ1.** *Do GNN-based models outperform other families of RSs under data minimization conditions?*
- **RQ2.** *Which data characteristics most impact the performance of GNN-based RSs in data minimization scenarios?*
- **RQ3.** *Are the most influential characteristics specific to the minimization strategies applied?*
- **RQ4.** *Can we augment the set of characteristics to reflect the criteria and retention levels used in data minimization?*

The main contributions of this chapter are as follows:

- We provide an extensive comparison of GNN-based and non-GNN recommender systems across multiple minimization strategies. Our results show that GNN-based models consistently exhibit superior robustness to data minimization, with performance advantages that are especially pronounced in highly reduced settings.
- Through an explanatory regression model, we demonstrate that a small set of global structural characteristics of the minimized data dominates the performance of GNN-based recommenders.
- We demonstrate that the factors driving performance under data minimization are substantially different from those that emerge under random reduction, highlighting that the significance of these factors is not universal, but is instead a direct consequence of the semantic criteria guiding the minimization process.
- We introduce and validate a novel set of semantic features that capture the logic of the minimization strategies. We show that integrating these features significantly improves our model's explanatory power, providing practical insight for designing more effective minimization strategies.

7.2 Data Minimization in Recommender Systems

The principle of *data minimization* advocates for learning effective models from the smallest possible amount of personal or behavioral data, aligning with modern

requirements of privacy preservation and resource efficiency. In recommender systems, data minimization can be operationalized as the selective reduction of the available user–item interactions used for model training, under the constraint of maintaining acceptable recommendation quality.

Formally, let $\mathcal{D}$ denote the original interaction set, typically partitioned into training, validation, and test subsets. A data minimization process transforms the training portion $\mathcal{D}_{\text{train}}$ into a reduced dataset $\mathcal{D}_{\text{train}}^{\alpha, \beta}$ according to two dimensions:

- **Minimization strategy** (α): defines the criterion or semantic principle used to determine which interactions are retained. Examples include strategies that prioritize recent interactions, high-rated items, or diverse preferences.
- **Retention level** (β): determines the number of interactions preserved per user, thus controlling the extent of the reduction applied to the dataset.

Each combination (α, β) induces a transformation of the empirical interaction distribution $p_{\mathcal{D}}(u, i)$ into a minimized distribution $p_{\mathcal{D}_{\text{train}}^{\alpha, \beta}}(u, i)$. This transformation alters the structural and semantic properties of the data — such as user activity, item popularity, and temporal balance — which in turn influence the learned representations and the resulting recommendation performance.

7.3 Dataset Characteristics

In this section, we introduce the dataset characteristics employed to explain the variations observed in the performance of GNN-based recommender models when trained under data minimization conditions. Prior studies [4, 68] have demonstrated that *structural* and *distributional* characteristics derived from the user–item interaction matrix exhibit a significant and well-established impact on recommendation performance. Furthermore, Malitesta et al. [161] have shown that specific *topological* properties of the user–item bipartite graph are effectively leveraged by GNN-based models to enhance recommendation quality. To ensure consistency with these findings, our study incorporates both structural and topological characteristics.

However, the empirical evidence that we will present in Section 7.6.1 indicates that GNN-based models tend to outperform other recommendation paradigms when trained on minimized datasets. Since different minimization strategies reduce the dataset according to distinct semantic criteria, we hypothesize that GNNs are capable of capturing and exploiting the underlying semantics of these minimization processes. To test this hypothesis, we extend the traditional set of data characteristics by introducing a novel family of *minimization-semantic* features that reflect the semantics of the minimization criteria adopted to produce the reduced datasets.

As a result, we categorize the dataset characteristics into three main families:

1. **Structural and distributional characteristics**, describing the fundamental properties of the user–item interaction matrix;

2. **Topological characteristics**, capturing higher-order connectivity patterns in the corresponding bipartite graph;
3. **Minimization-Semantic characteristics**, quantifying the semantic effects induced by the specific data minimization criteria.

The following subsections formally define all the dataset characteristics considered in this work, organized according to the three aforementioned categories. Furthermore, we discuss the trend that these characteristics should assume under data minimization conditions. In the rest of the section, we assume a recommendation dataset shaping a user-item interaction matrix (UIM) composed of $|\mathcal{U}|$ users, $|\mathcal{I}|$ items, and $|\mathcal{R}|$ user-item interactions.

7.3.1 Structural and distributional characteristics

Structural and distributional characteristics summarize the overall shape and sparsity of the UIM. They quantify to what extent the interactions are distributed across users and items, providing insight into the size, density, and balance of the dataset, as well as the inequality of user and item activity levels.

Space Size. This measure captures the potential interaction space of the dataset, i.e., the total number of possible user-item pairs. It reflects the upper bound of the matrix dimensionality.

Definition 7.1 (Space Size). *We define the Space Size of a UIM as:*

$$SpaceSize = \frac{|\mathcal{U}| \cdot |\mathcal{I}|}{sc} \quad (7.1)$$

where the scaling factor sc is set to limit the range of $|\mathcal{U}| \cdot |\mathcal{I}|$.

In this work, we set $sc = 1000$ [4, 72].

In the context of data minimization, the space size metric reflects both the applied strategy and the number of retained interactions per user. Since the number of users remains constant, variations in space size arise from changes in the number of items, which depend on how and how many each minimization strategy preserves user-item interactions. For instance, a strategy discarding the less-rated items will produce a dataset with a smaller interaction space.

Shape. This metric characterizes the relative proportion between users and items in the dataset, indicating whether the user-item matrix is more “user-heavy” or “item-heavy.”

Definition 7.2 (Shape). *We define the Shape of a UIM as:*

$$Shape = \frac{|\mathcal{U}|}{|\mathcal{I}|}. \quad (7.2)$$

A shape value greater than one implies that the dataset contains more users than items, typically corresponding to scenarios with many consumers interacting with a limited catalog. Conversely, a shape value smaller than one indicates that the number of items exceeds the number of users, which is common in domains where users interact with a vast collection of contents.

Under data minimization, the shape metric always reflects a more “user-heavy” user-item matrix compared to the original dataset since the number of items decreases according to the number of retained interactions. Furthermore, it also mirrors the logic underlying the interaction retention criterion. For example, a strategy preserving the most-rated items will produce a dataset with higher shape.

Density. This characteristic measures the proportion of observed interactions relative to all possible user–item pairs. It quantifies how populated the UIM is, providing a direct indicator of dataset sparsity.

Definition 7.3 (Density). *We define the Density of a UIM as:*

$$\text{Density} = \frac{|\mathcal{K}|}{|\mathcal{U}| \cdot |\mathcal{I}|}. \quad (7.3)$$

Low-density matrices are common in recommendation datasets and often require models to generalize from limited observations.

Regarding data minimization, the user-item interaction matrix is denser for a higher number of retained interactions. Furthermore, it reflects the semantics of the minimization criterion. As an example, a strategy conserving the most popular items will produce a denser dataset. Although density may increase when fewer but more concentrated interactions are retained, this does not necessarily imply higher informational richness, as minimization may reduce behavioral variability.

Gini User and Gini Item. While the above metrics describe global structural properties, Gini coefficients capture inequality in user and item activity.

Let $\mathcal{N}_u^{(1)}$ and $\mathcal{N}_i^{(1)}$ denote the set of neighboring nodes for user u and item i at distance 1, respectively. Then, the node degree for u and i (i.e., $\sigma_u = |\mathcal{N}_u^{(1)}|$ and $\sigma_i = |\mathcal{N}_i^{(1)}|$) corresponds to the number of items and users directly connected with them. To capture inequality in the distribution of these degrees, we define the following measures.

Definition 7.4 (Gini User). *We define the Gini User of a UIM as:*

$$\text{Gini-U} = \frac{\sum_{u=1}^{|\mathcal{U}|-1} \sum_{v=u+1}^{|\mathcal{U}|} \text{abs}(\sigma_u - \sigma_v)}{|\mathcal{U}| \sum_{u=1}^{|\mathcal{U}|} \sigma_u}, \quad (7.4)$$

where $\text{abs}(\cdot)$ denotes the absolute value function.

A high Gini User value indicates that a few users account for most of the interactions (i.e., highly active users dominate the dataset), whereas a low value reflects a more balanced participation. Hence, the lower is the number of retained interactions, the lower is the Gini User.

Definition 7.5 (Gini Item). *We define the Gini Item of a UIM as:*

$$Gini-I = \frac{\sum_{i=1}^{|I|-1} \sum_{v=i+1}^{|I|} \text{abs}(\sigma_i - \sigma_v)}{|I| \sum_{i=1}^{|I|} \sigma_i}, \quad (7.5)$$

where $\text{abs}(\cdot)$ denotes the absolute value function.

Similarly, Gini Item quantifies how unevenly items are consumed across the catalog—higher values suggest a concentration of interactions on a small subset of popular items. Hence, strategies promoting more diverse items will have lower values of Gini Item.

Depending on the minimization criterion, inequality may be reduced on the item side (e.g., when preserving diverse contents) but simultaneously increase on the user side, as the remaining interactions are dominated by active users.

7.3.2 Topological characteristics

Topological characteristics capture higher-order relationships emerging from the bipartite structure of the UIM. They describe how users and items are interconnected through shared interactions, revealing clustering tendencies and assortative mixing patterns that may affect information diffusion and recommendation quality.

Average User and Item Degree. These metrics represent the average number of connections per user and per item, respectively.

Definition 7.6 (Average User Degree). *The average user node degree is defined as:*

$$AvgDeg-U = \frac{1}{|\mathcal{U}|} \sum_{u \in \mathcal{U}} |\mathcal{N}_u^{(1)}|. \quad (7.6)$$

A higher average user degree indicates that users generally interact with many items (broad profiles). Therefore, the average user degree is a clear indicator of the number of retained interactions in the context of data minimization.

Definition 7.7 (Average Item Degree). *The average item node degree is defined as:*

$$AvgDeg-I = \frac{1}{|I|} \sum_{i \in I} |\mathcal{N}_i^{(1)}|. \quad (7.7)$$

A higher average item degree suggests that items are widely consumed across users. Indeed, this characteristic mirrors the criteria utilized for discarding the interactions in data minimization.

Average Clustering Coefficient. The clustering coefficient measures the extent to which nodes in the projected user or item graph tend to form tightly connected groups. These properties are related to redundancy and community structure in the interaction network.

Let v and w be two nodes belonging to the same partition (e.g., user nodes). Their similarity is computed as the ratio between the intersection and the union of their neighborhoods. By evaluating this measure for each node, we define:

$$\gamma_v = \frac{\sum_{w \in \mathcal{N}_v^{(2)}} \gamma_{v,w}}{|\mathcal{N}_v^{(2)}|}, \quad \text{with} \quad \gamma_{v,w} = \frac{|\mathcal{N}_v^{(1)} \cap \mathcal{N}_w^{(1)}|}{|\mathcal{N}_v^{(1)} \cup \mathcal{N}_w^{(1)}|}, \quad (7.8)$$

where $\mathcal{N}_v^{(2)}$ denotes the set of second-order neighbors of node v . In a bipartite graph, second-order neighborhoods are used since nodes within the same partition can only be connected through paths of even length.

Definition 7.8 (Average User Clustering Coefficient). *The average user clustering coefficient is defined as:*

$$AvgClustC-U = \frac{1}{|\mathcal{U}|} \sum_{u \in \mathcal{U}} \gamma_u, \quad (7.9)$$

In recommendation settings, a high user clustering coefficient implies that users with similar preferences are densely interconnected through shared items. Then, it can be expected that for a smaller number of retained interactions, users more probably share the same items, leading to low values of user clustering coefficient.

Definition 7.9 (Average Item Clustering Coefficient). *The average item clustering coefficient is defined as:*

$$AvgClustC-I = \frac{1}{|\mathcal{I}|} \sum_{i \in \mathcal{I}} \gamma_i. \quad (7.10)$$

A high item clustering coefficient suggests that items are frequently co-interacted by the same users. Hence, for a smaller number of retained interactions, this characteristic can importantly reflect the minimization strategy semantic.

While lower numbers of retained interactions typically reduce overall clustering, some minimization criteria can lead to higher local clustering if they preserve interactions among highly similar users or tightly connected items.

Degree Assortativity. Degree assortativity measures the tendency of nodes with similar degrees to connect with each other. Positive assortativity indicates that high-degree nodes tend to link with other high-degree nodes (homophily), while negative values suggest that highly connected nodes are preferentially linked to less connected ones (disassortativity).

Let $\mathcal{D} = \{d_1, d_2, \dots\}$ be the set of unique node degrees in the projected graph. Let e_{d_h, d_k} denote the fraction of edges that connect nodes of degrees d_h and d_k , and

let q_{d_h} be the probability of reaching a node of degree d_h by following a randomly selected edge (i.e., the *excess degree distribution*). The degree assortativity coefficient is then defined as:

$$\rho = \frac{\sum_{d_h, d_k} d_h d_k (e_{d_h, d_k} - q_{d_h} q_{d_k})}{\text{std}_q^2}, \quad (7.11)$$

where std_q denotes the standard deviation of the degree distribution q . Conceptually, ρ behaves similarly to a correlation coefficient (e.g., the Pearson correlation), indicating whether nodes with similar degrees tend to be connected.

Following the same rationale as for the clustering coefficient, we project the bipartite user–item graph into two homogeneous graphs—one for users ($\mathcal{G}_U$) and one for items ($\mathcal{G}_I$)—and compute the degree assortativity coefficient for each of them.

Definition 7.10 (User Degree Assortativity). *Let $\mathcal{G}_U$ be the user–user projection of the bipartite user–item graph. The User Degree Assortativity coefficient is defined as:*

$$\text{Assort-U} = \frac{\sum_{d_h, d_k} d_h d_k (e_{d_h, d_k}^U - q_{d_h}^U q_{d_k}^U)}{(\text{std}_{q^U})^2}, \quad (7.12)$$

where e_{d_h, d_k}^U and $q_{d_h}^U$ are respectively the joint degree distribution and the excess degree distribution computed over $\mathcal{G}_U$.

User assortativity can reflect the coherence of user behavior patterns.

Definition 7.11 (Item Degree Assortativity). *Let $\mathcal{G}_I$ be the item–item projection of the bipartite user–item graph. The Item Degree Assortativity coefficient is defined as:*

$$\text{Assort-I} = \frac{\sum_{d_h, d_k} d_h d_k (e_{d_h, d_k}^I - q_{d_h}^I q_{d_k}^I)}{(\text{std}_{q^I})^2}, \quad (7.13)$$

where e_{d_h, d_k}^I and $q_{d_h}^I$ are respectively the joint degree distribution and the excess degree distribution computed over $\mathcal{G}_I$.

Item assortativity reveals whether popular items are co-consumed by the same groups of users.

In recommendation, assortativity quantifies the tendency of active users to interact with either popular or niche items. When data minimization is applied, this property reflects how the retained interactions reshape the correlation structure of the bipartite graph. Strategies that preserve highly popular or representative interactions generally increase assortativity, reinforcing concentration on a few items. Conversely, strategies that retain a more diverse or randomly sampled subset of interactions tend to reduce assortativity, favoring more exploratory and less correlated interaction patterns.

7.3.3 Minimization-Semantic Characteristics

The third family of characteristics captures the semantic criteria underlying the data minimization strategies applied during the reduction process. These measures quantify how the retained subsets differ in terms of users' preference centrality, exposure to item popularity, rating variability, and temporal recency of interactions.

Preference Centrality. This metric captures the central tendency of user preferences within the retained subset. It quantifies whether users' rated items are, on average, closer to their most or least favored ones, considering the global rating range.

Definition 7.12 (Preference Centrality). *We define the Preference Centrality of a UIM as:*

$$\text{PrefCent} = \frac{1}{|\mathcal{U}|} \sum_{u \in \mathcal{U}} \frac{\frac{1}{|\mathcal{I}_u|} \sum_{i \in \mathcal{I}_u} r_{ui} - r_{\min}}{r_{\max} - r_{\min}}, \quad (7.14)$$

where $r_{\min}$ and $r_{\max}$ denote the minimum and maximum observed ratings and $\mathcal{I}_u$ is the set of items rated by user $u \in \mathcal{U}$.

High values indicate that the retained interactions correspond to users' most appreciated items, while low values suggest that less preferred items have been preserved.

Popularity Bias. This measure reflects how strongly the retained interactions are biased toward popular items. It evaluates whether users' rated items tend to have high or low degrees of popularity, computed from the number of ratings each item has received.

Definition 7.13 (Popularity Bias). *We define the Popularity Bias of a UIM as:*

$$\text{PopBias} = \frac{1}{|\mathcal{U}|} \sum_{u \in \mathcal{U}} \frac{1}{|\mathcal{I}_u|} \sum_{i \in \mathcal{I}_u} \frac{d_i}{\max_{j \in \mathcal{I}} d_j}, \quad (7.15)$$

where d_i denotes the number of ratings received by item i and $\mathcal{I}_u$ is the set of items rated by user $u \in \mathcal{U}$.

High values indicate that the minimized dataset preserves interactions with well-known or frequently rated items, while low values correspond to subsets emphasizing niche or long-tail items.

Rating Variability Exposure. This characteristic quantifies the exposure of users to items that received heterogeneous evaluations. It captures how much variability is present in the items retained for each user, reflecting whether the minimization preserves interactions with controversial or consistently rated items.

Definition 7.14 (Rating Variability Exposure). *We define the Rating Variability Exposure of a UIM as:*

$$\text{VarExp} = \frac{1}{|\mathcal{U}|} \sum_{u \in \mathcal{U}} \frac{1}{|\mathcal{I}_u|} \sum_{i \in \mathcal{I}_u} \frac{\text{std}_i}{\Delta_r}, \quad (7.16)$$

where std_i denotes the standard deviation of the ratings received by item i , and $\Delta_r = \max_{u,i} r_{ui} - \min_{u,i} r_{ui}$ is the global rating range.

Higher values indicate subsets containing items with diverse opinions among users, whereas lower values correspond to more uniformly rated items.

Temporal Recency. This measure describes the average temporal freshness of the retained interactions. It indicates whether the minimization process focuses on more recent or older user activities.

Definition 7.15 (Temporal Recency). *We define the Temporal Recency of a UIM as:*

$$\text{TempRec} = \frac{1}{|\mathcal{U}|} \sum_{u \in \mathcal{U}} \frac{\bar{t}_u}{T_{\max}}, \quad (7.17)$$

where $\bar{t}_u = \frac{1}{|\mathcal{I}_u|} \sum_{i \in \mathcal{I}_u} t_{ui}$ is the mean timestamp of user u 's interactions, and $T_{\max}$ is the maximum timestamp in the dataset.

A higher value of this metric corresponds to subsets composed mainly of recent interactions, while lower values imply that older data points are more prevalent.

Characteristic Deviation. This metric quantifies how distinctive the retained items are with respect to the global interaction pattern. It measures the average deviation of each item's interaction vector from the overall interaction tendency, normalized to account for scale differences. High values of this metric indicate that the minimized dataset preserves interactions with items whose consumption patterns strongly differ from the average user behavior, thus emphasizing more idiosyncratic or niche content.

Definition 7.16 (Characteristic Deviation). *We define the Characteristic Deviation of a UIM as:*

$$\text{CharDev} = \frac{1}{|\mathcal{U}|} \sum_{u \in \mathcal{U}} \frac{1}{|\mathcal{I}_u|} \sum_{i \in \mathcal{I}_u} \frac{\|\mathbf{x}_i - \bar{\mathbf{x}}\|_2}{\max_{j \in \mathcal{I}} \|\mathbf{x}_j - \bar{\mathbf{x}}\|_2}, \quad (7.18)$$

where $\mathbf{x}_i$ denotes the interaction vector of item i across users, $\bar{\mathbf{x}}$ is the average item interaction vector, and $\|\cdot\|_2$ represents the Euclidean norm.

A higher value of Characteristic Deviation suggests that retained interactions concern items whose interaction patterns are more distant from the dataset's global trend. Under data minimization, this characteristic reflects whether the applied strategy tends to preserve typical or distinctive behavioral signals. For instance, strategies that focus on preserving representative or popular interactions will yield lower values of this metric, while those emphasizing niche or semantically diverse content will result in higher Characteristic Deviation.

7.4 Methodology

Our goal is to investigate how the characteristics of minimized datasets influence the performance of GNN-based recommender systems trained on them. Following prior studies with a similar objective, we employ a simple explanatory model to uncover dependencies between dataset characteristics and recommendation performance. In this model, dataset characteristics serve as explanatory variables, while the recommendation performance acts as the target variable.

To achieve this goal, we perform the following steps: (i) collect multiple minimized datasets by applying different minimization strategies and cutoff thresholds; (ii) compute their corresponding data characteristics; (iii) train and evaluate GNN-based recommendation models on each minimized dataset; and (iv) fit the explanatory model to capture relationships between data characteristics and observed performance outcomes.

7.4.1 Data collection

To train our explanatory model, we require a collection of samples that pair dataset characteristics with the corresponding recommendation performance of trained models. As in previous studies [4, 68, 72, 161], this sample pool is typically generated by manipulating a single recommendation dataset through controlled sampling strategies. For instance, Malitesta et al. [161] employed ad-hoc graph sampling techniques (e.g., node and edge dropout) to produce samples that preserve the structural properties of the original dataset.

In contrast, under data minimization scenarios, the training dataset is reduced by applying a minimization strategy that selects a subset of candidate user–item interactions according to a specific criterion [31]. To generate a sufficiently large and diverse sample pool, we adopt several minimization strategies, each retaining a different number of candidate items per user. We explore the steps in the following².

Formally, let $\mathcal{D}$ denote the full recommendation dataset, which we split into training ($\mathcal{D}_{\text{train}}$), validation ($\mathcal{D}_{\text{validation}}$), and test ($\mathcal{D}_{\text{test}}$) sets. We reduce $\mathcal{D}_{\text{train}}$ using a minimization strategy $\alpha \in \{s_1, s_2, \dots, s_m\}$, retaining β interactions per user, where $\beta \in \{r_1, r_2, \dots, r_n\}$. This process yields $m \times n$ minimized training datasets, denoted as $\mathcal{D}_{\text{train}}^{\alpha, \beta}$.

We first train a recommendation model g on $\mathcal{D}_{\text{train}}$, tuning its hyperparameters on $\mathcal{D}_{\text{validation}}$ to obtain an optimal configuration ζ . We then retrain g with ζ on each minimized dataset $\mathcal{D}_{\text{train}}^{\alpha, \beta}$ and evaluate its recommendation performance on $\mathcal{D}_{\text{test}}$ using a performance metric y . This produces a performance score $y^{\alpha, \beta}$ associated with each minimized dataset $\mathcal{D}_{\text{train}}^{\alpha, \beta}$. Each minimized dataset constitutes the input for the next steps.

2. Full details of the experimental setup and minimization strategies are provided in Section 7.5.

7.4.2 Target Variable

The recommendation performance $y^{\alpha,\beta}$ represents the dependent variable of the explanatory framework. In this work, we measure the performance of the recommendation with the recall metric to have a direct comparison of the results from [161].

Definition 7.17. Given the top- k recommendation list $\mathcal{L}_u^{(1,\dots,k)}$ for a user u and the set of relevant items in the test set $\mathcal{I}_u^{+,test}$, the recall ($Recall@k$) is the average of the proportion of relevant items retrieved in the top- k of each user:

$$Recall@k = \frac{1}{|\mathcal{U}|} \sum_{u \in \mathcal{U}} \frac{|\mathcal{L}_u^{(1,\dots,k)} \cap \mathcal{I}_u^{+,test}|}{|\mathcal{I}_u^{+,test}|}. \quad (7.19)$$

7.4.3 Explanatory Variables

After collecting the minimized training datasets $\mathcal{D}_{train}^{\alpha,\beta}$ for each minimization strategy α and cutoff β , along with the corresponding recommendation performance $y^{\alpha,\beta}$ of model g , we compute their data characteristics. As detailed in Section 7.3, these characteristics encompass three complementary categories: *classical* [4], *topological* [161], and *strategy semantic-based* features.

Prior studies [4, 161] have demonstrated that both classical and topological characteristics exert a significant influence on the behavior and performance of collaborative filtering models, including GNN-based recommenders. In the context of data minimization, however, models are trained on datasets intentionally reduced according to specific selection criteria. We therefore hypothesize that characteristics encoding the semantics of such minimization criteria may further affect recommendation outcomes. To capture this aspect, we have introduced an additional category of features in our analysis. These characteristics, referred to as semantic strategy-based characteristics, should describe how the adopted minimization strategy shapes the retained data.

Furthermore, following prior works, we apply a base-10 logarithmic normalization (*log10-scale*) to selected characteristics in order to align their magnitudes and stabilize the training of the explanatory model. In addition, the dataset characteristics are mean-centered. Table 7.1 summarizes the notation used to denote each characteristic, where the prefix “log” indicates that the corresponding feature has been normalized using the log10-scale transformation. The outcome of this step process is the set of samples $\{ch^{\alpha,\beta}, y^{\alpha,\beta}\}$, where $ch^{\alpha,\beta} = \{ch_1^{\alpha,\beta}, ch_2^{\alpha,\beta}, \dots, ch_C^{\alpha,\beta}\}$ denotes the set of data characteristics corresponding to $\mathcal{D}_{train}^{\alpha,\beta}$ for each minimization strategy α and number of retained interactions β . This set is used to fit the explanatory model.

Table 7.1. Summary of the dataset characteristics analyzed in this work. Each metric is associated with its notation and a concise description of what it quantifies.

Characteristic	Notation	Description
Structural characteristics		
Space Size	$SpaceSize_{log}$	Reflects the overall number of potential user–item pairs, normalized by a scaling factor.
Shape	$Shape_{log}$	Captures the proportion between users and items, indicating dataset balance.
Density	$Density_{log}$	Fraction of observed interactions over all possible user–item pairs.
Gini User	$Gini-U$	Measures inequality in users’ activity levels (interaction counts).
Gini Item	$Gini-I$	Measures inequality in items’ popularity distribution.
Topological characteristics		
Average User Degree	$AvgDeg-U_{log}$	Mean number of items interacted with per user.
Average Item Degree	$AvgDeg-I_{log}$	Mean number of users interacting with each item.
Average User Clustering Coefficient	$AvgClustC-U_{log}$	Measures the local cohesiveness among users in the projected graph.
Average Item Clustering Coefficient	$AvgClustC-I_{log}$	Measures the local cohesiveness among items in the projected graph.
User Degree Assortativity	$Assort-U$	Indicates correlation between users’ degrees in the user–user projection.
Item Degree Assortativity	$Assort-I$	Indicates correlation between items’ degrees in the item–item projection.
Minimization-semantic characteristics		
Preference Centrality	$PrefCent$	Reflects whether retained interactions correspond to users’ most or least favored items.
Popularity Bias	$PopBias$	Quantifies how much the retained interactions favor popular items.
Rating Variability Exposure	$VarExp$	Measures the exposure to items receiving diverse (highly variable) ratings.
Temporal Recency	$TempRec$	Indicates the average recency of user interactions after minimization.
Characteristic Deviation	$CharDev$	Measures how distinctive the retained items are compared to the overall interaction patterns in the dataset.

7.4.4 Explanatory Model

Finally, we aim to fit an explanatory model on the set $\{ch^{\alpha,\beta}, y^{\alpha,\beta}\}$. Specifically, we seek to understand whether the underlying effects measured by the dataset characteristics $ch^{\alpha,\beta}$ of the reduced training set $\mathcal{D}_{train}^{\alpha,\beta}$ are the cause of the underlying effects measured by the model performance $y^{\alpha,\beta}$. To support the “interpretability” of the results and following prior works [4, 68, 72, 161], we employ a linear regression

framework to model this relationship. Formally:

$$y^{\alpha,\beta} = \epsilon^{\alpha,\beta} + \theta_0 + \sum_{c=1}^C \theta_c ch_c^{\alpha,\beta}, \quad (7.20)$$

where C is the number of data characteristics, θ_c is the regression coefficient of the explanatory variable $ch_c^{\alpha,\beta} \in ch^{\alpha,\beta}$ (i.e., the data characteristic), computed for the minimization strategy α and the number of retained interactions β , and $y^{\alpha,\beta}$ is the performance metric value of the model g trained in $\mathcal{D}_{\text{train}}^{\alpha,\beta}$. Considering all possible combinations of α and β , we obtain $ch = \{ch^{\alpha,\beta} | \forall \alpha \in \{s_1, s_2, \dots, s_m\}, \forall \beta \in \{r_1, r_2, \dots, r_n\}\}$, with $ch \in \mathbb{R}^{(m+n) \times C}$ and $y = \{y^{\alpha,\beta} | \forall \alpha \in \{s_1, s_2, \dots, s_m\}, \forall \beta \in \{r_1, r_2, \dots, r_n\}\}$, with $y \in \mathbb{R}^{(m+n)}$. Then, a more compact formulation is the following:

$$y = \epsilon + \theta_0 + \theta ch, \quad (7.21)$$

where $\theta = \{\theta_1, \dots, \theta_C\}$ is the vector of regression coefficients associated with each data characteristic. Under the assumption of mean-centered data, θ_0 represents the expected value of y , i.e., the expected recommendation performance. We optimize the regression coefficients through the Ordinary Least Squares (OLS) model:

$$(\theta_0^*, \theta^*) = \min_{\theta_0, \theta} \frac{1}{2} \|y - \theta_0 - \theta ch\|_2^2. \quad (7.22)$$

To methodically validate the relationships between the recommendation performance and the dataset characteristics, we compute the *coefficient of determination* R^2 , which measures the proportion of variance in the dependent variable y that is explained by the explanatory variables ch . This coefficient values range from 0 to 1, where a higher R^2 value indicates that the dataset characteristics account for a larger fraction of the variability in the model performance $y^{\alpha,\beta}$, thus reflecting a stronger explanatory capacity of the regression model. To account for the number of explanatory variables and avoid an overestimation of model fit, we also report the *adjusted coefficient of determination* R_{adj}^2 . In addition to the overall model fit, we analyze the *significance* of each estimated regression coefficient θ_c to determine which dataset characteristics have a statistically reliable impact on the model performance $y^{\alpha,\beta}$. Following standard regression analysis practice, we compute the p -value associated with each coefficient under the null hypothesis $H_0 : \theta_c = 0$. A low p -value (typically $p < 0.05$) indicates that the corresponding characteristic $ch_c^{\alpha,\beta}$ has a statistically significant contribution to explaining variations in $y^{\alpha,\beta}$, whereas higher p -values suggest that its effect may not be distinguishable from noise. Furthermore, we examine the *directionality* of the measured coefficients to interpret the qualitative effect of each data characteristic on model performance. A positive coefficient $\theta_c > 0$ implies that higher values of $ch_c^{\alpha,\beta}$ are associated with higher recommendation performance, while a negative coefficient $\theta_c < 0$ indicates an inverse relationship. By jointly analyzing the significance and directionality of the regression coefficients, we can identify which dataset characteristics positively or negatively influence the effectiveness of the model g under data minimization conditions.

Table 7.2. Key statistics of datasets.

Dataset	Users	Items	Ratings	Density (%)
Amazon-Book	14407	13164	601992	0.3174
Yelp	27999	22323	1389662	0.2223

7.5 Experimental Setup

To rigorously investigate the effects of data minimization on recommender systems, we designed a comprehensive experimental framework. This framework allows us to first establish the relative robustness of graph-based models and then conduct a deep, fine-grained analysis of their behavior under various data minimization strategies and topological configurations.

7.5.1 Datasets and Preprocessing

We selected two widely-adopted benchmark datasets from different domains to ensure the generalizability of our findings: *Amazon-Book*³, a large-scale product recommendation dataset, and *Yelp*⁴, a dataset from the business recommendation domain.

To study data minimization as a controlled intervention on user histories, we first ensure that the starting graphs provide sufficient interaction support. We first performed an iterative 20-core filtering on both datasets. This process removes all users and items with fewer than 20 interactions until no such users or items remain. This step is critical for isolating the variable of interest: data minimization. By beginning with a dense graph, we mitigate the confounding effects of extreme native sparsity often present in raw datasets. This ensures that any observed performance degradation is a direct consequence of our controlled interaction removal, rather than a pre-existing condition of the data, thereby establishing a reliable analytical baseline.

Subsequently, we partitioned the interaction data for each user into training (70%), validation (10%), and test (20%) sets. Unlike a global random split, this user-centric protocol guarantees that every user appearing in validation/test also appears in training, preserving the continuity of user profiles across partitions. This choice lets us attribute performance changes to the interactions reduction (data minimization) rather than to artifacts of the split (e.g., users present only in evaluation but absent from training), thus enabling a fair assessment of personalization under controlled data availability. The final statistics of the preprocessed datasets used in our experiments are detailed in Table 7.2.

3. https://mcauleylab.ucsd.edu/public_datasets/data/amazon_2023/benchmark/5score/rating_only/Books.csv.gz

4. <https://business.yelp.com/data/resources/open-dataset/>

7.5.2 *Models Under Investigations*

To address our research questions, we selected a diverse suite of eight models, categorized into two families based on their underlying architecture.

