UNIVERSITÀ DEGLI STUDI DI BARI ALDO MORO

DEPARTMENT OF COMPUTER SCIENCE

PHD PROGRAMME IN
COMPUTER SCIENCE AND MATHEMATICS
XXXVIII CYCLE
SCIENTIFIC ACADEMIC DISCIPLINE INF0-01/A - INFORMATICS

A Thesis Submitted in Fulfillment
of the Requirements for the Degree of
DOCTOR OF PHILOSOPHY

Knowledge-aware Recommendation Models based on Heterogeneous Embeddings

PhD Candidate:
Giuseppe Spillo

Coordinator:
Prof. Francesca Mazzia

Supervisor:
Prof. Cataldo Musto
Cosupervisor:
Prof. Marco de Gemmis

FINAL EXAM 2026

SWAP Research Group

Department of Computer Science

University of Bari Aldo Moro

Contents

Abstract **xi**

List of Figures **xiii**

List of Tables **xvii**

Publication Lists **xxi**

1 Introduction **1**

1.1 Taxonomy of Recommender Systems 2

1.1.1 Collaborative Filtering 2

1.1.2 Content-based Approaches 3

1.1.3 Knowledge-aware Approaches 3

1.1.4 Hybrid Approaches 4

1.2 Evaluating Recommender Systems 5

1.2.1 *Accuracy*: Measuring the Effectiveness of Recommender Systems 6

1.2.2 *Beyond-Accuracy*: A Sustainable and Multi-objective Perspective 8

1.3 Current Research Gaps in KARS and Contributions of this Thesis . 10

1.3.1 Chapter 3: Enriching Homogeneous Embeddings for Knowledge-aware Recommendations 11

1.3.2 Chapter 4: Recommendations Based on Heterogeneous Item Embeddings 13

1.3.3 Chapter 5: Sustainable and Multi-objective Perspective on Recommender Systems 15

1.3.4 Source Code and Reproducibility 18

2 Knowledge-aware Recommender Systems: Preliminaries **19**

2.1 Knowledge Sources and Encoding Strategies 20

2.1.1 Structured Knowledge 20
2.1.2 Unstructured Knowledge 23
2.2 State-of-the-art KARS Approaches 29
2.3 Formalization of the Problem 33
2.3.1 Interaction data 34
2.3.2 Knowledge Graph 34
2.3.3 Textual data 34
2.3.4 Image data 35
2.3.5 Audio data 35
2.3.6 Video data 35
2.3.7 Problem Statement 35
2.3.8 Datasets 36
MovieLens-1M 36
DBbook 39
Last.fm-2K 40
Amazon Books 42
Mind 43
2.3.9 Baselines 44

3 Enriching Homogeneous Embeddings for Knowledge-aware Recommendations 49
3.1 Knowledge Graph Embeddings with First-order Logic Rules 50
3.1.1 Introduction 50
3.1.2 Methodology 51
3.1.3 Experimental Setting 61
3.1.4 Results and Discussion 64
3.1.5 Take-Home Messages 79
3.2 *GAL-KARS*: Graph Augmentation through LLMs for Knowledge-aware Recommendations 80
3.2.1 Introduction 80
3.2.2 Methodology 82
3.2.3 Problem Formulation 87
3.2.4 Experimental Evaluation 88
3.2.5 Results Discussion 90
3.2.6 Take-Home Messages 94
3.3 Sentence Embeddings Based on LLM-generated User Profiles and Item Descriptions 97
3.3.1 Introduction 97

3.3.2 Methodology . . . 98
3.3.3 Problem Formulation . . . 107
3.3.4 Experimental Evaluation . . . 108
3.3.5 Results and Discussion . . . 109
3.3.6 Take-Home Messages . . . 111
3.4 Evaluating Multimodal Item Embeddings through User Feedback: An Empirical Evaluation . . . 111
3.4.1 Introduction . . . 111
3.4.2 The *Movie4Rec* Web-app . . . 112
3.4.3 Experimental Evaluation . . . 113
3.4.4 Results and Discussion . . . 118
3.4.5 Take-Home Messages . . . 120
3.5 Conclusion . . . 122

4 Recommendations based on Heterogeneous Item Embeddings 125
4.1 Fusion of Graph and Text Embeddings through Deep Architectures 126
4.1.1 Introduction . . . 126
4.1.2 Methodology . . . 127
4.1.3 Experimental Evaluation . . . 130
4.1.4 Take-Home Messages . . . 136
4.2 Content-based Pre-training Strategies for KARS . . . 138
4.2.1 Introduction . . . 138
4.2.2 Methodology . . . 140
4.2.3 Problem Statement . . . 143
4.2.4 Experimental Evaluation . . . 143
4.2.5 Results and Discussion . . . 144
4.2.6 Take-Home Messages . . . 147
4.3 Recommendation Dataset with Multimodal Fusion: a Comprehensive Benchmark . . . 148
4.3.1 Introduction . . . 148
4.3.2 Experimental Session . . . 150
4.3.3 Results Discussion . . . 152
4.3.4 Take-Home Messages . . . 158
4.4 *Gotta Embed Them All!* . . . 159
4.4.1 Introduction . . . 159
4.4.2 Methodology . . . 160
4.4.3 Discussion . . . 164
4.4.4 Problem Formulation . . . 166

4.4.5 Experimental Evaluation 166
4.4.6 Results Discussion 167
4.4.7 Take-Home Messages 175
4.5 Conclusions 176

5 Sustainable and Multi-objective Perspective on RSs 179
5.1 Improving Memory Efficeny of Multi-Modal Recommendation Models 180
5.1.1 Introduction 180
5.1.2 *URecJPQ* in Multi-modal Large-scale Scenarios 182
5.1.3 Basics of RecJPQ 182
5.1.4 Experimental Design 186
5.1.5 Results Discussion 188
5.1.6 Take-Home Messages 191
5.2 Benchmarking State-of-the-art Recommendation Model Carbon Footprint 192
5.2.1 Introduction and Motivations 192
5.2.2 Methodology 195
5.2.3 Experimental Evaluation 197
5.2.4 Results Discussion 199
5.2.5 Discussion and Limitations 207
5.2.6 Take-Home Messages 208
5.3 Data Reduction Approaches to Improve the Emissions-Performance Trade-off 209
5.3.1 Introduction 209
5.3.2 Methodology 210
5.3.3 Experimental Evaluation 213
5.3.4 Discussion of the Results 214
5.3.5 Take-Home Messages 224
5.4 Green Early Stopping Criterion for Sustainable Training 226
5.4.1 Introduction 226
5.4.2 Methodology 226
5.4.3 Experimental Session 229
5.4.4 Results Discussion 230
5.4.5 Take-Home Messages 233
5.5 Conclusions 235

6 Conclusions 239

6.1 Contributions of this Thesis 239
6.2 Future Works 242

Bibliography **245**

UNIVERSITY OF BARI ALDO MORO

Abstract

PhD Programme in Computer Science and Mathematics, XXXIII Cycle

Department of Computer Science

**Knowledge-aware
Recommendation Models based on
Heterogeneous Embeddings**

In recent decades, Recommender Systems have emerged to address information overload by providing personalized suggestions based on users' preferences. Early Recommender Systems used collaborative filtering, relying on user-item interactions, with matrix factorization as a state-of-the-art technique, though affected by the cold-start problem. Content-based Recommender Systems addressed this by using textual item descriptions to build profiles, while hybrid Recommender Systems combine Collaborative Filtering and Content-based approaches to leverage their advantages.

Deep learning has enabled Knowledge-Aware Recommender Systems to incorporate multimodal information sources (including knowledge graphs, text, images, audio, and video) through embedding techniques, improving recommendation precision and novelty. However, current state-of-the-art models often overlook combining these multimodal sources in effective manners.

This thesis fits into this research area and studies Knowledge-Aware Recommendation from different perspectives. First, it examines Knowledge-Aware Recommender Systems approaches based on homogeneous item embeddings, where a single knowledge source is used to enrich item representations and provide recommendations. Then, approaches based on heterogeneous embeddings are discussed, showing that fusing and combining diverse knowledge sources is an effective strategy to improve the overall performance of the recommendation model. Moreover, fusion strategy comparisons are investigated to understand the most appropriate way to combine such embeddings. Finally, beyond-accuracy aspects are explored, aiming to assess the underlying costs of performance improvements in terms of complexity and environmental sustainability.

List of Figures

1.1 Graphical summarization of the contributions provided by this thesis. 11

2.1 A tripartite knowledge graph. Different kinds of entities (users, items, properties) are highlighted with different colors. 20

3.1 Workflow carried out by our model. 51
3.2 Recommendation framework architecture 61
3.3 Values of Kendall-Tau's test and nDCG metric for the LAST.FM dataset 66
3.4 Values of Kendall-Tau's test and nDCG metric for the DBBOOK dataset 66
3.5 Values of Kendall-Tau's test and nDCG metric for the ML1M dataset 67
3.6 Our framework workflow, in which we first *(a)* infer new triples to build an augmented KG, from which we *(b)* learn the embeddings and provide recommendations. 82
3.7 Demographic information form. 113
3.8 Search for a movie. 114
3.9 Profile building. 115
3.10 First displayed list of suggested movies. 116
3.11 Survey instructions. 117
3.12 ResQue questionnaire. 118
3.13 Second displayed list of suggested movies. 119

4.1 Overview of the workflow 128
4.2 Sensitivity Analysis - Performances of Precision, Recall and NDCG on the datasets on varying of different hyper-parameters. 137
4.3 The workflow of our work 139
4.4 The Deep RS architecture. User and item embeddings are projected onto separate linear layers with ReLU as the activation function, then concatenated. Subsequently, a linear layer with a sigmoid activation predicts the recommendation score, which lies in [0,1]. . . . 142

4.5 Sensitivity analysis on the F1 and nDCG metrics at the varying number of training epochs. 147
4.6 Architecture of GETALL!. For each source (graph, text, images, audio, video), users and item embeddings are fused with Self-attention, and the resulting embedding are fused through Cross-Attention, in which unified graph and text embeddings serve as query tensors, all the multimedia content derived embeddings serve as key and value tensors. 162
4.7 Attention mechanisms adopted in GETALL! to fuse modality-specific embeddings. 162
4.8 Dimension of Graph Embeddings 173
4.9 Information Encoded in the Graph 173
4.10 Sentence Embedding Techniques 174
4.11 Image Embedding Techniques 174
4.12 Audio Embedding Techniques (only for ML1M) 174
4.13 Video Embedding Techniques (only for ML1M) 175

5.1 Embedding quantization in RecJPQ: an 8-dimensional embedding is approximated by the concatenation of 4 sub-embeddings, each 2-dimensional. 183
5.2 Comparison between the number of trainable parameters of the BPR model on the Baby23 dataset, for 143K users and 44K items. The vanilla version has 12M total trainable parameters, while URecJPQ has only 32K. 185
5.3 Model performance on the Baby23 and Sports23 datasets, in terms of Recall and NDCG. On the x-axis, the number of trainable parameters is reported, while on the y-axis the metric values are reported. The size of each dot indicates the size of the model checkpoint. . . . 188
5.4 Impact of number of sub-spaces on performance. 189
5.5 Experimental Protocol. 198
5.6 Comparison between carbon emissions (*X-axis*) and performance (*Y-axis*) on **Mind** data. 200
5.7 Comparison between carbon emissions (*X-axis*) and performance (*Y-axis*) on **MovieLens** data. 202
5.8 Comparison between carbon emissions (*X-axis*) and performance (*Y-axis*) on **Amazon-Books** data. 203