Graph-based Models. These models explicitly leverage the user-item graph structure to capture high-order collaborative signals, which is hypothesized to enhance robustness against data sparsity. We include three state-of-the-art representatives:

- **LightGCN** [115]: A simplified and powerful Graph Convolutional Network (GCN) that removes non-linearities and feature transformations, retaining only the core neighborhood aggregation for collaborative filtering. It has become a standard and highly effective baseline in graph-based recommendation
- **UltraGCN** [167]: An advanced GNN that further improves efficiency by forgoing the explicit message-passing paradigm. It directly approximates infinite-layer graph convolutions via a dedicated constraint loss, significantly accelerating training while maintaining strong performance.
- **GFCF** [218]: A model based on Graph Signal Processing (GSP) that treats recommendation as a signal denoising task. It applies graph filters to smooth user preferences across the graph, offering a distinct theoretical foundation from standard GCNs.

Non-graph-based Models. This group comprises strong and representative baselines from established recommendation paradigms, serving to benchmark the performance of graph-based approaches.

- **UserKNN** [198] - **ItemKNN** [212]: Classic k-Nearest Neighbors (k-NN) algorithms that infer preferences from the most similar items or users, respectively.
- **BPRMF** [140]: A seminal matrix factorization model optimized with the Bayesian Personalized Ranking (BPR) pairwise loss. It is a standard and highly influential baseline for implicit feedback scenarios.
- **DirectAU** [249]: Optimizes recommendation performance by explicitly promoting *alignment* between users and their interacted items and *uniformity* in the distribution of embeddings. The training objective is constructed to directly enforce this trade-off, enabling DirectAU to attain strong ranking accuracy even with a relatively simple encoder.
- **SimpleX** [165]: Introduces a simplified framework that explicitly disentangles user preferences from item characteristics. It defines a scoring function that incorporates the importance of items a user has previously interacted with and couples this with a contrastive learning objective.

7.5.3 Hyperparameter Optimization

To ensure a fair comparison, hyperparameters for all models were meticulously tuned on the full, non-minimized training datasets. We employed a Bayesian optimization approach using the Tree-structured Parzen Estimator (TPE) algorithm, conducting 10 trials per model. Search spaces were constructed in close alignment with the ranges and configurations recommended in the respective original publications. To prevent overfitting, we employed an early stopping mechanism with a patience of 10 epochs, ultimately selecting the hyperparameter configuration that yielded the highest Recall@20 on the validation set.

Crucially, the optimal set of hyperparameters, once found, was fixed for all subsequent experiments on both the minimized and the sub-sampled datasets. This strict methodology ensures that any observed performance differences are attributable solely to the effects of data minimization, rather than variations in model tuning.

7.5.4 Data Minimization Strategies

To simulate varying levels of data availability, we apply a set of structured data minimization strategies to each user’s training data. We adopt the comprehensive framework proposed by Biega et al. [31], which operationalizes minimization via simple heuristics that are computationally lightweight and reproducible, yet expressive enough to capture complementary facets of user evidence. For each user, a given strategy selects a subset of n interactions from their training history, where we explore a wide range of minimization levels by setting $\beta \in \{1, 3, 6, 9, 12, 15, 18, 50, 100\}$. The strategies are defined as follows:

Full Data. This is the baseline condition representing no data minimization. It utilizes the complete user training history (the 70% partition) and serves as the upper-bound performance reference against which all minimization strategies are compared.

Random. This strategy uniformly samples β interactions at random from each user’s training history. While simple, it serves as a crucial and robust baseline. Its strength lies in its neutrality; by sampling randomly, it tends to preserve the average statistical properties of a user’s original taste profile, making it a difficult benchmark to outperform.

Most Recent. This strategy selects the β most recent user interactions, ordered by timestamp. The underlying assumption is that a user’s present interests are better predicted by their most recent activities, reflecting a temporal dynamic in user preferences.

Most and Least Favorite. These two strategies select items based on explicit preference signals. *Most Favorite* retains the β items with the highest ratings provided by the user, isolating the strongest positive signals in their history. Conversely, *Least*

Favorite retains the β items with the lowest ratings, allowing us to investigate the utility of negative preference signals in model training under data scarcity.

Most Rated. This strategy selects, for each user, the β items from their history that are the most popular globally (i.e., rated most frequently by all users in the training set). This method directly tests the impact of popularity bias, creating user profiles that are heavily skewed towards mainstream items.

Most Characteristic. This strategy aims to retain items that are most representative of mainstream behavior across the entire system. To achieve this, each item is represented by a binary vector indicating which users have interacted with it. A system-wide average item profile vector is then computed. The strategy selects, for each user, the n items from their history that have the smallest Euclidean distance to this average profile. The intuition is to retain interactions that align with the platform’s “average” user, effectively filtering out niche or atypical preferences.

Highest Variance. This strategy prioritizes items that are most discriminative of user preferences, identified by a high variance in their global ratings. It selects, for each user, the β items from their history that have the highest rating variance, calculated globally. The assumption is that high-variance items are more informative and that retaining these signals may be more beneficial for learning nuanced and robust user representations.

7.6 Results

7.6.1 Performance of GNN-based Models under Data Minimization (RQ1)

To address the research question “*Do GNN-based models outperform other families of RSs under data minimization conditions?*”, we analyze the results reported in Tables 7.3 and 7.4, which present Recall values for a variety of recommendation models across two datasets (Yelp and Amazon Book) under multiple data minimization strategies, with varying numbers of retained interactions per user ($\beta \in \{1, 3, 6, 9, 12, 15, 18, 50, 100\}$).

Our empirical evaluation across the Yelp and Amazon-Book datasets reveals that GNN-based RSs—specifically LightGCN, UltraGCN, and GFCF—tend to achieve superior performance compared to traditional non-graph-based models such as UserKNN, ItemKNN, BPRMF, SimpleX, and DirectAU, particularly under more extreme data minimization scenarios (i.e., small β values).

On the Yelp dataset (Table 7.3), one of the GNN models achieve the highest (or second-to-highest) Recall in 95% of the 63 strategy/ β combinations (7 strategies $\times$ 9 β values), with bold markings indicating outright superiority over all non-graph models in 86% of cases and underlined markings (superior to all but one) in 9,5%. For instance, under the Random strategy, UltraGCN dominates across all β , with Recall

Table 7.3. Performance results in terms of Recall of all models on the Yelp dataset across different minimization strategies and values of β . Bold values indicate that a GNN-based model outperforms all non-GNN models for a given strategy and β . Underlined values indicate that a GNN-based model outperforms all but one non-GNN model.

	Model	$n = 1$	$n = 3$	$n = 6$	$n = 9$	$n = 12$	$n = 15$	$n = 18$	$n = 50$	$n = 100$
Random	UserKNN	0.0011	0.0208	0.0444	0.0583	0.0659	0.0757	0.0833	0.1007	0.1037
	ItemKNN	0.0013	0.0138	0.0163	0.0268	0.0406	0.0554	0.0666	0.1020	0.1041
	BPRMF	0.0051	0.0079	0.0402	0.0558	0.0762	0.0839	0.0885	0.0977	0.0989
	SimpleX	0.0012	0.0250	0.0470	0.0633	0.0743	0.0840	0.0889	0.1031	0.1047
	DirectAU	0.0022	0.0254	0.0478	0.0644	0.0753	0.0830	0.0883	0.0942	0.0958
	GFCF	0.0054	0.0155	0.0224	0.0541	0.0836	0.0987	0.1055	0.1184	0.1197
	UltraGCN	0.0145	0.0421	0.0632	0.0761	0.0862	0.0948	0.1016	0.1168	0.1187
	LightGCN	0.0053	0.0332	0.0602	0.0692	0.0807	0.0885	0.0942	0.1074	0.1108
Most Recent	UserKNN	0.0013	0.0199	0.0413	0.0539	0.0630	0.0729	0.0808	0.1007	0.1040
	ItemKNN	0.0013	0.0138	0.0174	0.0277	0.0403	0.0538	0.0639	0.1009	0.1045
	BPRMF	0.0039	0.0253	0.0356	0.0511	0.0727	0.0801	0.0842	0.0969	0.0997
	SimpleX	0.0011	0.0229	0.0425	0.0581	0.0712	0.0804	0.0871	0.1017	0.1050
	DirectAU	0.0014	0.0243	0.0447	0.0582	0.0706	0.0779	0.0849	0.0967	0.0978
	GFCF	<u>0.0038</u>	0.0154	0.0226	0.0501	0.0773	0.0941	0.1020	0.1187	0.1203
	UltraGCN	0.0098	0.0134	0.0574	0.0699	0.0809	0.0912	0.0974	0.1169	0.1187
	LightGCN	<u>0.0037</u>	0.0298	0.0537	0.0656	0.0777	0.0867	0.0923	0.1084	0.1099
Most Favorite	UserKNN	0.0016	0.0253	0.0430	0.0536	0.0653	0.0757	0.0831	0.1010	0.1034
	ItemKNN	0.0008	0.0155	0.0205	0.0302	0.0447	0.0588	0.0698	0.1019	0.1046
	BPRMF	0.0031	0.0290	0.0380	0.0625	0.0732	0.0817	0.0861	0.0976	0.0989
	SimpleX	0.0012	0.0276	0.0486	0.0635	0.0767	0.0838	0.0892	0.1036	0.1059
	DirectAU	0.0016	0.0260	0.0468	0.0608	0.0732	0.0809	0.0863	0.0961	0.0961
	GFCF	<u>0.0028</u>	0.0176	0.0294	0.0563	0.0807	0.0953	0.1031	0.1186	0.1200
	UltraGCN	0.0059	0.0338	0.0520	0.0682	0.0820	0.0923	0.0999	0.1169	0.1181
	LightGCN	0.0034	0.0326	0.0505	0.0661	0.0779	0.0875	0.0937	0.1077	0.1106
Least Favorite	UserKNN	0.0013	0.0189	0.0406	0.0540	0.0619	0.0731	0.0815	0.0999	0.1035
	ItemKNN	0.0016	0.0142	0.0188	0.0292	0.0419	0.0558	0.0661	0.1002	0.1041
	BPRMF	0.0032	0.0244	0.0339	0.0623	0.0715	0.0786	0.0842	0.0986	0.0994
	SimpleX	0.0011	0.0220	0.0429	0.0592	0.0707	0.0811	0.0869	0.1015	0.1046
	DirectAU	0.0014	0.0232	0.0450	0.0600	0.0713	0.0792	0.0855	0.0952	0.0971
	GFCF	0.0034	0.0152	0.0251	0.0535	0.0776	0.0938	0.1018	0.1188	0.1200
	UltraGCN	0.0082	0.0102	0.0521	0.0685	0.0801	0.0916	0.0983	0.1168	0.1184
	LightGCN	<u>0.0032</u>	0.0263	0.0499	0.0650	0.0749	0.0852	0.0909	0.1069	0.1103
Most Rated	UserKNN	0.0028	0.0521	0.0641	0.0709	0.0762	0.0816	0.0864	0.1001	0.1034
	ItemKNN	0.0035	0.0556	0.0662	0.0734	0.0779	0.0807	0.0851	0.1022	0.1043
	BPRMF	0.0084	0.0591	0.0698	0.0760	0.0799	0.0822	0.0866	0.0969	0.0996
	SimpleX	0.0052	0.0561	0.0691	0.0784	0.0856	0.0921	0.0959	0.1048	0.1057
	DirectAU	0.0044	0.0537	0.0666	0.0737	0.0789	0.0825	0.0854	0.0962	0.0973
	GFCF	0.0087	<u>0.0584</u>	0.0681	0.0743	<u>0.0801</u>	<u>0.0863</u>	<u>0.0925</u>	0.1155	0.1195
	UltraGCN	0.0140	<u>0.0590</u>	<u>0.0698</u>	<u>0.0776</u>	<u>0.0831</u>	<u>0.0887</u>	<u>0.0937</u>	0.1126	0.1174
	LightGCN	<u>0.0082</u>	<u>0.0567</u>	<u>0.0676</u>	<u>0.0767</u>	<u>0.0848</u>	<u>0.0899</u>	<u>0.0944</u>	0.1074	0.1113
Most Char.	UserKNN	0.0005	0.0102	0.0194	0.0296	0.0431	0.0571	0.0681	0.1006	0.1042
	ItemKNN	0.0003	0.0106	0.0206	0.0314	0.0432	0.0531	0.0634	0.0982	0.1033
	BPRMF	0.0004	0.0117	0.0197	0.0281	0.0401	0.0539	0.0656	0.0947	0.0992
	SimpleX	0.0004	0.0118	0.0216	0.0305	0.0425	0.0553	0.0672	0.0984	0.1045
	DirectAU	0.0004	0.0118	0.0215	0.0307	0.0417	0.0562	0.0669	0.0938	0.0967
	GFCF	0.0004	0.0109	0.0217	0.0349	0.0498	0.0677	0.0840	0.1175	0.1205
	UltraGCN	0.0003	0.0004	0.0230	0.0340	0.0481	0.0656	0.0785	0.1143	0.1173
	LightGCN	0.0004	0.0116	0.0214	<u>0.0309</u>	0.0433	0.0599	0.0741	0.1067	0.1105
Highest Variance	UserKNN	0.0010	0.0164	0.0337	0.0463	0.0563	0.0671	0.0747	0.0993	0.1033
	ItemKNN	0.0009	0.0157	0.0253	0.0367	0.0465	0.0550	0.0650	0.0988	0.1031
	BPRMF	0.0010	0.0198	0.0355	0.0471	0.0581	0.0696	0.0771	0.0954	0.0992
	SimpleX	0.0008	0.0192	0.0362	0.0507	0.0625	0.0732	0.0808	0.1018	0.1047
	DirectAU	0.0007	0.0193	0.0356	0.0502	0.0605	0.0714	0.0779	0.0943	0.0958
	GFCF	0.0009	0.0166	0.0315	0.0512	0.0675	0.0806	0.0912	0.1174	0.1201
	UltraGCN	0.0033	0.0054	0.0391	0.0539	0.0667	0.0781	0.0876	0.1159	0.1184
	LightGCN	0.0009	0.0188	<u>0.0360</u>	0.0509	0.0638	0.0758	0.0830	0.1059	0.1096

Table 7.4. Performance results in terms of Recall of all models on the Amazon-Book dataset across different minimization strategies and values of β . Bold values indicate that a GNN-based model outperforms all non-GNN models for a given strategy and β . Underlined values indicate that a GNN-based model outperforms all but one non-GNN model.

	Model	$n = 1$	$n = 3$	$n = 6$	$n = 9$	$n = 12$	$n = 15$	$n = 18$	$n = 50$	$n = 100$
Random	UserKNN	0.0021	0.0257	0.0646	0.0927	0.1137	0.1329	0.1456	0.1725	0.1747
	ItemKNN	0.0016	0.0188	0.0380	0.0725	0.1077	0.1352	0.1496	0.1846	0.1856
	BPRMF	0.0075	0.0123	0.0746	0.1008	0.1178	0.1321	0.1354	0.1533	0.1583
	SimpleX	0.0017	0.0295	0.0642	0.0915	0.1162	0.1339	0.1447	0.1664	0.1722
	DirectAU	0.0026	0.0300	0.0682	0.0965	0.1171	0.1330	0.1411	0.1572	0.1586
	GFCF	0.0079	0.0212	0.0440	0.0944	0.1308	0.1524	0.1633	0.1903	0.1951
	UltraGCN	0.0211	0.0214	0.0776	0.1114	0.1375	0.1571	0.1686	0.1947	0.1999
	LightGCN	0.0076	0.0295	<u>0.0738</u>	<u>0.0987</u>	0.1161	0.1286	0.1409	0.1649	0.1672
Most Recent	UserKNN	0.0020	0.0309	0.0717	0.1021	0.1248	0.1424	0.1534	0.1742	0.1748
	ItemKNN	0.0021	0.0262	0.0579	0.0942	0.1241	0.1456	0.1582	0.1853	0.1852
	BPRMF	0.0044	0.0361	0.0783	0.1043	0.1228	0.1318	0.1415	0.1582	0.1602
	SimpleX	0.0026	0.0346	0.0734	0.1039	0.1263	0.1438	0.1509	0.1705	0.1728
	DirectAU	0.0022	0.0346	0.0760	0.1051	0.1243	0.1359	0.1436	0.1589	0.1599
	GFCF	0.0049	0.0277	0.0623	0.1064	0.1366	0.1564	0.1673	0.1912	0.1950
	UltraGCN	0.0046	0.0346	0.0790	0.1140	0.1399	0.1600	0.1711	0.1978	0.2003
	LightGCN	<u>0.0043</u>	0.0382	0.0760	0.1069	0.1268	0.1402	0.1463	0.1679	0.1702
Most Favorite	UserKNN	0.0014	0.0388	0.0816	0.1092	0.1290	0.1426	0.1526	0.1727	0.1751
	ItemKNN	0.0024	0.0331	0.0636	0.0975	0.1290	0.1497	0.1616	0.1866	0.1867
	BPRMF	0.0038	0.0445	0.0831	0.1075	0.1275	0.1378	0.1424	0.1593	0.1578
	SimpleX	0.0021	0.0418	0.0815	0.1110	0.1326	0.1452	0.1534	0.1677	0.1728
	DirectAU	0.0022	0.0418	0.0814	0.1089	0.1278	0.1389	0.1445	0.1586	0.1603
	GFCF	0.0040	0.0346	0.0713	0.1124	0.1426	0.1608	0.1701	0.1930	0.1954
	UltraGCN	0.0058	0.0106	0.0884	0.1211	0.1452	0.1618	0.1730	0.1966	0.2011
	LightGCN	<u>0.0037</u>	0.0446	<u>0.0817</u>	0.1090	<u>0.1310</u>	0.1430	0.1490	0.1667	0.1702
Least Favorite	UserKNN	0.0016	0.0242	0.0599	0.0876	0.1099	0.1300	0.1434	0.1719	0.1750
	ItemKNN	0.0018	0.0200	0.0417	0.0749	0.1052	0.1329	0.1492	0.1839	0.1855
	BPRMF	0.0046	0.0283	0.0638	0.0881	0.1080	0.1246	0.1351	0.1577	0.1614
	SimpleX	0.0022	0.0267	0.0585	0.0858	0.1114	0.1303	0.1410	0.1670	0.1717
	DirectAU	0.0023	0.0266	0.0615	0.0864	0.1097	0.1266	0.1362	0.1564	0.1569
	GFCF	0.0057	0.0210	0.0468	0.0891	0.1234	0.1478	0.1610	0.1917	0.1954
	UltraGCN	0.0121	0.0308	0.0697	0.0986	0.1262	0.1484	0.1628	0.1949	0.1987
	LightGCN	0.0053	0.0314	<u>0.0634</u>	0.0873	0.1089	0.1229	0.1396	0.1627	0.1696
Most Rated	UserKNN	0.0035	0.0623	0.0918	0.1113	0.1273	0.1408	0.1505	0.1712	0.1747
	ItemKNN	0.0051	0.0671	0.0995	0.1231	0.1397	0.1527	0.1621	0.1856	0.1871
	BPRMF	0.0120	0.0697	0.0983	0.1166	0.1284	0.1368	0.1400	0.1573	0.1592
	SimpleX	0.0070	0.0621	0.0968	0.1203	0.1380	0.1513	0.1584	0.1712	0.1716
	DirectAU	0.0048	0.0650	0.0975	0.1172	0.1321	0.1412	0.1458	0.1588	0.1580
	GFCF	0.0150	<u>0.0685</u>	0.0998	<u>0.1209</u>	0.1376	0.1509	<u>0.1598</u>	0.1890	0.1943
	UltraGCN	0.0211	0.0743	0.1042	0.1285	0.1449	0.1591	0.1695	0.1972	0.2013
	LightGCN	0.0128	<u>0.0680</u>	0.0964	0.1170	0.1321	0.1410	0.1498	0.1661	0.1686
Most Char.	UserKNN	0.0009	0.0210	0.0523	0.0833	0.1101	0.1323	0.1471	0.1750	0.1761
	ItemKNN	0.0009	0.0229	0.0515	0.0834	0.1137	0.1375	0.1534	0.1836	0.1851
	BPRMF	0.0011	0.0248	0.0521	0.0735	0.0929	0.1130	0.1296	0.1550	0.1590
	SimpleX	0.0012	0.0244	0.0525	0.0754	0.0970	0.1190	0.1323	0.1665	0.1679
	DirectAU	0.0009	0.0248	0.0545	0.0765	0.0971	0.1181	0.1312	0.1557	0.1580
	GFCF	<u>0.0012</u>	0.0236	0.0538	0.0848	0.1142	0.1414	0.1589	0.1924	0.1958
	UltraGCN	0.0011	0.0258	0.0584	0.0879	0.1148	0.1424	0.1596	0.1948	0.1995
	LightGCN	0.0010	0.0251	0.0506	0.0737	0.0916	0.1173	0.1341	0.1638	0.1675
Highest Variance	UserKNN	0.0011	0.0265	0.0540	0.0781	0.0991	0.1221	0.1370	0.1708	0.1748
	ItemKNN	0.0016	0.0214	0.0479	0.0751	0.1030	0.1280	0.1458	0.1830	0.1858
	BPRMF	0.0027	0.0298	0.0586	0.0812	0.0978	0.1173	0.1306	0.1576	0.1591
	SimpleX	0.0020	0.0285	0.0561	0.0808	0.1056	0.1254	0.1380	0.1683	0.1698
	DirectAU	0.0018	0.0257	0.0566	0.0807	0.1019	0.1212	0.1311	0.1560	0.1577
	GFCF	0.0031	0.0236	0.0554	0.0884	0.1155	0.1382	0.1543	0.1901	0.1950
	UltraGCN	0.0087	0.0116	0.0131	0.0928	0.1190	0.1417	0.1586	0.1949	0.1984
	LightGCN	0.0032	<u>0.0292</u>	<u>0.0567</u>	0.0800	<u>0.1031</u>	0.1206	0.1350	0.1632	0.1650

gains up to almost 3 times at $\beta = 1$ (0.0145 vs. non-graph max of 0.0051). Similar patterns hold for Most Recent, Most Favorite, and Least Favorite strategies, where GFCF and UltraGCN excel at higher β . Even in challenging strategies like Most Rated, where non-graph models (e.g., SimpleX and BPRMF) compete closely at low β , GNNs prevail at $n \geq 50$, with GFCF reaching 0.1195 at $\beta = 100$ (vs. non-graph max of 0.1057). Strategies emphasizing mainstream and discriminative behaviors of users, i.e., Highest Variance and Most Characteristic, respectively, further highlight GNN advantages, with gains in sparse regimes ($\beta \leq 6$) attributed to their ability to propagate information through graph neighborhoods despite data scarcity.

The Amazon-Book dataset (Table 7.4) corroborates these findings, with at least one GNN-based model outperforming the other models in 59/63 combinations (bold cases), with this statistic increasing up to 62/63 when considering the second-to-best model. UltraGCN and GFCF consistently lead for $\beta \geq 6$. However, GNN-based models generally outperform less consistently than in the Yelp dataset.

These findings suggest that GNN-based models, which explicitly leverage the graph structure of user-item interactions, provide a significant advantage under data minimization conditions. By propagating information along the user-item interaction graph, GNNs can capture higher-order connectivity patterns better, thereby mitigating the performance degradation that typically affects non-graph models when the amount of available training data is reduced. Consequently, our results provide strong empirical evidence that GNN-based recommender systems are generally more effective than other model families when limited user interaction data constrain the recommendation task.

In summary, the results demonstrate that GNN-based recommender systems—particularly LightGCN, UltraGCN, and GFCF—consistently outperform non-graph models under data minimization. Their performance advantage is especially pronounced in highly reduced settings (low β), where their ability to exploit higher-order graph connectivity mitigates data sparsity. Overall, these findings confirm that GNNs are more resilient and effective than traditional RSs when interaction data are severely limited.

7.6.2 Relationship between Dataset Characteristics and Model Performance (RQ2)

In this section, our aim is to answer the research question **RQ2** “*Which data characteristics impact the most on the performance of RS based on GNN in data minimization scenarios?*”. We organize the answer as follows. First, we discuss on the sanity check we have conducted to ensure the result validity of the regression model. Then, we analyze (i) whether there is an underlying relationship between the presented set of dataset characteristics and the effectiveness of GNN-based recommender systems, and (ii) the significance and directionality of the regression coefficients.

Table 7.5. Variance Inflation Factor (VIF) values before and after removing highly collinear features for each dataset and model. Features with VIFs > 5 were excluded to ensure regression stability. After correction, all remaining characteristics show acceptable VIFs, confirming the absence of harmful multicollinearity.

Characteristics		LightGCN		UltraGCN		GFCG	
		VIF before	VIF after	VIF before	VIF after	VIF before	VIF after
Amazon Books	<i>SpaceSize_{log}</i>	inf	3.1	inf	2.6	inf	3.6
	<i>Shape_{log}</i>	inf	dropped	inf	dropped	inf	dropped
	<i>Density_{log}</i>	inf	2.5	inf	3.3	inf	3.3
	<i>Gini-U</i>	33.6	4.4	36.4	3.9	33.0	3.9
	<i>Gini-I</i>	28.8	3.9	28.3	3.5	30.1	3.5
	<i>AvgDeg-U_{log}</i>	inf	dropped	inf	dropped	inf	dropped
	<i>AvgDeg-I_{log}</i>	inf	dropped	inf	dropped	inf	dropped
	<i>AvgClustC-U_{log}</i>	809.2	dropped	786.1	dropped	648.4	dropped
	<i>AvgClustC-I_{log}</i>	237.2	dropped	209.9	dropped	203.0	dropped
	<i>Assort-U</i>	11.6	dropped	11.2	dropped	13.8	dropped
	<i>Assort-I</i>	8.6	2.4	7.9	2.2	7.4	2.1
Yelp	<i>SpaceSize_{log}</i>	inf	3.0	inf	2.6	inf	2.5
	<i>Shape_{log}</i>	inf	dropped	inf	dropped	inf	dropped
	<i>Density_{log}</i>	1508.6	2.5	1729.2	2.3	1763.1	2.4
	<i>Gini-U</i>	33.7	2.6	34.8	2.4	34.2	2.6
	<i>Gini-I</i>	13.7	2.1	12.8	2.0	11.7	1.9
	<i>AvgDeg-U_{log}</i>	inf	dropped	inf	dropped	inf	dropped
	<i>AvgDeg-I_{log}</i>	inf	dropped	inf	dropped	inf	dropped
	<i>AvgClustC-U_{log}</i>	111.1	dropped	114.7	dropped	111.8	dropped
	<i>AvgClustC-I_{log}</i>	19.2	dropped	14.1	dropped	13.5	dropped
	<i>Assort-U</i>	7.4	dropped	6.2	dropped	6.9	dropped
	<i>Assort-I</i>	8.1	2.0	7.8	1.7	7.9	1.7

Sanity Check

Before applying the regression analysis, we performed a sanity check to ensure the reliability of the explanatory model. This work considers structural, distributional, topological, and minimization-semantic-based characteristics (see Section 7.3). Except for the minimization-semantic-based ones, the relationship between model performance—not specifically trained under data minimization conditions—and the structural or topological properties of the data has already been investigated in the literature [4, 161]. Thus, we included these features in our study. However, under data minimization scenarios, such characteristics may exhibit new behaviors that have not been documented. Therefore, it is crucial to verify that the features are not affected by excessive multicollinearity, which could compromise the interpretability and robustness of the regression coefficients [4]. To this end, we followed the procedure proposed in [68] and computed each feature’s Variance Inflation Factor (VIF). The VIF quantifies the degree of multicollinearity in a multiple regression model by measuring how much the variance of an estimated regression coefficient

is inflated due to correlations with other predictors. A VIF value 1 indicates the absence of collinearity, whereas higher values reflect stronger inter-feature correlation. Standard thresholds in the literature suggest that VIF values exceeding 5 or 10 indicate potentially problematic multicollinearity that may require correction [201]. Accordingly, we inspected all features and removed those with particularly high VIF scores to mitigate redundancy and improve model stability. Table 7.5 reports the VIF values for each feature before and after the sanity check.

The initial analysis revealed severe multicollinearity among several variables, with VIF values approaching infinity—an indicator of perfect or near-perfect linear dependencies. This behavior was particularly evident among structural and topological characteristics. In particular, the Shape, Average Degree, and Average Clustering Coefficient features exhibited extremely high VIF values (often above 100 or infinite) across all models and datasets. This was expected, as these measures are mathematically or conceptually intertwined:

- Shape is directly derived from the ratio between users and items, and is therefore inherently related to Space Size and Density, which are both functions of user–item cardinalities.
- Average Degree and Average Clustering Coefficient are second-order topological properties that depend on the same adjacency structure of the user–item bipartite graph, leading to strong linear relationships.
- Similarly, the User Assortativity index correlated with these metrics, as it partially reflects user connectivity patterns captured by degree-based descriptors.

After removing these highly collinear features, the remaining variables presented acceptable VIF values (below 5 in all cases), ensuring numerical stability and interpretability of the regression coefficients. The retained characteristics capture complementary and non-redundant aspects of dataset structure and heterogeneity. Interestingly, the removal process preserved at least one representative from each major family of characteristics (structural, distributional, and topological), preventing the explanatory model from being biased toward a specific descriptor type.

Importance of Classical and Topological Characteristics

The sanity check was crucial to ensure that the subsequent regression analyses reflect genuine relationships between dataset properties and the performance of GNN-based recommender systems under data minimization conditions, rather than artifacts introduced by feature interdependence.

Table 7.6 shows the results of the explanatory model. Particularly, we report the values of R^2 and R^2_{adj} coefficients, the constant term, and the learned regression coefficients for each dataset/model combination (refer to “Data Minimization” columns). For each coefficient, we report its statistical significance, derived from the p -value

Table 7.6. Comparison between regression models fitted on subsampled datasets and under data minimization conditions for LightGCN, UltraGCN, and GFCF, across the Amazon-Book and Yelp datasets. We consider structural, distributional, and topological characteristics to train the regression model. The table reports coefficients, significance levels (* for $p < 0.05$, ** for $p < 0.01$, and *** for $p < 0.001$), and goodness-of-fit metrics (R^2 , R^2_{adj}). The symbol “X” indicates that the corresponding feature is removed after the sanity check due to multicollinearity.

	Characteristics	Subsample			Data Minimization		
		LightGCN	UltraGCN	GFCF	LightGCN	UltraGCN	GFCF
Amazon Books	R^2	0.8582	0.4876	0.8016	0.9700	0.9529	0.9666
	R^2_{adj}	0.8573	0.4844	0.8004	0.9664	0.9480	0.9628
	Constant	0.0457***	0.0321***	0.0349***	0.1154***	0.1319***	0.1279***
	$SpaceSize_{log}$	0.0213***	0.0489***	-0.0034***	0.2895***	0.4184***	0.2900***
	$Shape_{log}$	0.0145	0.4642***	-0.1031***	X	X	X
	$Density_{log}$	X	X	X	0.1414***	0.1871***	0.1997***
	$Gini-U$	-0.5705***	-0.8064***	-0.5590***	0.0404	0.0532	0.0045
	$Gini-I$	X	X	X	0.0955***	0.1058**	0.0310
	$AvgDeg-U_{log}$	X	X	X	X	X	X
	$AvgDeg-I_{log}$	X	X	X	X	X	X
	$AvgClustC-U_{log}$	X	X	X	X	X	X
	$AvgClustC-I_{log}$	X	X	X	X	X	X
	$Assort-U$	-0.1008***	-0.0662***	-0.0932***	X	X	X
	$Assort-I$	-0.1163***	-0.0831***	-0.1270***	0.0093	0.0530	0.0109
Yelp	R^2	0.9458	0.8401	0.4264	0.9346	0.9321	0.9465
	R^2_{adj}	0.9454	0.8392	0.4228	0.9268	0.9250	0.9404
	Constant	0.0416***	0.0335***	0.0247***	0.0754***	0.0787***	0.0751***
	$SpaceSize_{log}$	0.0305***	0.0491***	0.0004	0.1648***	0.1763***	0.1334***
	$Shape_{log}$	0.0219	-0.0558*	0.0687**	X	X	X
	$Density_{log}$	X	X	X	0.0795***	0.0984***	0.1185***
	$Gini-U$	-0.2088***	-0.6569***	-0.0960***	0.0222*	0.0084	0.0067
	$Gini-I$	X	X	X	0.0703***	0.0635***	0.0358*
	$AvgDeg-U_{log}$	X	X	X	X	X	X
	$AvgDeg-I_{log}$	X	X	X	X	X	X
	$AvgClustC-U_{log}$	X	X	X	X	X	X
	$AvgClustC-I_{log}$	X	X	X	X	X	X
	$Assort-U$	-0.0298***	-0.0398***	-0.0365***	X	X	X
	$Assort-I$	-0.0634***	-0.0637***	-0.1072***	0.0024	-0.0249	-0.0170

of a t-test, using the signs * ($p < 0.05$), ** ($p < 0.01$), and *** ($p < 0.001$). We answer to RQ2 in two parts.

The first part aims to answer the following question: (RQ2a) “*Is there an underlying relationship between the presented set of dataset characteristics and the effectiveness of GNN-based recommender systems under data minimization conditions?*”. To answer this question, we examine the values of R^2 and R^2_{adj} in Table 7.6. The results demonstrate a strong underlying relationship between the

selected dataset characteristics (e.g., space size, density, Gini coefficients for users and items, and item assortativity) and the effectiveness of GNN-based recommender systems in minimized data scenarios. This is evidenced by the consistently high coefficient of determination (R^2 and R^2_{adj}) across the Amazon Books and Yelp datasets. For Amazon Books, the models explain 95–97% of the variance in performance (R^2_{adj} ranging from 0.9480 to 0.9664), while for Yelp, the explanatory power remains robust at 93–94% (R^2_{adj} from 0.9250 to 0.9404). These values empirically demonstrate that the retained characteristics capture the majority of factors influencing GNN performance when data is minimized independently from the original dataset and the recommendation model trained.

After validating the relationship between the dataset characteristics and the performance, we move to answering the following: **(RQ2b) “How significant is the impact of each characteristic on the effectiveness of GNN-based recommender systems under data minimization? What is the directionality of this impact (positive or negative)?”** Examining the regression coefficients reveals varying degrees of significance and directionality in how individual characteristics influence GNN effectiveness under data minimization. Across both datasets and all models, the most impactful features exhibit positive directionality and high statistical significance ($p < 0.001$ in most cases), while others show weaker or insignificant effects. This suggests a hierarchy of importance, with certain characteristics acting as performance anchors in minimized graphs.

First, we observe that structural characteristics play a pivotal role in the effectiveness of GNN-based recommender systems when trained on minimized datasets, surpassing distributional and topological properties in terms of impact. Specifically, $SpaceSize_{\log}$ emerges as the most influential characteristic, with large positive coefficients (e.g., 0.2895*** to 0.4184*** on Amazon Books; 0.1334*** to 0.1763*** on Yelp) across all models. This indicates that increasing the number of items within the reduced dataset brings performance benefits. We attribute this impact to two main causes: (i) $SpaceSize_{\log}$ reflects the overall number of retained interactions—when more interactions are preserved, the learning process becomes more effective and stable, thereby increasing recommendation accuracy; (ii) A higher $SpaceSize_{\log}$ value implies that more long-tail items are included in the training set. This allows GNN-based models to capture users’ nuanced preferences and to recommend diverse and niche items that remain relevant in the test set.

The second key observation concerns $Density_{\log}$ (e.g., 0.1414*** to 0.1997*** on Amazon Books; 0.0795*** to 0.1185*** on Yelp) and $Gini-I$ (e.g., 0.0955*** for LightGCN on Amazon Books; 0.0703*** for LightGCN on Yelp), both showing positive and highly significant coefficients—particularly for $Density_{\log}$. Higher density, often resulting from a skewed distribution where popular items dominate, implies denser user–item connections and fewer missing links. This consistently improves accuracy across all datasets, echoing findings in the long-tail recommendation literature, where popularity bias can enhance learning efficiency [2]. However, the

stronger impact of $SpaceSize_{log}$ suggests that minimization strategies preserving a larger number of interactions and maintaining some representation of the long-tail—rather than focusing solely on popular items—yield better accuracy. This provides a novel insight for designing data minimization techniques that explicitly trade off between the presence of popular and niche items after fixing the number of retained interactions per user. We conjecture that this occurs because the resulting minimized dataset is more representative of the test distribution,⁵ thus facilitating better generalization.

In contrast, $Gini-U$ (user-side inequality) exerts a much weaker and less consistent influence. Coefficients are small and rarely significant (e.g., 0.0222* for LightGCN on Yelp, insignificant elsewhere), suggesting that user activity imbalance is not a decisive factor under minimized conditions. This finding aligns with the intuition that GNNs, through neighborhood aggregation, are more sensitive to item-side structures and can partially compensate for uneven user engagement levels.

Similarly, $Assort-I$ exhibits minimal and inconsistent effects, with coefficients oscillating between small positive and negative values (e.g., -0.0249 for UltraGCN on Yelp), none of which reach statistical significance. This implies that the homophily structure of the bipartite interaction graph—i.e., the extent to which similar items or users connect—is largely irrelevant once dataset sparsity and density are controlled. These observations are consistent with prior evidence [107] showing that homophily plays a secondary role in graph-based recommender performance compared to global connectivity patterns.

Overall, these findings indicate that the effectiveness of GNN-based recommenders under data minimization is primarily driven by global structural characteristics rather than local or distributional properties such as assortativity or user inequality. This highlights a key design principle: minimized datasets that preserve sufficient density and item diversity can sustain high GNN performance while substantially reducing data volume.

In summary, the results reveal that GNN-based recommender performance under data minimization is predominantly driven by global structural characteristics. Features such as $SpaceSize_{log}$, $Density_{log}$, and $Gini-I$ consistently emerge as strong positive predictors, indicating that preserving dataset scale, connectivity, and item diversity enhances model effectiveness. In contrast, user inequality and assortativity exhibit negligible effects, confirming that structural properties outweigh local or distributional ones in determining GNN performance on minimized data.

5. We recall that, in our experimental setup, the test set is derived from the original, non-minimized data, simulating a realistic deployment scenario.

7.6.3 Specificity of Influential Characteristics to Data Minimization Strategies (RQ3)

In this section, we aim to answer the research question *RQ3* “*Are the most influential characteristics for GNN-based recommender performance under data minimization specific to the minimization strategies?*”. We assess whether the influential characteristics identified in the previous section are specific to this scenario of data minimization or generalize to random data reduction methods. To this end, we train our explanatory model on a set of samples of dataset characteristics collected by following the procedure proposed by [161]. We generate 900 subsamples for each of the two original datasets by sampling them through the node-dropout and edge-dropout strategies with a dropout rate uniformly taken in the range $[0.7, 0.9]$.⁶ These techniques indiscriminately remove interactions without preserving a fixed number of interactions per user and without reflecting a specific criterion, as instead done in data minimization. We contrast the regression results from the “Data Minimization” columns with those from the “Subsample” columns in Table 7.6.

This comparison reveals notable differences in the retained characteristics (due to multicollinearity handling), their coefficients, significance, and directionality, underscoring that the patterns observed are indeed specific to data minimization.

First, the overall relationship between dataset characteristics and GNN-based recommender systems performance is weaker under random subsampling, as indicated by lower R^2 and R^2_{adj} values. For Amazon Books, subsampling yields R^2_{adj} ranging from 0.4844 (UltraGCN) to 0.8573 (LightGCN), with particularly low values for UltraGCN (0.4844) and GFCE (0.8004). On Yelp, the range is 0.4228 (GFCE) to 0.9454 (LightGCN), again showing diminished explanatory power for GFCE. In contrast, data minimization consistently achieves $R^2_{\text{adj}} > 0.92$ across all models and datasets.

Regarding the differences in characteristics impact, the most influential characteristics under data minimization—such as $SpaceSize_{\log}$, $Density_{\log}$, and $Gini-I$ —do not hold the same prominence or directionality in the subsample case. Instead, subsampling highlights different features with often opposing effects, indicating that the positive, structure-preserving influences in minimization are not replicated in random reduction.

For instance, while positive and highly significant in minimization, $SpaceSize_{\log}$ shows mixed and weaker impacts in subsampling. On Amazon Books, it is positive for LightGCN (0.0213^{***}) and UltraGCN (0.0489^{***}), but negative for GFCE (-0.0034^{***}). On Yelp, it remains positive but less pronounced (0.0305^{***} for LightGCN; insignificant for GFCE).

In stark contrast, $Gini-U$ emerges as a dominant negative influence in subsampling, with large magnitudes (e.g., -0.5705^{***} to -0.8064^{***} on Amazon Books; -0.0960^{***} to -0.6569^{***} on Yelp). This indicates that higher user inequality under

6. For further details, we recommend the reader reading the work by [161]

random reduction harms performance, likely by amplifying imbalances and creating sparser user neighborhoods. Under minimization, however, its effect is weakly positive and often insignificant (e.g., 0.0222*).

Regarding assortativity, both *Assort-U* and *Assort-I* are negative and significant in subsampling (e.g., *Assort-U*: -0.0298*** to -0.1008***; *Assort-I*: -0.0634*** to -0.1270***), implying that higher assortativity (homophily) under random reduction correlates with performance drops, perhaps due to fragmented clusters. In minimization, *Assort-U* is excluded, and *Assort-I* is weakly positive or negative but insignificant.

A critical factor underlying the observed differences between data minimization and random subsampling is the fixed test set, extracted from the original, non-reduced dataset and held constant across all experiments. This design choice reflects a realistic deployment scenario—where a system is trained on a minimized or subsampled dataset but evaluated on full user behavior—but introduces an asymmetric generalization challenge that interacts differently with the two reduction cases. Under data minimization, retaining a fixed number of interactions per user with a specific criterion promotes distributional alignment between the reduced training graph and the fixed test set. This alignment explains the strong positive impact of *SpaceSize_{log}*, *Density_{log}*, and *Gini-I* in the regression models: increases in these characteristics directly enhance the model’s ability to generalize to the fixed test distribution. The GNN learns embeddings from a graph that, despite being smaller, remains structurally representative of the evaluation domain. In contrast, random subsampling (via edge or node dropout) induces uncontrolled distribution shift. As a result, characteristics such as *Gini-U* and assortativity emerge as negative predictors: higher values signal greater fragmentation and deviation from the test domain, degrading generalization.

Thus, the fixed test set acts as a structural anchor that amplifies the benefits of distribution-aware reduction (data minimization) while exposing the fragility of random methods. The influential characteristics under data minimization are not universally dominant but emerge as performance-critical precisely because the minimization strategy mitigates distribution shift.

In summary, our findings show that the characteristics influencing GNN-based recommender performance are specific to data minimization rather than generalizable to random subsampling. Under random reduction, the explanatory power of the regression model declines, and key predictors such as SpaceSize_{log}, Density_{log}, and Gini-I lose significance or reverse direction. These results confirm that data minimization preserves structural and distributional alignment with the test set, enabling more stable and interpretable performance patterns.

7.6.4 Augmenting the Feature Space with Minimization-Semantic Characteristics (RQ4)

In this section, we address the research question *RQ4*: “*Can we augment the set of characteristics so that the criteria used for retaining interactions during data minimization are also considered?*” To examine whether the explanatory power of our regression model can be enhanced by incorporating minimization-semantic characteristics that directly encode the rationale underlying data minimization—we extend the original feature set with the five metrics introduced in Section 7.3.3: Preference Centrality (*PrefCent*), Popularity Bias (*PopBias*), Rating Variability Exposure (*VarExp*), Temporal Recency (*TempRec*), and Characteristic Deviation (*CharDev*). These features capture the *semantics* of how interactions are retained, complementing structural graph descriptors with minimization-aware signals. We retrain the linear regression models by including this extended feature set under both data minimization and sub-sampling conditions, and we compare the results with the baseline model in Table 7.6. The augmented results, obtained after the sanity check, are reported in Table 7.7. Incorporating semantic characteristics improves both R^2 and R^2_{adj} in five out of six cases, with particularly notable gains on the Yelp dataset. The augmented model explains up to 97.5% of the performance variance, indicating that minimization criteria provide a distinct and non-redundant contribution. Despite this augmentation, $\text{SpaceSize}_{\log}$ and $\text{Density}_{\log}$ remain the most influential and stable predictors. The coefficient of $\text{SpaceSize}_{\log}$ decreases slightly but remains highly significant, while $\text{Density}_{\log}$ remains nearly unchanged. This suggests partial information overlap between structural and semantic features but reaffirms that dataset scale and density are fundamental determinants of performance. Overall, structural characteristics form the backbone of model explainability, while semantic criteria add complementary, higher-level insight. Focusing on minimization-semantic features, we observe that their statistical significance is not uniform across models. However, their directionality remains consistent across all model–dataset combinations, revealing practical design insights:

- The negative directionality of *PopBias* corroborates the results discussed in Section 7.6.2: retaining only popular items within the minimized dataset diminishes the performance of GNN-based recommender systems;
- The strong positive directionality of *TempRec* suggests that prioritizing recent interactions yields performance gains, highlighting the value of temporal information retention;
- The positive directionality of *PrefCent* indicates that preserving user preference representativeness can enhance model performance.

In summary, the results show that, while structural dataset characteristics remain the primary drivers of performance under data minimization, the inclusion of minimization-semantic characteristics provides additional explanatory power and supports the formulation of actionable guidelines for designing more effective minimization strategies.

7.7 Conclusion

This chapter systematically investigated the performance of Graph Neural Network (GNN)-based Recommender Systems (RSs) under data minimization conditions. Leveraging a comprehensive explanatory framework that integrates structural, topological, and novel minimization-semantic dataset characteristics, we conducted an extensive empirical analysis across diverse datasets, minimization strategies, and retention levels. This dual focus—performance evaluation and characteristic-driven interpretation—has yielded the results summarized as follows:

- GNN-based models consistently outperform traditional RS families even when trained on significantly reduced interaction subsets.
- We identified the most influential dataset characteristics impacting GNN performance under minimization. Among structural, distributional, and topological features, structural characteristics, i.e., Space Size and Density, emerged as the most influential characteristics on the performance of GNN-based RSs.
- We explained that the predictive power of these characteristics is specific to the data minimization scenario. In contrast, random pruning yielded different rankings of feature importance.
- We validated the introduction of a novel family of minimization-semantic characteristics, which quantify the alignment between retained interactions and the underlying retention criterion. These features improved the explanatory power of our framework, supporting the formulation of actionable guidelines for designing more effective minimization strategies.

Our findings provide actionable guidance for addressing regulations related to data minimization. GNN-based RSs can achieve better performance with less training data. Furthermore, we also provide actionable guidance for practitioners seeking to design novel minimization strategies. We have found that maintaining some representation of the long-tail items within the reduced datasets promotes distributional alignment between the reduced datasets and the test set, resulting in better accuracy of the trained models. Therefore, it is crucial to trade off the retention of interactions involving popular items to achieve a higher density of the graph versus those involving long-tail items.

Table 7.7. Comparison between regression models fitted on subsampled datasets and under data minimization conditions for LightGCN, UltraGCN, and GFCF, across the Amazon-Book and Yelp datasets. Beyond structural, distributional, and topological characteristics, we consider also minimization-semantic characteristics to train the regression model. The table reports coefficients, significance levels (* for $p < 0.05$, ** for $p < 0.01$, and *** for $p < 0.001$), and goodness-of-fit metrics (R^2 , R^2_{adj}). The symbol “X” indicates that the corresponding feature is removed after the sanity check due to multicollinearity.

	Characteristics	Subsample			Data Minimization		
		LightGCN	UltraGCN	GFCF	LightGCN	UltraGCN	GFCF
Amazon Books	R^2	0.8593	0.4890	0.8041	0.9791	0.9555	0.9730
	R^2_{adj}	0.8580	0.4846	0.8024	0.9748	0.9476	0.9677
	Intercept	0.0458***	0.0321***	0.0349***	0.1158***	0.1317***	0.1276***
	$SpaceSize_{log}$	0.0208***	0.0495***	-0.0042***	0.2563***	0.3910***	0.2616***
	$Shape_{log}$	0.0253	0.4631***	-0.0906***	X	X	X
	$Density_{log}$	X	X	X	0.1384***	0.1866***	0.1973***
	Gini-U	-0.5744***	-0.8159***	-0.5604***	0.0324	0.0471	-0.0008
	Gini-I	X	X	X	0.0675***	0.0865*	0.0110
	$AvgDeg-U_{log}$	X	X	X	X	X	X
	$AvgDeg-I_{log}$	X	X	X	X	X	X
	$AvgClustC-U_{log}$	X	X	X	X	X	X
	$AvgClustC-I_{log}$	X	X	X	X	X	X
	Assort-U	-0.1002***	-0.0672***	-0.0930***	X	X	X
	Assort-I	-0.1147***	-0.0858***	-0.1233***	0.0092	0.0511	0.0013
	PrefCent	-0.0958	-0.3070	0.0398	0.0513**	0.0342	0.0598
	VarExp	X	X	X	X	X	X
	CharDev	X	X	X	X	X	X
	PopBias	X	X	X	-0.0337*	-0.0230	-0.0102
	TempRec	-0.3910	0.3522	-0.5864*	0.1860	0.1676	0.1588
Yelp	R^2	0.9462	0.8402	0.4358	0.9530	0.9492	0.9512
	R^2_{adj}	0.9457	0.8388	0.4309	0.9434	0.9402	0.9417
	Intercept	0.0416***	0.0335***	0.0247***	0.0747***	0.0786***	0.0753***
	$SpaceSize_{log}$	0.0310***	0.0494***	0.0017	0.1428***	0.1679***	0.1288***
	$Shape_{log}$	0.0129	-0.0622*	0.0477	X	X	X
	$Density_{log}$	X	X	X	0.0847***	0.0943***	0.1203***
	Gini-U	-0.2159***	-0.6617***	-0.1127***	0.0131	0.0089	0.0059
	Gini-I	X	X	X	0.0518***	0.0425*	0.0287*
	$AvgDeg-U_{log}$	X	X	X	X	X	X
	$AvgDeg-I_{log}$	X	X	X	X	X	X
	$AvgClustC-U_{log}$	X	X	X	X	X	X
	$AvgClustC-I_{log}$	X	X	X	X	X	X
	Assort-U	-0.0307***	-0.0403***	-0.0384***	X	X	X
	Assort-I	-0.0629***	-0.0629***	-0.1051***	0.0153	0.0138	0.0013
	PrefCent	0.0915*	0.0660	0.2343***	0.0053	0.0211*	0.0030
	VarExp	X	X	X	X	X	X
	CharDev	X	X	X	X	X	X
	PopBias	X	X	X	-0.0612***	-0.0635***	-0.0221
	TempRec	-0.1021	-0.0239	-0.2186	0.1928**	0.0327	0.2617

Part V

**Closing the Reproducibility Gap in
Recommendation**

Chapter 8

Toward Standardized and Reproducible Data Management in Recommender Systems

Recommender systems have demonstrated significant impact across diverse domains, yet ensuring the reproducibility of experimental findings remains a persistent challenge. A primary obstacle lies in the fragmented and often opaque data management strategies employed during the preprocessing stage, where decisions about dataset selection, filtering, and splitting can substantially influence outcomes. To address these limitations, we introduce DATAREC, an open-source Python-based library specifically designed to unify and streamline data handling in recommender system research. By providing reproducible routines for dataset preparation, data versioning, and seamless integration with other frameworks, DATAREC promotes methodological standardization, interoperability, and comparability across different experimental setups. Our design is informed by an in-depth review of 55 state-of-the-art recommendation studies ensuring that DATAREC adopts best practices while addressing common pitfalls in data management. Ultimately, our contribution facilitates fair benchmarking, enhances reproducibility, and fosters greater trust in experimental results within the broader recommender systems community. The DATAREC library, documentation, and examples are freely available at <https://github.com/sisinflab/DataRec>.¹

1. This chapter is based on the work published in the *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval* (SIGIR 2025) “Datarec: A python library for standardized and reproducible data management in recommender systems”.

8.1 Introduction

Over the past decade, recommender systems have become a pivotal area of research and innovation, driving transformative advancements across diverse domains, including e-commerce, entertainment, healthcare, and education. By tailoring content and services to individual preferences, these systems not only enhance user experiences but also generate significant economic value, establishing their development and evaluation as critical priorities for researchers and practitioners in both academia and industry.

Despite this progress, achieving fair, reliable, and reproducible results remains a persistent challenge in recommender system research. Addressing this challenge and avoiding biased comparisons, unreliable findings, and virtual scientific progress requires rigorous methodologies that encompass every aspect of the recommendation pipeline, from data management and model training to performance evaluation [28, 64, 232, 300]. Central to these issues is the principle of reproducibility, a cornerstone of scientific inquiry that enables the validation of findings, ensures fair comparisons, and fosters trust within the research community. To better understand the origins of these reproducibility concerns in recommender systems, it is crucial to examine the initial stages of the recommendation pipeline. Indeed, the data management phase emerges as a particularly influential factor in shaping the reliability and comparability of experimental results [204].

The role of this phase extends far beyond simple preprocessing. Instead, it encompasses a complex sequence of tasks, from dataset preparation and filtering to splitting and formatting, each of which plays a pivotal role in shaping the outcomes of experimental evaluations. While existing recommender system frameworks have undeniably contributed to improving the transparency and comparability of model training and evaluation [9, 53, 81, 100, 135, 158, 175, 205, 232, 250, 311, 315], their framework-specific implementations of data handling strategies become a significant source of concern. Although effective within their own ecosystems, these implementations often diverge significantly, leading to inconsistent experimental results and, critically, undermining the very foundation of comparability in research findings [63, 151]. Adding to this challenge, the prevalence of unique, non-interchangeable data management strategies across different frameworks further exacerbates the problem, severely limiting cross-framework interoperability and, consequently, the potential for robust reproducibility. This pervasive fragmentation unequivocally highlights the urgent need for a unifying approach to data management, one that transcends the limitations of individual frameworks and fosters essential methodological consistency across the entire field.

To address all the above-mentioned challenges, we propose **DATAREC**, an open-source Python-based library specifically designed to align, unify, and streamline the data management phase of recommender system research. Unlike traditional frameworks, which often aim to provide comprehensive solutions for the entire

recommendation pipeline, DATAREC is a library designed to complement them by serving as a shared foundation for reproducible and interoperable data handling. By unifying the diverse strategies currently employed across frameworks, DATAREC provides researchers with a comprehensive and open toolkit that facilitates both the standalone preprocessing of data and the coherent implementation of the pipelines of existing and new frameworks. DATAREC's key features include support for widely used data formats, direct access to 18 built-in research datasets with explicit references, and reproducible pre-processing and splitting strategies. It can be easily integrated into standalone projects and offers interfaces for exporting datasets in formats compatible with other frameworks. By prioritizing interoperability and modularity, DATAREC streamlines best practices in data management, addresses a key challenge in the reproducibility of recommender systems, and simplifies the development process.

DATAREC design is guided by a systematic analysis of 55 recommendation papers published in leading venues between 2020 and 2024. This analysis identifies common strategies, inconsistencies, and challenges in data handling, and its findings directly inform the design of DATAREC, ensuring that the tool provides robust solutions to these issues while facilitating transparency, comparability, and replicability. By addressing the often-overlooked yet critical data management phase, DATAREC advances the reproducibility of recommender system research, complementing existing frameworks and fostering more rigorous and trustworthy scientific practices. In particular, our contributions are threefold:

- A comprehensive survey of data-related practices in state-of-the-art recommender systems research, addressing prevalent challenges in dataset preparation, filtering, and splitting.
- A fine-grained analysis of how the most widely used frameworks support common data-related operations.
- The release of DATAREC, an open-source Python-based library for reproducible and interoperable data management in recommender systems research, publicly available at <https://github.com/sisinflab/DataRec>.

8.2 Overview of Data Handling and Processing in Recommender Systems

Managing and processing recommendation datasets is fundamental to recommender system research, with strategies that can vary from study to study. On the one hand, to identify the most commonly adopted approaches in state-of-the-art research, we surveyed recent papers published at top-tier conferences in search and recommendation and collected data-handling strategies involved in their proposed recommendation models. Then, on the other side, we critically analysed leading frameworks collecting several recommender system techniques and whether they

Table 8.1. Datasets, pre-processing and splitting strategies adopted in 55 research papers published between 2020 and 2024. Only datasets and strategies appearing in at least 3 research papers are shown.

Paper	Venue	Year	Datasets														Pre-processing					Splitting											
			Yelp	Amazon Books	MovieLens 1M	Amazon Beauty	Last.fm	Gowalla	Amazon Toys and Games	Tmall	Amazon Games	Yelp2018	Alibaba-iFashion	Amazon Clothing	Amazon Sports and Outdoors	Epinions	MovieLens 20M	Others	None	Binarization	k-Core Iterative	k-Core User	k-Core Item	Session length	Others	Random Hold-Out	Temporal Leave-1-out	Hold-Out Fixed	Random Leave-1-out	Hold-Out Handcrafted	Others	NA	
[110]	TOIS	2023	✓	✓			✓									✓		✓								✓							
[220]	SIGIR	2022	✓					✓					✓				✓				✓					✓							
[132]	KDD	2023	✓				✓										✓		✓							✓							
[297]	SIGIR	2022		✓								✓					✓		✓							✓							
[281]	WWW	2023	✓	✓			✓										✓									✓							
[108]	RecSys	2020															✓														✓		
[265]	CIKM	2022				✓			✓								✓		✓	✓							✓						
[91]	Cyb IEEE	2021			✓												✓	✓		✓												✓	
[280]	MM	2023	✓	✓	✓	✓											✓		✓		✓					✓							
[257]	SIGIR	2023	✓	✓	✓												✓		✓							✓							
[193]	SIGIR	2023		✓				✓		✓							✓											✓					
[302]	CIKM	2021															✓		✓					✓		✓							
[104]	SIGKDD	2021								✓							✓		✓											✓			
[303]	TKDE	2022				✓					✓						✓		✓							✓							
[75]	SIGIR	2023			✓	✓			✓								✓		✓							✓							
[285]	ICDE	2021			✓	✓					✓						✓		✓					✓		✓							
[112]	TKDE	2023							✓	✓										✓	✓	✓				✓							
[152]	RecSys	2020				✓				✓	✓						✓		✓	✓						✓							
[134]	PKDD	2022															✓		✓							✓							
[274]	FCS	2022		✓													✓				✓					✓							
[307]	TOIS	2020				✓	✓				✓						✓	✓	✓	✓	✓	✓	✓	✓		✓						✓	
[282]	WWW	2023	✓	✓				✓									✓		✓							✓							
[49]	WSDM	2023	✓													✓	✓												✓				
[283]	SIGIR	2022	✓	✓													✓		✓							✓							
[34]	ESwA	2022															✓						✓	✓		✓							
[162]	RecSys	2023			✓												✓		✓	✓						✓							
[289]	SIGIR	2022			✓							✓					✓									✓							
[288]	KDD	2023					✓						✓				✓		✓							✓							
[251]	WSDM	2023		✓			✓										✓			✓						✓							
[126]	AAAI	2021	✓														✓		✓												✓		
[237]	SIGIR	2022	✓	✓	✓												✓		✓	✓						✓							
[39]	ICLR	2023	✓	✓				✓		✓							✓		✓							✓							
[115]	SIGIR	2020		✓				✓				✓					✓									✓							
[268]	CIKM	2022		✓				✓	✓				✓				✓				✓					✓							
[320]	SIGIR	2022			✓		✓										✓		✓							✓							
[270]	WWW	2023															✓									✓							
[279]	DASFAA	2022								✓							✓		✓							✓							
[145]	SIGIR	2023	✓														✓															✓	
[189]	NIPS	2024				✓			✓								✓			✓						✓							
[48]	AAAI	2020		✓					✓								✓									✓							
[105]	SIGIR	2021											✓				✓		✓							✓						✓	
[313]	CIKM	2020	✓			✓	✓		✓								✓									✓							
[284]	CIKM	2022								✓							✓					✓	✓			✓							
[275]	SIGIR	2021										✓					✓									✓							
[101]	IJCAI	2022	✓														✓		✓							✓							
[296]	WWW	2021	✓				✓										✓		✓							✓						✓	
[45]	SIGIR	2021															✓			✓	✓					✓					✓		
[143]	Inf. Sci.	2021						✓									✓				✓		✓	✓		✓						✓	
[233]	UMUAI	2020															✓									✓							
[223]	SIGIR	2021	✓														✓		✓							✓							
[295]	KDD	2022	✓				✓										✓		✓							✓							
[23]	ICBD	2020														✓	✓									✓						✓	
[94]	WSDM	2021			✓												✓		✓							✓							
[141]	Inf. Sci.	2021															✓							✓		✓						✓	
[294]	TKDE	2022										✓	✓				✓		✓							✓							
			17	14	10	9	9	8	7	7	4	4	3	3	3	3	40	25	15	13	9	4	4	7	21	13	5	4	3	10	1		

integrate (or do not) these functionalities. These analyses offer an overview of current practices in data handling for recommendation and serve as the motivational basis for the functionalities implemented in DATARec. We deepen into these two investigations in the following paragraphs.

8.2.1 Current Trends in Recommender Research

Techniques and procedures for data handling in recommender systems are diverse and may depend on the specific dataset, domain, study, or implementation. To identify the most commonly adopted approaches, we analyzed 55 papers published between 2020 and 2024, extracted from five surveys covering diverse domains in recommendation: **graph neural networks** [276], **contrastive learning** [133], **temporal modeling** [33], **reinforcement learning** [5], and **generative approaches** [69]. These papers were selected from the cited surveys to ensure they are relevant to the community, come from diverse domains, and represent state-of-the-art research. Table 8.1 summarizes the results of our survey.

Acknowledging the diversity of existing approaches, our work provides a comprehensive overview of current practices and their shortcomings, setting the stage for a unified approach to data management. In subsequent sections, we detail our findings in each key area, dataset usage, filtering strategies, and data-splitting methods and discuss their broader implications for the reproducibility of recommender systems research. This systematic investigation not only underscores the pressing need for methodological consistency but also motivates the development of DATARec as a solution to these challenges. Below, we present the results of our study, highlighting the differences in dataset selection, binarization and filtering techniques adopted, and splitting strategies for model evaluation.

Datasets

Table 8.2. Dataset referencing in surveyed papers. The table reports the different ways datasets are cited, including references to the original data source, copies of the dataset, citations of the dataset’s original paper or other scholarly papers, missing references, and broken links.

Reference Type	# Usages (Percentage)
Original data source	63 (35.2%)
Copy a new version of the dataset	58 (32.4%)
Dataset’s original paper	27 (15.1%)
Other scholarly papers	3 (1.7%)
No reference	19 (10.6%)
Broken link	9 (5.0%)
Total	179 (100%)

The surveyed papers reveal a remarkable breadth in dataset usage, with **79 distinct datasets** identified. Within these, the ones appearing in at least three different datasets are explicitly reported in Table 8.1. This extensive utilization of diverse datasets underscores the inherent adaptability and domain-agnostic nature of recommender systems, showcasing their capacity to address a wide array of real-world applications. Indeed, this heterogeneity reflects the field’s maturity and its responsiveness to varied application contexts.

However, our review raises a significant issue in how recommendation datasets are referenced, as evidenced by the further analysis reported in Table 8.2. While papers using open datasets should reference the **original data source**, only 35.2% of the 179 dataset usages analyzed provide such an explicit reference. Among the remaining cases, 32.4% fail to cite the original source entirely, instead offering a **copy of a new version the dataset**—often already processed—without clarifying its origin. Although this practice may support reproducibility, it does not ensure a clear reproduction of the filtering methods applied, compromising the validation of these approaches. Additionally, 16.8% cite only the **dataset’s original paper** or **other scholarly papers**, which does not guarantee clarity about the dataset used, especially when multiple online versions exist. Finally, 15.6% of the datasets are irretrievable, either due to the **absence of any reference** or because the provided **links are broken**, e.g., they are no longer functional or have been blocked.

This analysis highlights the importance of tracking and versioning recommendation datasets. By maintaining a direct reference to the original or trusted sources and by providing tools to reproduce filtering and splitting approaches, complete reproducibility can be ensured. For this reason, DATAREC includes classes and methods for direct and traceable management of the most widely used datasets in the analyzed papers, specifically, those appearing in at least three different articles, as reported in Table 8.1.

Binarization and Filtering

Pre-processing strategies are critical in every experimental setting, as meaningful evaluations require aligning datasets with the experimental design. Consequently, ensuring the reproducibility of these algorithms is essential for valid comparisons across different experiments. Our analysis identified the most commonly adopted strategies — retaining only those that appear in at least three datasets — which are reported in Table 8.1.

One fundamental pre-processing strategy is **binarization**, which converts explicit feedback into implicit feedback. This transformation is designed to facilitate the training and evaluation of models that predict the likelihood of interactions rather than precise rating values. This task has become increasingly prevalent in contemporary recommender systems [125, 195, 234]. The ***k*-Core** method is widely employed to remove cold users (***k*-Core User**), items (***k*-Core Item**), or iteratively both of them (***k*-Core Iterative**). Indeed, due to the lack of detailed preference in-

formation, these interactions are often considered noisy and are removed to prevent them from negatively affecting model performance. Other strategies we identified are primarily associated with session-based recommendation experiments, where it could be useful to filter data based on the **session length**, in order to retain neither short nor long sessions. Due to their wide adoption, we have included these techniques in DATAREC and other classical approaches. However, we do not include session-based approaches because they are not yet supported within the library.

Splitting

Data splitting strategies partition datasets for training, evaluating, and testing models. Our survey confirms prior findings [173, 230] that there is no universally accepted splitting method. Instead, studies employ diverse strategies, contributing to inconsistent rankings of recommender systems, even when the same datasets and metrics are used [64, 304].