5.9 **Mind Dataset.** Cost of the hyperparameter tuning in terms of CO_2 emissions (kg) and performances obtained in terms of recall (dots). Vertical bars refer to the emissions, while red dots refer to the considered metric. 205
5.10 **ML1M Dataset.** Cost of the hyperparameter tuning in terms of CO_2 emissions (kg) and performances obtained in terms of recall (dots). Vertical bars refer to the emissions, while red dots refer to the considered metric. 206
5.11 **Amazon-Books Dataset.** Cost of the hyperparameter tuning in terms of CO_2 emissions (kg) and performances obtained in terms of recall (dots). Vertical bars refer to the emissions, while red dots refer to the considered metric. 207
5.12 Overview of the main steps in our methodology pipeline. 211
5.13 Overview of the Data Reduction Strategies. 211
5.14 Comparisons of the emissions on the *complete* AmazonBooks and ML1M datasets. 215
5.15 Impact of data reduction on CO_2 emissions of DGCF on AmazonBooks (left) and ML1M (right). 216
5.16 Impact of data reduction on CO_2 emissions of NGCF on AmazonBooks (left) and ML1M (right). 217
5.17 Impact of data reduction on CO_2 emissions of LightGCN on AmazonBooks (left) and ML1M (right). 217
5.18 Effect of the Data Reduction Approaches on the Recall on AmazonBooks. 219
5.19 Effect of the Data Reduction Approaches on the Recall on ML1M. . 220
5.20 Effect of the Data Reduction Approaches on the Average Popularity on AmazonBooks. 221
5.21 Effect of the Data Reduction Approaches on the Average Popularity on ML1M. 222
5.22 An example of the accuracy/emission curve graph 228
5.23 Results obtained by applying our GES criterion, with *high* treshold and *low* tolerance. *a)* compares the emissions produced by each model whether the GES criterion is applied (green) or not (red); *b)* and *c)* focus on the trade-off between $C0_2$ emissions (*X-axis*) and recommendation performance *(Y-axis)*. 231

List of Tables

1.1 Summary of source code repositories and software artifacts referenced in this thesis. 18

2.1 Statistics of the ML1M dataset, including KG, textual and multimedia data. 37
2.2 Statistics of the DBbook dataset, including KG, textual and image data. 39
2.3 Statistics of the Last.fm dataset, including KG and multimedia data (not available). 41
2.4 Statistics of the Amazon Books dataset, including KG (multimedia data is not available). 43
2.5 Statistics of the Mind dataset, for which KG and multimedia data are not available. 43

3.1 Some examples of rules mined by our framework. In the tables, we also report the metrics we used for the definition of the heuristics . 56
3.2 Mapping of triples in a KG in FOL formalism. 57
3.3 Statistics about the used subsets of FOL rules 63
3.4 Top-k real recommendations provided by different configurations of our model in the movie field, for Users *6039*, *5743*, and *12*; for this example, we set *k=5*. 65
3.5 Results of our model on the Last.FM dataset. In **bold**, the best overall score, while the best score obtained by a configuration based on FOL rules is underlined. 69
3.6 Results of our model on the DBbook dataset. In **bold**, the best overall score, while the best score obtained by a configuration based on FOL rules is underlined. 71
3.7 Results of our model on the ML1M dataset. In **bold**, the best overall score, while the best score obtained by a configuration based on FOL rules is underlined. 73

3.8 Comparisons to baselines in the music field (Last.fm dataset). The best overall value is highlighted in bold, while the best baseline value is underlined. (*) means the differences are statistically significant. 75
3.9 Comparisons to baselines in the book field (DBbook dataset). The best overall value is highlighted in bold, while the best baseline value is underlined. (*) means the differences are statistically significant. 76
3.10 Comparisons to baselines in the movie field (ML1M dataset). The best overall value is highlighted in bold, while the best baseline value is underlined. (*) means the differences are statistically significant with $p < 0.05$. 78
3.11 Real example of *zero-shot* prompting to the Item Feature Inference Module, with a subset of the obtained KG triples 83
3.12 Real example of *zero-shot* prompting to the User Preference Inference Module, with a subset of the obtained KG triples 85
3.13 Knowledge graph statistics for DBBOOK and ML1M. 89
3.14 Statistics about side information in different KGs for DBBOOK and for ML1M. 90
3.15 A comparison between the triples encoded for the book 'The Dark Knight Strikes Again'. We omit the head entity, as it is the same for all the triples (the item itself). 92
3.16 Ablation results on DBBOOK. Best results are in **bold**, second-best in *italic*; * denotes significance ($p = 0.05$) vs. $\mathcal{KG}$. 93
3.17 Ablation results on ML1M. Best results are in **bold**, second-best in *italic*; * denotes significance ($p = 0.05$) vs. $\mathcal{KG}$. 94
3.18 Comparison to baselines on DBBOOK. Best results are in **bold**, second-best in *italic*; * means statistically significant ($p = 0.05$). . . . 95
3.19 Comparison to baselines on ML1M. Best results are in **bold**, second-best in *italic*; * means statistically significant ($p = 0.05$). For fairness, KARS models use the augmented graph $\mathcal{KG}_{\mathcal{I}}^{+}$. 96
3.20 Single source configuration performance on the ML1M dataset. . . 109
3.21 Single source configuration performance on the DBbook dataset. . 109
3.22 Multiple sources configuration performance on the ML1M dataset. 110
3.23 Multiple sources configuration performance on the DBbook dataset. 110
3.24 Demographic Information of the participants involved in the study. 120
3.25 Average ratings for each question across different information sources. The bold values indicate the highest score for each row. 121

4.1 Comparison to baselines on ML1M data. Best-performing baseline is reported in *italics*. Overall best performing configuration is highlighted in **bold**. Significant improvements are emphasized with * (for $p < 0.05$) and ** (for $p < 0.01$). 132
4.2 Comparison to baselines on DBBOOK data. Best-performing baseline is reported in *italics*. Overall best performing configuration is highlighted in **bold**. Significant improvements are emphasized with * (for $p < 0.05$) and ** (for $p < 0.01$). 133
4.3 Results of the ablation test on ML1M data. Best-performing configuration is reported in **bold**. Each line indicates the module which is removed from the architecture. Significant improvements are emphasized with (*) (for $p < 0.05$) and (**) (for $p < 0.01$). 134
4.4 Results of the ablation test on DBBOOK. Best-performing configuration is reported in **bold**. Each line indicates the module which is removed from the architecture. Significant improvements are emphasized with (*) (for $p < 0.05$) and (**) (for $p < 0.01$). 135
4.5 Statistics of the datasets. 144
4.6 Comparison between the full model, in which both user and item embeddings are initialized with PTCB embeddings, a variant in which only item embeddings are initialized with PTCB embeddings, and the conventional setting (no pre-training). Best results are in **bold**; (*) and (**) indicate statistically significant differences with p-value < 0.05 and < 0.01, respectively. 145
4.7 Baseline comparison on ML1M. Best results in **bold**; (**) means $p < 0.01$. 145
4.8 Baseline comparison on DBBOOK. Best results in **bold**; (**) means $p < 0.01$. 146
4.9 Encoders we used per modality 150
4.10 Datasets used for the experiment, in which we kept only the items for which *all* the modalities are available. 153
4.11 ML1M *uni-modal* results. The best results per model are colored, while the best overall result is in **bold.** 153
4.12 DBbook *uni-modal* results. The best results per model are colored, while the best overall result is in **bold.** 153
4.13 Last.fm-2K *uni-modal* results. The best results per model are colored, while the best overall result is in **bold.** 154
4.14 ML1M *multi-modal* results. The best results per model are colored, while the best overall result is in **bold.** 154

4.15 DBbook *multi-modal* results. The best overall result is in **bold**. . . . 155
4.16 Last.fm-2K *multi-modal* results. The best results per model are colored, while the best overall result is in **bold.** 155
4.17 Comparison to baselines in the movie domain (ML1M dataset. The best overall value is highlighted in bold, while the best baseline value is underlined. (*) means the differences are statistically significant with $p < 0.01$. 168
4.18 Comparison to baselines in the movie domain (DBBOOK dataset). The best overall value is highlighted in bold, while the best baseline value is underlined. (*) means the differences are statistically significant with $p < 0.01$. 169
4.19 Ablation tests performed on the movie domain (ML1M dataset). Best values are in **bold**, second best values are underlined. Differences statistically significant are highlighted with * ($p < 0.01$) and ** ($p < 0.05$). 171
4.20 Ablation tests performed on the book domain (DBBOOK dataset). Best values are in **bold**, second best values are underlined. Differences statistically significant are highlighted with * ($p < 0.01$) and ** ($p < 0.05$). 172

5.1 Comparison of our URecJPQ with other quantization approaches, highlighting the novelty of the proposed model. 186
5.2 Statistics of the datasets used in the experiments (on the left), compared to previous and smaller versions of the same datasets (on the right). 186
5.3 Impact of code assignment strategy on performance (Recall@20). . 190
5.4 Percentage variation of Emissions and Recall@10 for the Amazon-Books datasets. 223
5.5 Percentage variation of Emissions and Recall@10 for the ML1M datasets. 223
5.6 Sensitivity analysis under **high threshold** settings for different levels of user tolerance. Values represent percentage change in emissions, recall, and Gini index. 232
5.7 Sensitivity analysis under **low threshold** settings for different levels of user tolerance. Values represent percentage change in emissions, recall, and Gini index. 232

Publication Lists

International Journal Papers

- Allegra De Filippo et al. "Recommender systems and sustainability: a dual perspective". In: *Computer Science Review* 60 (2026), p. 100912. **Q1** (Scimago).
- Giuseppe Spillo et al. "Balancing carbon footprint and algorithm performance in recommender systems: A comprehensive benchmark". In: *Sustainable Computing: Informatics and Systems* (2025), p. 101286. **Q1** (Scimago).
- Giuseppe Spillo et al. "*Gotta Embed Them All!* - Knowledge-aware Recommendations Fusing Heterogeneous Multi-Modal Item Embeddings". In: *Journal of Intelligent Information Systems* (2025), pp. 1–27. **Q2** (Scimago).
- Giuseppe Spillo et al. "Comparing data reduction strategies for energy-efficient green recommender systems". In: *Journal of Intelligent Information Systems* (2025), pp. 1–27. **Q2** (Scimago).
- Giuseppe Spillo et al. "Recommender systems based on neuro-symbolic knowledge graph embeddings encoding first-order logic rules". In: *User Modeling and User-Adapted Interaction* 34.5 (2024), pp. 2039–2083. **Q1** (Scimago).
- Cataldo Musto, Giuseppe Spillo, Giovanni Semeraro, et al. "Harnessing distributional semantics to build context-aware justifications for recommender systems". In: *User Modeling and User-Adapted Interaction* (2023). **Q1** (Scimago).

International Conference Papers

- Giuseppe Spillo et al. "See the Movie, Hear the Song, Read the Book: Extending MovieLens-1M, Last. fm-2K, and DBbook with Multimodal Data". In: *Proceedings of the Nineteenth ACM Conference on Recommender Systems*. 2025, pp. 847–856. **Rank A** (Core2026).
- Giuseppe Spillo et al. "Training Green and Sustainable Recommendation Models: Introducing Carbon Footprint Data into Early Stopping Criteria".

In: *Proceedings of the 33rd ACM Conference on User Modeling, Adaptation and Personalization*. 2025, pp. 341–346. **Rank A** (Core2026).

- Giuseppe Spillo et al. "GAL-KARS: Exploiting LLMs for Graph Augmentation in Knowledge-Aware Recommender Systems". In: *Proceedings of the 33rd ACM Conference on User Modeling, Adaptation and Personalization*. 2025, pp. 73–82. **Rank A** (Core2026).
- Giuseppe Spillo et al. "Towards Green Recommender Systems: Investi- gating the Impact of Data Reduction on Carbon Footprint and Model Performances". In: *Proceedings of the 18th ACM Conference on Recommender Systems*. 2024. **Rank A** (Core2026).
- Giuseppe Spillo et al. "Evaluating Content-based Pre-Training Strategies for a Knowledge-aware Recommender System based on Graph Neural Networks". In: *Proceedings of the 32nd ACM Conference on User Modeling, Adaptation and Personalization*. 2024, pp. 165–171. **Rank A** (Core2026).
- Alessandro Petruzzelli et al. "Improving transformer-based sequential conversational recommendations through knowledge graph embeddings". In: *Proceedings of the 32nd ACM Conference on User Modeling, Adaptation and Personalization*. 2024, pp. 172–182. **Rank B** (Core2023).
- Giuseppe Spillo. "Knowledge-Aware Recommender Systems based on Multi-Modal Information Sources". In: *Proceedings of the 17th ACM Conference on Recommender Systems*. 2023, pp. 1312–1317. **Rank A** (Core2023).
- Giuseppe Spillo et al. "Towards sustainability-aware recommender systems: analyzing the trade-off between algorithms performance and carbon footprint". In: *Proceedings of the 17th ACM Conference on Recommender Systems*. 2023, pp. 856–862. **Rank A** (Core2023).
- Giuseppe Spillo. "Combining Heterogeneous Embeddings for Knowledge-Aware Recommendation Models". In: *Proceedings of the 31st ACM Conference on User Modeling, Adaptation and Personalization*. 2023, pp. 269–273. **Rank B** (Core2023).
- Giuseppe Spillo et al. "Combining graph neural networks and sentence encoders for knowledge-aware recommendations". In: *Proceedings of the 31st ACM Conference on User Modeling, Adaptation and Personalization*. 2023, pp. 1–12. **Rank B** (Core2023).

- Giuseppe Spillo et al. “Knowledge-aware Recommendations Based on Neuro-Symbolic Graph Embeddings and First-Order Logical Rules”. In: *Proceedings of the 16th ACM Conference on Recommender Systems*. 2022, pp. 616–621. **Rank A** (Core2023).

Tutorials in International Conferences

- Allegra De Filippo, Ludovico Boratto, and Giuseppe Spillo. “Human-Centered and Sustainable Recommender Systems”. In: *Adjunct Proceedings of the 33rd ACM Conference on User Modeling, Adaptation and Personalization*. 2025, pp. 10–12. **Rank A** (Core2026).

International Workshop Papers

- Antonio Raffaele Iacovazzi et al. “E-Mealio: An LLM-Powered Conversational Agent for Sustainable and Healthy Food Recommendation”. In: *Second International Workshop on Recommender Systems for Sustainability and Social Good. In press.* 2025
- Alessandro Vicenti et al. “Estimating Product Carbon Footprint via Large Language Models for Sustainable Recommender Systems”. In: *Second International Workshop on Recommender Systems for Sustainability and Social Good. In press.* 2025
- Giuseppe Spillo et al. “Graph Augmentation with LLMs for Knowledge-Aware Recommender Systems”. In: *Proceedings of the 5th Italian Information Retrieval Workshop*. Vol. 4026. CEUR Workshop Proceedings. CEUR-WS.org, 2025
- Giuseppe Spillo et al. “Recsys CarbonAtor: Predicting carbon footprint of recommendation system models”. In: *International Workshop on Recommender Systems for Sustainability and Social Good*. Springer. 2024, pp. 98–110
- Giuseppe Spillo et al. “Exploiting Neuro-Symbolic Graph Embeddings based on First-Order Logical Rules for Knowledge-aware Recommendations.” In: *DP@ AI* IA*. 2022, pp. 1–11

International Conference Submitted Papers

- Giuseppe Spillo et al. “URecJPQ: Memory-efficient Multimodal Recommendation Models through RecJPQ in Large-Scale Scenarios”. In: *Submitted to*

ACM International Conference on User Modeling, Adaptation and Personalization (UMAP). 2026. 2026. **Rank A** (Core2023).

Chapter 1

Introduction

Recommender Systems (RS) are becoming a crucial part of everyone's daily life in many diverse situations [147]. The reason for their success lies in the fact that they are a solution to the *information overload* [81] problem faced daily while surfing the Internet: due to the vast amount of information available, filtering irrelevant information can be a hard and long process. From this perspective, RSs act as filters by providing personalized suggestions of relevant information, based on previous interactions.

As shown by several pieces of evidence, people typically use this technology to get suggestions about movies to enjoy, music to listen to, places to visit, and, in general, *to make decisions* [81]. Over the years, such systems have shown their effectiveness in both *low-risk* domains as well as in more complex and riskier scenarios, including finance or healthcare [34].

In a nutshell, these systems take as input information about user preferences together with some side information (*i.e.*, descriptive features of the items, contextual features, etc.) and provide as output a ranked list of items that can be of interest to the user [148].

This chapter serves as an introduction to the field and will analyze the categories of RSs at the *state-of-the-art*, along with common evaluation approaches used to measure how accurate RSs are.

1.1 Taxonomy of Recommender Systems

1.1.1 Collaborative Filtering

The current spread of RSs is the consequence of a very long path. Indeed, most of the models that were originally proposed tended to *oversimplify* the recommendation process, since they just relied on *user-item* interactions [86], and this approach is defined as *Collaborative Filtering* (CF) [81]. CF is based on the assumption that users with similar preferences, reflected by their consumption of common items, are likely to enjoy similar content. As a result, CF recommends items to a user based on those rated by similar users. The main limitation of this approach is the *cold-start* [79]: in the case of a new user, the recommendation cannot be provided, since their interaction history is empty; the same happens in the case of new items, as the absence of previous ratings makes them impossible to be recommended.

A widely adopted way to apply CF is matrix factorization [86], which can be implemented in several ways. Historically, the first approaches exploited *distance-based* methods, such as the *k*-nearest neighbors. For example, UserKNN and ItemKNN [50] suggest items to users based on the similarity between users (in the former case) and the similarity of liked items (in the latter case).

Subsequent approaches to implement CF were based on the *Singular Value Decomposition* (SVD) [154, 7]. In this setting, the user-item interaction matrix Y is approximated by the product of three matrices U, Σ, V^T. U and V^T represent the user and item latent features, which allow a RS to rank user preferences and predict new recommendations. Matrix factorization has evolved over time, and new approaches based on neural networks and deep learning were proposed as well, such as *Deep Matrix Factorization* (DMF) [235] or Neural Matrix Factorization (NeuMF) [71]. These models use linear layers coming from neural networks to replace or approximate the dot product used to compute the recommendation scores, widely used in the classic matrix factorization.

Currently, among the most popular methods to implement CF, there is a class of neural networks called *Graph Neural Networks* [232] (GNNs). In particular, *Graph Convolutional Networks* [249] (GCNs) showed prominent results in the recommendation field. The idea behind the application of GCNs for the recommendation task consists in modeling the interaction matrix Y as a bipartite user-item graph. Then, GCNs perform the *message-passing* process, in which each node shares its information with its neighbors. This way, each node is able to update its own representation by performing a convolution operation on the information received by the set of its neighbors. LightGCN [70] and its variants [115] are among the most prominent approaches to exploit GCNs to implement CF.

1.1.2 Content-based Approaches

Another approach to tackling the recommendation problem is *content-based filtering*. In this case, the RS builds a profile of the user by analyzing the items that have already been rated, and, based on the descriptive attributes (which are typically in textual form), similar and unseen items are recommended [2, 124].

A content-based RS is typically composed of a *content analyzer*, a *profile learner*, and a *filtering component* [107]. The content analyzer exploits encoding techniques to learn meaningful representations of the items, based on their textual descriptive attributes (e.g., the *genre* of movies). These representations are used by the profile learner to represent users based on the items they like and on their characteristics. Finally, the filtering component uses the available descriptions of the users (encoded by the content analyzer) and the user profiles (generated by the profile learner) to provide recommendations about new items in which users might be interested.

However, content-based RSs have drawbacks as well: first, in the case of a new user for which no information is available, there are no ratings nor item attribute preferences for building the user's profile, and this issue is partially shared with CF. The second problem is called *overspecialization*: given the nature of content-based RS, users can only receive recommendations similar to their earlier experiences [2], and this might negatively affect the diversity of the provided recommendations.

1.1.3 Knowledge-aware Approaches

All the above-mentioned approaches are very effective at providing recommendations. However, they ignore the huge amount of information that is currently available in knowledge bases and Knowledge Graphs (KGs) such as Wikidata [208] and DBpedia [8]. Knowledge-Aware Recommender Systems (KARS) aim to fill this gap, since they use explicit knowledge about the domain of interest to generate recommendations. In this setting, knowledge can be both structured and non-structured, since KARS exploit knowledge that is generally encoded into ontological and logic-based knowledge bases (KBs), KGs, or the semantics emerging from the analysis of unstructured sources [6], including plain text and multimedia information (images, audio, video).

As shown by the research line investigating *semantics-aware recommender systems* [44] and KARS [62], features from KGs and KBs can be very useful for enriching item and user representations and for obtaining more precise and effective recommendation models. This intuition is confirmed by several works [125, 188, 5, 246], generally showing that the injection of *knowledge elements* typically increases the performance of RSs. In this scenario, a state-of-the-art research line concerns the exploitation of *knowledge graph embeddings* [24]. Basically, graph embedding (GE) techniques take as input a graph and return as output a set of vectors (embeddings) representing nodes and relations. Such representation depends on the nature of the chosen embedding model, and it tends to preserve the structural equivalence between nodes. As shown in the literature, such techniques have obtained very good performance in a broad range of scenarios where data can be modeled as a graph, such as biology and social networks [58], and RSs are no exception [128, 188, 163].

Moreover, *unstructured knowledge* should also be taken into consideration as a knowledge source for KARS. Several studies have shown how *user-generated reviews* encapsulate rich semantic information, such as possible explanations of users' preferences and descriptions of specific item attributes, thus representing a rich source of information about users' preferences, and can be exploited to build fine-grained user profiles and, in turn, to generate recommendations [124]. Similarly, state-of-the-art models to handle text, like BERT [67], SBERT [145], and their derivatives, have given a major improvement in state-of-the-art strategies for content representation. Finally, multimedia content can also be used to generate effective recommendations. On the one hand, audio, video, and images can be used to recommend multimedia items (such as movies or music). On the other hand, non-multimedia items can be recommended by leveraging multimedia content related to them (e.g., clothes based on the visual appearance of respective photos) [124].

1.1.4 Hybrid Approaches

However, regardless of the specific approach to build KARS, a crucial aspect of this recommendation paradigm is the learning of a precise representation of users and items. To this end, it is often necessary to combine several information sources, such as information deriving from KGs, text (user reviews or a textual description of the item), and multimedia content (e.g., the movie poster or the trailer of the movie) into a unique and comprehensive representation.

There are also other approaches to integrate different sources. For example, in this line of research, several works have shown that fusing heterogeneous item representations improves the overall performance of KARS. The simplest strategy consists of concatenating embeddings from different sources, including graph and text [136]. Moreover, considering multimedia embeddings (audio, video, images, referred to as *modalities*), has shown performance improvements as well [258, 247].

However, also in this setting, most of research rely on simple concatenation as fusion strategy [69, 257]. In this line of research, a key issue is the alignment of different modalities learned from different sources: different modalities provide features that cannot be simply concatenated to improve feature coverage, but they need to be aligned in a more sophisticated way for the task of recommendation [238].

1.2 Evaluating Recommender Systems

The evaluation of RSs is useful for measuring the quality of the suggestions in the recommendation lists provided by the model. Metrics commonly adopted in the field do not focus solely on the *accuracy* of the model, which measures how precisely the RS provides good suggestions, but also on *beyond-accuracy* metrics that measure other aspects, such as biases in the items recommended or the environmental impact of the model, providing a wider perspective on RSs' capabilities. In general, given a recommendation list that suggests, for each user, at most n items, several metrics can be computed to assess both accuracy and beyond-accuracy metrics.