The most commonly strategies, as reported in Table 8.1 include:

- **Random Holdout Splitting:** Randomly partitions the dataset into training, validation, and test sets based on predefined percentages. This is the most common approach.
- **Temporal Leave-1-Out:** It is the second most common. Extracts the last transaction per user for test and one for validation.
- **Random Leave-1-Out:** Similarly, the temporal version extracts one transaction per user randomly chosen from the user history.
- **Pre-computed:** One common approach consists of releasing a copy of the train, test and validation splits used for the experiments.
- **Temporal Fixed:** Splits data based on timestamps, either globally (e.g., interactions after a fixed time point) or per user (e.g., a percentage of the most recent interactions).

Like the aforementioned filtering strategies, these have also been included in DATAREC along with other classic approaches.

8.2.2 Existing Approaches in Recommendation Frameworks

Recommendation frameworks foster reproducibility and replicability of research, accelerate the development of recommender systems by eliminating the need for developers to reimplement algorithms from scratch, and contribute to more reliable and impactful research. Many of these tools provide end-to-end architectures that manage the entire recommendation pipeline, while others focus on specific recommendation tasks.

Following the frameworks suggested by the ACM RecSys conference², we analyze the most widely adopted ones. General-purpose end-to-end frameworks in-

2. <https://github.com/ACMRecSys/recsys-evaluation-frameworks>

Table 8.3. Comparison of DATAREC with existing recommendation frameworks based on data input/output, pre-filtering and dataset splitting strategies.

	Data Input/Output				Prefiltering							Splitting																				
	I/O		Avail. Data	Ext. frameworks output	Filter- by-rating		<i>k</i> -Core			Temporal Hold-out				Random																		
	Formats	Tabular			Numerical	Distributional	User Dist.	User	Item	Iterative	Iter- <i>n</i> -rounds	Cold-Users	Fixed Timestamp	By-Ratio Sys.	By-Ratio User	Leave-1-out	Leave- <i>n</i> -in	Leave- <i>n</i> -out	By-Ratio Sys.	By-Ratio User	Leave-1-out	Leave- <i>n</i> -in	Leave- <i>n</i> -out	By-Ratio Sys.	By-Ratio User	Leave-1-out	Leave- <i>n</i> -in	Leave- <i>n</i> -out	<i>k</i> -folds Sys.	<i>k</i> -folds User	Pre-computed	
																																Inline
ClayRS [158]	✓	✓	✓	1	✓									✓						✓										✓	✓	✓
Cornac [205]	✓	✓		13	✓			✓	✓											✓	✓									✓	✓	✓
Elliot [9]				1	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
FuxiCTR [315]	✓			0																✓											✓	✓
LensKit Python [81]	✓			1																										✓	✓	✓
Mab2Rec [135]	✓			0																												✓
Recommenders [100]	✓			7	✓				✓	✓	✓			✓	✓	✓				✓	✓											✓
RecBole [311]				66	✓				✓	✓	✓	✓	✓		✓	✓	✓	✓	✓					✓								✓
ReChorus [250]	✓			3																												✓
RecPack [175]	✓			11					✓	✓				✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
DATAREC	✓	✓	✓	18	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

clude **ClayRS** [158], **DaisyRec** [232], **Elliot** [9], **LensKit** [81], **RecBole** [311], **Recommenders** [100], and **RecPack** [175], each designed to support different recommendation paradigms. Other frameworks specialize in specific recommendation tasks. **Cornac** [205] targets multimodal recommendation, **FuxiCTR** [315] focuses on click-through rate prediction, and **ReChorus** [250] is tailored for sequential recommendation. In contrast, **RecList** [53] is not designed for pipeline management but serves as a behavioral testing framework. One of the most recent frameworks, **Mab2Rec** [135], addresses the lack of modularity in existing frameworks, which the authors argue hinders usability. To overcome this limitation, they introduce a modular framework specifically for multi-armed bandit recommenders.

Although these frameworks share models, algorithms, and design principles, they often function as isolated systems, each managing the entire recommendation pipeline. This results in monolithic architectures that limit interoperability and complicate the reproduction of data transformations across frameworks, posing a particular challenge for researchers aiming to experiment with models available only within specific ecosystems.

In addition, dealing with all elements of a recommendation pipeline requires constant maintenance and updating, leading to the potential oversight of some components that still need to be completed, updated, or even missing. This limitation is further exacerbated by the lack of explicit interoperability between frameworks, which prevents the reuse of procedures already implemented in other projects. The resulting landscape is a collection of frameworks that re-implement existing solutions or, even worse, fail to implement some of the most common data handling strategies in recommendation. This is highlighted in Table 8.3.

Conversely, in DATAREC, we not only implemented the reported strategies, but thanks to our library’s modularity and the explicit interfaces for exporting datasets in

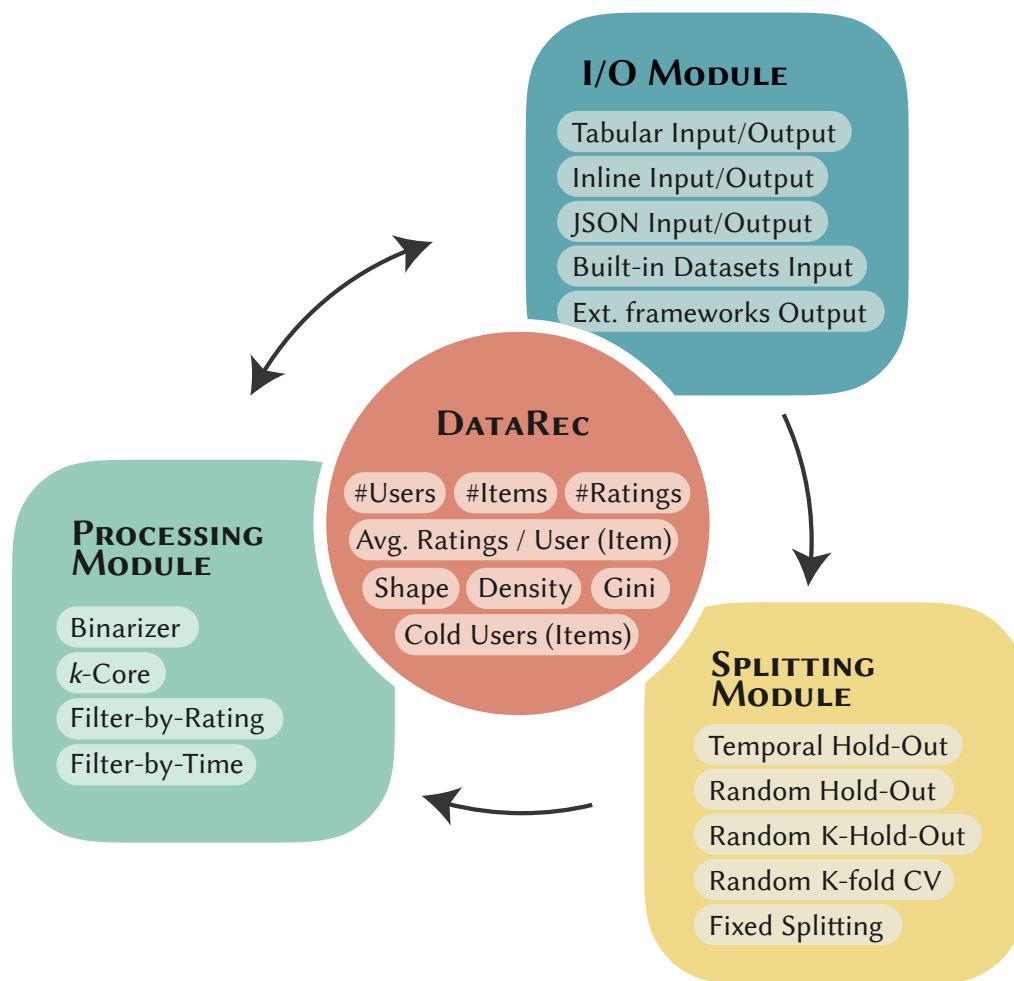


Figure 8.1. Overview of the DATARec architecture. The DataRec class provides key dataset metrics and interacts with three modules: I/O for handling different data formats, processing for dataset transformations, and splitting for partitioning into training, validation, and test sets.

a format compatible with other frameworks, these strategies can be easily integrated into existing frameworks, thus contributing to the overall reproducibility landscape of recommender systems. It is essential to underline that these analyses could change as recommendation frameworks evolve over time. RecList [53] was excluded from the analysis because it only handles the evaluation phase.

8.3 DATARec

DATARec is an open-source Python library for handling recommendation datasets. In addition to providing the tools needed to manage datasets, DATARec aims to ensure reproducibility, ease of use, data traceability, modularity, and interoperability. Figure 8.1 illustrates the fundamental components of the library that interact

through the main class `DataRec`, which serves as the primary element for dataset manipulation. In the following sections a more in-depth description of the `DATARec` main functionalities: the input/output module, the `DataRec` Class, the processing and splitting modules, and the reproducibility strategies.

8.3.1 *The I/O Module*

This module handles the operations for reading and writing datasets in `DATARec`. Datasets can be read from files in the most common formats and saved in the same formats. Moreover, to facilitate the export of datasets for direct use within other frameworks, export methods are provided for several frameworks. To ensure traceability and versioning, `DATARec` includes the most widely used datasets within its scope according to state-of-the-art practices identified in our analysis. For clarity, each of these aspects is discussed in detail below.

Data Formats

`DATARec` supports the three main formats for representing recommendation datasets that arose from our analysis:

- **Tabular:** Each row in the dataset represents a record of a user-item interaction, which may also include an explicit rating and a timestamp.
- **Inline:** This format represents implicit feedback between users and items. Each row represents a user's history, where the first element is the user and the following elements (in a variable number) represent the items in their history.
- **JSON:** This format facilitates the representation of unstructured content associated with interactions. It is structured as a sequence of objects, where each object represents a user-item interaction and its metadata, including an explicit rating and a timestamp.

For each of these formats, the library provides reading and writing methods, allowing conversion of a dataset from one format to another. During the input/output process, `DATARec` uses the `RawData` class to represent the data. This class is a simplified version of the `DataRec` class and serves as a unified interface for I/O operations.

Reading and Writing

```
from datarec.io import read_tabular, write_tabular

data = read_tabular(
    path, sep='::', user_col=0, header=None)

write_tabular(
    data, path=output_path, sep='\t', header=None)
```

Table 8.4. Overview of the datasets implemented in DATAREC. The table lists the dataset name, file format, available versions, and source links.

Datasets	Format	Versions	Source
Alibaba-iFashion [50]	Inline	2019	[link] ³
Amazon Reviews [124, 176]	Tabular	2018	[link] ⁴
	Tabular	2023	[link] ⁵
CiaoDVD [103]	Inline	2013	[link] ⁶
Epinions [200]	Inline	2003	[link] ⁷
Gowalla [55]	Inline	2011	[link] ⁸
Last.fm [41]	Tabular	2011	[link] ⁹
Mind [273]	Tabular	2020	[link] ¹⁰
MovieLens [113]	Tabular	2006 (1M)	[link] ¹¹
	Tabular	2016 (20M)	[link] ¹²
Tmall [238]	Tabular	2018	[link] ¹³
Yelp	JSON	2023	[link] ¹⁴

Built-in Datasets

Based on the findings of our survey, DATAREC incorporates the **16 most widely used recommendation datasets** [41, 50, 55, 113, 114, 123, 169, 170, 176, 238, 273, 291], each appearing in at least three different papers (along with other additional datasets). To mitigate versioning issues, we maintain the reference to the original data source whenever possible and implement available previous versions for backward compatibility. A summary of available datasets is available in Table 8.4.

When a dataset is directly available in DATAREC, the library allows the user to download it and use it in a ready-to-use DataRec object. For each version of each dataset available, DATAREC provides a public checksum to validate the correctness of the resource downloaded from the referenced source.

3. <https://drive.google.com/drive/folders/1xFdx5xuNXHGsuVVG2VIohFTXf9S7G5veq>

4. https://cseweb.ucsd.edu/~jmcauley/datasets/amazon_v2/

5. <https://amazon-reviews-2023.github.io/>

6. <https://guoguibing.github.io/librec/datasets.html>

7. <https://snap.stanford.edu/data/soc-Epinions1.html>

8. <https://snap.stanford.edu/data/loc-gowalla.html>

9. <https://grouplens.org/datasets/hetrec-2011/>

10. <https://msnews.github.io/>

11. <https://grouplens.org/datasets/movielens/1m/>

12. <https://grouplens.org/datasets/movielens/20m/>

13. <https://tianchi.aliyun.com/dataset/53?t=1716541860503>

14. <https://www.yelp.com/dataset>

Built-in Dataset

```
from datarec.datasets import MovieLens

data = MovieLens(version='1m')
```

Interoperability

To facilitate the integration of library functionalities and results into existing recommendation frameworks, DATAREC provides methods for exporting a DataRec object in formats compatible with the following recommendation frameworks: **ClayRS** [158], **Cornac** [205], **DaisyRec** [232], **Elliot** [9], **LensKit** [81], **RecBole** [311], **ReChorus** [250], **Recommenders** [100], and **RecPack** [175]. In addition, DATAREC includes utilities designed to streamline integration with each of these frameworks.

Data Export

```
from datarec.io import FrameworkExporter

exporter = FrameworkExporter(output_path=path)
exporter.to_elliot(train, test, val)
```

8.3.2 The DataRec Class

The DataRec class is the main interface within the library for managing recommendation datasets. Internally, each dataset is stored as a pandas DataFrame object, which allows access to optimized methods for dataset management. The pandas DataFrame is structured as a table with at least two columns, one for the user and one for the item, with each record corresponding to a user-item interaction. If an interaction includes an explicit record, it is stored numerically in a third column. The fourth and final column, when provided, manages timestamps to capture temporal information.

Among the other functionalities of the DataRec class is the calculation of metrics for analyzing recommendation datasets, which have been shown to impact model performance [4, 54, 72]. The implemented metrics include **space size**, **shape**, **density**, user and item **Gini coefficient**, and **average ratings** per user and per item. Finally, users and items can be classified into four categories based on their **popularity** within the dataset: “most popular,” “popular,” “common,” and “long tail.” The classification is automatically determined by computing the quartile values of the user and item distributions.

8.3.3 Processing Module

This module implements the tools necessary for transforming a recommendation dataset. The implemented approaches are:

- **Binarizer:** Converts explicit ratings into implicit feedback based on user-definable thresholds.
- **k -Core:** Filters out users and/or items with fewer than k recorded interactions. The iterative version (k -Core iterative) repeatedly applies this filtering until all remaining users and items meet the k -Core criterion. If convergence cannot be achieved, DATARec allows setting a maximum number of iterations (Iter- n -rounds). Additionally, the framework includes an option to filter out cold users by retaining only those with a minimum level of prior interactions.
- **Filter-by-Rating:** This method eliminates user-item interactions when the preference score falls below a specified threshold. The threshold can be defined as: (1) a fixed numerical value (e.g., 2.5); (2) a global metric (e.g., the dataset-wide average rating); or (3) a user-specific measure (e.g., the user's mean rating).
- **Filter-by-Time:** Retains only interactions before or after a given time threshold.

8.3.4 Splitting Module

Once filtered, the data is partitioned using the splitting module, which supports several strategies for creating train-test sets:

- **Temporal Splitting:** Partitions interactions based on timestamps. This method supports fixed time thresholds, optimal cutoff points, or a hold-out (HO) mechanism. HO approaches can rely on specified ratios or removing the most recent items.
- **Random Splitting:** Similar to temporal splitting, this method includes a hold-out option. Additionally, using K -repeated hold-out (K-HO) and cross-validation (CV), it can generate multiple train-test partitions.
- **Pre-computed Splitting:** Uses predefined data partitions, which is helpful for benchmarks that require consistent train-test sets.

When applicable, these strategies are applied in a **user-stratified manner**, meaning that the splitting logic is executed on each user's history individually rather than on the entire dataset.

8.3.5 Reproducibility

To ensure reproducibility in not-deterministic procedures, DATARec allows users to set a random seed. Additionally, DATARec transparently tracks every operation performed on any DataRec object, maintaining a complete history. This history can be

exported as a configuration file, which, when provided to `DATARec`, enables all previous operations to be reproduced accurately. Furthermore, each operation records the checksum of the `DataRec` object's state to verify its correctness. In Listing 1, an example configuration file is generated from a `DataRec` history.

8.4 Conclusion and Future Work

As observed from the related and recent literature on recommender systems, a critical limitation is represented by the lack of standardized procedures for recommendation data management, both in new proposed methodologies and in recommendation frameworks collecting them. To this end, we designed `DATARec` to empower recommender systems developers and foster ongoing discussions and research aimed at enhancing dataset management. It offers a comprehensive suite of resources and tools for data management and processing, with an emphasis on ease of use, seamless integration, reproducibility, and traceability of data sources. Furthermore, to promote standardization and code reuse, `DATARec` includes routines that facilitate integration with existing recommendation frameworks. As an open project, `DATARec` is committed to continuous improvement, evolving to incorporate emerging datasets and novel strategies. In the near future, we plan to extend this work with more in-depth analyses and enhanced support for integrated side information management.

Listing 1. example_configuration.yml.

```
pipeline:
- name: load
  operation: MovieLens
  params:
    version: 1m
  checksum: c4d9eecfca2ab87c1945afe126590906
- name: process
  operation: Binarize
  params:
    threshold: 4
  checksum: 0c5a5e05efb79e561a2d9c6b087980ff
- name: process
  operation: UserItemIterativeKCore
  params:
    cores: 2
  checksum: ef1a1bca94111c164d17b03a1a5c1314
- name: split
  operation: RandomHoldOut
  params:
    test_ratio: 0.2
    val_ratio: 0.1
    seed: 42
  checksum:
    test: 81e4150e5230a15d7c0d97b3371ffab1
    val: 65c04aa6c326c832891dfe4815465855
    train: 9a6760e3da74a1984d6d0057739b14ad
- name: export
  operation: Elliot
  params:
    output_path: ./elliot/
```

Part VI

Conclusion

Chapter 9

Closing Remarks

Traditional research in Recommender Systems (RSs) has long identified data sparsity as one of the most persistent and foundational challenges. The lack of sufficient user-item interactions severely degrades the quality of recommendations, limiting the effectiveness of Recommender Systems models. To overcome this hurdle, research has shifted towards integrating side information, with Knowledge Graphs (KGs) emerging as a valuable semantic resource. This dissertation positions itself within this context, addressing the multifaceted challenge of data sparsity through a structured journey from downstream structural mitigation to upstream semantic completion.

Structural Mitigation with Knowledge Graphs. The first part of this work focused on developing novel methodologies to leverage external knowledge to enhance recommendations. In Chapter 3, we introduced an innovative framework that learns user-aligned knowledge signals by modeling the individual decision process to achieve fine-grained personalization. This approach allowed for the enrichment of each interaction with an explicit semantic meaning. Building on this, Chapter 4 addressed the need for efficiency by proposing a lightweight simplification demonstrating that simpler, carefully designed architectures can achieve competitive or even superior performance.

Semantic Completion of Knowledge Graphs. Recognizing that the effectiveness of knowledge-aware models hinges on the quality of the underlying KG, the second part of this thesis shifted the focus upstream to tackle KG incompleteness. In chapter 5, we presented a novel inductive link prediction framework. This approach leverages Pre-trained Language Models (PLMs) to enrich node representations with implicit type signals, enabling robust semantic completion even in environments characterized by high structural and type sparsity. By doing so, it builds a more reliable semantic foundation, enhancing any downstream recommendation task that relies on KG features.

Operating under Imposed Data Sparsity. The third part of this research addressed the emerging challenge of data sparsity not as an intrinsic feature, but as an

imposed constraint. In chapter 6, we investigated how data minimization influences recommendation quality and the complex trade-offs that emerge between competing system goals. Chapter 7, deepened this analysis by developing an explanatory framework to identify the specific dataset characteristics that drive model performance in these resource-constrained settings, providing actionable guidelines for designing more effective minimization strategies.

Closing the Reproducibility Gap in Recommendation. Finally, recognizing that scientific progress is hindered by the lack of standardized data management practices, the final part of the thesis tackled this methodological challenge. In Chapter 8, we introduced an open-source library designed to standardize and make reproducible the data management in recommender systems.

Future directions. The researched presented in this dissertation, while addressing key aspect of data sparsity, simultaneously opens up several avenues for future investigation. Building upon the user-aligned knowledge distillation , future work could explore alternative formalism beyond decision trees to achieve a more nuanced understanding of user rationale. A significant extension would also be to incorporate temporal dynamics, modeling how a user’s decision criteria evolve over time. Shifting the focus from leveraging existing knowledge to enhancing its very foundation, the next logical step is to pursue a joint optimization framework where the KG completion process, as introduced in Chapter 5, is trained end-to-end with the downstream recommendation task. Finally, our analytical results on data minimization revealed the superior robustness of graph-based models under imposed sparsity and highlighted the key dataset characteristics that drive their performance. The natural evolution of this work is therefore twofold: first, to design novel minimization strategies specifically tailored to exploit the structural properties of GNNs; second, given the inherent resilience of knowledge-aware models, to extend these data minimization audits to KGCF architectures, creating a more complete picture of RSs in resource-constrained settings.

Chapter A

Appendix: Reconciling Latent Representations with Interpretability

This appendix investigates a challenge complementary to the main themes of this dissertation: the opacity of modern recommender systems, briefly introduced in Section 2.2.1.

Collaborative filtering is the most prominent paradigm in the realm of recommendation, having driven significant advancements in personalization by leveraging user-item interactions. Latent factor models within this framework have achieved state-of-the-art performance by capturing complex relationships in embedding spaces; however, their inherent opacity poses challenges for interpretability. In contrast, linear models offer transparent recommendations by explicitly modeling item-item relationships, though often at the cost of predictive performance and with potential scalability challenges during training. To bridge this gap, we introduce ILF-Rec, an Interpretable Linear Factor model for Recommendation that combines the strengths of both paradigms. ILF-Rec dynamically constructs user representations via a learnable, low-dimensional linear transformation of historical interactions, while preserving expressive latent item embeddings. This hybrid design guarantees interpretability without sacrificing scalability or accuracy. Extensive experiments on three real-world datasets demonstrate that ILF-Rec achieves competitive performance against strong collaborative filtering baselines. Additionally, both theoretical and empirical analyses support the interpretability of its recommendations.¹

1. The work presented in this chapter is being prepared for submission to the *49th ACM SIGIR Conference on Research and Development in Information Retrieval*. (SIGIR 2026) “Bridging Latent Embeddings and Interpretability with a Linear Factor Model for Recommender Systems”.

A.1 Introduction

Recommender Systems (RSs) are one of the most pervasive technologies across a wide range of real-world applications, including e-commerce, digital retail, media streaming platforms, and online advertising ecosystem [215]. Personalization lies at the core of their success, i.e., the system’s ability to surface relevant and engaging content tailored to individual users. Over the past two decades, Collaborative Filtering (CF) [83] has emerged as the dominant modeling paradigm in RSs, evolving significantly from its early reliance on neighborhood-based techniques [83, 198, 211] to more expressive approaches grounded in latent factor modeling [140, 197, 278]. This shift has been further amplified by the advent of deep learning, which has enabled models to capture non-linear user-item interactions beyond the capabilities of traditional matrix factorization. Latent factor models have consistently demonstrated state-of-the-art performance across diverse recommendation tasks [6, 10]. From classic Matrix Factorization [140] to more recent neural and graph-based models—including Variational Autoencoders [149, 214] and Graph Convolutional Networks [115, 168, 260, 262]—these methods provide not only superior accuracy but also strong scalability and flexibility. Their capacity to accommodate batch-wise training, incorporate complex optimization objectives, and serve as modular backbones for further extensions has cemented their role as foundational components in modern recommendation pipelines. However, this predictive power significantly compromises the interpretability of the recommendations [179]. Interpretability is more than a desirable property—it is becoming a functional requirement for modern RSs. Transparent models are essential for fostering user trust [306]: when users understand why an item is recommended, they are more likely to perceive the system as intelligent, fair, and aligned with their preferences. Latent factor models inherently operate in high-dimensional embedding spaces, which unfortunately hinders the transparency of their decision-making process for both end users and system designers. Recent efforts focusing on disentangling latent representations [159, 262] have made strides in attributing semantic meaning to components of the latent space. Nevertheless, the models remain opaque and difficult to audit or explain.

In contrast, linear models such as EASE^R [229] and SLIM [177] offer high interpretability by directly modeling item-item relationships through explicit correlation matrices. This approach improves interpretability at the recommendation level by expressing recommendations in terms of items the user has previously interacted with. However, adopting this strategy comes at the cost of sacrificing the rich representations learned by latent factor models. Moreover, learning the item-item correlation matrix imposes a significant computational burden, particularly when working with large item sets. This trade-off led us to ask: *Can we design a model that provides explicitly interpretable recommendations without compromising the capabilities of latent factor models?*

In this work, we take a step toward addressing this challenge by introducing

ILF-Rec, an Interpretable Linear Factor model for Recommendation. Our approach is designed to preserve the expressive power of latent factor models while delivering inherently interpretable recommendations. To achieve this, ILF-Rec dynamically constructs user representations by applying a linear transformation to the user’s historical interaction vector using a learnable, low-dimensional matrix. At the same time, it retains latent item representations to capture complex semantic relationships. This enables ILF-Rec to combine the key strengths of both linear and latent factor models. From the linear perspective, we provide a theoretical analysis showing that ILF-Rec recommendations can be expressed as explicit functions of a user’s past interactions, making the recommendation process transparent. From the latent factor side, the model learns expressive item embeddings, enabling both scalability and strong predictive performance.

We validate the effectiveness of ILF-Rec through extensive experiments on three real-world datasets, demonstrating that it achieves competitive accuracy compared to state-of-the-art CF baselines. Beyond accuracy, we showcase the model’s interpretability through a practical case study using real recommendations.

The main contributions of this study consist of:

- We propose ILF-Rec, a novel framework that unifies the strengths of linear and latent factor models, delivering recommendations that are simultaneously accurate, interpretable, and scalable.
- We provide a theoretical analysis demonstrating that ILF-Rec produces recommendations as an explicit function of a user’s interaction history, thereby offering formal guarantees of interpretability.
- We conduct extensive experiments on three real-world datasets, benchmarking ILF-Rec against eight strong CF baselines. In addition, we perform empirical analyses to support the architectural design of ILF-Rec and demonstrate the practical interpretability of its recommendations through a human-understandable case study.

A.2 Related Work

We provide a formal definition of the Collaborative Filtering (CF) problem, followed by an overview of the main categories of CF techniques. Finally, we investigate the interpretability of CF models.

A.2.1 Collaborative Filtering Techniques

Collaborative filtering (CF) [15] is a fundamental technique in recommendation systems that leverages collaborative information among users and items to predict user preferences. It is typically formulated by a user-item interaction matrix $R \in \mathbb{R}^{|\mathcal{U}| \times |\mathcal{I}|}$, where an entry $r_{u,i} = 1$ indicates that user u has interacted with item i , while $r_{u,i} = 0$ indicates no interaction. This binary user-item interaction framework

enables an implicit feedback recommendation scenario, where only engagement signals, such as whether a user clicked or purchased, are available rather than explicit feedback (e.g. user ratings or number of clicks).

Neighborhood-based Approaches.

Neighborhood-based [247] models estimate user preferences based on the similarity from neighboring users and/or items. Some of the earliest k -nearest-neighbor CF approaches, such as UserkNN [198] and ItemKNN [211], fall into this category. In the latter, the prediction for a user-item pair relies on the similarity between the target item and the items in the user history, which bears a direct analogy to our approach. Recent work [87] has shown that, despite their reliance on simple heuristics, nearest-neighbor methods are still capable of achieving performance comparable to more recent, neural-based models.

Matrix Factorization Methods.

Matrix Factorization (MF) [140] techniques decompose the user-item interaction matrix into two low-dimensional matrices, representing latent factors for users and items separately. BPRMF [195] introduces a sampling-based training procedure for MF models, minimizing a personalized ranking loss function through stochastic gradient descent. ENMF [46] is a neural matrix factorization framework that forgoes negative sampling by leveraging whole-data-based learning. NeuMF [117] combines the general framework of MF with neural networks to learn more complex, non-linear user-item interactions beyond what can be achieved through simple inner products. NeuWMF [118] extends WMF by introducing a weighting scheme to better handle the uncertainty inherent in implicit feedback data.

Autoencoder-based Models.

The autoencoder paradigm [214] has been investigated in the context of CF, framing the item recommendation task as the reconstruction of the user feedback matrix. Autoencoder models are typically trained end-to-end and can consist of multiple stacked layers, which encode a given input into a dense, potentially multi-layered, latent representation. Additionally, non-linearities can be incorporated into the encoding-decoding architecture to enhance representational capacity. Multi-VAE [149] samples a latent representation for a user, which is transformed via a non-linear function to a multinomial probability distribution, from which the sparse user implicit feedback is assumed to be sampled. MacridVAE [159] is a variational autoencoder that enforces disentanglement over the latent representations, so that each latent factor refers to a distinct low-level factor.

In contrast, linear autoencoder models, such as SLIM [177] and EASE^R [229], reconstruct the binary user-item interaction matrix as the product of the sparse user-item binary interaction matrix and a learnable item-item correlation matrix.

The final reconstructed feedback for a target item is, therefore, a linear combination of user feedbacks on all the other items, weighted by the item-item correlations. ELSA [243] builds upon the concept of linear autoencoder by introducing a low-rank approximation of the full item-item correlation matrix learning a single low-dimensional matrix, significantly reducing the number of parameters and improving efficiency for large-scale recommendation tasks.

Other Approaches.

Graph-based approaches for CF model user-item interactions as a bipartite, undirected graph [15]. Random walk [57] CF models frame recommendation as n-hop random walks on the interaction graph. Notably, RP_β^3 [57] computes 3-hop random walks on the user-item graph, adjusting the transition probabilities with an inverse popularity weighting to mitigate the dominance of popular items and promote the visibility of long-tail items. Building on the graph-based CF paradigm, Graph Convolutional Networks (GCN), such as NGCF [260], learn latent user and item representations by propagating information across multiple layers of neighboring nodes, effectively capturing high-order relationships that traditional CF methods are unable to model. LightGCN [115] improves upon NGCF by replacing complex operations like feature transformations and non-linear activations with simple weighted sum aggregations, achieving state-of-the-art performance. DGCF [262] learns disentangled representations by modeling a distribution over intents for each user-item interaction. UltraGCN [168] improves upon LightGCN by avoiding explicit multi-layer message passing, approximating infinite-layer graph convolutions through a constraint loss. SimpleX [166] combines the simplicity of matrix factorization with an aggregation mechanism that incorporates interacted items using either mean pooling or graph attention [245], designed to enable finer user latent representations.

A.2.2 Interpretability in Collaborative Filtering

Interpretability in RSs refers to the ability to understand why a model recommends certain items to users, which is crucial for user trust and model refinement. While many CF approaches achieve interpretability through ad-hoc mechanisms [85, 305], training constraints [159] or post-hoc analysis [52], we focus on interpretability that arises from the intrinsic properties of the scoring function or model architecture. We can broadly classify existing interpretability approaches in two categories: *feature-level* and *recommendation-level*. While *feature-level* interpretability refers to the ability to link latent features to human-interpretable features, *recommendation-level* interpretability refers to the ability to explain specific recommendations in terms of which users/items influenced the decision. Many CF models, such as matrix factorization and neural-based approaches, remain difficult to interpret due to their reliance on low-dimensional latent representations or complex, non-linear transformations, which obscure their decision-making process [43, 184]. Despite this challenge, non-

linear models like MacridVAE [159] are capable of providing interpretability at a feature-level by enforcing feature disentanglement in their training procedure. Differently, linear autoencoder models such as EASE^R [229] and SLIM [177], which learn a full item-item correlation matrix, are inherently interpretable at a recommendation-level, as the final score is obtained as a sum of item-item correlations. We show (Section A.3.4) that, while still learning low-dimensional latent features, our model maintains the advantage of providing interpretable recommendations.

A.3 Methodology

This section introduces the Interpretable Linear Factor model for Recommendation (ILF-Rec). Firstly, we define the recommendation problem. Then, we detail the architecture of ILF-Rec and a theoretical demonstration of its interpretability. Finally, we conduct ILF-Rec’s analysis shedding light on its relationships with state-of-the-art models.

A.3.1 Problem Definition

Let $\mathcal{U} = \{u_1, \dots, u_{|\mathcal{U}|}\}$ be the set of $|\mathcal{U}|$ users and $\mathcal{I} = \{i_1, \dots, i_{|\mathcal{I}|}\}$ be the set of $|\mathcal{I}|$ items. We observe unary feedback data in the form of a binary interaction matrix $R \in \{0, 1\}^{|\mathcal{U}| \times |\mathcal{I}|}$ [129], where an element $r_{u,i}$ of R is:

$$r_{u,i} = \begin{cases} 1 & \text{if user } u \text{ has interacted with item } i, \\ 0 & \text{otherwise.} \end{cases} \quad (\text{A.1})$$

Our objective is to learn a parametric function $s_{u,i} = f(u, i; \Theta)$ that predicts a relevance score of item i for user u , where Θ denotes the set of model parameters.