Among the most commonly used metrics are:

- **Accuracy**: Precision, Mean Average Precision (MAP), Recall, Mean Average Recall (MAR), F1, Normalized Discounted Cumulative Gain (nDCG);
- **Beyond-accuracy**: Gini index, Expected Popularity Complement (EPC), Average Percentage of Long-Tail Items (APLT), Carbon Footprint.

Furthermore, many evaluation metrics can be computed at a cutoff k (commonly referred to as @k metrics), by considering only the top-k elements in the recommendation list. This notation is commonly adopted in the literature and will be used in this thesis as well.

Now, each metric will be described and analyzed in more detail.

1.2.1 *Accuracy*: Measuring the Effectiveness of Recommender Systems

The following are typical metrics used to measure the accuracy of RSs:

- Precision: The ratio of true positives to the sum of true positives and false positives. It ranges from 0 (worst value) to 1 (best value):

  $$Precision = \frac{TP}{TP + FP}$$

  When considering a *k*-cutoff, it is computed as follows:

  $$Precision@k = \frac{TP@k}{TP@k + FP@k}$$

- **Mean Average Precision (MAP)**: Measures the model's ability to rank relevant items higher in the list of predictions. It ranges from 0 (worst) to 1 (best):

  $$\text{MAP@}k = \frac{1}{|U|}\sum_{u=1}^{|U|} \text{AP@}k_u$$

  where $|U|$ is the number of users, and AP@k_u is the Average Precision at cutoff k for user u, defined as:

  $$\text{AP@}k = \frac{1}{\min(m, k)}\sum_{i=1}^{k} P(i) \cdot \text{rel}(i)$$

  Here, m is the number of relevant items for the user, $P(i)$ is the precision at rank i, and rel(i) is a binary relevance indicator (1 if the item at position i is relevant, 0 otherwise).

- Recall: The true positive rate, or sensitivity. It is the ratio between true positives and the sum of true positives and false negatives. It ranges from 0 (worst value) to 1 (best value).

  $$Recall = \frac{TP}{TP + FN}$$

  When considering a *k*-cutoff, it is computed as follows:

$$Recall@k = \frac{TP@k}{TP@k + FN@k}$$

- Mean Average Recall (MAR): The ability to recall all items that are relevant for users. It ranges from 0 (worst value) to 1 (best value):

$$MAR@k = \frac{1}{|U|}\sum_{u=1}^{|U|}(Recall@k)_u$$

- F-measure: The F-measure combines Precision and Recall. It is computed as follows:

$$F = (1+\beta^2)\cdot\frac{Precision \cdot Recall}{(\beta^2 \cdot Precision) + Recall}$$

  A common F-measure is the F1, which represents the harmonic mean of precision and recall and is obtained as follows:

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

  When considering a *k*-cutoff, it is computed as follows:

$$F1@k = 2\cdot\frac{Precision@k \cdot Recall@k}{Precision@k + Recall@k}$$

- Normalized Discounted Cumulative Gain (nDCG): A ranking quality score that ranges from 0 (worst value) to 1 (best value):

$$nDCG@k = \frac{DCG@k}{IDCG@k}$$

  where

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

  and

$$IDCG@k = \sum_{i=1}^{|REL@k|}\frac{rel_i}{\log_2(i+1)}$$

1.2.2 *Beyond-Accuracy*: A Sustainable and Multi-objective Perspective

Beyond-Accuracy metrics are not directly related to accuracy, but they are essential to provide a more comprehensive perspective on the RS. These aspects can include explanations [250], transparency [23], fairness [53], and, more generally, sustainability [18, 11, 42].

The following beyond-accuracy metrics are used to evaluate the recommendation lists in this thesis:

- Gini index: it is a *diversity* metric that measures the inequality in the recommendation lists. A perfectly equal RS would have every item recommended the same number of times, in which case the Gini index would be equal to 0. The more unequal the RS is in providing suggestions, the closer the Gini index is to 1. It is computed as follows:

  $$GiniIndex = \frac{\sum_i (2i - n - 1) \cdot x_i}{n \cdot \sum_i x_i}$$

  where n is the total number of distinct items that are being recommended and x_i is the number of times that the item i has been recommended.

- The Expected Popularity Complement (EPC): it is *novelty* metric that expresses the expected number of seen relevant recommended items not previously seen. It ranges from 0 to 1, and a higher value indicates greater novelty of the recommendation list:

  $$EPC = C \sum_{i_k \in R} disc(k) p(rel|i_k, u)(1 - p(i|seen, 0))$$

  where C is a normalization constant that scales the EPC value to be between 0 and 1, R is the recommendation list containing items i_k at various cutoffs k, $disc(k)$ is a discount function applied to the item's cutoff k typically giving higher importance to items appearing earlier in the list, $p(rel|i_k, u)$ is the probability that item i_k is relevant to user u, $p(i|seen, 0)$ is the probability that item i has already been seen by the user prior to this recommendation, and the term $(1 - p(i|seen, 0))$ captures the novelty of the item, giving the likelihood that the user has not encountered it before.

- Serendipity: a metric that evaluates both *novelty* and *relevance* simultaneously. It reflects the extent to which the recommended items are both relevant and surprising to users, or the likelihood that users receive unexpected suggestions that also match their interests. High values of serendipity express that users are recommended with relevant and new items.

  When considering a *k*-cutoff, it is computed as follows:

  $$Serendipity@k = (\sum_{i=1}^{k} max(p(i) - pu(i), 0) * rel(i))/k$$

  where $p(i)$ is the probability to recommend an item to the user i position i, $pu(i)$ is the probability to recommend to *any* user the same item in position i, $rel(i)$ is the relevance of the item to user (thus, it is 1 if the item in position i is relevant, 0 otherwise).

- The Average Percentage of Long-Tail Items (APLT): the percentage of items belonging to the long-tail that have been recommended.

  When considering a *k*-cutoff, it is computed as follows:

  $$APLT@k = \frac{1}{|U|}\Sigma_{u\in U}\frac{|i, i \in R(u) \cap LongTail|}{|R(u)|}$$

  where U is the set of users, $R(u)$ is the recommendation list provided to user u, and $LongTail$ is the set of items in the long tail.

- Carbon footprint: this metric is not directly related to the recommendation lists, but refers to the electrical power required to carry out any computation (both model training and inference). It is expressed in terms of CO_2-eq grams emitted by the computation. One of the contributions of this thesis is to highlight the importance of CO_2 emissions resulting from the training of recommendation models. It is typically computed as follows:

  $$CO_2\text{-eq} = CarbonIntensity \cdot PowerConsumption$$

  $CarbonIntensity$ represents the amount of CO_2-eq emitted per unit of energy consumed, reflecting the energy source's environmental impact, and is expressed as g/kWh; $PowerConsumption$ refers to the total energy used to carry out the computation, and is expressed as kWh.

1.3 Current Research Gaps in KARS and Contributions of this Thesis

Given this context discussed above, current research in KARS has scarcely investigated effective approaches to fuse heterogeneous knowledge sources into a single, richer representation, mostly relying on simple strategies such as plain concatenation of the representations. However, different knowledge sources (e.g., knowledge graphs, text, images) may provide complementary information that cannot be effectively combined through simple concatenation, as this may also introduce noise. More sophisticated fusion strategies are therefore needed to highlight the most relevant features from each modality for the recommendation task.

In addition, the current literature mostly focus on improving the accuracy of the designed models, ignoring other relevant aspects, such as diversity and novelty of the recommendation lists, or the energy efficiency of the designed models, an issue that has become increasingly relevant in the contemporary global context.

Accordingly, this manuscript investigates which are the performances of RSs relying on homogeneous item embeddings, and how these representation can be improved to obtain more effective recommendations. Then, the manuscripts investigates strategies to effectively fuse the knowledge provided by the heterogeneous sources to obtain better recommendations. Finally, it investigates beyond-accuracy aspects of RSs, with a focus on resource and energy efficiency of the recommendation models, which are aspects typically overlooked in the literature RSs evaluation.

In line with these objectives, this manuscript addresses the following research questions (RQs):

1. **RQ1 - Contribution of Individual Knowledge Sources:** How does each knowledge source, considered individually, contribute to the overall performance of the recommendation? Which are the most effective strategies to enrich the representations associated to these knowledge sources?

2. **RQ2 - Fusion of Heterogeneous Embeddings:** How do heterogeneous embeddings affect the overall performance of the recommendations, when they are fused together? How do different fusion strategies affect the performance?

3. **RQ3 - Beyond-accuracy Implications:** How do heterogeneous embeddings influence the beyond-accuracy performance of the recommendation?

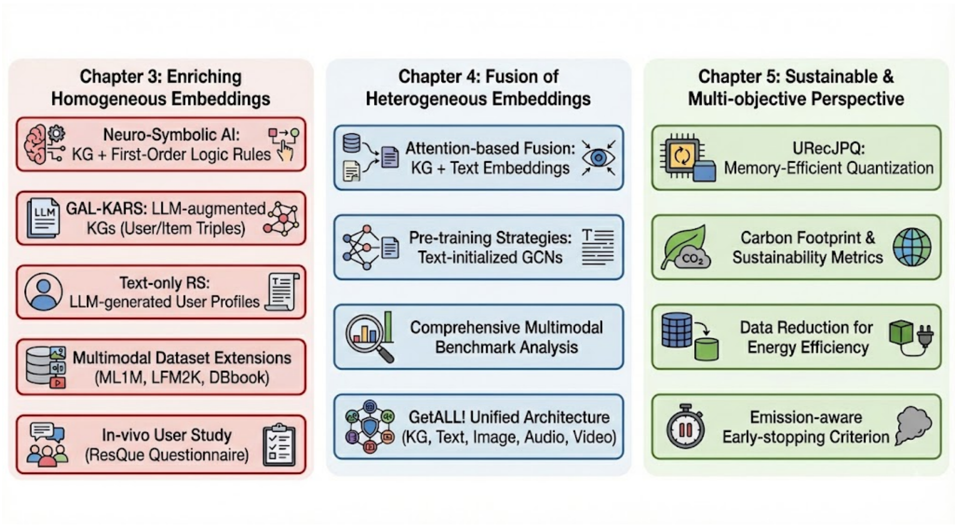

Chapter 3: Enriching Homogeneous Embeddings
Neuro-Symbolic AI: KG + First-Order Logic Rules
GAL-KARS: LLM-augmented KGs (User/Item Triples)
Text-only RS: LLM-generated User Profiles
Multimodal Dataset Extensions (ML1M, LFM2K, DBbook)
In-vivo User Study (ResQue Questionnaire)
Chapter 4: Fusion of Heterogeneous Embeddings
Attention-based Fusion: KG + Text Embeddings
Pre-training Strategies: Text-initialized GCNs
Comprehensive Multimodal Benchmark Analysis
GetALL! Unified Architecture (KG, Text, Image, Audio, Video)
Chapter 5: Sustainable & Multi-objective Perspective
URecJPQ: Memory-Efficient Quantization
Carbon Footprint & Sustainability Metrics
Data Reduction for Energy Efficiency
Emission-aware Early-stopping Criterion

FIGURE 1.1: Graphical summarization of the contributions provided by this thesis.

Each research question is discussed and investigated in the subsequent chapters of this thesis, whose structure is discussed in the following. Chapter 2 provides background information related to knowledge representations and their applications in the recommendation domain. Then, Chapter 3 discusses approaches based on homogeneous item embeddings, and how such representations can be enriched to provide more effective recommendations. Chapter 4 focuses on heterogeneous item embeddings and approaches to effectively fuse them, in order to provide more precise recommendations. Chapter 5 discusses the sustainable and multi-objective perspectives of knowledge-aware recommendations, focusing not only on pure accuracy, but also on beyond-accuracy metrics for a broader evaluation perspective. Finally, Chapter 6 draws conclusions and sketches future research directions.