A.3.2 Proposed Approach

Our goal is to design a model that delivers interpretable recommendations while retaining the expressive power of latent representations for capturing user intents. ILF-Rec achieves this by introducing a novel user representation that enhances interpretability in relevance prediction.

User Representation

Instead of learning user embeddings directly as in conventional factorization-based RSs—a process that limits interpretability—ILF-Rec computes the user embedding by applying a linear transformation to the user’s interaction vector.

Formally, let $\mathbf{r}_u \in \{0, 1\}^{1 \times |\mathcal{I}|}$ be the u -th row of the interaction matrix R corresponding to the user u . We define a linear transformation matrix $W \in \mathbb{R}^{k \times |\mathcal{I}|}$, where k is a hyper-parameter denoting the dimensionality of the latent space. The user

embedding e_u is computed as:

$$e_u = (\beta_u \odot r_u) \cdot W^T, \quad (\text{A.2})$$

where “ $\odot$ ” is the element-wise product, “ $\cdot$ ” is the dot product, and $\beta_u \in \mathbb{R}^{1 \times |\mathcal{I}|}$ is a vector whose element $\beta_{u,i}$ is a scaling factor of the interaction $r_{u,i} \in r_u$ between the user u and the item i . The rationale behind β_u is that raw interaction vectors r_u are inherently skewed by disparities in item popularity. Consequently, interactions involving highly popular items can dominate the learning process. This disproportionate influence overshadows less frequent but potentially significant interactions, causing the resulting user embedding e_u to reflect popular trends rather than capturing the nuanced individual preferences of the user. The vector β_u aims to weigh each interaction, ensuring a more equitable contribution across items.

To operationalize β_u , we explore three distinct scaling schemes inspired by Kipf et al. [137], each addressing different sources of bias and imbalance. Let N_u denote the number of items user u has interacted with, and N_i the number of users who have interacted with item i . We define each i -th element $\beta_{u,i} \in \beta_u$ as follows:

$$\beta_{u,i} = \begin{cases} \frac{1}{\sqrt{|N_u||N_i|}} & \text{for symmetric scaling,} \\ \frac{1}{\sqrt{|N_i|}} & \text{for item-centric scaling,} \\ \frac{1}{\sqrt{|N_u|}} & \text{for user-centric scaling.} \end{cases} \quad (\text{A.3})$$

The symmetric scaling approach jointly considers user activity and item popularity to modulate interaction weights. In contrast, the item-centric scaling isolates the influence of item popularity, compensating for variations in the number of interactions received by each item. Lastly, user-centric scaling accounts solely for the user’s interaction frequency, addressing discrepancies in user engagement levels across the dataset.

Item Representation

ILF-Rec adopts classic latent representations for item embeddings to enhance the predictive power. We associate each item i with a latent embedding $e_i \in E_i$ where $E_i \in \mathbb{R}^{|\mathcal{I}| \times k}$ is the matrix of item embeddings and k is the same hyper-parameter defined for W .

Scoring Function

The relevance score $s_{u,i}$ for user u and item i is calculated as the inner product of the user ($e_u \in \mathbb{R}^{1 \times k}$) and item ($e_i \in \mathbb{R}^{1 \times k}$) embeddings:

$$s_{u,i} = e_u \cdot e_i^T = ((\beta_u \odot r_u) \cdot W^T) \cdot e_i^T. \quad (\text{A.4})$$

This score is used for ranking the items at recommendation time. We theoretically demonstrate the interpretability of this scoring function in Section A.3.4.

A.3.3 Training Objectives

Let $\Theta = \{W, E_i\}$ denote the trainable parameters of ILF-Rec. To optimize the model using a contrastive approach, we adopt the Bayesian Personalized Ranking (BPR) [195] loss function. This pairwise ranking method encourages the model to assign a higher relevance score to a positive item than to a negative one for any given user.

Formally, let $\mathcal{D} = \{(u, i, j) \mid r_{u,i} = 1, r_{u,j} = 0\}$ be the set of training triplets. The BPR loss function is defined as:

$$\mathcal{L}_{\text{BPR}}(\Theta) = - \sum_{(u,i,j) \in \mathcal{D}} \log \sigma(s_{u,i} - s_{u,j}), \quad (\text{A.5})$$

where $\sigma(\cdot)$ denotes the sigmoid function and $s_{u,i}$ is the predicted relevance for user-item pair (u, i) .

Additionally, inspired by Raziperchikolaei et al. [191], we apply an orthogonality loss to the transformation matrix W by penalizing deviations from $WW^T \approx I$. This constraint enforces the rows of W to be orthonormal, thus ensuring that each factor of the user embedding e_u represents a distinct and non-redundant facet of the user's interaction history. The orthogonality loss is defined as:

$$\mathcal{L}_{\text{ORTHO}}(W) = \|WW^T - I\|_F^2, \quad (\text{A.6})$$

where $\|\cdot\|_F$ refers to the Frobenius norm of a matrix.

Overall, we minimize the previously discussed losses by adding an L2-regularization term to mitigate overfitting:

$$\mathcal{L}(\Theta) = \mathcal{L}_{\text{BPR}}(\Theta) + \lambda_o \mathcal{L}_{\text{ORTHO}}(W) + \lambda_r \|\Theta\|_2^2, \quad (\text{A.7})$$

where λ_o and λ_r are two hyper-parameters controlling the orthogonality loss and the regularization term, respectively. The general training procedure of ILF-Rec is shown in Algorithm 1.

A.3.4 Interpretability of the Scoring Function

Beyond training objectives, we now theoretically investigate the interpretability of the ILF-Rec scoring function. Specifically, we show how Eq. (A.4)—and consequently the ILF-Rec recommendations—can be interpreted on the basis of the user's history.

Let us consider a user-item pair (u, i) and its predicted score $s_{u,i}$ computed with Eq. (A.4). We define $w_{x,y}$ as the element in the x -th row and y -th column of matrix W , and e_{i_x} as the x -th element of the embedding e_i of item i .

Theorem A.1 (Interpretability). *The ILF-Rec predicted relevance score $s_{u,i}$ for user u and item i is a linear combination of the items the user u has interacted with:*

$$s_{u,i} = \sum_{y=0}^{|I|} r_{u,y} \alpha_{i_y}, \quad (\text{A.8})$$

where $r_{u,y}$ is 1 if user u interacted with item y , 0 otherwise, and $\alpha_{i_y} = \beta_{u_i} \sum_{x=0}^k e_{i_x} w_{x,y}$.

Algorithm 1: Training Procedure of ILF-Rec

Inputs : $R \leftarrow$ interaction matrix
 $k \leftarrow$ dimension of W and E_i
 $\lambda_r \leftarrow$ regularization factor
 $\lambda_o \leftarrow$ orthogonality loss factor
 $n_{epoch} \leftarrow$ number of epochs
 $n_{batch} \leftarrow$ dimension of batches
 $\mathcal{D} \leftarrow$ set of training triplets

Output: Θ

- 1 Random initialization: $\Theta = \{W, E_i\}$ // Model parameters
- 2 **for** $epoch \in \{1, \dots, n_{epoch}\}$ **do**
- 3 sample $\mathcal{D}_{batch} \in \mathcal{D}$ of dimension n_{batch}
- 4 **for** $\{u, i, j\} \in \mathcal{D}_{batch}$ **do**
- 5 compute β_u (Eq. (A.3))
- 6 compute $e_u = (\beta_u \odot r_u) \cdot W^\top$ (Eq. (A.2))
- 7 compute $s_{u,i} = e_u \cdot e_i^\top$
- 8 compute $s_{u,j} = e_u \cdot e_j^\top$
- 9 construct $\mathcal{L}(\Theta) = \mathcal{L}_{BPR}(\Theta) + \lambda_o \mathcal{L}_{ORTHO}(W) + \lambda_r ||\Theta||_2^2$
- 10 compute $\nabla_{\Theta} \mathcal{L}(\Theta)$
- 11 update Θ // Back-propagation

Proof. The predicted relevance score $s_{u,i}$ for user u and item i is computed through Eq. (A.4), where the user embedding e_u is defined as $(\beta_u \odot r_u)$ (see Eq. (A.2)). Consequently, the generic x -th factor e_{u_x} of e_u , with $x \in \{1, \dots, k\}$ is computed as:

$$e_{u_x} = \sum_{y=0}^{|I|} w_{x,y} \beta_{u_y} r_{u,y}. \quad (\text{A.9})$$

Let us consider the generic x -th factor e_{i_x} of e_i , $x \in \{1, \dots, k\}$. Explicitly, the score $s_{u,i}$ is computed as:

$$s_{u,i} = e_u \cdot e_i^\top = \sum_{x=0}^k e_{i_x} \left(\sum_{y=0}^{|I|} w_{x,y} \beta_{u_y} r_{u,y} \right) = \quad (\text{A.10})$$

$$= \sum_{y=0}^{|I|} \sum_{x=0}^k e_{i_x} w_{x,y} \beta_{u_y} r_{u,y} = \sum_{y=0}^{|I|} r_{u,y} \underbrace{\sum_{x=0}^k e_{i_x} w_{x,y} \beta_{u_y}}_{\alpha_{i_y}} = \quad (\text{A.11})$$

$$= \sum_{y=0}^{|I|} r_{u,y} \alpha_{i_y}. \quad (\text{A.12})$$

□

From the formula above, we have that each recommendation score is explicitly expressed as a weighted sum of the user’s past interactions. In particular, each term $\alpha_{i,y}$ in the summation quantifies the contribution of a previously consumed item y to the score of the suggested item i . This observation facilitates a straightforward post-hoc analysis, allowing one to identify and rank the contributions of individual interactions, thus enhancing model transparency and enabling the generation of explanations.

A.3.5 Model Analysis

In this section, we analyze the space complexity of ILF-Rec and examine its relation with existing state-of-the-art CF approaches.

ILF-Rec Model Complexity

The trainable parameters of ILF-Rec are $W \in \mathbb{R}^{k \times |\mathcal{I}|}$ and $E_i \in \mathbb{R}^{|\mathcal{I}| \times k}$. Thus, the space complexity of ILF-Rec is $2|\mathcal{I}| \times k$, resulting in $\mathcal{O}(|\mathcal{I}|)$ by treating the constants k and 2 as negligible compared to $|\mathcal{I}|$.

Relation to ItemKNN

ILF-Rec shares conceptual similarities with ItemKNN [211], computing user-item scores by summing the similarities between the target item and its top-K most similar items. In contrast, our approach aggregates correlations with all items (see Eq. (A.8)), eliminating the need for a fixed K, which can be dataset-dependent. Furthermore, ItemKNN requires precomputing and storing a full $|\mathcal{I}| \times |\mathcal{I}|$ similarity matrix, resulting in a space complexity of $\mathcal{O}(|\mathcal{I}|^2)$. Instead, our model captures item-item interactions implicitly through learnable transformations in a latent k -dimensional space, offering greater scalability. By not relying on a predefined similarity metric, our approach enables flexible modeling of item relationships via learnable transformations.

Relation to MF

Our model closely aligns with Matrix Factorization (MF) [140], as MF represents items and users in a shared latent space with item embeddings $E_i \in \mathbb{R}^{|\mathcal{I}| \times k}$ and user embeddings $E_u \in \mathbb{R}^{|\mathcal{U}| \times k}$, resulting in a space complexity of $\mathcal{O}(|\mathcal{I}| + |\mathcal{U}|)$. Unlike traditional MF, which learns both user and item embeddings, our approach learns only item representations, deriving user embeddings dynamically from interaction histories via a transformation matrix $W \in \mathbb{R}^{k \times |\mathcal{I}|}$. This design is particularly advantageous in real-world settings, where the user set is typically larger and more dynamic than the item set, enabling efficient, on-the-fly user representation.

Table A.1. Comparison of CF baselines in terms of scoring function and space complexity (S.C.). We refer the reader to the original papers for additional details.

Model	Scoring Function ($s_{u,i} = f(u, i; \Theta)$)	S.C.
RP $^3_\beta$ [57]	$\sum_{j=1}^{ I } \sum_{v=1}^{ \mathcal{U} } \frac{a_{ju} \cdot a_{jv} \cdot a_{iv}}{d_{ii}^{1-\lambda} \cdot d_{jj}^\lambda \cdot d_{vv}}$	$O(I ^2)$
UserKNN [198]	$\sum_{v \in \mathcal{N}_k(u)} \text{sim}(u, v) \cdot r_{vi}$	$O(\mathcal{U} ^2)$
ItemKNN [211]	$\sum_{v \in \mathcal{N}_k(i)} \text{sim}(i, v) \cdot r_{uv}$	$O(I ^2)$
SLIM [177]	$\sum_{j \in \mathcal{I}_u} B_{ij}$	$O(I ^2)$
EASE ^{R} [229]	$\sum_{j \in \mathcal{I}_u} B_{ij}$	$O(I ^2)$
MultiVAE [149]	$f_{\text{dec}}(z_u)_i; z_u = f_{\text{enc}_\mu}(x_u) + \epsilon \odot f_{\text{enc}_\sigma}(x_u); \epsilon \sim \mathcal{N}(0, \mathbf{I}_k)$	$O(I)$
BPRMF [195]	$\mathbf{e}_u^\top \cdot \mathbf{e}_i$	$O(\mathcal{U} + I)$
LightGCN [115]	$\mathbf{e}_u^\top \cdot \mathbf{e}_i; \mathbf{e}_u = \sum_{l=0}^L \alpha_l \cdot \mathbf{e}_u^{(l)}; \mathbf{e}_i = \sum_{l=0}^L \alpha_l \cdot \mathbf{e}_i^{(l)}$	$O(\mathcal{U} + I)$
ILF-Rec	$\mathbf{e}_u \cdot \mathbf{e}_i^\top = ((\beta_u \odot \mathbf{r}_u) \cdot \mathbf{W}^\top) \cdot \mathbf{e}_i^\top$	$O(I)$

Relation to Shallow Linear Autoencoders.

Our score function can be viewed as a sum of different item contributions in the user history (see Eq. (A.8)). This has direct analogies to the scoring function on linear shallow autoencoders, such as SLIM [177], EASE ^{R} [229], and ELSA [243]. Unlike SLIM and EASE ^{R} , which have a space complexity of $O(|I|^2)$, our model does not rely on a full item-item correlation matrix, significantly reducing space and computational requirements, particularly for large item sets. Differently from ELSA, our method factorizes the item-item correlation matrix via two independent learnable matrices, enhancing representation expressiveness.

A.4 Experimental Setup

We conducted a comprehensive experimental evaluation to rigorously assess the proposed ILF-Rec framework and address several key research questions. In this section we detail the experimental setup employed to investigate the following research questions:

- RQ1:** How does ILF-Rec perform compared to state-of-the-art collaborative filtering approaches?
- RQ2:** How do the different components of ILF-Rec impact its performance?
- RQ3:** To what extent the hyper-parameters of ILF-Rec affect its performance?
- RQ4:** Can ILF-Rec provide easily interpretable recommendations?

Table A.2. Performance comparison of ILF-Rec with the baselines. Boldface and underline refer to best and second-to-best values, respectively. All the differences between ILF-Rec and the baselines are statistically significant based on paired t-tests ($p < 0.05$) except for the values denoted with $\dagger$.

Model	LastFM			ML-1M			AS&O		
	nDCG@10	Recall@10	Precision@10	nDCG@10	Recall@10	Precision@10	nDCG@10	Recall@10	Precision@10
UserKNN	0.2168	0.2141	0.1606	0.2922	0.1309	0.2654	0.0277	0.0398	0.0074
ItemKNN	0.2208	0.2192	0.1632	0.2744	0.1169	0.2518	0.0335	0.0476	0.0098
BPRMF	0.2239	0.2196	0.1638	0.2914	0.1300	0.2648	0.0228	0.0330	0.0069
SLIM	0.1848	0.1814	0.1397	0.1949	0.0707	0.1745	0.0179	0.0245	0.0052
MultiVAE	0.2326	0.2279	0.1699	0.2818	0.1305	0.2570	0.0248	0.0381	0.0077
RP $^3_\beta$	0.1359	0.1473	0.1075	0.2740	0.1213	0.2542	0.0069	0.0116	0.0022
EASE ^{R}	0.2089	0.2041	0.1561	0.3121	0.1403†	0.2841	0.0360†	0.0517	<u>0.0106†</u>
LightGCN	0.2443†	0.2416†	0.1794†	0.2966	0.1344	0.2694	0.0344	<u>0.0528†</u>	0.0105 †
ILF-Rec	<u>0.2428</u>	<u>0.2391</u>	<u>0.1776</u>	<u>0.3073</u>	<u>0.1402</u>	<u>0.2770</u>	<u>0.0356</u>	0.0534	0.0106

A.4.1 Datasets

We evaluate ILF-Rec on three widely used benchmark datasets spanning different domains and data characteristics: MovieLens 1M (ML-1M), HetRec LastFM (LastFM), and Amazon Sports & Outdoors (AS&O), which represent the movie, music, and e-commerce domains, respectively. Following standard practice in recommender systems evaluation [125], we binarize explicit feedback by treating all observed interactions as positive implicit signals.

Given the differences in sparsity and user-item interaction patterns across datasets, we apply tailored p -core filtering strategies to ensure meaningful comparisons and a comprehensive assessment of ILF-Rec’s robustness across varying levels of interaction density:

- **ML-1M** [113]: We apply iterative 5-core filtering, retaining users and items with at least five interactions each. The resulting dataset contains 6,040 users, 3,416 items, and 999,611 interactions, with a density of 4.85%. This dataset serves as our representative of dense interaction scenarios.
- **LastFM** [26]: We use iterative 10-core filtering, resulting in a dataset with 1,797 users, 1,507 items, and 62,376 interactions, corresponding to a density of 2.30%. This setup allows us to assess performance under moderate sparsity.
- **AS&O** [169]: We apply iterative 5-core filtering, yielding a dataset with 35,598 users, 18,357 items, and 296,337 interactions. With an extremely low density of 0.045%, it poses a highly challenging scenario for recommendation in sparse environments.

A.4.2 *Baselines*

To assess the effectiveness of the proposed ILF-Rec model, we compare its performance with a diverse set of state-of-the-art baseline methods that span multiple recommendation paradigms, including:

- **RP³** [57] exploits random walk-based methods to estimate relevance in the user-item graph.
- **UserKNN** [198] is a nearest-neighbor (NN) approach that computes scores by aggregating the preferences of similar users.
- **ItemKNN** [211] is a NN method that recommends items using item similarities derived from user co-interaction patterns.
- **SLIM** [177] is a sparse linear method that directly learns an item-item similarity matrix.
- **EASE^R** [229] is a linear reconstruction model that computes a closed-form item-item similarity matrix.
- **MultiVAE** [149] employs a variational autoencoder framework to learn non-linear latent representations of user preferences, capturing complex relationships within a probabilistic model.
- **BPRMF** [195] is a MF method that optimizes a pairwise ranking objective via BPR loss function. It learns latent user and item representations, serving as a classic latent factor baseline.
- **LightGCN** [115] is a graph-based model that simplifies traditional GCN by removing feature transformation and non-linear activation functions, effectively capturing the collaborative signal.

A comparison of these baselines in terms of scoring function and space complexity is provided in Table A.1.

A.4.3 *Reproducibility, Evaluation, and Metrics*

In the following, we detail the evaluation protocol, hyper-parameter optimization, and performance metrics. All models are implemented and trained using the Elliot reproducibility framework [9], ensuring consistent experimental conditions and standardized evaluation.

Evaluation Protocol

We adopt the all-unrated-items protocol [227], treating any item not previously interacted with by a user as a recommendable candidate. Each dataset in the implicit feedback setting is partitioned using a 60-20-20 splitting hold-out strategy. The models are trained with a fixed batch size of 1024 and a maximum of 1000 epochs. They are evaluated every epoch by selecting the best ones on nDCG@10 computed on the validation set, with early stopping applied using a patience of 5 epochs.

Reproducibility Details

Models’ hyper-parameters are tuned via grid search to ensure a fair comparison across methods. For models adopting latent factors (i.e., LightGCN, BPRMF, ILF-Rec), we tune the embedding size in $\{64, 128, 256\}$, the learning rates in $\{1e^{-2}, 1e^{-3}, 1e^{-4}\}$, and regularization weights in $\{1e^{-4}, 1e^{-5}, 1e^{-6}, 1e^{-7}\}$. In addition, we set the number of layers in LightGCN equal to 2 or 3. Regarding neighborhood-based methods—UserKNN and ItemKNN—the neighborhood size is tuned in $\{20, 30, 50, 100\}$, with similarity measures alternating between cosine similarity and dot product. EASE^R optimizes its regularization factor λ within $\{100, 500, 1000\}$. For RP_β^3 , the neighborhood size is searched over $\{10, 100, 200, 300\}$, with $\alpha \in \{0.6, 1.0, 1.5, 2.0\}$ and beta in $\beta \in \{0.8, 1.5\}$. SLIM adjusts its L1 and L2 regularization parameters in $\{1e^{-4}, 1e^{-5}, 1e^{-6}\}$. For ILF-Rec, the orthogonality loss weight λ_o is tuned in $\{1e^{-3}, 1e^{-4}, 1e^{-5}, 1e^{-6}\}$. Finally, MultiVAE optimizes its latent dimensionality within $\{100, 200, 300, 400\}$ and hidden layer sizes within $\{400, 500, 600, 700, 800\}$.

Metrics

The final performance of the models is assessed on the test set using three standard accuracy metrics, namely nDCG, Precision, and Recall. All metrics are computed with a cutoff of 10, focusing on the top 10 recommended items per user.

A.5 Results and Discussion

In this section, we present a comprehensive evaluation of ILF-Rec, organized around our four key research questions. First, we compare its recommendation accuracy against a diverse set of state-of-the-art collaborative filtering methods (**RQ1**) to examine ILF-Rec’s effectiveness in producing reliable and accurate recommendations. Next, we conduct ablation studies (**RQ2**) to assess the impact of critical architectural components. This is followed by a hyper-parameter analysis (**RQ3**) that explores the sensitivity of ILF-Rec to key configuration choices, to further demonstrate its robustness and adaptability. Finally, we examine the interpretability of ILF-Rec (**RQ4**) to highlight how its design facilitates transparent recommendations.

A.5.1 Performance Comparison (RQ1)

This section is devoted to comparing the performance of our model ILF-Rec against the baselines across all three datasets. Table A.2 presents the overall accuracy performance comparison.

The empirical results suggest that ILF-Rec achieves highly competitive performance compared to a diverse set of established collaborative filtering approaches. Indeed, across all three datasets, ILF-Rec consistently ranks as either the best or the

second-best model for every evaluation metric (nDCG@10, Recall@10, Precision@10). Notably, ILF-Rec outperforms traditional neighborhood-based methods (UserKNN, ItemKNN), latent factor models (BPRMF, MultiVAE), and simpler linear models (SLIM, $RP^3\beta$). This substantial performance gap underscores the effectiveness of ILF-Rec’s hybrid architecture, which successfully balances robust predictive accuracy with interpretability. Regarding the remaining baselines, while specific state-of-the-art models (e.g., LightGCN or EASE^R) occasionally achieve higher scores, the differences relative to ILF-Rec are minimal and often not statistically significant. In dense environments like ML-1M, although EASE^R achieves marginally higher performance, ILF-Rec still delivers comparable results, and in sparser contexts such as LastFM and AS&O, ILF-Rec not only surpasses EASE^R in key metrics but also matches or exceeds the performance of LightGCN. These results indicate that ILF-Rec exhibits a consistently competitive performance, underscoring its robustness across dense and sparse environments.

In addressing RQ1, our findings show that ILF-Rec is a recommender that reliably ranks among the top performers under all tested conditions. By bridging the gap between latent factor models and interpretable linear approaches, ILF-Rec delivers high predictive accuracy alongside clear, interpretable recommendations—a combination making it a robust and versatile framework for recommendation tasks.

A.5.2 Ablation Studies (RQ2)

This section investigates the rationale behind the architecture of ILF-Rec. We assess the impact of key components of ILF-Rec on its accuracy performance, comprising the role of the orthogonality loss, the behavior of the different mechanisms of scaling the user-item interactions, and the usefulness of adopting dedicated latent representations for the items.

Impact of Orthogonality Loss

As detailed in Section A.3.3, the orthogonality loss ($\mathcal{L}_{\text{ORTHO}}$) aims to promote representational diversity within user embeddings by penalizing correlations between latent factors. The underlying hypothesis is that encouraging factor distinctness leads to more informative representations of user preferences. To validate this hypothesis, we conducted an ablation experiment where we removed the influence of $\mathcal{L}_{\text{ORTHO}}$ by setting its weight λ_o to zero and trained this version of the model (ILF-Rec w/o $\mathcal{L}_{\text{ORTHO}}$). We then compared its performance against the complete ILF-Rec configuration (ILF-Rec w/ $\mathcal{L}_{\text{ORTHO}}$) across all three datasets and reported this results in Figure A.1. A clear and consistent trend emerges: the full ILF-Rec consistently outperforms the variant lacking the orthogonality loss. This outcome empirically confirms the utility of the $\mathcal{L}_{\text{ORTHO}}$ component in improving the recommendation quality and highlighting the significant positive contribution of the orthogonality constraint to the overall effectiveness of ILF-Rec.

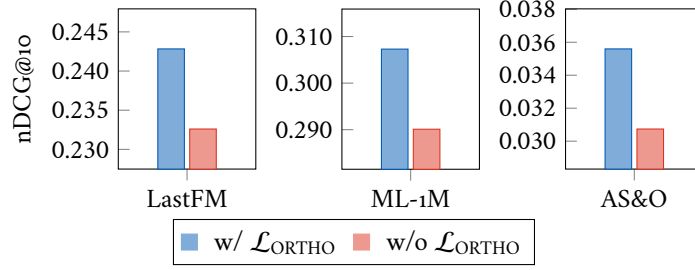


Figure A.1. Performance comparison of ILF-Rec with the full formulation (w/ $\mathcal{L}_{\text{ORTHO}}$) versus its variant excluding the orthogonality term (w/o $\mathcal{L}_{\text{ORTHO}}$). All differences are statistically significant, as determined by paired t-tests ($p < 0.05$).

Table A.3. nDCG@10 performance comparison of ILF-Rec under different scaling schemes. Results are shown for no scaling (ILF-Rec w/ $\beta_{u_i} = 1$), symmetric scaling (ILF-Rec w/ $\beta_{u_i} = 1/\sqrt{|\mathcal{N}_u||\mathcal{N}_i|}$), user-centric scaling (ILF-Rec w/ $\beta_{u_i} = 1/\sqrt{|\mathcal{N}_u|}$), and item-centric scaling (ILF-Rec w/ $\beta_{u_i} = 1/\sqrt{|\mathcal{N}_i|}$). Differences relative to the no-scaling variant are statistically significant ($p < 0.05$, paired t-test), except for values marked with $\dagger$. Boldface denotes the best nDCG values.

Model	LastFM	ML-1M	AS&O
ILF-Rec w/ $\beta_{u_i} = 1$	0.2368	0.3014	0.0310
ILF-Rec w/ $\beta_{u_i} = \frac{1}{\sqrt{ \mathcal{N}_u \mathcal{N}_i }}$	0.2356 †	0.2947	0.0354
ILF-Rec w/ $\beta_{u_i} = \frac{1}{\sqrt{ \mathcal{N}_u }}$	0.2395 †	0.3061	0.0309 †
ILF-Rec w/ $\beta_{u_i} = \frac{1}{\sqrt{ \mathcal{N}_i }}$	0.2428	0.3073	0.0356

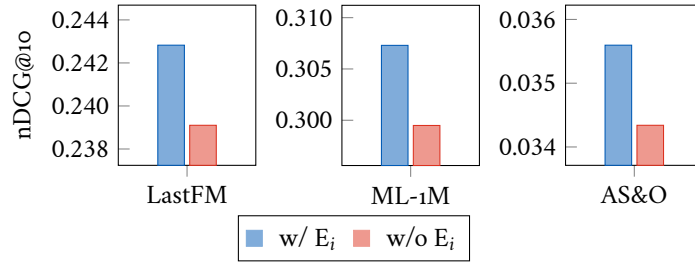


Figure A.2. Performance comparison of ILF-Rec with and without dedicated item embeddings. The full model (w/ E_i) employs a separate item embedding matrix, whereas the ablated variant (w/o E_i) uses the transformation matrix W for both projection and item representation. All differences are statistically significant, as determined by paired t-tests ($p < 0.05$).

Impact of Scaling of scaling factor β_{u_i}

Section A.3.2 introduces a scaling factor β_{u_i} to weigh user-item interactions, aiming to mitigate the overwhelming influence of popular items on the model's training.

Our hypothesis is that such scaling enhances representation learning and consequently improves recommendation accuracy. To validate this and determine the most effective scaling strategy, we compared the effectiveness of different formulations of β_{u_i} . Specifically, we evaluated ILF-Rec variants using: (1) no scaling ($\beta_{u_i} = 1$), (2) symmetric scaling ($\beta_{u_i} = 1/\sqrt{|\mathcal{N}_u||\mathcal{N}_i|}$), (3) user-centric scaling ($\beta_{u_i} = 1/\sqrt{|\mathcal{N}_u|}$), and (4) item-centric scaling ($\beta_{u_i} = 1/\sqrt{|\mathcal{N}_i|}$). Table A.3 reports the nDCG@10 results on the three datasets. The findings clearly favor the item-centric scaling approach, which consistently outperforms all other variants. Crucially, the model without any scaling (ILF-Rec w/ $\beta_{u_i} = 1$) achieves less optimal performance on any dataset often ranking last among the variants. Even when it is superior, the differences are not statistically significant. This analysis underscores the importance of employing a suitable scaling strategy to enhance the model’s accuracy.

Impact of Item Embeddings

ILF-Rec employs a transformation matrix W that projects the user’s interaction vector into the latent space and a dedicated item embedding matrix E_i . Our rationale for introducing distinct matrices lies in the hypothesis that decoupling these components provides greater representational capacity than models that conflate their roles such as ELSA [243]. Specifically, W focuses on how items contribute to user profiles, while E_i on modeling the items themselves.

To empirically validate the contribution of these dedicated model’s parameters, we conduct an ablation study comparing the full model, i.e., ILF-Rec w/ E_i against its variant, denoted as ILF-Rec w/o E_i , where E_i is omitted. In this setting, the columns of W share conceptual similarities with ELSA and play a dual role acting both as transformation weights and item embeddings. Formally, the user-item score $s_{u,i}$ in Eq. (A.4) is computed by using W as both transformation matrix and the source of item factors:

$$s_{u,i} = ((\beta_u \odot r_u) \cdot W^T) \cdot W_{:,i},$$

where $W_{:,i}$ represents the i -th column of the matrix W .

The results in Figure A.2 show consistent performance advantages of the full ILF-Rec across all datasets, thus demonstrating the benefit of maintaining separate matrices for user embedding transformation and item representation.

*Answering **RQ2**, the ablation studies systematically validate our core design choices, demonstrating that the orthogonality loss, the interaction scaling, and the dedicated item embeddings each provide significant contributions to the overall performance of ILF-Rec.*

A.5.3 Hyper-parameter Analysis (RQ3)

This section studies to what extent ILF-Rec is robust to its hyper-parameters tuning. A desirable property of any recommendation model is maintaining strong performance even without precise calibration of its hyper-parameters. Figure A.3 reports the nDCG scores obtained by varying the values for the dimensionality of item

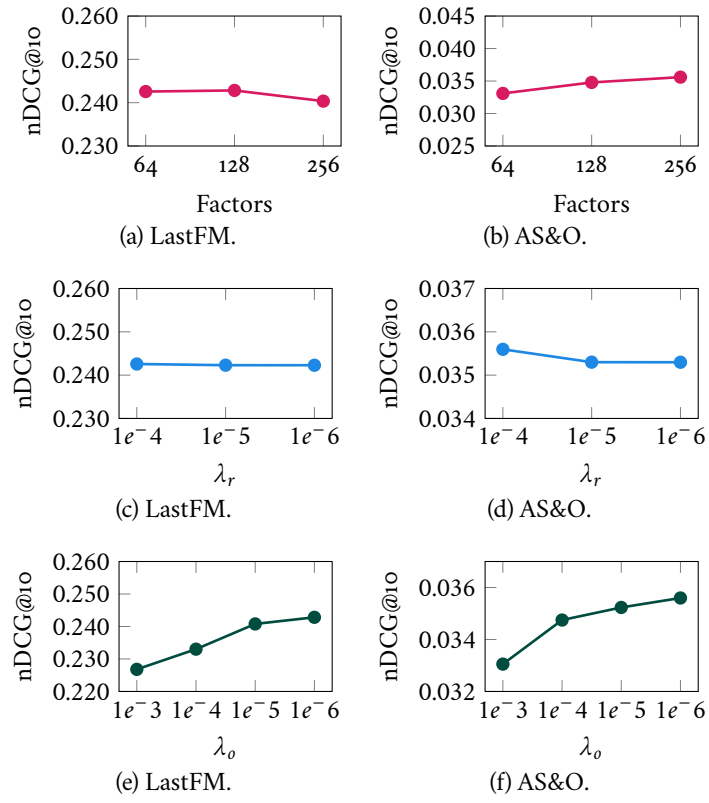


Figure A.3. nDCG@10 performance of ILF-Rec by varying the number of factors k and the coefficients λ_r and λ_o .