A graphical representation summarizing the contributions provided by this thesis is reported in Figure 1.1.

1.3.1 Chapter 3: Enriching Homogeneous Embeddings for Knowledge-aware Recommendations

This chapter discusses works based on homogeneous item embeddings. It investigates how homogeneous knowledge sources can be exploited in RSs and how they affect the recommendation performance, both from accuracy and non-accuracy perspectives. In addition, each section investigates how to enrich such representations in novel manner that have never been explored in the KARS literature.

These topics are related to different works conducted during the PhD and here discussed. In the first discussed work (Section 3.1), KGs are used to infer First-Order Logic Rules to be injected during the embedding process and subsequently used for recommendations. This approach falls in the general category of Neuro-Symbolic AI [153], a topic scarcely investigated in RSs. The core idea here is to exploit First-Order Logic Rules, inferred from the KG, to improve the embeddings learnt from the KG itself, thus improving the performance of the RS.

The findings of this work have been deeply discussed in two published papers [173, 174]: *Knowledge-aware recommendations based on neuro-symbolic graph embeddings and first-order logical rules* [173], **published** in the Proceedings of the 16th ACM Conference on Recommender Systems (RecSys 23). **Rank A** (Core2023), and *Recommender systems based on neuro-symbolic knowledge graph embeddings encoding first-order logic rules* [174], **published** in Journal of User Modeling and User-Adapted Interaction (UMUAI). **Q1** (Scimago).

The second work discussed in this chapter is GAL-KARS (Section 3.2). Similarly to the previous case, the core idea of this work is to improve the embeddings derived from KGs. To this aim, LLMs are exploited to infer additional KG triples describing user preferences on one hand and item features on the other hand. The augmented KG is then encoded with a KG encoder, and the the resulting embeddings are used to provide recommendations.

The findings of this work have been discussed in the paper *GAL-KARS: Exploiting LLMs for Graph Augmentation in Knowledge-Aware Recommender Systems*, **published** in the Proceedings of the 33rd ACM Conference on User Modeling, Adaptation and Personalization (UMAP 25). **Rank A** (Core Core2026).

Another work based on LLMs is described in Section 3.3. In this case, instead of relying on KGs, a recommendation model based on pure text is designed. Also in this case, LLMs are used to obtain textual descriptions of user profiles based on their preferences. Then, these profiles are encoded together with item descriptions to obtain user and item embeddings, which are finally used to provide recommendations.

Then, multimedia knowledge sources (images, audio, video) are discussed. In this research area, the literature lacks of relevant, high quality and comprehensive recommendation benchmark datasets for which several sources are available. One of the contributions carried out during the PhD and discussed in this thesis is the extension of three popular state-of-the-art benchmark datasets with multimedia features. The procedure of such extension is discussed through the next chapter, in the Problem Formalization section 2.3, where all the datasets used in the experiments discussed in thesis are introduced. A benchmark analysis investigating the effectiveness of each individual data source, and a comparison with these feature combined, are reported in the Chapter 4. This structure facilitates a clearer evaluation of the contribution introduced by multimodal feature fusion.

These results have also been discussed more in details in the paper [176] *See the Movie, Hear the Song, Read the Book: Extending MovieLens-1M, Last.fm-2K, and DB-book with Multimodal Data* [176], **published** in the Proceedings of the 19th ACM Conference on Recommender Systems (ACM RecSys 25). **Rank A** (Core2026).

Finally, with one of the extended datasets for which the highest number of modalities were available, a user study has been conducted, and presented in Section 3.4. The purpose of this study is to systematically investigate the advantages and disadvantages of providing suggestions by relying on single knowledge sources. The comparative analysis focused on both standard quantitative performance metrics (accuracy) and a broader range of qualitative and impact factors (beyond-accuracy considerations), both evaluated through the ResQue questionnaire [137]. This represents a relevant advance in the literature, as *in-vivo* experiments, directly involving real users, are conducted far less frequently than *in-vitro* experiments, primarily assessing the technical effectiveness of a methodology.

1.3.2 Chapter 4: Recommendations Based on Heterogeneous Item Embeddings

This chapter focuses on recommendations based on the fusion of heterogeneous knowledge embeddings coming from different sources, including KGs, textual, and multimedia content (images, audio, video).

Each section discusses works that have been carried out during the PhD.

In the first work, discussed in Section 4.1, a novel fusion strategy is proposed to combine knowledge graph embeddings and textual embeddings into a unified representation. This strategy is based on deep architectures and attention mechanisms. The resulting embeddings are then used to provide recommendations. This work represent a relevant improvement with respect to the current literature, which only focus on simpler strategies, such as the concatenation or the sum of embeddings.

The findings of this work have been deeply discussed in the paper [167]: *Combining graph neural networks and sentence encoders for knowledge-aware recommendations* [167], **published** in the Proceedings of the 31st ACM Conference on User Modeling, Adaptation and Personalization (ACM UMAP 23). **Rank B** (Core2023). This paper has also been awared with the **James Chen Best Student Paper Award**[1].

Another approach to fuse heterogeneous embeddings is discussed in Section 4.2. Here, pre-training strategies are investigated. Instead of randomly initializing the embeddings of a graph encoder representing users and items, as it is common in the field, in this work the core idea is to initialize them with pre-trained text-based embeddings. This way, the rich semantics coming from the text is exploited and refined by the message passing of the GCNs and the graph structure, and the resulting embeddings are then used to provide recommendations. Pre-training strategies exploited in this way are poorly investigated in the KARS literature to perform knowledge sources fusion, whereas the common strategies to do so, as discussed above, are commonly based on embedding concatenation.

The findings of this work have been published in the paper [169] *Evaluating Content-based Pre-Training Strategies for a Knowledge-aware Recommender System based on Graph Neural Networks* [169], **published** in the Proceedings of the 32nd ACM Conference on User Modeling, Adaptation and Personalization (ACM UMAP 24). **Rank B** (Core2023).

The third work (Section 4.3) discussed in this chapter is a comprehensive benchmark performed to explore how multimedia embeddings affect recommendation performance. In particular, the extended datasets ML1M, DBbook, Last.fm-2K introduced in Section 2.3 have been considered. The benchmark analysis first covers how the various modalities perform when treated individually, then they are fused together to assess how modality fusion affects the performance, and this structure allows to better assess this aspect. Such an analysis on the mentioned benchmark datasets is currently missing in the literature, as these datasets were never released with the discussed multimedia sources.

These results have been discussed in the paper [176] *See the Movie, Hear the Song, Read the Book: Extending MovieLens-1M, Last.fm-2K, and DBbook with Multimodal*

[1]`https://www.um.org/awards/james-chen-best-student-paper-awards`

Data [176], **published** in the Proceedings of the 19th ACM Conference on Recommender Systems (ACM RecSys 25). **Rank A** (Core2026).

Finally, a novel recommendation architecture, *Gotta Embed Them All!* (GETALL! for short) is discussed in Section 4.4. This work focuses on the fusion of all the available heterogeneous embeddings (KGs, text, image, audio, video) into a unique representation. To this aim, a deep architecture based on attention mechanisms has been designed and developed. While the literature has explored the combination of embeddings derived from different multimedia content, the integration of such data with KGs within a single, unified architecture remains largely unexplored.

The findings of this work has been deeply discussed in the paper [177] *Gotta Embed Them All! - Knowledge-aware Recommendations Fusing Heterogeneous Multi-Modal Item Embeddings* [177], **published** in the Journal of Intelligent Information Systems. **Q2** (Scimago).

1.3.3 Chapter 5: Sustainable and Multi-objective Perspective on Recommender Systems

This chapter focuses on beyond-accuracy aspects of the KARS, with a particular focus on resource and energy efficiency. This is a relevant aspect that should be considered, since the current literature in KARS mostly focuses on improving the accuracy, but overlooks beyond-accuracy metrics and the general efficiency of the designed models.

The first work described (Section 5.1) focuses on the application of RecJPQ [132] in large-scale and multimodal recommendation scenarios. The core idea is to quantize user and item embeddings in order to reduce the number of trainable parameters, improving the memory-efficiency of recommendation models, while keeping good accuracy performance. To this end, URecJPQ is introduced as the first approach in the KARS literature to dramatically reduce the number of trainable parameters and the checkpoint sizes of the trained models, while previous approaches only focused on sequential CF recommendation.

This research was conducted during a visiting period at the University of Glasgow, and the findings have been discussed in the paper *URecJPQ: Memory-efficient Multimodal Recommendation Models through RecJPQ in Large-Scale Scenarios* [181], **submitted** at the 34th ACM International Conference on User Modeling, Adaptation and Personalization (UMAP). **Rank A** (Core2026).

Then, novel aspects in the evaluation of recommendations, the environmental sustainability and the carbon footprint, are introduced in Section 5.2. To this end, a work discussing the trade-off between state-of-the-art RSs (including KARS) and emissions during training is presented. The carbon footprint analysis extends conventional evaluation frameworks by introducing a new dimension for assessing beyond-accuracy factors, which has not been previously explored in the literature. This contribution offers a novel multi-objective perspective for evaluating KARS.

The findings of this analysis have been discussed in the papers [179, 166]: *Towards green recommender systems: Investigating the impact of data reduction on carbon footprint and algorithm performance* [178], **published** in the Proceedings of the 18th ACM Conference on Recommender Systems (ACM RecSys 24). **Rank A** (Core2023) and *Balancing Carbon Footprint and Algorithm Performance in Recommender Systems: a Comprehensive Benchmark* [166], **published** in the Journal of Sustainable Computing: Informatics and Systems. **Q1** (Scimago). In addition, a tutorial about these topics [42] has been delivered at ACM UMAP 25: *Human-Centered and Sustainable Recommender Systems* [42], **published** in the Adjunt Proceedings of the 33rd ACM Conference on User Modeling, Adaptation and Personalization (ACM UMAP 25). **Rank A** (Core2026).

Finally, some approaches to improve the environmental sustainability of recommender systems, an aspect largely missing in the literature, are discussed, building upon the research directions first introduced in previous works. In this research line, the first work (Section 5.3) investigates how applying data reduction strategies to the train datasets affects the overall trade-off between performance and emissions of RSs and KARS. The findings of this work have been published in the papers [178, 168] *Towards green recommender systems: Investigating the impact of data reduction on carbon footprint and algorithm performance* [178], **published** in the Proceedings of the 18th ACM Conference on Recommender Systems (ACM RecSys 24). **Rank A** (Core2023) and *Comparing data reduction strategies for energy-efficient green recommender systems* [168], **published** in Journal of Intelligent Information Systems. **Q2** (Scimago).

The second work belonging to the research line, focusing on how to improve the trade-off between RS emissions and performance, is discussed in Section 5.4. This work investigates if emission data can be considered in order to early-stop the recommendation training process when the increase in the emissions does not justify the improvements in the validation score.

The findings of this work have been discussed in the paper [180] *Training Green* *and Sustainable Recommendation Models: Introducing Carbon Footprint Data into Early Stopping Criteria* [180], **published** in the Proceedings of the 33rd ACM Conference on User Modeling, Adaptation and Personalization (ACM UMAP 25). **Rank A** (Core2026).

1.3.4 Source Code and Reproducibility

To support the reproducibility of the research presented in this thesis, the underlying source code and software repositories have been made publicly available. Table 1.1 provides a consolidated summarization of these resources. Furthermore, direct links to the repositories are included within the specific sections where the corresponding contributions are discussed.