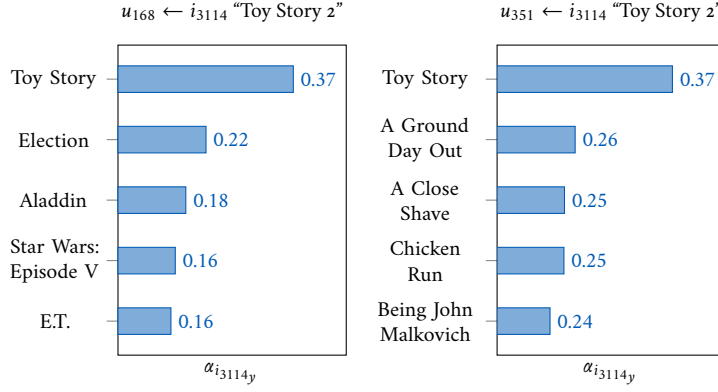


Figure A.4. Examples of interpretable recommendations of ILF-Rec. The lists correspond to the top five influencing movies to the recommendation of i_{3114} (Toy Story 2) to the users u_{168} and u_{351} .

embeddings k and the coefficients λ_r and λ_o on the LastFM and AS&O datasets (we observed similar trends for the learning rate and on the ML-1M dataset). The results indicate that ILF-Rec consistently achieves similar accuracy performance regardless of the specific hyper-parameter values, demonstrating a high degree of robustness and reduced sensitivity to hyper-parameter choices especially for k and λ_r . Concerning λ_o , setting $\lambda_o = 1e^{-6}$ consistently emerges as the best choice.

*To answer **RQ3**, ILF-Rec can achieve competitive predictive performance without requiring a precise tuning of the hyper-parameters.*

A.5.4 Interpretability Study (RQ4)

In Section A.3.4, we demonstrated that the architecture of ILF-Rec inherently enables interpretable recommendations. In particular, its scoring function expresses the predicted score of item i for a user u as a weighted sum over the items previously interacted with by u (see Eq. A.8). This process explains why item i was recommended by highlighting the past interactions that strongly influenced the model's decision. Here, each weight α_{i_y} quantifies the influence of a specific past interaction on the final recommendation, making it possible to discern which interactions drive ILF-Rec's decisions.

To illustrate this interpretability in practice, we conducted a case study on the ML-1M dataset, which includes movie metadata. We focused on two users, u_{168} and u_{351} , for whom ILF-Rec recommended item i_{3114} "Toy Story 2". Figure A.4 shows the top five historical items that contributed most positively to the recommendation score of i_{3114} for both users. In both cases, the top contributing item was i_1 , i.e., "Toy Story", despite the fact that ILF-Rec operates purely on collaborative signals without accessing item content. This outcome suggests that the model implicitly captures semantic relationships between items based solely on user interaction patterns.

When we extend the analysis to the top five contributing items, we notice that all

of them share at least one genre with “Toy Story 2”, further confirming that ILF-Rec captures implicit correlations among items and translates these into interpretable recommendations. Moreover, examining the most frequent contributors to the recommendation of i_{3114} , we consistently found, across all users, that i_1 “Toy Story” and i_{2355} “A Bug’s Life”—both Disney-Pixar productions with overlapping genres—rank among the top influences. By linking recommendations to specific, human-understandable signals—even without explicit content features—ILF-Rec offers a level of interpretability that can significantly enhance user trust.

In summary, our findings addressing RQ4 underscore ILF-Rec’s ability to generate semantically coherent and explainable recommendations, demonstrating its potential to produce outputs that are both rational and interpretable.

A.6 Conclusion and Future Work

In this work, we introduced ILF-Rec, a novel recommender system that combines the strengths of latent factor and linear models to deliver both accurate and interpretable recommendations. Extensive experiments on three real-world datasets demonstrated that ILF-Rec consistently outperforms several state-of-the-art collaborative filtering methods in terms of accuracy and achieves comparable performance with LightGCN and EASE^R. By combining the expressive power of latent factor models with the transparency of linear models, ILF-Rec effectively balances accuracy, scalability, and interpretability, offering a compelling solution for practical deployment. Future work will explore improving model efficiency, enhancing cross-domain generalization, and incorporating richer user-item interactions to further boost recommendation quality.

Bibliography

- [1] Himan Abdollahpouri and Robin Burke. “Multistakeholder Recommender Systems”. In: *Recommender Systems Handbook*. Ed. by Francesco Ricci, Lior Rokach, and Bracha Shapira. Springer US, 2022, pp. 647–677. DOI: 10.1007/978-1-0716-2197-4_17.
- [2] Himan Abdollahpouri, Robin Burke, and Bamshad Mobasher. “Controlling Popularity Bias in Learning-to-Rank Recommendation”. In: *Proceedings of the Eleventh ACM Conference on Recommender Systems, RecSys 2017, Como, Italy, August 27-31, 2017*. Ed. by Paolo Cremonesi, Francesco Ricci, Shlomo Berkovsky, and Alexander Tuzhilin. ACM, 2017, pp. 42–46. DOI: 10.1145/3109859.3109912.
- [3] Gediminas Adomavicius and Alexander Tuzhilin. “Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions”. In: *IEEE Trans. Knowl. Data Eng.* 17.6 (2005), pp. 734–749.
- [4] Gediminas Adomavicius and Jingjing Zhang. “Impact of data characteristics on recommender systems performance”. In: *ACM Trans. Manag. Inf. Syst.* 3.1 (2012), 3:1–3:17.
- [5] Mohammad Mehdi Afsar, Trafford Crump, and Behrouz H. Far. “Reinforcement Learning based Recommender Systems: A Survey”. In: *ACM Comput. Surv.* 55.7 (2023), 145:1–145:38.
- [6] Mohammed Fadhel Aljunid, Manjaiah Doddaghatta Huchaiiah, Mohammad Kazim Hooshmand, Wasim A. Ali, Amrithkala M. Shetty, and Sadiq Qaid Alzoubah. “A collaborative filtering recommender systems: Survey”. In: *Neurocomputing* 617 (2025), p. 128718. DOI: 10.1016/J.NEUCOM.2024.128718.
- [7] Vito Walter Anelli, Pierpaolo Basile, Gerard de Melo, Francesco Maria Donini, Antonio Ferrara, Cataldo Musto, Fedelucio Narducci, Azzurra Ragone, and Markus Zanker. “Fourth Knowledge-aware and Conversational Recommender Systems Workshop (KaRS)”. In: *RecSys*. ACM, 2022, pp. 663–666.

- [8] Vito Walter Anelli, Luca Belli, Yashar Deldjoo, Tommaso Di Noia, Antonio Ferrara, Fedelucio Narducci, and Claudio Pomo. “Pursuing Privacy in Recommender Systems: the View of Users and Researchers from Regulations to Applications”. In: *RecSys*. ACM, 2021, pp. 838–841.
- [9] Vito Walter Anelli, Alejandro Bellogín, Antonio Ferrara, Daniele Malitesta, Felice Antonio Merra, Claudio Pomo, Francesco Maria Donini, and Tommaso Di Noia. “Elliot: A Comprehensive and Rigorous Framework for Reproducible Recommender Systems Evaluation”. In: *SIGIR*. ACM, 2021, pp. 2405–2414.
- [10] Vito Walter Anelli, Alejandro Bellogín, Tommaso Di Noia, and Claudio Pomo. “Reenvisioning the comparison between Neural Collaborative Filtering and Matrix Factorization”. In: *RecSys '21: Fifteenth ACM Conference on Recommender Systems, Amsterdam, The Netherlands, 27 September 2021 - 1 October 2021*. Ed. by Humberto Jesús Corona Pampín, Martha A. Larson, Martijn C. Willemsen, Joseph A. Konstan, Julian J. McAuley, Jean Garcia-Gathright, Bouke Huurnink, and Even Oldridge. ACM, 2021, pp. 521–529. doi: 10.1145/3460231.3475944.
- [11] Vito Walter Anelli, Yashar Deldjoo, Tommaso Di Noia, Daniele Malitesta, and Felice Antonio Merra. “A Study of Defensive Methods to Protect Visual Recommendation Against Adversarial Manipulation of Images”. In: *SIGIR*. ACM, 2021, pp. 1094–1103.
- [12] Vito Walter Anelli, Yashar Deldjoo, Tommaso Di Noia, Daniele Malitesta, Vincenzo Paparella, and Claudio Pomo. “Auditing Consumer- and Producer-Fairness in Graph Collaborative Filtering”. In: *Advances in Information Retrieval - 45th European Conference on Information Retrieval, ECIR 2023, Dublin, Ireland, April 2-6, 2023, Proceedings, Part I*. Ed. by Jaap Kamps, Lorraine Goeuriot, Fabio Crestani, Maria Maistro, Hideo Joho, Brian Davis, Cathal Gurrin, Udo Kruschwitz, and Annalina Caputo. Vol. 13980. Lecture Notes in Computer Science. Springer, 2023, pp. 33–48. doi: 10.1007/978-3-031-28244-7_3.
- [13] Vito Walter Anelli, Yashar Deldjoo, Tommaso Di Noia, Eugenio Di Sciascio, Antonio Ferrara, Daniele Malitesta, and Claudio Pomo. “Reshaping Graph Recommendation with Edge Graph Collaborative Filtering and Customer Reviews”. In: *DL4SR@CIKM*. Vol. 3317. CEUR Workshop Proceedings. CEUR-WS.org, 2022.
- [14] Vito Walter Anelli, Antonio Ferrara, Cataldo Musto, Fedelucio Narducci, Azzurra Ragone, and Markus Zanker. “Sixth knowledge-aware and conversational recommender systems workshop (kars)”. In: *Proceedings of the 18th ACM Conference on Recommender Systems*. 2024, pp. 1245–1249.
- [15] Vito Walter Anelli, Daniele Malitesta, Claudio Pomo, Alejandro Bellogin, Eugenio Di Sciascio, and Tommaso Di Noia. “Challenging the Myth of Graph Collaborative Filtering: a Reasoned and Reproducibility-driven Analysis”. In:

- Proceedings of the 17th ACM Conference on Recommender Systems*. RecSys '23. Singapore, Singapore: Association for Computing Machinery, 2023, pp. 350–361. ISBN: 9798400702419. DOI: 10.1145/3604915.3609489.
- [16] Vito Walter Anelli, Daniele Malitesta, Claudio Pomo, Alejandro Bellogín, Eugenio Di Sciascio, and Tommaso Di Noia. “Challenging the Myth of Graph Collaborative Filtering: a Reasoned and Reproducibility-driven Analysis”. In: *RecSys*. ACM, 2023, pp. 350–361.
 - [17] Vito Walter Anelli, Tommaso Di Noia, Eugenio Di Sciascio, Antonio Ferrara, and Alberto Carlo Maria Mancino. “Sparse Feature Factorization for Recommender Systems with Knowledge Graphs”. In: *RecSys*. ACM, 2021, pp. 154–165.
 - [18] Vito Walter Anelli, Tommaso Di Noia, Eugenio Di Sciascio, Azzurra Ragone, and Joseph Trotta. “How to Make Latent Factors Interpretable by Feeding Factorization Machines with Knowledge Graphs”. In: *ISWC (1)*. Vol. 11778. Lecture Notes in Computer Science. Springer, 2019, pp. 38–56.
 - [19] Vito Walter Anelli, Tommaso Di Noia, Eugenio Di Sciascio, Azzurra Ragone, and Joseph Trotta. “Semantic Interpretation of Top-N Recommendations”. In: *IEEE Trans. Knowl. Data Eng.* 34.5 (2022), pp. 2416–2428.
 - [20] Vito Walter Anelli et al. “Report on the 3rd workshop of knowledge-aware and conversational recommender systems (KARS/ComplexRec) at RecSys 2021”. In: *SIGIR Forum* 55.2 (Mar. 2022). ISSN: 0163-5840. DOI: 10.1145/3527546.3527566.
 - [21] Lei Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. “Layer Normalization”. In: *CoRR* abs/1607.06450 (2016).
 - [22] Lars Backstrom and Jure Leskovec. “Supervised random walks: predicting and recommending links in social networks”. In: *WSDM*. ACM, 2011, pp. 635–644.
 - [23] Ting Bai, Youjie Zhang, Bin Wu, and Jian-Yun Nie. “Temporal Graph Neural Networks for Social Recommendation”. In: *IEEE BigData*. IEEE, 2020, pp. 898–903.
 - [24] Robert M. Bell and Yehuda Koren. “Lessons from the Netflix prize challenge”. In: *SIGKDD Explor.* 9.2 (2007), pp. 75–79. DOI: 10.1145/1345448.1345465.
 - [25] Alessandro De Bellis, **Salvatore Bufi**, Giovanni Servedio, Vito Walter Anelli, Tommaso Di Noia, and Eugenio Di Sciascio. “Type-Less yet Type-Aware Inductive Link Prediction with Pretrained Language Models”. In: *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, EMNLP 2025*. Ed. by Christodoulopoulos Christos, Chakraborty Tanmoy, Rose Carolyn, and Peng Violet. Suzhou, China: Association for Computational Linguistics, Nov. 2025, pp. 27181–27197. ISBN: 979-8-89176-332-6.

- [26] Alejandro Bellogín, Iván Cantador, and Pablo Castells. “A study of heterogeneity in recommendations for a social music service”. In: *HetRec@RecSys*. ACM, 2010, pp. 1–8.
- [27] Alejandro Bellogín, Pablo Castells, and Iván Cantador. “Statistical biases in Information Retrieval metrics for recommender systems”. In: *Inf. Retr. J.* 20.6 (2017), pp. 606–634. doi: 10.1007/S10791-017-9312-Z.
- [28] Alejandro Bellogín and Alan Said. “Improving accountability in recommender systems research through reproducibility”. In: *User Model. User Adapt. Interact.* 31.5 (2021), pp. 941–977.
- [29] James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. “Algorithms for Hyper-Parameter Optimization”. In: *Advances in Neural Information Processing Systems 24: 25th Annual Conference on Neural Information Processing Systems 2011. Proceedings of a meeting held 12-14 December 2011, Granada, Spain*. Ed. by John Shawe-Taylor, Richard S. Zemel, Peter L. Bartlett, Fernando C. N. Pereira, and Kilian Q. Weinberger. 2011, pp. 2546–2554.
- [30] Asia J. Biega, Krishna P. Gummadi, and Gerhard Weikum. “Equity of Attention: Amortizing Individual Fairness in Rankings”. In: *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, SIGIR 2018, Ann Arbor, MI, USA, July 08-12, 2018*. Ed. by Kevyn Collins-Thompson, Qiaozhu Mei, Brian D. Davison, Yiqun Liu, and Emine Yilmaz. ACM, 2018, pp. 405–414. doi: 10.1145/3209978.3210063.
- [31] Asia J. Biega, Peter Potash, Hal Daumé III, Fernando Diaz, and Michèle Finck. “Operationalizing the Legal Principle of Data Minimization for Personalization”. In: *SIGIR*. ACM, 2020, pp. 399–408.
- [32] Jesús Bobadilla, Fernando Ortega, Antonio Hernando, and Jesus Bernal. “A collaborative filtering approach to mitigate the new user cold start problem”. In: *Knowl. Based Syst.* 26 (2012), pp. 225–238.
- [33] Veronika Bogina, Tsvi Kuflik, Dietmar Jannach, Mária Bielíková, Michal Kompan, and Christoph Trattner. “Considering temporal aspects in recommender systems: a survey”. In: *User Model. User Adapt. Interact.* 33.1 (2023), pp. 81–119.
- [34] Veronika Bogina, Yuri Variat, Tsvi Kuflik, and Eyal Dim. “Incorporating time-interval sequences in linear TV for next-item prediction”. In: *Expert Syst. Appl.* 192 (2022), p. 116284.
- [35] Antoine Bordes, Nicolas Usunier, Alberto García-Durán, Jason Weston, and Oksana Yakhnenko. “Translating Embeddings for Modeling Multi-relational Data”. In: *NIPS*. 2013, pp. 2787–2795.
- [36] Tom B. Brown et al. “Language Models are Few-Shot Learners”. In: *NeurIPS*. 2020.

- [37] Sebastian Bruch, Xuanhui Wang, Michael Bendersky, and Marc Najork. “An Analysis of the Softmax Cross Entropy Loss for Learning-to-Rank with Binary Relevance”. In: *ICTIR*. ACM, 2019, pp. 75–78.
- [38] Peter Brusilovsky, Alfred Kobsa, and Wolfgang Nejdl, eds. *The Adaptive Web, Methods and Strategies of Web Personalization*. Vol. 4321. Lecture Notes in Computer Science. Springer, 2007.
- [39] Xuheng Cai, Chao Huang, Lianghao Xia, and Xubin Ren. “LightGCL: Simple Yet Effective Graph Contrastive Learning for Recommendation”. In: *ICLR*. OpenReview.net, 2023.
- [40] *California Consumer Privacy Act (CCPA)*. <https://cloud.google.com/security/compliance/ccpa>. accessed 2024-02-13.
- [41] Iván Cantador, Peter Brusilovsky, and Tsvi Kuflik, eds. *Proceedings of the 2nd International Workshop on Information Heterogeneity and Fusion in Recommender Systems, HetRec ’11, Chicago, Illinois, USA, October 27, 2011*. ACM, 2011.
- [42] Yixin Cao, Xiang Wang, Xiangnan He, Zikun Hu, and Tat-Seng Chua. “Unifying Knowledge Graph Learning and Recommendation: Towards a Better Understanding of User Preferences”. In: *The World Wide Web Conference, WWW 2019, San Francisco, CA, USA, May 13–17, 2019*. Ed. by Ling Liu, Ryen W. White, Amin Mantrach, Fabrizio Silvestri, Julian J. McAuley, Ricardo Baeza-Yates, and Leila Zia. ACM, 2019, pp. 151–161. DOI: 10.1145/3308558.3313705.
- [43] Tommaso Carraro, Mirko Polato, and Fabio Aioli. “A Look Inside the Black-Box: Towards the Interpretability of Conditioned Variational Autoencoder for Collaborative Filtering”. In: *Adjunct Publication of the 28th ACM Conference on User Modeling, Adaptation and Personalization*. UMAP ’20 Adjunct. Genoa, Italy: Association for Computing Machinery, 2020, pp. 233–236. ISBN: 9781450379502. DOI: 10.1145/3386392.3399305.
- [44] Pablo Castells, Neil Hurley, and Saúl Vargas. “Novelty and Diversity in Recommender Systems”. In: *Recommender Systems Handbook*. Ed. by Francesco Ricci, Lior Rokach, and Bracha Shapira. Springer US, 2022, pp. 603–646. DOI: 10.1007/978-1-0716-2197-4_16.
- [45] Jianxin Chang, Chen Gao, Yu Zheng, Yiqun Hui, Yanan Niu, Yang Song, Depeng Jin, and Yong Li. “Sequential Recommendation with Graph Neural Networks”. In: *SIGIR*. ACM, 2021, pp. 378–387.
- [46] Chong Chen, Min Zhang, Yongfeng Zhang, Yiqun Liu, and Shaoping Ma. “Efficient Neural Matrix Factorization without Sampling for Recommendation”. In: *ACM Trans. Inf. Syst.* 38.2 (Jan. 2020). ISSN: 1046-8188. DOI: 10.1145/3373807.
- [47] Jiajun Chen, Huarui He, Feng Wu, and Jie Wang. “Topology-Aware Correlations Between Relations for Inductive Link Prediction in Knowledge Graphs”. In: *AAAI*. AAAI Press, 2021, pp. 6271–6278.

- [48] Lei Chen, Le Wu, Richang Hong, Kun Zhang, and Meng Wang. “Revisiting Graph Based Collaborative Filtering: A Linear Residual Graph Convolutional Network Approach”. In: *AAAI*. AAAI Press, 2020, pp. 27–34.
- [49] Mengru Chen, Chao Huang, Lianghao Xia, Wei Wei, Yong Xu, and Ronghua Luo. “Heterogeneous Graph Contrastive Learning for Recommendation”. In: *WSDM*. ACM, 2023, pp. 544–552.
- [50] Wen Chen, Pipei Huang, Jiaming Xu, Xin Guo, Cheng Guo, Fei Sun, Chao Li, Andreas Pfadler, Huan Zhao, and Binqiang Zhao. “POG: Personalized Outfit Generation for Fashion Recommendation at Alibaba iFashion”. In: *KDD*. ACM, 2019, pp. 2662–2670.
- [51] Yankai Chen, Yaming Yang, Yujing Wang, Jing Bai, Xiangchen Song, and Irwin King. “Attentive Knowledge-aware Graph Convolutional Networks with Collaborative Guidance for Personalized Recommendation”. In: *ICDE*. IEEE, 2022, pp. 299–311.
- [52] Weiyu Cheng, Yanyan Shen, Linpeng Huang, and Yanmin Zhu. “Incorporating Interpretability into Latent Factor Models via Fast Influence Analysis”. In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. KDD ’19. Anchorage, AK, USA: Association for Computing Machinery, 2019, pp. 885–893. ISBN: 9781450362016. DOI: 10.1145/3292500.3330857.
- [53] Patrick John Chia, Jacopo Tagliabue, Federico Bianchi, Chloe He, and Brian Ko. “Beyond NDCG: Behavioral Testing of Recommender Systems with RecList”. In: *WWW (Companion Volume)*. ACM, 2022, pp. 99–104.
- [54] Jin Yao Chin, Yile Chen, and Gao Cong. “The Datasets Dilemma: How Much Do We Really Know About Recommendation Datasets?” In: *WSDM*. ACM, 2022, pp. 141–149.
- [55] Eunjoon Cho, Seth A. Myers, and Jure Leskovec. “Friendship and mobility: user movement in location-based social networks”. In: *KDD*. ACM, 2011, pp. 1082–1090.
- [56] Richard Chow, Hongxia Jin, Bart P. Knijnenburg, and Gökalp Saldamli. “Differential data analysis for recommender systems”. In: *Seventh ACM Conference on Recommender Systems, RecSys ’13, Hong Kong, China, October 12–16, 2013*. Ed. by Qiang Yang, Irwin King, Qing Li, Pearl Pu, and George Karypis. ACM, 2013, pp. 323–326. DOI: 10.1145/2507157.2507190.
- [57] Fabian Christoffel, Bibek Paudel, Chris Newell, and Abraham Bernstein. “Blockbusters and Wallflowers: Accurate, Diverse, and Scalable Recommendations with Random Walks”. In: *RecSys*. ACM, 2015, pp. 163–170.
- [58] Gemma Galdon Clavell, Mariano Martín Zamorano, Carlos Castillo, Oliver Smith, and Aleksandar Matic. “Auditing Algorithms: On Lessons Learned and the Risks of Data Minimization”. In: *AIES*. ACM, 2020, pp. 265–271.

- [59] Ludovik Coba, Roberto Confalonieri, and Markus Zanker. “RecoXplainer: A Library for Development and Offline Evaluation of Explainable Recommender Systems”. In: *IEEE Comput. Intell. Mag.* 17.1 (2022), pp. 46–58.
- [60] Paul Covington, Jay Adams, and Emre Sargin. “Deep Neural Networks for YouTube Recommendations”. In: *RecSys*. ACM, 2016, pp. 191–198.
- [61] Paolo Cremonesi and Dietmar Jannach. “Progress in Recommender Systems Research: Crisis? What Crisis?”. In: *AI Mag.* 42.3 (2021), pp. 43–54. DOI: 10.1609/AIMAG.V42I3.18145.
- [62] Paolo Cremonesi, Yehuda Koren, and Roberto Turrin. “Performance of recommender algorithms on top-n recommendation tasks”. In: *Proceedings of the 2010 ACM Conference on Recommender Systems, RecSys 2010, Barcelona, Spain, September 26-30, 2010*. Ed. by Xavier Amatriain, Marc Torrens, Paul Resnick, and Markus Zanker. ACM, 2010, pp. 39–46. DOI: 10.1145/1864708.1864721.
- [63] Maurizio Ferrari Dacrema, Simone Boglio, Paolo Cremonesi, and Dietmar Jannach. “A Troubling Analysis of Reproducibility and Progress in Recommender Systems Research”. In: *ACM Trans. Inf. Syst.* 39.2 (2021), 20:1–20:49.
- [64] Maurizio Ferrari Dacrema, Paolo Cremonesi, and Dietmar Jannach. “Are we really making much progress? A worrying analysis of recent neural recommendation approaches”. In: *RecSys*. ACM, 2019, pp. 101–109.
- [65] Enyan Dai, Tianxiang Zhao, Huaisheng Zhu, Junjie Xu, Zhimeng Guo, Hui Liu, Jiliang Tang, and Suhang Wang. “A Comprehensive Survey on Trustworthy Graph Neural Networks: Privacy, Robustness, Fairness, and Explainability”. In: *Mach. Intell. Res.* 21.6 (2024), pp. 1011–1061.
- [66] Daniel Daza, Michael Cochez, and Paul Groth. “Inductive Entity Representations from Text via Link Prediction”. In: *WWW. ACM / IW3C2*, 2021, pp. 798–808.
- [67] Yashar Deldjoo, Vito Walter Anelli, Hamed Zamani, Alejandro Bellogín, and Tommaso Di Noia. “A flexible framework for evaluating user and item fairness in recommender systems”. In: *User Model. User Adapt. Interact.* 31.3 (2021), pp. 457–511. DOI: 10.1007/S11257-020-09285-1.
- [68] Yashar Deldjoo, Alejandro Bellogín, and Tommaso Di Noia. “Explaining recommender systems fairness and accuracy through the lens of data characteristics”. In: *Inf. Process. Manag.* 58.5 (2021), p. 102662. DOI: 10.1016/J.IPM.2021.102662.
- [69] Yashar Deldjoo, Zhankui He, Julian J. McAuley, Anton Korikov, Scott Sanner, Arnau Ramisa, René Vidal, Maheswaran Sathiamoorthy, Atoosa Kasirzadeh, and Silvia Milano. “A Review of Modern Recommender Systems Using Generative Models (Gen-RecSys)”. In: *CoRR abs/2404.00579* (2024).

- [70] Yashar Deldjoo, Dietmar Jannach, Alejandro Bellogín, Alessandro Difonzo, and Dario Zanzonelli. “Fairness in recommender systems: research landscape and future directions”. In: *User Model. User Adapt. Interact.* 34.1 (2024), pp. 59–108.
- [71] Yashar Deldjoo, Tommaso Di Noia, and Felice Antonio Merra. “A Survey on Adversarial Recommender Systems: From Attack/Defense Strategies to Generative Adversarial Networks”. In: *ACM Comput. Surv.* 54.2 (2022), 35:1–35:38.
- [72] Yashar Deldjoo, Tommaso Di Noia, Eugenio Di Sciascio, and Felice Antonio Merra. “How Dataset Characteristics Affect the Robustness of Collaborative Recommendation Models”. In: *SIGIR*. ACM, 2020, pp. 951–960.
- [73] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”. In: *NAACL-HLT (1)*. Association for Computational Linguistics, 2019, pp. 4171–4186.
- [74] Jingtao Ding, Yuhan Quan, Xiangnan He, Yong Li, and Depeng Jin. “Reinforced Negative Sampling for Recommendation with Exposure Data”. In: *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10–16, 2019*. Ed. by Sarit Kraus. ijcai.org, 2019, pp. 2230–2236. DOI: 10.24963/IJCAI.2019/309.
- [75] Hanwen Du, Huanhuan Yuan, Pengpeng Zhao, Fuzhen Zhuang, Guanfeng Liu, Lei Zhao, Yanchi Liu, and Victor S. Sheng. “Ensemble Modeling with Contrastive Knowledge Distillation for Sequential Recommendation”. In: *SIGIR*. ACM, 2023, pp. 58–67.
- [76] Yuntao Du, Xinjun Zhu, Lu Chen, Baihua Zheng, and Yunjun Gao. “HAKG: Hierarchy-Aware Knowledge Gated Network for Recommendation”. In: *SIGIR*. ACM, 2022, pp. 1390–1400.
- [77] Abhimanyu Dubey et al. “The Llama 3 Herd of Models”. In: *CoRR* abs/2407.21783 (2024).
- [78] Travis Ebesu, Bin Shen, and Yi Fang. “Collaborative Memory Network for Recommendation Systems”. In: *SIGIR*. ACM, 2018, pp. 515–524.
- [79] Tobias Eichinger and Axel Küpper. “Distributed Data Minimization for Decentralized Collaborative Filtering Systems”. In: *ICDCN*. ACM, 2023, pp. 140–149.
- [80] Tobias Eichinger and Axel Küpper. “On data minimization and anonymity in pervasive mobile-to-mobile recommender systems”. In: *Pervasive Mob. Comput.* 103 (2024), p. 101951.
- [81] Michael D. Ekstrand. “LensKit for Python: Next-Generation Software for Recommender Systems Experiments”. In: *CIKM*. ACM, 2020, pp. 2999–3006.

- [82] Michael D. Ekstrand, Anubrata Das, Robin Burke, and Fernando Diaz. “Fairness in Recommender Systems”. In: *Recommender Systems Handbook*. Ed. by Francesco Ricci, Lior Rokach, and Bracha Shapira. Springer US, 2022, pp. 679–707. DOI: 10.1007/978-1-0716-2197-4_18.
- [83] Michael D. Ekstrand, John Riedl, and Joseph A. Konstan. “Collaborative Filtering Recommender Systems”. In: *Found. Trends Hum. Comput. Interact.* 4.2 (2011), pp. 175–243. DOI: 10.1561/11000000009.
- [84] Kawin Ethayarajh. “How Contextual are Contextualized Word Representations? Comparing the Geometry of BERT, ELMo, and GPT-2 Embeddings”. In: *EMNLP/IJCNLP (1)*. Association for Computational Linguistics, 2019, pp. 55–65.
- [85] Leigh Eytan, Veronika Bogina, Irad Ben-Gal, and Noam Koenigstein. “KPAR: Knowledge-aware Path-based Attentive Recommender with Interpretability”. In: *ACM Trans. Recomm. Syst.* 3.3 (Mar. 2025). DOI: 10.1145/3673243.
- [86] Antonio Ferrara, Vito Walter Anelli, Alberto Carlo Maria Mancino, Tommaso Di Noia, and Eugenio Di Sciascio. “KGFlex: Efficient Recommendation with Sparse Feature Factorization and Knowledge Graphs”. In: *ACM Trans. Recomm. Syst.* (Apr. 2023). Just Accepted. DOI: 10.1145/3588901.
- [87] Maurizio Ferrari Dacrema, Paolo Cremonesi, and Dietmar Jannach. “Are we really making much progress? A worrying analysis of recent neural recommendation approaches”. In: *Proceedings of the 13th ACM Conference on Recommender Systems*. RecSys ’19. Copenhagen, Denmark: Association for Computing Machinery, 2019, pp. 101–109. ISBN: 9781450362436. DOI: 10.1145/3298689.3347058.
- [88] Matthias Feurer and Frank Hutter. “Hyperparameter Optimization”. In: *Automated Machine Learning - Methods, Systems, Challenges*. Ed. by Frank Hutter, Lars Kotthoff, and Joaquin Vanschoren. The Springer Series on Challenges in Machine Learning. Springer, 2019, pp. 3–33. DOI: 10.1007/978-3-030-05318-5_1.
- [89] Michèle Finck and Asia Biega. “Reviving Purpose Limitation and Data Minimisation in Personalisation, Profiling and Decision-Making Systems”. In: *CoRR abs/2101.06203* (2021).
- [90] M. Fleischer. “The Measure of Pareto Optima”. In: *EMO*. Vol. 2632. Lecture Notes in Computer Science. Springer, 2003, pp. 519–533.
- [91] Mingsheng Fu, Anubha Agrawal, Athirai A. Irissappane, Jie Zhang, Liwei Huang, and Hong Qu. “Deep Reinforcement Learning Framework for Category-Based Item Recommendation”. In: *IEEE Trans. Cybern.* 52.11 (2022), pp. 12028–12041.
- [92] Prakhar Ganesh, Cuong Tran, Reza Shokri, and Ferdinando Fioretto. “The Data Minimization Principle in Machine Learning”. In: *CoRR abs/2405.19471* (2024).