Chapter	Section	Repository URL
Chapter 3	Section 3.1	`https://github.com/giuspillo/RepoNeSyRecSys2022` `https://github.com/swapUniba/KARS_NeSy_KGE_with_FOL_rules`
	Section 3.2	`https://github.com/swapUniba/gal-kars`
Chapter 4	Section 4.1	`https://github.com/swapUniba/UMAP23_KARSCombiningGNNsEncSentenceEnc`
	Section 4.2	`https://github.com/swapUniba/PTCB_wikiembs_KARS`
	Section 4.3	`https://github.com/swapUniba/multimodal_ml1m_dbbook_lfm2k`
	Section 4.4	`https://github.com/giuspillo/gotta_embed_them_all`
Chapter 5	Section 5.2	`https://github.com/swapUniba/CarbonRecommenderRecSys23` `https://github.com/swapUniba/RecSysCarbonFootprint`
	Section 5.3	`https://github.com/swapUniba/datared-green-recsys` `https://github.com/swapUniba/recsys_carbonfoot_data_reduction`
	Section 5.4	`https://github.com/swapUniba/GES-green-early-stop`

TABLE 1.1: Summary of source code repositories and software artifacts referenced in this thesis.

Chapter 2

Knowledge-aware Recommender Systems: Preliminaries

This chapter focuses on KARS, discussing how to represent knowledge source in the field, and how *state-of-the-art* KARS use such sources.

In particular, the first section focuses on the knowledge sources, both structured and unstructured, together with *state-of-the-art* encoders to obtain effective representations.

The second section discusses how the KARS literature exploits and treats such knowledge sources. This is done by reviewing the most popular knowledge-aware recommendation models on the market, highlighting the way the sources are used in the studies.

Finally, this chapter formalizes the common framework for KARS that is shared among all the works discussed in the next chapters of this manuscript. The formalization of the problem also includes the discussion of the datasets used to carry out the experiments reported in this thesis, and briefly discusses the baseline models used in the experimental settings reported in this thesis.

2.1 Knowledge Sources and Encoding Strategies

This section examines the different types of knowledge sources leveraged in KARS, encompassing both structured sources, such as KGs and ontologies, and unstructured sources, including textual descriptions, reviews, and other multimedia content-based data relevant for this manuscript, such as images, audio, and video. It further discusses state-of-the-art encoding techniques used to transform these heterogeneous forms of knowledge into effective representations suitable for recommendation tasks. By addressing both the nature of the knowledge and the corresponding encoding approaches, this section establishes the basis for understanding how representation learning contributes to more relevant recommendations.

2.1.1 Structured Knowledge

One of the most common knowledge sources used in this field is represented by KGs [39], whose effectiveness in KARS has already been acknowledged [121, 122, 108]. KGs encode side information associated with items (*e.g.*, the genres of a movie or the author of a book) and are represented as *triples* of the form *(item, relation, entity)*. An example of KG in the movie domain is depicted in Figure 2.1.

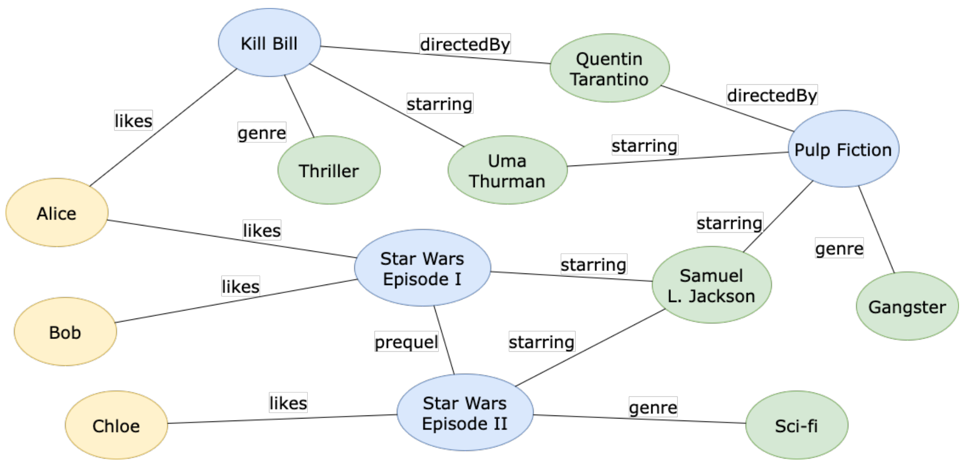


FIGURE 2.1: A tripartite knowledge graph. Different kinds of entities (users, items, properties) are highlighted with different colors.

The *state of the art* in KG encoding is represented by Knowledge Graph Embedding (KGE) techniques [215]. The main idea behind KGE is to represent entities (nodes) and relations (edges) in a continuous vector space, where the geometry of the embeddings preserves the semantic structure of the original KG. In this way, KGE methods enable efficient computation and reasoning over complex graph relationships, facilitating tasks such as link prediction, entity classification, and semantic similarity estimation. Common KGE models include translational approaches (e.g., TransE [19], TransH [244], TransR [101]) and semantic matching approaches (e.g., DistMult [237], ComplEx [197], ConvE [51]), which differ in how they model the interactions between entities and relations. These techniques have become a cornerstone in modern RSs, as they allow the integration of structured knowledge with latent feature representations, improving the system's ability to capture implicit relationships and enhance recommendation quality.

KGE models have evolved significantly over time, moving from simple translational and semantic matching approaches to more sophisticated neural architectures. Among the most prominent recent developments there are GNNs [233] and their variants, such as GCNs [249], which represent KGs in an embedding form while preserving their structural and relational properties. These models exploit message-passing mechanisms to aggregate information from neighboring nodes, allowing them to capture higher-order dependencies and contextual semantics. As a result, GNN- and GCN-based KGE methods have shown remarkable performance improvements in tasks such as link prediction [215], node classification [85], and recommendation [70], where relational reasoning and context-awareness are crucial.

In many works conducted during the PhD and presented in this thesis, a specific GCN model was employed to encode graph-based data, including both user–item interaction graphs and KGs. Accordingly, in this section, first GCNs are discussed in general, highlighting their functionality and limitations, and then CompGCN [202] is introduced, which serves as the foundation for several studies presented in this thesis.

As discussed above, the key idea behind GCNs is to aggregate information propagated by the neighborhood of a node. In particular, a single GCN layer captures information from one-hop neighbors for each node, while stacking k GCN layers enables the capture of information from k-hop neighbors.

Generally speaking, let $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{R})$ be a directed multi-relational graph, where $\mathcal{V}$ is the set of nodes, $\mathcal{R}$ is the set of relation types, and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{R} \times \mathcal{V}$ is the set of edges consisting of triplets (v, r, u) representing a directed connection of type r from source node v to target node u.

Then, given a node $v \in \mathcal{V}$, with $N(v)$ denoting the set of its one-hop neighbors, let $r \in \mathcal{R}$ be the relation connecting v to a neighbor $u \in N(v)$. The representation

$\vec{v} \in \mathbb{R}^d$ of the node v is given by:

$$\vec{v} = f\left(\sum_{(u,r)\in N(v)} W_r \vec{u}\right) \tag{2.1}$$

where $W_r \in \mathbb{R}^{d_1 \times d_0}$ is the relation-specific weight matrix for relation r (with d_0 and d_1 the dimensions of the two stacked layers), and f is an activation function.

Equation 2.1 models the behavior of a single layer of a GCN, which captures the information coming from one-hop neighbors. To capture information coming from N-hop nodes it is necessary to stack N layers of the same type.

However, such a basic formulation has some weaknesses. First, it considers only undirected edges, while in some cases, it may be necessary to treat the directions of relations differently. Second, when the number of relations encoded in the graph is large, one weight matrix W is learned for each relation r, which makes GCNs inefficient and prone to *over-parametrization*. Given that many different relations are available in a KG, standard GCNs are not optimal in our setting.

To better capture relationships between nodes and to learn more precise graph embeddings, CompGCN [202] extends classic GCNs by learning a weight relation matrix specific to each relation and to each direction of edges through a function $\lambda(\cdot)$ that returns the *direction* of the relation r (outgoing, incoming, self-loop).

Accordingly, following CompGCN's paper [202], Equation 2.2 can be rewritten as follows:

$$\vec{v} = \sum_{(u,r)\in N(v)} W_{\lambda(r)} \Phi\left(\vec{u}, \vec{r}\right) \tag{2.2}$$

where $\vec{u}$ and $\vec{r}$ are the embeddings of node u and relation r, respectively; Φ is a composition operator, $W_{\lambda(r)}$ is a weight relation matrix and $\lambda\left(r\right)$ is the function discussed above.

In this formulation, the embedding associated with node v is learned by first considering each neighbor u connected by relation r. Then, a composition operator (*i.e.,* the subtraction as in TransE [19], but many other operators can be considered) is applied to combine the embeddings of the neighbor and the embedding of the relation. The resulting embedding is multiplied by a weight relation matrix $W_{\lambda(r)}$, learned by the model, and dependent on the direction of the edge connecting the two nodes.

Formally, given a node $v \in \mathcal{V}$, and $N(v)$ the set of its one-hop nodes, let $r \in \mathcal{R}$ be the relation connecting v to a neighbor $u \in N(v)$. When stacking k CompGCN layers, the node updating formula for the node v can be written as:

$$\vec{v}^{k+1} = \sum_{(u,r) \in N(v)} W^{k}_{\lambda(r)} \Phi \left(\vec{u}^{k}, \vec{r}^{k} \right) \tag{2.3}$$

where $\vec{u}^k$ and $\vec{r}^k$ are the embeddings of node u and relation r learned at the k-th layer, respectively. Φ is a composition operator, $W_{\lambda(r)}$ is a weight relation matrix and $\lambda(\cdot)$ is the function discussed above.

This methodology can also be applied in the recommendation setting. Given a bipartite graph representing user–item interactions or a knowledge graph encoding both user–item interactions and item properties (both formalized in Section 2.3), Equation 2.3 is used to learn embeddings $\vec{u_g}$ and $\vec{i_g}$ for each user $u \in \mathcal{U}$ and item $i \in \mathcal{I}$. these embeddings can be combined with side information to generate recommendations, or directly utilized as the basis for recommendation generation.

2.1.2 Unstructured Knowledge

Text Encoders Similar to what was discussed regarding graph encoding, text embeddings provide a way to represent textual data in a continuous vector space, capturing semantic and syntactic relationships between words, sentences, or documents.

In many of the works conducted during the PhD and presented in this thesis, specific text embedding techniques were employed to encode textual information. Accordingly, this section first reviews general embedding methods for text, and then introduces the specific approach used in the studies, which serves as a foundation for several works presented in this thesis.

In this research line, early attempts started with approaches such as Word2Vec [118]. Currently, the state of the art is represented by models based on *transformers*, whose strength lies in their ability to learn precise *word embeddings* that capture the importance of each word with respect to all preceding and following words. Examples of models in this research area are BERT [52] and its derivatives, such as RoBERTa [106] (which employs more robust pre-training on a larger corpus of text and hyperparameter optimization), or MPNet [162], which adds information related to the position of each token in the sentence.

An extension of word embedding models is represented by sentence embedding models: while BERT focuses on representing individual words, SBERT [145] aims to learn *sentence embeddings* by adopting a siamese neural network [40], enabling the model to compute a similarity score between sentences. To improve the overall efficiency of these models, Microsoft proposed MiniLM [217], a distilled transformer model (that is, a model with similar performance to the teacher model but with lower computational cost).

Given a textual description, such as the plot of a movie, it can be modeled as a sequence of terms $\{w_1, w_2, \ldots, w_T\}$. Following [52], the embedding of each term is learned by summing three embeddings (token, position, segment). This represents the first layer of a Transformer. Formally, let $H^0 = \{h_t^0\} \, \forall t \in [1, T]$ be the set of embeddings learned at the first layer.

Next, Transformers stack N encoding blocks, in which each block acts as a *many-to-many encoder* that maps each word embedding learned at the previous layer to all the word embeddings learned at the current layer. Moreover, through the use of Multi-Head Attention [203], multiple sets of attention scores are used for different parts of the input (the *heads*).

As stated in [203], Transformers exploit attention mechanisms to learn a representation that becomes more accurate at each encoding layer. Intuitively, thanks to attention mechanisms, different portions of the input text influence each other based on their semantics, and this leads to representations that become more accurate at each encoding layer.

After stacking N encoding layers, the output is the set of embeddings $H^N = \{h_1^N, \ldots, h_T^N\}$ that represent the final representations of the words. In order to obtain an embedding of the entire sentence, SBERT [145] is employed. In particular, SBERT employs a Siamese Neural Network [40], an architecture with two or more identical sub-networks joined together to process different input data. This enables sentences to be represented starting from the words they are composed of and facilitates the learning of semantic similarity.

Let $\mathcal{I}$ be the set of items, and let each item $i \in \mathcal{I}$ be associated with a textual description t_i modeled as a sequence of terms $w_1, w_2, \ldots, w_T$. For the purposes here, given t_i, the text encoder returns an embedding $\vec{i_t}$ based on SBERT and obtained by the process described above, such that

$$\vec{i_t} = SBERT(t_i) \tag{2.4}$$

Next, it is possible to learn user embeddings based on the textual information by exploiting the interaction matrix Y, which encodes the user preferences. Given the user set $\mathcal{U}$, user preferences can be modeled as a user-item matrix $\mathcal{Y} \in \mathbb{R}^{n \times m}$. If user u_j liked item i_k, then $y_{j,k} = 1$, 0 otherwise.

Given $u \in \mathcal{U}$, the set of items the user has liked is collected.

Formally: $L = \{i \in \mathcal{Y} | y_{u,i} = 1\}$. Then, user embedding $\vec{u_t}$ is obtained as the centroid of the text embeddings of the items in L:

$$\vec{u_t} = \frac{\sum_{i \in L} \vec{i_t}}{|L|} \tag{2.5}$$

Accordingly, text-based embeddings for all users and items, $\vec{u_t}$ and $\vec{i_t}$, in the graph are obtained.

Image Encoders Image embeddings provide a way to represent visual data in an embedding form, capturing essential features. As in the previous section, several studies were conducted using the same encoder model, which is described in this section.

Early works in image encoding relied on CNNs, such as VGGNet [160] or ResNet [68], which showed good performance on image classification tasks. The current state of the art is represented by ViT (Vision Transformers) [54], which borrowed the principles of Transformers for image processing. A recent work [200] showed that ViT outperformed popular image encoding strategies based on CNNs, including VGGNet [160] and ResNet [68].

ViT learns image representations by applying Transformer principles to vision tasks, and it is based on the idea that an image can be treated as a sequence of fixed-size patches (tokens).

Let $x_i \in \mathbb{R}^{C \times H \times W}$ denote the image associated with item $i \in \mathcal{I}$, where $H \times W$ is its dimension in pixels[1] and C the number of channels (*i.e.*, in the classical RGB format, $C = 3$).

The resulting image is $\tilde{x_i} \in \mathbb{R}^{N \times P^2 C}$, where (P, P) is the dimension (in pixels) of the fixed-sized patches applied to the images, C is the number of channels, and $N = HW/P^2$ is the number of patches applied to the image. Since ViT uses a transformer architecture, the patches are flattened and projected into a

[1]ViT requires that the original size of the images be a multiple of 16.

d-dimensional learnable linear space, and this representation is the input to the sequence. Then, a learnable [CLS] token is prepended, and this represents the embedding of the whole sequence. In addition, a learnable 1D position embedding is added, and the resulting sequence is the input of the transformer architecture, where each encoder block consists of a multi-head self-attention layer followed by a feed-forward MLP, with residual connections around both layers and Layer Normalization applied before each. Each patch is fed into a standard Transformer architecture, and the CLS embedding obtained at the last layer of the encoder is considered as the final image embedding.

Next, user embeddings based on the image data can be learned by exploiting the interaction matrix Y, in an analogous manner as in the previous section. Given a user $u \in \mathcal{U}$, the set of items liked by the user is collected.

Formally: $L = \{i \in \mathcal{Y} \mid y_{u,i} = 1\}$. Then, the user embedding $\vec{u}_{av}^{img}$ is obtained as the centroid of the image embeddings of the items in L:

$$\vec{u}_{av}^{img} = \frac{\sum_{i \in L} \vec{i}_{av}^{img}}{|L|} \tag{2.6}$$

Accordingly, image-based embeddings for all users and items $\vec{u}_t$ and $\vec{i}_t$ in the graph are learned.

As regards this aspect, it is important to point out that both *text-based* and *image-based* user embeddings are built upon the same set of items, *i.e.,* those the user liked. However, while the set of the items is the same, the resulting *image-based embeddings* will be different from *text-based embeddings*, since they start from different uni-modal embeddings. Accordingly, each representation of the user will encode different *modality-specific* preferences, and the same holds for audio and video embeddings that are described next.

Audio Encoders For audio encoding, the state of the art is represented by VGGish [72], a variant of VGG adapted to audio signals. VGGish has emerged as more efficient with respect to other comparable models such as Whisper [142], a sequence-to-sequence transformer-based model. This section provides a description of the VGGish encoding model, which is used in different works conducted during the PhD.

Suppose that each item $i \in \mathcal{I}$ is provided with an audio track. First, it is necessary to convert audio into images by computing the *log-mel spectrogram* of multiple frames, from which 2D image-like patches are created. Then, the patches are passed to the VGGish architecture, which follows the VGG19 [160] architecture with slight modifications.

Let $y_i \in \mathbb{R}^{C \times T \times F}$ be the log-mel spectrogram associated with item i, where T and F are the time frames and frequency bins, respectively, and C is the number of channels (typically $C = 1$ for mono audio). The input y_i is processed through a series of convolutional layers organized in blocks, where each block consists of one or more 3×3 convolutional layers, each followed by a ReLU activation, and a max-pooling layer at the end of the block that reduces the temporal and frequency resolution. The number of feature channels typically doubles after each block, allowing the network to learn hierarchical audio features. After the final convolutional block, the resulting feature maps are flattened along the spatial dimensions and passed through one or more fully connected layers, producing a fixed-length embedding vector $\vec{i}^{aud}_{av}$ that represents the item i. Dropout is often applied in the fully connected layers to improve generalization.

Formally, the audio-based embedding of item i, based on the audio signal y_i, is obtained as:

$$\vec{i}^{aud}_{av} = VGGish(y_i) \tag{2.7}$$

where $VGGish$ is the architecture described above.

As with text and image data, user embeddings from the audio data can be obtained as follows. Let $L = \{i \in \mathcal{Y} | y_{u,i} = 1\}$. Then, the user embedding $\vec{u}^{aud}_{av}$ is obtained as the centroid of the audio embeddings of the items in L:

$$\vec{u}^{av}_{aud} = \frac{\sum_{i \in L} \vec{i}^{aud}_{av}}{|L|} \tag{2.8}$$

Accordingly, this yields audio-based embeddings for all users and items $\vec{u}_t$ and $\vec{i}_t$ in the graph.

In this way, audio-specific features related to user preferences and item characteristics can be encoded, which would be overlooked in audio knowledge-source-agnostic scenarios.

Video Encoders The last A/V source considered in this thesis and in the works carried out during the PhD is video. To encode video data, such works consider

R(2+1)D [195].

In this research area, among the most relevant works are C3D [196], a 3D CNN that learns spatio-temporal features for the action recognition task, and I3D [29], which starts with image classification and then considers temporal information.

R(2+1)D [195] encodes spatial and temporal features in separate steps, through 2D *(spatial)* convolution followed by a 1D *(temporal)* convolution. As shown in the original paper [195], the method is designed to be approximately equivalent to the number of parameters of a full 3D convolution, as in methods such as C3D [196] and I3D [29].

The key intuition behind R(2+1)D is to replace full 3D convolutions (typically used in other approaches for video embeddings, such as I3D [29]) with a 2D convolution followed by a 1D convolution, decomposing spatial and temporal modeling into two separate steps. Accordingly, N_i 3D convolution filters of size $N_{i-1} \times t \times d \times d$ (where N_{i-1} is the number of convolutional filters applied to the $i-1$-th block, t is the number of frames, d is the dimension of the frame) can be replaced with 2D convolutional filters (with size $N_{i-1} \times 1 \times d \times d$) to capture the visual information, and 1D filters $M_i \times t \times 1 \times 1$ to capture the temporal information. In addition, M_i is designed to be approximately equivalent to the number of parameters of a full 3D convolution, in order to avoid information loss. Formally, given the item $i \in \mathcal{I}$ (see Section 2.3 for the formal definition of the $\mathcal{I}$ item set), the corresponding A/V embedding from the video signal z_i is obtained by adopting the above-mentioned architecture, and can be written as follows:

$$\vec{i}_{av}^{vid} = R2P1D(z_i) \tag{2.9}$$

where $R2P1D$ is the architecture described above.

User embeddings $\vec{u}_{av}^{vid}$ are obtained as discussed in the previous sections.

Formally, let $L = \{i \in \mathcal{Y} \mid y_{u,i} = 1\}$. Then, the user embedding $\vec{u}_{av}^{vid}$ is obtained as the centroid of the video embeddings of the items in L:

$$\vec{u}_{av}^{vid} = \frac{\sum_{i \in L} \vec{i}_{av}^{vid}}{|L|} \tag{2.10}$$

Accordingly, this yields video-based embeddings for all users and items $\vec{u}_t$ and $\vec{i}_t$ in the graph.

In this way, this approach encodes *video-specific* features related to user preferences and item characteristics that would be overlooked in scenarios agnostic of video knowledge-source.

2.2 State-of-the-art KARS Approaches

This section examines the ways in which KARS exploit and process different knowledge sources. The discussion focuses on the most widely adopted recommendation models, with particular attention to how structured and unstructured knowledge is incorporated into their architectures and exploited to improve recommendation performance. By analyzing these approaches, this section provides insight into the practical application of knowledge sources within the current KARS literature.

During the last decade, several works exploited KGs to improve KARS performance. Among these, CKE (Collaborative Knowledge Base Embedding) [246], MKR [212], and CFKG (Collaborative Filtering on Knowledge Graphs) [251] exploit simple KGE approaches to provide recommendations.

In particular, CKE [246] is a recommendation model that exploits a KG to enrich the Collaborative Filtering (CF) data with the side information of the items provided by a KG. CKE learns, on one side, item embeddings from the KG by using TransE [19], and, on the other side, it performs the CF task by applying the Bayesian Personalized Ranking loss [146]. The two tasks are jointly optimized to learn effective side-information item embeddings adapted for the recommendation task. MKR [212] exploits similar intuitions: it performs both recommendation and KG embedding tasks jointly by employing a *cross&compress unit* to learn better item embeddings, which are shared between the two tasks. CFKG [251] extends the classical CF task by incorporating side information encoded in a KG. This model also uses TransE [19] to represent items and entities in the same latent space, and CFKG finds paths in the KGs connecting users and items to provide explanations.

With the advent of Graph Neural Networks [233] (GNNs), several KARS started exploiting these technologies. Graph Convolutional Networks [249] (GCNs) and Graph Attention Networks (GATs) [207] are now widely used in the KARS field.

As already discussed, the core idea of GCNs is to learn effective node representations by exploiting the message passing process: nodes propagate their representations to their set of neighbors, allowing them to update their own representations accordingly. This process is repeated n times, meaning that each node can propagate its representation up to n-hops in the graph. In this context, n represents the number of layers in the GCN model.