- [93] Bin Gao, Tie-Yan Liu, Wei Wei, Taifeng Wang, and Hang Li. “Semi-supervised ranking on very large graphs with rich metadata”. In: *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Diego, CA, USA, August 21-24, 2011*. Ed. by Chid Apté, Joydeep Ghosh, and Padhraic Smyth. ACM, 2011, pp. 96–104. DOI: 10.1145/2020408.2020430.
- [94] Yingqiang Ge et al. “Towards Long-term Fairness in Recommendation”. In: *WSDM*. ACM, 2021, pp. 445–453.
- [95] Aryo Pradipta Gema, Dominik Grabarczyk, Wolf De Wulf, Piyush Borole, Javier Antonio Alfaro, Pasquale Minervini, Antonio Vergari, and Ajitha Rajan. “Knowledge Graph Embeddings in the Biomedical Domain: Are They Useful? A Look at Link Prediction, Rule Learning, and Downstream Polypharmacy Tasks”. In: *CoRR* abs/2305.19979 (2023).
- [96] Marco de Gemmis, Pasquale Lops, Cataldo Musto, Fedelucio Narducci, and Giovanni Semeraro. “Semantics-Aware Content-Based Recommender Systems”. In: *Recommender Systems Handbook*. Springer, 2015, pp. 119–159.
- [97] Xavier Glorot and Yoshua Bengio. “Understanding the difficulty of training deep feedforward neural networks”. In: *AISTATS*. Vol. 9. JMLR Proceedings. JMLR.org, 2010, pp. 249–256.
- [98] Nadav Golbandi, Yehuda Koren, and Ronny Lempel. “Adaptive bootstrapping of recommender systems using decision trees”. In: *WSDM*. ACM, 2011, pp. 595–604.
- [99] Elizabeth Gómez, David Contreras, Ludovico Boratto, and Maria Salamó. “AMBAR: A dataset for Assessing Multiple Beyond-Accuracy Recommenders”. In: *RecSys*. ACM, 2024, pp. 137–147.
- [100] Scott Graham, Jun-Ki Min, and Tao Wu. “Microsoft recommenders: tools to accelerate developing recommender systems”. In: *RecSys*. ACM, 2019, pp. 542–543.
- [101] Shuyun Gu, Xiao Wang, Chuan Shi, and Ding Xiao. “Self-supervised Graph Neural Networks for Multi-behavior Recommendation”. In: *IJCAI*. ijcai.org, 2022, pp. 2052–2058.
- [102] Asela Gunawardana and Guy Shani. “Evaluating Recommender Systems”. In: *Recommender Systems Handbook*. Springer, 2015, pp. 265–308.
- [103] Guibing Guo, Jie Zhang, Daniel Thalmann, and Neil Yorke-Smith. “ETAF: An extended trust antecedents framework for trust prediction”. In: *2014 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining, ASONAM 2014, Beijing, China, August 17-20, 2014*. Ed. by Xindong Wu, Martin Ester, and Guandong Xu. IEEE Computer Society, 2014, pp. 540–547. DOI: 10.1109/ASONAM.2014.6921639.

- [104] Wei Guo, Rong Su, Renhao Tan, Huifeng Guo, Yingxue Zhang, Zhirong Liu, Ruiming Tang, and Xiuqiang He. “Dual Graph enhanced Embedding Neural Network for CTR Prediction”. In: *KDD*. ACM, 2021, pp. 496–504.
- [105] Deepesh V. Hada, Vijaikumar M, and Shirish K. Shevade. “ReXPlug: Explainable Recommendation using Plug-and-Play Language Model”. In: *SIGIR*. ACM, 2021, pp. 81–91.
- [106] William L. Hamilton. *Graph Representation Learning*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2020.
- [107] William L. Hamilton, Zhitao Ying, and Jure Leskovec. “Inductive Representation Learning on Large Graphs”. In: *NIPS*. 2017, pp. 1024–1034.
- [108] Casper Hansen, Christian Hansen, Lucas Maystre, Rishabh Mehrotra, Brian Brost, Federico Tomasi, and Mounia Lalmas. “Contextual and Sequential User Embeddings for Large-Scale Music Recommendation”. In: *RecSys*. ACM, 2020, pp. 53–62.
- [109] Christian Hansen, Rishabh Mehrotra, Casper Hansen, Brian Brost, Lucas Maystre, and Mounia Lalmas. “Shifting Consumption towards Diverse Content on Music Streaming Platforms”. In: *WSDM '21, The Fourteenth ACM International Conference on Web Search and Data Mining, Virtual Event, Israel, March 8-12, 2021*. Ed. by Liane Lewin-Eytan, David Carmel, Elad Yom-Tov, Eugene Agichtein, and Evgeniy Gabrilovich. ACM, 2021, pp. 238–246. DOI: 10.1145/3437963.3441775.
- [110] Bowen Hao, Hongzhi Yin, Jing Zhang, Cuiping Li, and Hong Chen. “A Multi-strategy-based Pre-training Method for Cold-start Recommendation”. In: *ACM Trans. Inf. Syst.* 41.2 (2023), 31:1–31:24. DOI: 10.1145/3544107.
- [111] Shibo Hao, Bowen Tan, Kaiwen Tang, Bin Ni, Xiyan Shao, Hengzhe Zhang, Eric P. Xing, and Zhiting Hu. “BertNet: Harvesting Knowledge Graphs with Arbitrary Relations from Pretrained Language Models”. In: *ACL (Findings)*. Association for Computational Linguistics, 2023, pp. 5000–5015.
- [112] Yongjing Hao, Tingting Zhang, Pengpeng Zhao, Yanchi Liu, Victor S. Sheng, Jiajie Xu, Guanfeng Liu, and Xiaofang Zhou. “Feature-Level Deeper Self-Attention Network With Contrastive Learning for Sequential Recommendation”. In: *IEEE Trans. Knowl. Data Eng.* 35.10 (2023), pp. 10112–10124.
- [113] F. Maxwell Harper and Joseph A. Konstan. “The MovieLens Datasets: History and Context”. In: *ACM Trans. Interact. Intell. Syst.* 5.4 (2016), 19:1–19:19.
- [114] Ruining He and Julian J. McAuley. “Ups and Downs: Modeling the Visual Evolution of Fashion Trends with One-Class Collaborative Filtering”. In: *WWW*. ACM, 2016, pp. 507–517.

- [115] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yong-Dong 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, SIGIR 2020, Virtual Event, China, July 25-30, 2020*. Ed. by Jimmy X. Huang, Yi Chang, Xueqi Cheng, Jaap Kamps, Vanessa Murdock, Ji-Rong Wen, and Yiqun Liu. ACM, 2020, pp. 639–648. DOI: 10.1145/3397271.3401063.
- [116] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. “Neural Collaborative Filtering”. In: *WWW*. ACM, 2017, pp. 173–182.
- [117] 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. WWW ’17*. Perth, Australia: International World Wide Web Conferences Steering Committee, 2017, pp. 173–182. ISBN: 9781450349130. DOI: 10.1145/3038912.3052569.
- [118] Stan Hennekes and Flavius Frasincar. “Weighted Neural Collaborative Filtering: Deep Implicit Recommendation with Weighted Positive and Negative Feedback”. In: *Proceedings of the 38th ACM/SIGAPP Symposium on Applied Computing. SAC ’23*. Tallinn, Estonia: Association for Computing Machinery, 2023, pp. 1799–1808. ISBN: 9781450395175. DOI: 10.1145/3555776.3577619.
- [119] Jonathan L. Herlocker, Joseph A. Konstan, Loren G. Terveen, and John Riedl. “Evaluating collaborative filtering recommender systems”. In: *ACM Trans. Inf. Syst.* 22.1 (2004), pp. 5–53. DOI: 10.1145/963770.963772.
- [120] Yassine Himeur, Shahab Saquib Sohail, Faycal Bensaali, Abbes Amira, and Mamoun Alazab. “Latest trends of security and privacy in recommender systems: A comprehensive review and future perspectives”. In: *Comput. Secur.* 118 (2022), p. 102746.
- [121] Aidan Hogan et al. *Knowledge Graphs*. Synthesis Lectures on Data, Semantics, and Knowledge. Morgan & Claypool Publishers, 2021.
- [122] Aidan Hogan et al. “Knowledge Graphs”. In: *ACM Comput. Surv.* 54.4 (2022), 71:1–71:37. DOI: 10.1145/3447772.
- [123] Yupeng Hou, Jiacheng Li, Zhankui He, An Yan, Xiusi Chen, and Julian McAuley. “Bridging Language and Items for Retrieval and Recommendation”. In: *arXiv preprint arXiv:2403.03952* (2024).
- [124] Yupeng Hou, Jiacheng Li, Zhankui He, An Yan, Xiusi Chen, and Julian J. McAuley. “Bridging Language and Items for Retrieval and Recommendation”. In: *CoRR abs/2403.03952* (2024). DOI: 10.48550/ARXIV.2403.03952. arXiv: 2403.03952.
- [125] Yifan Hu, Yehuda Koren, and Chris Volinsky. “Collaborative Filtering for Implicit Feedback Datasets”. In: *ICDM*. IEEE Computer Society, 2008, pp. 263–272.

- [126] Chao Huang, Huance Xu, Yong Xu, Peng Dai, Lianghao Xia, Mengyin Lu, Liefeng Bo, Hao Xing, Xiaoping Lai, and Yanfang Ye. “Knowledge-aware Coupled Graph Neural Network for Social Recommendation”. In: *AAAI*. AAAI Press, 2021, pp. 4115–4122.
- [127] Ruoran Huang, Chuanqi Han, and Li Cui. “Entity-aware Collaborative Relation Network with Knowledge Graph for Recommendation”. In: *CIKM*. ACM, 2021, pp. 3098–3102.
- [128] Dietmar Jannach, Michael Jugovac, and Ingrid Nunes. “Explanations and User Control in Recommender Systems”. In: *ABIS@HT*. ACM, 2019, p. 31.
- [129] Dietmar Jannach, Lukas Lerche, and Markus Zanker. “Recommending Based on Implicit Feedback”. In: *Social Information Access*. Vol. 10100. Lecture Notes in Computer Science. Springer, 2018, pp. 510–569.
- [130] Sharmila Rani Jetti and MHM Krishna Prasad. “Knowledge Graphs and Neural Networks in Recommendation Systems: A Comprehensive Survey and Future Directions”. In: *2025 3rd International Conference on Intelligent Data Communication Technologies and Internet of Things (IDCIoT)*. IEEE, 2025, pp. 1163–1170.
- [131] Yitong Ji, Aixin Sun, Jie Zhang, and Chenliang Li. “Do Loyal Users Enjoy Better Recommendations?: Understanding Recommender Accuracy from a Time Perspective”. In: *ICTIR '22: The 2022 ACM SIGIR International Conference on the Theory of Information Retrieval, Madrid, Spain, July 11 - 12, 2022*. Ed. by Fabio Crestani, Gabriella Pasi, and Éric Gaussier. ACM, 2022, pp. 92–97. DOI: 10.1145/3539813.3545124.
- [132] Yangqin Jiang, Chao Huang, and Lianghao Huang. “Adaptive Graph Contrastive Learning for Recommendation”. In: *KDD*. ACM, 2023, pp. 4252–4261.
- [133] Mengyuan Jing, Yanmin Zhu, Tianzi Zang, and Ke Wang. “Contrastive Self-supervised Learning in Recommender Systems: A Survey”. In: *ACM Trans. Inf. Syst.* 42.2 (2024), 59:1–59:39.
- [134] Mengyuan Jing, Yanmin Zhu, Tianzi Zang, Jiadi Yu, and Feilong Tang. “Graph Contrastive Learning with Adaptive Augmentation for Recommendation”. In: *ECML/PKDD (1)*. Vol. 13713. Lecture Notes in Computer Science. Springer, 2022, pp. 590–605.
- [135] Serdar Kadioglu and Bernard Kleynhans. “Building Higher-Order Abstractions from the Components of Recommender Systems”. In: *AAAI*. AAAI Press, 2024, pp. 22998–23004.
- [136] Marius Kaminskis and Derek Bridge. “Diversity, Serendipity, Novelty, and Coverage: A Survey and Empirical Analysis of Beyond-Accuracy Objectives in Recommender Systems”. In: *ACM Trans. Interact. Intell. Syst.* 7.1 (2017), 2:1–2:42.

- [137] Thomas N. Kipf and Max Welling. “Semi-Supervised Classification with Graph Convolutional Networks”. In: *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24–26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.
- [138] Yehuda Koren. “Factorization meets the neighborhood: a multifaceted collaborative filtering model”. In: *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Las Vegas, Nevada, USA, August 24–27, 2008*. Ed. by Ying Li, Bing Liu, and Sunita Sarawagi. ACM, 2008, pp. 426–434. DOI: 10.1145/1401890.1401944.
- [139] Yehuda Koren. “Factor in the neighbors: Scalable and accurate collaborative filtering”. In: *ACM Trans. Knowl. Discov. Data* 4.1 (2010), 1:1–1:24.
- [140] Yehuda Koren, Robert M. Bell, and Chris Volinsky. “Matrix Factorization Techniques for Recommender Systems”. In: *Computer* 42.8 (2009), pp. 30–37. DOI: 10.1109/MC.2009.263.
- [141] Dominik Kowald, Elisabeth Lex, and Markus Schedl. “Utilizing Human Memory Processes to Model Genre Preferences for Personalized Music Recommendations”. In: *CoRR* abs/2003.10699 (2020).
- [142] Walid Krichene and Steffen Rendle. “On Sampled Metrics for Item Recommendation”. In: *KDD*. ACM, 2020, pp. 1748–1757.
- [143] Sara Latifi, Noemi Mauro, and Dietmar Jannach. “Session-aware recommendation: A surprising quest for the state-of-the-art”. In: *Inf. Sci.* 573 (2021), pp. 291–315.
- [144] Jian Li, Zhuoming Xu, Yan Tang, Bo Zhao, and Haimei Tian. “Deep Hybrid Knowledge Graph Embedding for Top-N Recommendation”. In: *WISA*. Vol. 12432. Lecture Notes in Computer Science. Springer, 2020, pp. 59–70.
- [145] Lei Li, Yongfeng Zhang, and Li Chen. “Personalized Prompt Learning for Explainable Recommendation”. In: *ACM Trans. Inf. Syst.* 41.4 (2023), 103:1–103:26.
- [146] Muzhi Li, Cehao Yang, Chengjin Xu, Zixing Song, Xuhui Jiang, Jian Guo, Hofung Leung, and Irwin King. “Context-aware Inductive Knowledge Graph Completion with Latent Type Constraints and Subgraph Reasoning”. In: *AAAI*. AAAI Press, 2025, pp. 12102–12111.
- [147] Nian Li, Chen Gao, Depeng Jin, and Qingmin Liao. “Disentangled Modeling of Social Homophily and Influence for Social Recommendation”. In: *IEEE Trans. Knowl. Data Eng.* 35.6 (2023), pp. 5738–5751.
- [148] Dawen Liang, Rahul G. Krishnan, Matthew D. Hoffman, and Tony Jebara. “Variational Autoencoders for Collaborative Filtering”. In: *WWW*. ACM, 2018, pp. 689–698.

- [149] Dawen Liang, Rahul G. Krishnan, Matthew D. Hoffman, and Tony Jebara. “Variational Autoencoders for Collaborative Filtering”. In: *Proceedings of the 2018 World Wide Web Conference*. WWW ’18. Lyon, France: International World Wide Web Conferences Steering Committee, 2018, pp. 689–698. ISBN: 9781450356398. DOI: 10.1145/3178876.3186150.
- [150] Jianghao Lin, Xinyi Dai, Yunjia Xi, Weiwen Liu, Bo Chen, Hao Zhang, Yong Liu, Chuhan Wu, Xiangyang Li, Chenxu Zhu, et al. “How can recommender systems benefit from large language models: A survey”. In: *ACM Transactions on Information Systems* (2023).
- [151] Jimmy Lin. “The Neural Hype and Comparisons Against Weak Baselines”. In: *SIGIR Forum* 52.2 (2018), pp. 40–51. DOI: 10.1145/3308774.3308781.
- [152] Jing Lin, Weike Pan, and Zhong Ming. “FISSA: Fusing Item Similarity Models with Self-Attention Networks for Sequential Recommendation”. In: *RecSys*. ACM, 2020, pp. 130–139.
- [153] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. “Learning Entity and Relation Embeddings for Knowledge Graph Completion”. In: *AAAI*. AAAI Press, 2015, pp. 2181–2187.
- [154] Greg Linden, Brent Smith, and Jeremy York. “Amazon.com Recommendations: Item-to-Item Collaborative Filtering”. In: *IEEE Internet Comput.* 7.1 (2003), pp. 76–80. DOI: 10.1109/MIC.2003.1167344.
- [155] Qin Liu, Xuan Feng, Tianlong Gu, and Xiaoli Liu. “FairCRS: Towards User-oriented Fairness in Conversational Recommendation Systems”. In: *RecSys*. ACM, 2024, pp. 126–136.
- [156] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. “RoBERTa: A Robustly Optimized BERT Pretraining Approach”. In: *CoRR* abs/1907.11692 (2019).
- [157] Yong Liu, Peilin Zhao, Aixin Sun, and Chunyan Miao. “A Boosting Algorithm for Item Recommendation with Implicit Feedback”. In: *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*. Ed. by Qiang Yang and Michael J. Wooldridge. AAAI Press, 2015, pp. 1792–1798.
- [158] Pasquale Lops, Marco Polignano, Cataldo Musto, Antonio Silletti, and Giovanni Semeraro. “ClayRS: An end-to-end framework for reproducible knowledge-aware recommender systems”. In: *Inf. Syst.* 119 (2023), p. 102273.
- [159] Jianxin Ma, Chang Zhou, Peng Cui, Hongxia Yang, and Wenwu Zhu. “Learning Disentangled Representations for Recommendation”. In: *NeurIPS*. 2019, pp. 5712–5723.

- [160] Sijie Mai, Shuangjia Zheng, Yuedong Yang, and Haifeng Hu. “Communicative Message Passing for Inductive Relation Reasoning”. In: *AAAI*. AAAI Press, 2021, pp. 4294–4302.
- [161] Daniele Malitesta, Claudio Pomo, Vito Walter Anelli, Alberto Carlo Maria Mancino, Tommaso Di Noia, and Eugenio Di Sciascio. “A Novel Evaluation Perspective on GNNs-based Recommender Systems through the Topology of the User-Item Graph”. In: *RecSys*. ACM, 2024, pp. 549–559.
- [162] Alberto Carlo Maria Mancino, Antonio Ferrara, **Salvatore Bufi**, Daniele Malitesta, Tommaso Di Noia, and Eugenio Di Sciascio. “KGTORe: Tailored Recommendations through Knowledge-aware GNN Models”. In: *Proceedings of the 17th ACM Conference on Recommender Systems, RecSys 2023, Singapore, Singapore, September 18–22, 2023*. Ed. by Jie Zhang, Li Chen, Shlomo Berkovsky, Min Zhang, Tommaso Di Noia, Justin Basilico, Luiz Pizzato, and Yang Song. ACM, 2023, pp. 576–587. DOI: 10.1145/3604915.3608804.
- [163] Alberto Carlo Maria Mancino, **Salvatore Bufi**, Angela Di Fazio, Antonio Ferrara, Daniele Malitesta, Claudio Pomo, and Tommaso Di Noia. “DataRec: A Python Library for Standardized and Reproducible Data Management in Recommender Systems”. In: *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2025, Padua, Italy, July 13–18, 2025*. Ed. by Nicola Ferro, Maria Maistro, Gabriella Pasi, Omar Alonso, Andrew Trotman, and Suzan Verberne. ACM, 2025, pp. 3478–3487. DOI: 10.1145/3726302.3730320.
- [164] Masoud Mansoury, Bamshad Mobasher, Robin Burke, and Mykola Pechenizkiy. “Bias Disparity in Collaborative Recommendation: Algorithmic Evaluation and Comparison”. In: *Proceedings of the Workshop on Recommendation in Multi-stakeholder Environments co-located with the 13th ACM Conference on Recommender Systems (RecSys 2019), Copenhagen, Denmark, September 20, 2019*. Ed. by Robin Burke, Himan Abdollahpouri, Edward C. Malthouse, K. P. Thai, and Yongfeng Zhang. Vol. 2440. CEUR Workshop Proceedings. CEUR-WS.org, 2019.
- [165] Kelong Mao, Jieming Zhu, Jinpeng Wang, Quanyu Dai, Zhenhua Dong, Xi Xiao, and Xiuqiang He. “SimpleX: A Simple and Strong Baseline for Collaborative Filtering”. In: *CIKM*. ACM, 2021, pp. 1243–1252.
- [166] Kelong Mao, Jieming Zhu, Jinpeng Wang, Quanyu Dai, Zhenhua Dong, Xi Xiao, and Xiuqiang He. “SimpleX: A Simple and Strong Baseline for Collaborative Filtering”. In: *Proceedings of the 30th ACM International Conference on Information & Knowledge Management. CIKM ’21. Virtual Event, Queensland, Australia: Association for Computing Machinery, 2021*, pp. 1243–1252. ISBN: 9781450384469. DOI: 10.1145/3459637.3482297.

- [167] Kelong Mao, Jieming Zhu, Xi Xiao, Biao Lu, Zhaowei Wang, and Xiuqiang He. “UltraGCN: Ultra Simplification of Graph Convolutional Networks for Recommendation”. In: *CIKM*. ACM, 2021, pp. 1253–1262.
- [168] Kelong Mao, Jieming Zhu, Xi Xiao, Biao Lu, Zhaowei Wang, and Xiuqiang He. “UltraGCN: Ultra Simplification of Graph Convolutional Networks for Recommendation”. In: *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. CIKM ’21. Virtual Event, Queensland, Australia: Association for Computing Machinery, 2021, pp. 1253–1262. ISBN: 9781450384469. DOI: 10.1145/3459637.3482291.
- [169] Julian J. McAuley and Jure Leskovec. “Hidden factors and hidden topics: understanding rating dimensions with review text”. In: *RecSys*. ACM, 2013, pp. 165–172.
- [170] Julian J. McAuley, Christopher Targett, Qinfeng Shi, and Anton van den Hengel. “Image-Based Recommendations on Styles and Substitutes”. In: *SIGIR*. ACM, 2015, pp. 43–52.
- [171] Sean M. McNee, John Riedl, and Joseph A. Konstan. “Being accurate is not enough: how accuracy metrics have hurt recommender systems”. In: *Extended Abstracts Proceedings of the 2006 Conference on Human Factors in Computing Systems, CHI 2006, Montréal, Québec, Canada, April 22–27, 2006*. Ed. by Gary M. Olson and Robin Jeffries. ACM, 2006, pp. 1097–1101. DOI: 10.1145/1125451.1125659.
- [172] Christian Meilicke, Manuel Fink, Yanjie Wang, Daniel Ruffinelli, Rainer Gemulla, and Heiner Stuckenschmidt. “Fine-Grained Evaluation of Rule- and Embedding-Based Systems for Knowledge Graph Completion”. In: *ISWC (1)*. Vol. 11136. Lecture Notes in Computer Science. Springer, 2018, pp. 3–20.
- [173] Zaiqiao Meng, Richard McCreadie, Craig Macdonald, and Iadh Ounis. “Exploring Data Splitting Strategies for the Evaluation of Recommendation Models”. In: *RecSys*. ACM, 2020, pp. 681–686.
- [174] Zeyuan Meng, Iadh Ounis, Craig Macdonald, and Zixuan Yi. “Knowledge Graph Cross-View Contrastive Learning for Recommendation”. In: *ECIR (3)*. Vol. 14610. Lecture Notes in Computer Science. Springer, 2024, pp. 3–18.
- [175] Lien Michiels, Robin Verachtert, and Bart Goethals. “RecPack: An(other) Experimentation Toolkit for Top-N Recommendation using Implicit Feedback Data”. In: *RecSys*. ACM, 2022, pp. 648–651.
- [176] Jianmo Ni, Jiacheng Li, and Julian J. McAuley. “Justifying Recommendations using Distantly-Labeled Reviews and Fine-Grained Aspects”. In: *EMNLP/IJCNLP (1)*. Association for Computational Linguistics, 2019, pp. 188–197.
- [177] Xia Ning and George Karypis. “SLIM: Sparse Linear Methods for Top-N Recommender Systems”. In: *2011 IEEE 11th International Conference on Data Mining*. 2011, pp. 497–506. DOI: 10.1109/ICDM.2011.134.

- [178] Xia Ning and George Karypis. “Sparse linear methods with side information for top-n recommendations”. In: *RecSys*. ACM, 2012, pp. 155–162.
- [179] Deng Pan, Xiangrui Li, Xin Li, and Dongxiao Zhu. “Explainable Recommendation via Interpretable Feature Mapping and Evaluation of Explainability”. In: *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*. Ed. by Christian Bessiere. ijcai.org, 2020, pp. 2690–2696. DOI: 10.24963/IJCAI.2020/373.
- [180] Vincenzo Paparella, Vito Walter Anelli, Franco Maria Nardini, Raffaele Perego, and Tommaso Di Noia. “Post-hoc Selection of Pareto-Optimal Solutions in Search and Recommendation”. In: *CIKM*. ACM, 2023, pp. 2013–2023.
- [181] Vincenzo Paparella, Dario Di Palma, Vito Walter Anelli, and Tommaso Di Noia. “Broadening the Scope: Evaluating the Potential of Recommender Systems beyond prioritizing Accuracy”. In: *Proceedings of the 17th ACM Conference on Recommender Systems, RecSys 2023, Singapore, Singapore, September 18–22, 2023*. Ed. by Jie Zhang, Li Chen, Shlomo Berkovsky, Min Zhang, Tommaso Di Noia, Justin Basilico, Luiz Pizzato, and Yang Song. ACM, 2023, pp. 1139–1145. DOI: 10.1145/3604915.3610649.
- [182] Bibek Paudel, Fabian Christoffel, Chris Newell, and Abraham Bernstein. “Updatable, Accurate, Diverse, and Scalable Recommendations for Interactive Applications”. In: *ACM Trans. Interact. Intell. Syst.* 7.1 (2017), 1:1–1:34.
- [183] Michael J. Pazzani and Daniel Billsus. “Content-Based Recommendation Systems”. In: *The Adaptive Web*. Vol. 4321. Lecture Notes in Computer Science. Springer, 2007, pp. 325–341.
- [184] Georgina Peake and Jun Wang. “Explanation Mining: Post Hoc Interpretability of Latent Factor Models for Recommendation Systems”. In: *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. KDD ’18*. London, United Kingdom: Association for Computing Machinery, 2018, pp. 2060–2069. ISBN: 9781450355520. DOI: 10.1145/3219819.3220072.
- [185] Matthew E. Peters, Mark Neumann, Robert L. Logan IV, Roy Schwartz, Vidur Joshi, Sameer Singh, and Noah A. Smith. “Knowledge Enhanced Contextual Word Representations”. In: *EMNLP/IJCNLP (1)*. Association for Computational Linguistics, 2019, pp. 43–54.
- [186] Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander H. Miller. “Language Models as Knowledge Bases?” In: *EMNLP/IJCNLP (1)*. Association for Computational Linguistics, 2019, pp. 2463–2473.
- [187] Jay Pujara, Eriq Augustine, and Lise Getoor. “Sparsity and Noise: Where Knowledge Graph Embeddings Fall Short”. In: *EMNLP*. Association for Computational Linguistics, 2017, pp. 1751–1756.

- [188] Hossein A. Rahmani, Mohammadmehdi Naghiaei, and Yashar Deldjoo. “A Personalized Framework for Consumer and Producer Group Fairness Optimization in Recommender Systems”. In: *Trans. Recomm. Syst.* 2.3 (2024), 19:1–19:24.
- [189] Shashank Rajput et al. “Recommender Systems with Generative Retrieval”. In: *NeurIPS*. 2023.
- [190] Theresia Veronika Rampisela, Maria Maistro, Tuukka Ruotsalo, and Christina Lioma. “Evaluation Measures of Individual Item Fairness for Recommender Systems: A Critical Study”. In: *CoRR* abs/2311.01013 (2023).
- [191] Ramin Raziperchikolaei and Young-joo Chung. “One-class recommendation systems with the hinge pairwise distance loss and orthogonal representations”. In: *RecSys*. ACM, 2024, pp. 1033–1038.
- [192] *Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation)*. <https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32016R0679>. accessed 2024-03-15.
- [193] Xubin Ren, Lianghao Xia, Jiashu Zhao, Dawei Yin, and Chao Huang. “Disentangled Contrastive Collaborative Filtering”. In: *SIGIR*. ACM, 2023, pp. 1137–1146.
- [194] Steffen Rendle. “Factorization Machines”. In: *ICDM*. IEEE Computer Society, 2010, pp. 995–1000.
- [195] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. “BPR: Bayesian Personalized Ranking from Implicit Feedback”. In: *UAI*. AUAI Press, 2009, pp. 452–461.
- [196] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. “BPR: Bayesian Personalized Ranking from Implicit Feedback”. In: *CoRR* abs/1205.2618 (2012).
- [197] Steffen Rendle, Walid Krichene, Li Zhang, and John R. Anderson. “Neural Collaborative Filtering vs. Matrix Factorization Revisited”. In: *RecSys 2020: Fourteenth ACM Conference on Recommender Systems, Virtual Event, Brazil, September 22–26, 2020*. Ed. by Rodrygo L. T. Santos, Leandro Balby Marinho, Elizabeth M. Daly, Li Chen, Kim Falk, Noam Koenigstein, and Edleno Silva de Moura. ACM, 2020, pp. 240–248. doi: 10.1145/3383313.3412488.
- [198] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. “GroupLens: An Open Architecture for Collaborative Filtering of Netnews”. In: *CSCW ’94, Proceedings of the Conference on Computer Supported Cooperative Work, Chapel Hill, NC, USA, October 22–26, 1994*. Ed. by John B. Smith, F. Donelson Smith, and Thomas W. Malone. ACM, 1994, pp. 175–186. doi: 10.1145/192844.192905.

- [199] Francesco Ricci, Lior Rokach, and Bracha Shapira, eds. *Recommender Systems Handbook*. Springer US, 2022.
- [200] Matthew Richardson, Rakesh Agrawal, and Pedro M. Domingos. “Trust Management for the Semantic Web”. In: *The Semantic Web - ISWC 2003, Second International Semantic Web Conference, Sanibel Island, FL, USA, October 20-23, 2003, Proceedings*. Ed. by Dieter Fensel, Katia P. Sycara, and John Mylopoulos. Vol. 2870. Lecture Notes in Computer Science. Springer, 2003, pp. 351–368. doi: 10.1007/978-3-540-39718-2_23.
- [201] Cecil D. Robinson and Randall Schumacker. “Interaction Effects: Centering, Variance Inflation Factor, and Interpretation Issues”. In: 2009.
- [202] Andrea Rossi, Denilson Barbosa, Donatella Firmani, Antonio Matinata, and Paolo Merialdo. “Knowledge Graph Embedding for Link Prediction: A Comparative Analysis”. In: *ACM Trans. Knowl. Discov. Data* 15.2 (2021), 14:1–14:49.
- [203] Evgenia Rusak, Patrik Reizinger, Attila Juhos, Oliver Bringmann, Roland S. Zimmermann, and Wieland Brendel. “InfoNCE: Identifying the Gap Between Theory and Practice”. In: *International Conference on Artificial Intelligence and Statistics, AISTATS 2025, Mai Khao, Thailand, 3-5 May 2025*. Ed. by Yingzhen Li, Stephan Mandt, Shipra Agrawal, and Mohammad Emtiyaz Khan. Vol. 258. Proceedings of Machine Learning Research. PMLR, 2025, pp. 4159–4167.
- [204] Alan Said and Alejandro Bellogín. “Comparative recommender system evaluation: benchmarking recommendation frameworks”. In: *RecSys*. ACM, 2014, pp. 129–136.
- [205] Aghiles Salah, Quoc-Tuan Truong, and Hady W. Lauw. “Cornac: A Comparative Framework for Multimodal Recommender Systems”. In: *J. Mach. Learn. Res.* 21 (2020), 95:1–95:5.
- [206] **Salvatore Bufi**, Alberto Carlo Maria Mancino, Antonio Ferrara, Daniele Malitesta, Tommaso Di Noia, and Eugenio Di Sciascio. “KGUF: Simple Knowledge-Aware Graph-Based Recommender with User-Based Semantic Features Filtering”. In: *Advances on Graph-Based Approaches in Information Retrieval, IRonGraphs 2024, Glasgow, UK, March 24, 2024, Proceedings*. Ed. by Ludovico Boratto, Daniele Malitesta, Mirko Marras, Giacomo Medda, Cataldo Musto, and Erasmo Purificato. Vol. 2197. Communications in Computer and Information Science. Springer, 2024, pp. 41–59. doi: 10.1007/978-3-031-71382-8_4.
- [207] **Salvatore Bufi**, Vincenzo Paparella, Vito Walter Anelli, Franco Maria Nardini, Raffaele Perego, and Tommaso Di Noia. “Explaining GNN-based Recommender Systems Performance under Data Minimization: The Role of Data Characteristics”. Manuscript submitted for publication to the journal *Expert Systems with Applications*. 2025.

- [208] **Salvatore Bufi**, Vincenzo Paparella, Vito Walter Anelli, and Tommaso Di Noia. “Legal but Unfair: Auditing the Impact of Data Minimization on Fairness and Accuracy Trade-off in Recommender Systems”. In: *Proceedings of the 33rd ACM Conference on User Modeling, Adaptation and Personalization, UMAP 2025, New York City, NY, USA, June 16-19, 2025. Best Student Paper Award*. ACM, 2025, pp. 114–123. DOI: 10.1145/3699682.3728356.
- [209] **Salvatore Bufi**, Vincenzo Paparella, Vito Walter Anelli, and Tommaso Di Noia. “Less Data, More Questions: Fairness and Accuracy Under Data Minimization in Recommender Systems”. In: *Proceedings of the 15th Italian Information Retrieval Workshop (IIR 2025)*. Ed. by Ludovico Boratto, Allegra De Filippo, Stefano Marchesin, and Mirko Marras. Cagliari, Italy: CEUR Workshop Proceedings, 2025, pp. 129–135.
- [210] Abdulaziz Samra, Evgeny Frolov, Alexey Vasilev, Alexander Grigorevskiy, and Anton Vakhrushev. “Cross-Domain Latent Factors Sharing via Implicit Matrix Factorization”. In: *RecSys*. ACM, 2024, pp. 309–317.
- [211] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. “Item-based collaborative filtering recommendation algorithms”. In: *Proceedings of the 10th International Conference on World Wide Web. WWW ’01*. Hong Kong, Hong Kong: Association for Computing Machinery, 2001, pp. 285–295. ISBN: 1581133480. DOI: 10.1145/371920.372071.
- [212] Badrul Munir Sarwar, George Karypis, Joseph A. Konstan, and John Riedl. “Item-based collaborative filtering recommendation algorithms”. In: *WWW*. ACM, 2001, pp. 285–295.
- [213] Michael Sejr Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. “Modeling Relational Data with Graph Convolutional Networks”. In: *ESWC*. Vol. 10843. Lecture Notes in Computer Science. Springer, 2018, pp. 593–607.
- [214] Suvash Sedhain, Aditya Krishna Menon, Scott Sanner, and Lexing Xie. “AutoRec: Autoencoders Meet Collaborative Filtering”. In: *Proceedings of the 24th International Conference on World Wide Web. WWW ’15 Companion*. Florence, Italy: Association for Computing Machinery, 2015, pp. 111–112. ISBN: 9781450334730. DOI: 10.1145/2740908.2742726.
- [215] Guy Shani and Asela Gunawardana. “Evaluating Recommendation Systems”. In: *Recommender Systems Handbook*. Springer, 2011, pp. 257–297.
- [216] Divya Shanmugam, Fernando Diaz, Samira Shabanian, Michèle Finck, and Asia Biega. “Learning to Limit Data Collection via Scaling Laws: A Computational Interpretation for the Legal Principle of Data Minimization”. In: *FACCT*. ACM, 2022, pp. 839–849.

- [217] Faisal Shehzad and Dietmar Jannach. “Everyone’s a Winner! On Hyperparameter Tuning of Recommendation Models”. In: *Proceedings of the 17th ACM Conference on Recommender Systems, RecSys 2023, Singapore, Singapore, September 18–22, 2023*. Ed. by Jie Zhang, Li Chen, Shlomo Berkovsky, Min Zhang, Tommaso Di Noia, Justin Basilico, Luiz Pizzato, and Yang Song. ACM, 2023, pp. 652–657. DOI: 10.1145/3604915.3609488.
- [218] Yifei Shen, Yongji Wu, Yao Zhang, Caihua Shan, Jun Zhang, Khaled B. Letaief, and Dongsheng Li. “How Powerful is Graph Convolution for Recommendation?” In: *CIKM*. ACM, 2021, pp. 1619–1629.
- [219] Wentao Shi, Jiawei Chen, Fuli Feng, Jizhi Zhang, Junkang Wu, Chongming Gao, and Xiangnan He. “On the Theories Behind Hard Negative Sampling for Recommendation”. In: *Proceedings of the ACM Web Conference 2023, WWW 2023, Austin, TX, USA, 30 April 2023 - 4 May 2023*. Ed. by Ying Ding, Jie Tang, Juan F. Sequeda, Lora Aroyo, Carlos Castillo, and Geert-Jan Houben. ACM, 2023, pp. 812–822. DOI: 10.1145/3543507.3583223.
- [220] Jie Shuai, Kun Zhang, Le Wu, Peijie Sun, Richang Hong, Meng Wang, and Yong Li. “A Review-aware Graph Contrastive Learning Framework for Recommendation”. In: *SIGIR*. ACM, 2022, pp. 1283–1293.
- [221] Ashudeep Singh and Thorsten Joachims. “Fairness of Exposure in Rankings”. In: *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018, London, UK, August 19–23, 2018*. Ed. by Yike Guo and Faisal Farooq. ACM, 2018, pp. 2219–2228. DOI: 10.1145/3219819.3220088.
- [222] Nasim Sonboli, Sipei Li, Mehdi Elahi, and Asia Biega. “The trade-off between data minimization and fairness in collaborative filtering”. In: *CoRR* abs/2410.07182 (2024).
- [223] Changhao Song, Bo Wang, Qinxue Jiang, Yehua Zhang, Ruifang He, and Yuexian Hou. “Social Recommendation with Implicit Social Influence”. In: *SIGIR*. ACM, 2021, pp. 1788–1792.
- [224] Giuseppe Spillo, Allegra De Filippo, Cataldo Musto, Michela Milano, and Giovanni Semeraro. “Towards Green Recommender Systems: Investigating the Impact of Data Reduction on Carbon Footprint and Algorithm Performances”. In: *RecSys*. ACM, 2024, pp. 866–871.
- [225] Giuseppe Spillo, Cataldo Musto, Marco de Gemmis, Pasquale Lops, and Giovanni Semeraro. “Exploiting Neuro-Symbolic Graph Embeddings based on First-Order Logical Rules for Knowledge-aware Recommendations”. In: *DP@AI*IA*. Vol. 3419. CEUR Workshop Proceedings. CEUR-WS.org, 2022, pp. 1–11.
- [226] Robin Staab, Nikola Jovanovic, Mislav Balunovic, and Martin T. Vechev. “From Principle to Practice: Vertical Data Minimization for Machine Learning”. In: *SP*. IEEE, 2024, pp. 4733–4752.

- [227] Harald Steck. “Evaluation of recommendations: rating-prediction and ranking”. In: *RecSys*. ACM, 2013, pp. 213–220.
- [228] Harald Steck. “Embarrassingly Shallow Autoencoders for Sparse Data”. In: *WWW*. ACM, 2019, pp. 3251–3257.
- [229] Harald Steck. “Embarrassingly Shallow Autoencoders for Sparse Data”. In: *The World Wide Web Conference*. WWW ’19. San Francisco, CA, USA: Association for Computing Machinery, 2019, pp. 3251–3257. ISBN: 9781450366748. DOI: 10.1145/3308558.3313710.
- [230] Aixin Sun. “Take a Fresh Look at Recommender Systems from an Evaluation Standpoint”. In: *SIGIR*. ACM, 2023, pp. 2629–2638.
- [231] Zhu Sun, Hui Fang, Jie Yang, Xinghua Qu, Hongyang Liu, Di Yu, Yew-Soon Ong, and Jie Zhang. “DaisyRec 2.0: Benchmarking Recommendation for Rigorous Evaluation”. In: *IEEE Trans. Pattern Anal. Mach. Intell.* 45.7 (2023), pp. 8206–8226. DOI: 10.1109/TPAMI.2022.3231891.
- [232] Zhu Sun, Di Yu, Hui Fang, Jie Yang, Xinghua Qu, Jie Zhang, and Cong Geng. “Are We Evaluating Rigorously? Benchmarking Recommendation for Reproducible Evaluation and Fair Comparison”. In: *RecSys*. ACM, 2020, pp. 23–32.
- [233] Panagiotis Symeonidis, Lidija Kirjackaja, and Markus Zanker. “Session-aware news recommendations using random walks on time-evolving heterogeneous information networks”. In: *User Model. User Adapt. Interact.* 30.4 (2020), pp. 727–755.
- [234] Gábor Takács, István Pilászy, and Domonkos Tikk. “Applications of the conjugate gradient method for implicit feedback collaborative filtering”. In: *RecSys*. ACM, 2011, pp. 297–300.
- [235] Komal K. Teru, Etienne G. Denis, and William L. Hamilton. “Inductive Relation Prediction by Subgraph Reasoning”. In: *ICML*. Vol. 119. Proceedings of Machine Learning Research. PMLR, 2020, pp. 9448–9457.
- [236] *The California Privacy Rights Act of 2020*. <https://thecpra.org/>. accessed 2024-02-13.
- [237] Changxin Tian, Yuexiang Xie, Yaliang Li, Nan Yang, and Wayne Xin Zhao. “Learning to Denoise Unreliable Interactions for Graph Collaborative Filtering”. In: *SIGIR*. ACM, 2022, pp. 122–132.
- [238] Tianchi. *IJCAI-16 Brick-and-Mortar Store Recommendation Dataset*. 2018.
- [239] Ilaria Tiddi, Freddy Lécué, and Pascal Hitzler, eds. *Knowledge Graphs for eXplainable Artificial Intelligence: Foundations, Applications and Challenges*. Vol. 47. Studies on the Semantic Web. IOS Press, 2020.
- [240] Riku Togashi, Mayu Otani, and Shin’ichi Satoh. “Alleviating Cold-Start Problems in Recommendation through Pseudo-Labeling over Knowledge Graph”. In: *WSDM*. ACM, 2021, pp. 931–939.

- [241] Matús Tomlein, Branislav Pecher, Jakub Simko, Ivan Srba, Róbert Móro, Elena Stefancova, Michal Kompan, Andrea Hrkova, Juraj Podrouzek, and Mária Bielíková. “An Audit of Misinformation Filter Bubbles on YouTube: Bubble Bursting and Recent Behavior Changes”. In: *RecSys '21: Fifteenth ACM Conference on Recommender Systems, Amsterdam, The Netherlands, 27 September 2021 - 1 October 2021*. Ed. by Humberto Jesús Corona Pampín, Martha A. Larson, Martijn C. Willemsen, Joseph A. Konstan, Julian J. McAuley, Jean Garcia-Gathright, Bouke Huurnink, and Even Oldridge. ACM, 2021, pp. 1–11. DOI: 10.1145/3460231.3474241.
- [242] *Translation: Personal Information Protection Law of the People's Republic of China – Effective Nov. 1, 2021*. <https://digichina.stanford.edu/work/translation-personal-information-protection-law-of-the-peoples-republic-of-china-effective-nov-1-2021/>. accessed 2024-02-13.
- [243] Vojtěch Vančura, Rodrigo Alves, Petr Kasalický, and Pavel Kordík. “Scalable Linear Shallow Autoencoder for Collaborative Filtering”. In: *Proceedings of the 16th ACM Conference on Recommender Systems*. RecSys '22. Seattle, WA, USA: Association for Computing Machinery, 2022, pp. 604–609. ISBN: 9781450392785. DOI: 10.1145/3523227.3551482.
- [244] Shikhar Vashishth, Soumya Sanyal, Vikram Nitin, and Partha P. Talukdar. “Composition-based Multi-Relational Graph Convolutional Networks”. In: *ICLR*. OpenReview.net, 2020.
- [245] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. “Graph Attention Networks”. In: *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.
- [246] Tobias Vente, Lukas Wegmeth, Alan Said, and Joeran Beel. “From Clicks to Carbon: The Environmental Toll of Recommender Systems”. In: *Proceedings of the 18th ACM Conference on Recommender Systems, RecSys 2024, Bari, Italy, October 14-18, 2024*. Ed. by Tommaso Di Noia, Pasquale Lops, Thorsten Joachims, Katrien Verbert, Pablo Castells, Zhenhua Dong, and Ben London. ACM, 2024, pp. 580–590. DOI: 10.1145/3640457.3688074.
- [247] Maksims Volkovs and Guang Wei Yu. “Effective Latent Models for Binary Feedback in Recommender Systems”. In: *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '15. Santiago, Chile: Association for Computing Machinery, 2015, pp. 313–322. ISBN: 9781450336215. DOI: 10.1145/2766462.2767716.
- [248] D. Wallach and B. Goffinet. “Mean squared error of prediction as a criterion for evaluating and comparing system models”. In: *Ecological Modelling* 44:3

- (1989), pp. 299–306. ISSN: 0304-3800. DOI: [https://doi.org/10.1016/0304-3800\(89\)90035-5](https://doi.org/10.1016/0304-3800(89)90035-5).
- [249] Chenyang Wang, Yuanqing Yu, Weizhi Ma, Min Zhang, Chong Chen, Yiqun Liu, and Shaoping Ma. “Towards Representation Alignment and Uniformity in Collaborative Filtering”. In: *KDD '22: The 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, August 14 - 18, 2022*. Ed. by Aidong Zhang and Huzefa Rangwala. ACM, 2022, pp. 1816–1825. DOI: 10.1145/3534678.3539253.
- [250] Chenyang Wang, Min Zhang, Weizhi Ma, Yiqun Liu, and Shaoping Ma. “Make It a Chorus: Knowledge- and Time-aware Item Modeling for Sequential Recommendation”. In: *SIGIR*. ACM, 2020, pp. 109–118.
- [251] Hao Wang, Yao Xu, Cheng Yang, Chuan Shi, Xin Li, Ning Guo, and Zhiyuan Liu. “Knowledge-Adaptive Contrastive Learning for Recommendation”. In: *WSDM*. ACM, 2023, pp. 535–543.
- [252] Hongwei Wang, Fuzheng Zhang, Jialin Wang, Miao Zhao, Wenjie Li, Xing Xie, and Minyi Guo. “RippleNet: Propagating User Preferences on the Knowledge Graph for Recommender Systems”. In: *CIKM*. ACM, 2018, pp. 417–426.
- [253] Hongwei Wang, Fuzheng Zhang, Jialin Wang, Miao Zhao, Wenjie Li, Xing Xie, and Minyi Guo. “Exploring High-Order User Preference on the Knowledge Graph for Recommender Systems”. In: *ACM Trans. Inf. Syst.* 37.3 (2019), 32:1–32:26.
- [254] Hongwei Wang, Fuzheng Zhang, Mengdi Zhang, Jure Leskovec, Miao Zhao, Wenjie Li, and Zhongyuan Wang. “Knowledge-aware Graph Neural Networks with Label Smoothness Regularization for Recommender Systems”. In: *KDD*. ACM, 2019, pp. 968–977.
- [255] Hongwei Wang, Miao Zhao, Xing Xie, Wenjie Li, and Minyi Guo. “Knowledge Graph Convolutional Networks for Recommender Systems”. In: *WWW*. ACM, 2019, pp. 3307–3313.
- [256] Shuyao Wang, Yongduo Sui, Chao Wang, and Hui Xiong. “Unleashing the Power of Knowledge Graph for Recommendation via Invariant Learning”. In: *WWW*. ACM, 2024, pp. 3745–3755.
- [257] Wenjie Wang, Yiyan Xu, Fuli Feng, Xinyu Lin, Xiangnan He, and Tat-Seng Chua. “Diffusion Recommender Model”. In: *SIGIR*. ACM, 2023, pp. 832–841.
- [258] Xiang Wang, Xiangnan He, Yixin Cao, Meng Liu, and Tat-Seng Chua. “KGAT: Knowledge Graph Attention Network for Recommendation”. In: *KDD*. ACM, 2019, pp. 950–958.

- [259] 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, SIGIR 2019, Paris, France, July 21-25, 2019*. Ed. by Benjamin Piwowarski, Max Chevalier, Éric Gaussier, Yoelle Maarek, Jian-Yun Nie, and Falk Scholer. ACM, 2019, pp. 165–174. DOI: 10.1145/3331184.3331267.
- [260] 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, SIGIR’19, Paris, France: Association for Computing Machinery, 2019*, pp. 165–174. ISBN: 9781450361729. DOI: 10.1145/3331184.3331267.
- [261] Xiang Wang, Tinglin Huang, Dingxian Wang, Yancheng Yuan, Zhenguang Liu, Xiangnan He, and Tat-Seng Chua. “Learning Intents behind Interactions with Knowledge Graph for Recommendation”. In: *WWW. ACM / IW3C2*, 2021, pp. 878–887.
- [262] Xiang Wang, Hongye Jin, An Zhang, Xiangnan He, Tong Xu, and Tat-Seng Chua. “Disentangled Graph Collaborative Filtering”. In: *SIGIR*. ACM, 2020, pp. 1001–1010.
- [263] Xiaozhi Wang, Tianyu Gao, Zhaocheng Zhu, Zhengyan Zhang, Zhiyuan Liu, Juanzi Li, and Jian Tang. “KEPLER: A Unified Model for Knowledge Embedding and Pre-trained Language Representation”. In: *Trans. Assoc. Comput. Linguistics* 9 (2021), pp. 176–194.
- [264] Yifan Wang, Weizhi Ma, Min Zhang, Yiqun Liu, and Shaoping Ma. “A Survey on the Fairness of Recommender Systems”. In: *ACM Trans. Inf. Syst.* 41.3 (2023), 52:1–52:43.
- [265] Yu Wang, Hengrui Zhang, Zhiwei Liu, Liangwei Yang, and Philip S. Yu. “ContrastVAE: Contrastive Variational AutoEncoder for Sequential Recommendation”. In: *CIKM*. ACM, 2022, pp. 2056–2066.
- [266] Ze Wang, Guangyan Lin, Huobin Tan, Qinghong Chen, and Xiyang Liu. “CKAN: Collaborative Knowledge-aware Attentive Network for Recommender Systems”. In: *SIGIR*. ACM, 2020, pp. 219–228.
- [267] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. “Knowledge Graph Embedding by Translating on Hyperplanes”. In: *AAAI*. AAAI Press, 2014, pp. 1112–1119.
- [268] Ziyang Wang, Huoyu Liu, Wei Wei, Yue Hu, Xian-Ling Mao, Shaojian He, Rui Fang, and Danyang Chen. “Multi-level Contrastive Learning Framework for Sequential Recommendation”. In: *CIKM*. ACM, 2022, pp. 2098–2107.
- [269] Wanxu Wei, Yitong Song, and Bin Yao. “Enhancing Heterogeneous Knowledge Graph Completion with a Novel GAT-based Approach”. In: *ACM Trans. Knowl. Discov. Data* 18.4 (2024), 104:1–104:20.

- [270] Wei Wei, Chao Huang, Lianghao Xia, and Chuxu Zhang. “Multi-Modal Self-Supervised Learning for Recommendation”. In: *WWW*. ACM, 2023, pp. 790–800.
- [271] Yanbin Wei, Qiushi Huang, Yu Zhang, and James T. Kwok. “KICGPT: Large Language Model with Knowledge in Context for Knowledge Graph Completion”. In: *EMNLP (Findings)*. Association for Computational Linguistics, 2023, pp. 8667–8683.
- [272] Hongyi Wen, Longqi Yang, Michael Sobolev, and Deborah Estrin. “Exploring recommendations under user-controlled data filtering”. In: *RecSys*. ACM, 2018, pp. 72–76.
- [273] Fangzhao Wu et al. “MIND: A Large-scale Dataset for News Recommendation”. In: *ACL*. Association for Computational Linguistics, 2020, pp. 3597–3606.
- [274] Jiancan Wu, Xiangnan He, Xiang Wang, Qifan Wang, Weijian Chen, Jianxun Lian, and Xing Xie. “Graph convolution machine for context-aware recommender system”. In: *Frontiers Comput. Sci.* 16.6 (2022), p. 166614.
- [275] Jiancan Wu, Xiang Wang, Fuli Feng, Xiangnan He, Liang Chen, Jianxun Lian, and Xing Xie. “Self-supervised Graph Learning for Recommendation”. In: *SIGIR*. ACM, 2021, pp. 726–735.
- [276] Shiwen Wu, Fei Sun, Wentao Zhang, Xu Xie, and Bin Cui. “Graph Neural Networks in Recommender Systems: A Survey”. In: *ACM Comput. Surv.* 55.5 (2023), 97:1–97:37.
- [277] Yao Wu, Jian Cao, Guandong Xu, and Yudong Tan. “TFROM: A Two-sided Fairness-Aware Recommendation Model for Both Customers and Providers”. In: *SIGIR ’21: The 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, Virtual Event, Canada, July 11-15, 2021*. Ed. by Fernando Diaz, Chirag Shah, Torsten Suel, Pablo Castells, Rosie Jones, and Tetsuya Sakai. ACM, 2021, pp. 1013–1022. DOI: 10.1145/3404835.3462882.
- [278] Yao Wu, Christopher DuBois, Alice X. Zheng, and Martin Ester. “Collaborative Denoising Auto-Encoders for Top-N Recommender Systems”. In: *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining, San Francisco, CA, USA, February 22-25, 2016*. Ed. by Paul N. Bennett, Vanja Josifovski, Jennifer Neville, and Filip Radlinski. ACM, 2016, pp. 153–162. DOI: 10.1145/2835776.2835837.
- [279] Yiqing Wu, Ruobing Xie, Yongchun Zhu, Xiang Ao, Xin Chen, Xu Zhang, Fuzhen Zhuang, Leyu Lin, and Qing He. “Multi-view Multi-behavior Contrastive Learning in Recommendation”. In: *DASFAA (2)*. Vol. 13246. Lecture Notes in Computer Science. Springer, 2022, pp. 166–182.
- [280] Zihao Wu, Xin Wang, Hong Chen, Kaidong Li, Yi Han, Lifeng Sun, and Wenwu Zhu. “Diff4Rec: Sequential Recommendation with Curriculum-scheduled Diffusion Augmentation”. In: *ACM Multimedia*. ACM, 2023, pp. 9329–9335.

- [281] Lianghao Xia, Chao Huang, Chunzhen Huang, Kangyi Lin, Tao Yu, and Ben Kao. “Automated Self-Supervised Learning for Recommendation”. In: *WWW*. ACM, 2023, pp. 992–1002.
- [282] Lianghao Xia, Chao Huang, Jiao Shi, and Yong Xu. “Graph-less Collaborative Filtering”. In: *Proceedings of the ACM Web Conference 2023, WWW 2023, Austin, TX, USA, 30 April 2023 - 4 May 2023*. Ed. by Ying Ding, Jie Tang, Juan F. Sequeda, Lora Aroyo, Carlos Castillo, and Geert-Jan Houben. ACM, 2023, pp. 17–27. doi: 10.1145/3543507.3583196.
- [283] Lianghao Xia, Chao Huang, Yong Xu, Jiashu Zhao, Dawei Yin, and Jimmy X. Huang. “Hypergraph Contrastive Collaborative Filtering”. In: *SIGIR*. ACM, 2022, pp. 70–79.
- [284] Xin Xia, Hongzhi Yin, Junliang Yu, Yingxia Shao, and Lizhen Cui. “Self-Supervised Graph Co-Training for Session-based Recommendation”. In: *CIKM*. ACM, 2021, pp. 2180–2190.
- [285] Xu Xie, Fei Sun, Xiaoyong Yang, Zhao Yang, Jinyang Gao, Wenwu Ou, and Bin Cui. “Explore User Neighborhood for Real-time E-commerce Recommendation”. In: *ICDE*. IEEE, 2021, pp. 2464–2475.
- [286] Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. “Representation Learning on Graphs with Jumping Knowledge Networks”. In: *ICML*. Vol. 80. Proceedings of Machine Learning Research. PMLR, 2018, pp. 5449–5458.
- [287] Fan Yang, Zhilin Yang, and William W. Cohen. “Differentiable Learning of Logical Rules for Knowledge Base Reasoning”. In: *NIPS*. 2017, pp. 2319–2328.
- [288] Yuhao Yang, Chao Huang, Lianghao Xia, and Chunzhen Huang. “Knowledge Graph Self-Supervised Rationalization for Recommendation”. In: *KDD*. ACM, 2023, pp. 3046–3056.
- [289] Yuhao Yang, Chao Huang, Lianghao Xia, and Chenliang Li. “Knowledge Graph Contrastive Learning for Recommendation”. In: *SIGIR*. ACM, 2022, pp. 1434–1443.
- [290] Liang Yao, Chengsheng Mao, and Yuan Luo. “KG-BERT: BERT for Knowledge Graph Completion”. In: *CoRR* abs/1909.03193 (2019).
- [291] Yelp. *Yelp Recommendation Dataset*. 2018–2022.
- [292] Hongzhi Yin, Bin Cui, Jing Li, Junjie Yao, and Chen Chen. “Challenging the Long Tail Recommendation”. In: *Proc. VLDB Endow*. 5.9 (2012), pp. 896–907.
- [293] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L. Hamilton, and Jure Leskovec. “Graph Convolutional Neural Networks for Web-Scale Recommender Systems”. In: *KDD*. ACM, 2018, pp. 974–983.

- [294] Junliang Yu, Xin Xia, Tong Chen, Lizhen Cui, Nguyen Quoc Viet Hung, and Hongzhi Yin. "XSimGCL: Towards Extremely Simple Graph Contrastive Learning for Recommendation". In: *IEEE Trans. Knowl. Data Eng.* 36.2 (2024), pp. 913–926. DOI: 10.1109/TKDE.2023.3288135.
- [295] Junliang Yu, Hongzhi Yin, Min Gao, Xin Xia, Xiangliang Zhang, and Nguyen Quoc Viet Hung. "Socially-Aware Self-Supervised Tri-Training for Recommendation". In: *KDD*. ACM, 2021, pp. 2084–2092.
- [296] Junliang Yu, Hongzhi Yin, Jundong Li, Qinyong Wang, Nguyen Quoc Viet Hung, and Xiangliang Zhang. "Self-Supervised Multi-Channel Hypergraph Convolutional Network for Social Recommendation". In: *WWW*. ACM / IW3C2, 2021, pp. 413–424.
- [297] Junliang Yu, Hongzhi Yin, Xin Xia, Tong Chen, Lizhen Cui, and Quoc Viet Hung Nguyen. "Are Graph Augmentations Necessary?: Simple Graph Contrastive Learning for Recommendation". In: *SIGIR*. ACM, 2022, pp. 1294–1303.
- [298] Keping Yu, Zhiwei Guo, Yu Shen, Wei Wang, Jerry Chun-Wei Lin, and Takuro Sato. "Secure Artificial Intelligence of Things for Implicit Group Recommendations". In: *IEEE Internet Things J.* 9.4 (2022), pp. 2698–2707.
- [299] Fajie Yuan, Xiangnan He, Alexandros Karatzoglou, and Liguang Zhang. "Parameter-Efficient Transfer from Sequential Behaviors for User Modeling and Recommendation". In: *SIGIR*. ACM, 2020, pp. 1469–1478.
- [300] Eva Zangerle and Christine Bauer. "Evaluating Recommender Systems: Survey and Framework". In: *ACM Comput. Surv.* 55.8 (2023), 170:1–170:38.
- [301] Hanwen Zha, Zhiyu Chen, and Xifeng Yan. "Inductive Relation Prediction by BERT". In: *AAAI*. AAAI Press, 2022, pp. 5923–5931.
- [302] Junwei Zhang, Min Gao, Junliang Yu, Lei Guo, Jundong Li, and Hongzhi Yin. "Double-Scale Self-Supervised Hypergraph Learning for Group Recommendation". In: *CIKM*. ACM, 2021, pp. 2557–2567.
- [303] Mengqi Zhang, Shu Wu, Xueli Yu, Qiang Liu, and Liang Wang. "Dynamic Graph Neural Networks for Sequential Recommendation". In: *IEEE Trans. Knowl. Data Eng.* 35.5 (2023), pp. 4741–4753.
- [304] Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. "Deep Learning Based Recommender System: A Survey and New Perspectives". In: *ACM Comput. Surv.* 52.1 (2019), 5:1–5:38.
- [305] Wei Zhang, Junbing Yan, Zhuo Wang, and Jianyong Wang. "Neuro-Symbolic Interpretable Collaborative Filtering for Attribute-based Recommendation". In: *Proceedings of the ACM Web Conference 2022*. WWW '22. Virtual Event, Lyon, France: Association for Computing Machinery, 2022, pp. 3229–3238. ISBN: 9781450390965. DOI: 10.1145/3485447.3512042.

- [306] Yongfeng Zhang and Xu Chen. “Explainable Recommendation: A Survey and New Perspectives”. In: *Found. Trends Inf. Retr.* 14.1 (2020), pp. 1–101. DOI: 10.1561/15000000066.
- [307] Yuan Zhang, Fei Sun, Xiaoyong Yang, Chen Xu, Wenwu Ou, and Yan Zhang. “Graph-based Regularization on Embedding Layers for Recommendation”. In: *ACM Trans. Inf. Syst.* 39.1 (2020), 2:1–2:27. DOI: 10.1145/3414067.
- [308] Zhiyuan Zhang, Xiaoqian Liu, Yi Zhang, Qi Su, Xu Sun, and Bin He. “Pretrain-KGE: Learning Knowledge Representation from Pretrained Language Models”. In: *EMNLP (Findings)*. Vol. EMNLP 2020. Findings of ACL. Association for Computational Linguistics, 2020, pp. 259–266.
- [309] Huan Zhao, Quanming Yao, Jianda Li, Yangqiu Song, and Dik Lun Lee. “Meta-Graph Based Recommendation Fusion over Heterogeneous Information Networks”. In: *KDD*. ACM, 2017, pp. 635–644.
- [310] Wayne Xin Zhao, Zihan Lin, Zhichao Feng, Pengfei Wang, and Ji-Rong Wen. “A Revisiting Study of Appropriate Offline Evaluation for Top- N Recommendation Algorithms”. In: *ACM Trans. Inf. Syst.* 41.2 (2023), 32:1–32:41. DOI: 10.1145/3545796.
- [311] Wayne Xin Zhao et al. “RecBole: Towards a Unified, Comprehensive and Efficient Framework for Recommendation Algorithms”. In: *CIKM*. ACM, 2021, pp. 4653–4664.
- [312] Yuhao Zhao, Rui Chen, Qilong Han, Hongtao Song, and Li Chen. “Unlocking the Hidden Treasures: Enhancing Recommendations with Unlabeled Data”. In: *Proceedings of the 18th ACM Conference on Recommender Systems, RecSys 2024, Bari, Italy, October 14-18, 2024*. Ed. by Tommaso Di Noia, Pasquale Lops, Thorsten Joachims, Katrien Verbert, Pablo Castells, Zhenhua Dong, and Ben London. ACM, 2024, pp. 247–256. DOI: 10.1145/3640457.3688149.
- [313] Kun Zhou, Hui Wang, Wayne Xin Zhao, Yutao Zhu, Sirui Wang, Fuzheng Zhang, Zhongyuan Wang, and Ji-Rong Wen. “S3-Rec: Self-Supervised Learning for Sequential Recommendation with Mutual Information Maximization”. In: *CIKM*. ACM, 2020, pp. 1893–1902.
- [314] Wentao Zhou, Jun Zhao, Tao Gui, Qi Zhang, and Xuanjing Huang. “Inductive Relation Inference of Knowledge Graph Enhanced by Ontology Information”. In: *EMNLP (Findings)*. Association for Computational Linguistics, 2023, pp. 6491–6502.
- [315] Jieming Zhu, Jinyang Liu, Shuai Yang, Qi Zhang, and Xiuqiang He. “Open Benchmarking for Click-Through Rate Prediction”. In: *CIKM*. ACM, 2021, pp. 2759–2769.
- [316] Xinjun Zhu, Yuntao Du, Yuren Mao, Lu Chen, Yujia Hu, and Yunjun Gao. “Knowledge-refined Denoising Network for Robust Recommendation”. In: *SIGIR*. ACM, 2023, pp. 362–371.

- [317] Yu Zhu, Ziyu Guan, Shulong Tan, Haifeng Liu, Deng Cai, and Xiaofei He. “Heterogeneous hypergraph embedding for document recommendation”. In: *Neurocomputing* 216 (2016), pp. 150–162.
- [318] Yuqi Zhu, Xiaohan Wang, Jing Chen, Shuofei Qiao, Yixin Ou, Yunzhi Yao, Shumin Deng, Huajun Chen, and Ningyu Zhang. “LLMs for knowledge graph construction and reasoning: recent capabilities and future opportunities”. In: *World Wide Web (WWW)* 27.5 (2024), p. 58.
- [319] Ziwei Zhu, Jianling Wang, and James Caverlee. “Measuring and Mitigating Item Under-Recommendation Bias in Personalized Ranking Systems”. In: *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, SIGIR 2020, Virtual Event, China, July 25–30, 2020*. Ed. by Jimmy X. Huang, Yi Chang, Xueqi Cheng, Jaap Kamps, Vanessa Murdock, Ji-Rong Wen, and Yiqun Liu. ACM, 2020, pp. 449–458. DOI: 10.1145/3397271.3401177.
- [320] Ding Zou, Wei Wei, Xian-Ling Mao, Ziyang Wang, Minghui Qiu, Feida Zhu, and Xin Cao. “Multi-level Cross-view Contrastive Learning for Knowledge-aware Recommender System”. In: *SIGIR*. ACM, 2022, pp. 1358–1368.
- [321] Ding Zou, Wei Wei, Ziyang Wang, Xian-Ling Mao, Feida Zhu, Rui Fang, and Danyang Chen. “Improving Knowledge-aware Recommendation with Multi-level Interactive Contrastive Learning”. In: *CIKM*. ACM, 2022, pp. 2817–2826.
- [322] Ding Zou, Wei Wei, Feida Zhu, Chuanyu Xu, Tao Zhang, and Chengfu Huo. “Knowledge Enhanced Multi-intent Transformer Network for Recommendation”. In: *WWW (Companion Volume)*. ACM, 2024, pp. 1–9.