On the other hand, GATs can learn, for each node, the most relevant nodes in its neighborhood by computing *attention scores*, which provide the relevance of a node with respect to all the others. These scores are then used to update the node embeddings.

KGCN (Knowledge Graph Convolutional Networks) [211] and KGAT (Knowledge Graph Attention Networks) [219] are KARS models that exploit GNNs. KGCN adopts GCNs to aggregate information across the KG, enabling it to capture higher-order neighbor relationships. In this way, KGCN exploits the graph structure to incorporate local and global information to improve recommendations. KGAT [219] exploits Graph Attention Networks [207] to model high-order connections between entities and items in a KG. In this way, item embeddings are refined, and attention mechanisms are used to discriminate the relevance of the neighbors. Then, both methods use such item embeddings obtained by exploiting side information encoded in KGs to perform the recommendation task.

There are also KARS that exploit KGs to better model user preferences, such as KTUP [27] (Knowledge-enhanced Translation-based User Preference), KGNNLS [210] (Knowledge-aware Graph Neural Networks with Label Smoothness), and KGIN (Knowledge Graph-based Intent Network) [220].

KTUP [27] assumes that the KG might be incomplete and uses TransH [223] to learn KG embeddings and perform KG completion. The embeddings are then transferred to the user modeling module, which performs the recommendation. In this way, both KG completion and recommendation tasks are jointly optimized, and the KG is used to improve the user embeddings as well. KGNNLS [210] uses KGs to learn user-specific item embeddings by identifying relevant relationships in the KG for that user. In this way, the KG is transformed into a user-specific graph, which is finally used to provide recommendations. KGIN [220] focuses on modeling user intents as a weighted combination of KG relationships. Moreover, KGIN uses a novel graph aggregation model that recursively integrates the relation sequences of long-range connectivity, leading to improved user modeling and better recommendations.

Recent KARS literature largely exploit unstructured knowledge as well. Unstructured knowledge sources typically refer to knowledge encoded in a format that does not adhere to a predefined schema or ontology and that lacks explicit semantic relationships between entities [6]. These sources include natural language text, images, audio, and videos. In contrast to structured knowledge bases, including KGs, which explicitly encode entities and their relationships and support reasoning tasks [36], unstructured sources provide knowledge in a form that needs to be encoded into a suitable format (typically embeddings) to be used in downstream tasks, including recommendation [104].

In addition, the knowledge encoded by unstructured sources might be complementary to that encoded by KGs. For example, in the movie domain, a KG might encode information about the genre or the cast of a movie. However, users might find it useful to rely on user reviews to decide whether to watch a movie. Users also typically read the movie synopsis, or watch the movie trailer to better grasp other aspects of the movie, such as the soundtrack, cinematography, and directing, that might influence their choice.

Indeed, several KARS consider this kind of information. The full body of Content-based RSs (CBRSs) [107] can fall in this category, as they typically exploit textual item descriptions to provide knowledge. Moreover, there are approaches that combine structured knowledge encoded in KGs with textual knowledge encoded in item descriptions [136, 167].

In CBRSs, user and item attributes, typically in textual form, can be exploited in several ways. In [159], the authors propose a research paper RS in which TF-IDF (Term Frequency Inverse Document Frequency) [10] is used to obtain user queries and item representations, which are then matched using cosine similarity to provide recommendations. In [41], textual user and item attributes are utilized to extend the original SVD algorithm. This work proposes a novel content-based RS that adapts the properties of SVD to a content-based context. To this end, the SVD is applied to the user–item attribute matrix instead of the rating matrix, showing promising results in cold-start and high-sparsity scenarios. In [44], KGs and word embeddings are exploited to extend content-based RSs. This approach helps overcome a common issue related to word ambiguity in item attribute representations, as such systems typically rely on basic keyword-matching strategies. Along this research line, the use of word embeddings [97] has also been explored. For instance, in [123], *word2vec* [118] is employed to obtain item attribute representations matched with user preferences, demonstrating better performance than collaborative filtering (CF) approaches in highly sparse recommendation scenarios.

However, CBRSs can exploit other knowledge sources beyond textual to provide recommendations as well, including visual or acoustic signals. In [47], the authors propose a content-based video recommender system that automatically extracts visual features from videos to improve recommendation accuracy. The approach analyzes stylistic aspects of video content, such as lighting, color, and motion, to generate representative feature embeddings. Evaluation against traditional content-based systems that rely on explicit features, such as movie genre, shows that the proposed method provides more accurate recommendations. The technique is effective when applied to both full-length videos and trailers and can be used independently or in combination with conventional content-based approaches. This enables recommendations for videos that lack metadata, addressing a common limitation of platforms where user-uploaded content may not include descriptive information. In [138], Siamese Neural Networks [40] are exploited to assess similarity between audio clips. Each clip is converted into a Mel-Spectrogram and processed by two identical CNNs, with their outputs compared to determine song similarity. Then, a query-by-multiple-example RS uses this similarity metric to suggest songs matching user preferences.

More recently, a new trend in the field has started exploiting multimedia information, typically encoded in images, audio and video knowledge sources.

Multimodal Recommendation [198, 104] refers to the research line in which different multimedia (or multimodal[2]) knowledge sources are used together to provide recommendations. Examples of multimodal recommendation models are VBPR [69], Lattice [247], Freedom [258], MMGCN [226], SLMRec [190] and MMGCL [240], and here they are discussed.

VBPR [69] represents the adaptation of the popular Bayesian Personalization Ranking (BPR) [146] model to visual features, in which the items are represented by the combination of the visual features associated with that item and its latent representation. In Lattice [247], the authors proposed a model based on visual and textual features adapted for a CF task. In particular, starting from the embeddings learnt from visual and textual features, an item-item graph is constructed by exploiting a similarity measure. Then, the embeddings are refined through a GCN, and the resulting embeddings are summed up with those coming from any

[2]Both these two terms will used throughout this thesis

other CF model. Finally, a BPR loss function is used to provide the recommendation score. Freedom [258] proposes to improve the efficiency of Lattice [247] by denoising the original interaction matrix, dropping noisy edges and freezing the original feature embeddings (instead of refining them). Then, graph convolution is performed on the item-item graph on one side, and on the (denoised) user-item graph on the other side. The item embeddings resulting from the two convolutions are summed to obtain a final representation of the items, while the user embedding resulting from the convolution on the user-item graph is used as the final user embedding. Finally, the BPR loss is used to perform the optimization, and the dot product is used to predict the recommendation score.

MMGCN [226] is a model that provides micro-video recommendations. It first considers each modality for micro-videos (visual, textual, acoustic), and builds a bipartite user-item graph for each modality, so as to capture user modality-specific preferences. Then, a GCN learns the embeddings from each modality, which are then combined through element-wise product or concatenation, resulting in a single embedding that is finally used to provide the recommendations. SLMRec (Self-supervised Learning for Multimedia Recommendation) [190] is a self-supervised graph learning model that uses the available modalities to generate supervised signals, which are then used as components of a contrastive loss, ultimately aligning the various modality spaces. MMGCL (Multi-Modal Graph Contrastive Learning) [241]: a self-supervised multi-task graph learning model that performs edge dropout and modality masking to better model user preferences, learning the correlation between different modalities. It uses both a self-supervised loss and a BPR loss.

However, these models focus only on multimedia feature, ignoring the structured knowledge encoded in KGs, which might be complementary and further improve users' and items' representations, leading to more accurate recommendations. On the other hand, most work focusing on KGs ignore multimedia data, showing the same mentioned drawback. In the next chapters of this thesis, this problem is tackled and a single architecture focusing on all these knowledge sources is proposed to fill this gap in the literature.

2.3 Formalization of the Problem

This section introduces the formal framework that will be used throughout the thesis and defines the recommendation problem, together with the datasets used in the experimental settings, the procedure to gather KGs and unstructured knowledge sources, and the models moslty used as baselines in the experiments.

2.3.1 Interaction data

Given a set of users $\mathcal{U} = \{u_1, u_2, \ldots, u_n\}$ and a set of items $\mathcal{I} = \{i_1, i_2, \ldots, i_m\}$, the interaction data (clicks, preferences, ratings, etc.) can be modeled as a user-item matrix $\mathcal{Y} \in \mathbb{R}^{n \times m}$. If user u_j liked item i_k, then $y_{j,k} = 1$, 0 otherwise. The matrix $\mathcal{Y}$ can also be modeled as a directed *interaction graph* $\mathcal{IG} = \langle \mathcal{V}, \mathcal{C} \rangle$, where $|\mathcal{V}| = |\mathcal{U}| + |\mathcal{I}|$ (users and items are encoded as *nodes*) and $\mathcal{C}$ is a set of directed edges connecting users and items. In particular, for all the 1s and 0s encoded in $\mathcal{Y}$, edges connecting users and items labeled with *'like'* and *'dislike'*, respectively, are created. If the user did not interact with an item, no edges are created.

2.3.2 Knowledge Graph

A knowledge graph $\mathcal{KG}$ encodes structured properties of the items. Formally, let $\mathcal{R}$ be a set of relations (*e.g., directed-by*, *genre*, etc.), and $\mathcal{P}$ a set of properties (*e.g., Scorsese*, *thriller*, etc.). A knowledge graph can be modeled as $\mathcal{KG} = \left\langle \mathcal{V}^{'}, \mathcal{C}^{'} \right\rangle$, where $|\mathcal{V}^{'}| = |\mathcal{I}| + |\mathcal{P}|$ (both items and properties are encoded as nodes). If an item $i \in \mathcal{I}$ is described by the property $p \in \mathcal{P}$ through a relation $r \in \mathcal{R}$, *e.g., (The Departed, genre, thriller)*, an edge connecting i and p labeled by r is created in $\mathcal{KG}$.

Next, the interaction graph $\mathcal{IG}$ and the knowledge graph $\mathcal{KG}$ can be merged into a unique tripartite graph $\mathcal{G}$ by exploiting the items $\mathcal{I}$ as overlapping nodes. The nodes of the resulting graph are users, items, and properties, while the edges encode both user preferences and item properties.

2.3.3 Textual data

In addition, each item $i \in \mathcal{I}$ is associated with textual information, such as a movie or book plot, or user reviews (one per item). Formally, let $\mathcal{T}$ be the knowledge source related to textual information, and let t_i denote the textual data corresponding to item i. Accordingly,

$$\mathcal{T} = \{t_1, t_2, \ldots, t_m\}$$

represents the set of information derived from textual data related to all the items.

2.3.4 Image data

Moreover, each item $i \in \mathcal{I}$ comes with visual data, in particular, images. As an example, in the movie domain, there is a poster for a movie, whereas in the book domain there is a book cover. Each item is associated with one image. Formally, let $\mathcal{IMG}$ be the knowledge source related to images, then let I_i be the image data of item i. Accordingly,

$$\mathcal{IMG} = \{img_1, img_2, \dots img_m\}$$

represents the set of the information from image data related to all the items.

2.3.5 Audio data

In a similar manner as before, each item $i \in \mathcal{I}$ is provided with audio data. In the movie domain this could be represented by the soundtrack of the movie, or by the soundtrack of the movie trailer. Each item is associated with one audio track. Formally, let $\mathcal{A}$ be the knowledge source related to audio signals, then let a_i be the audio data related to item i. Then,

$$\mathcal{A} = \{a_1, a_2, \dots a_m\}$$

represents the set of the information from audio data related to all the items.

2.3.6 Video data

Similar to the previous cases, each item $i \in \mathcal{I}$ is provided with video data, such as the movie trailer. Each item is associated with one video. Again, let $\mathcal{V}$ be the knowledge source related to video signals, then let v_i be the video data associated with item i. Accordingly,

$$\mathcal{V} = \{v_1, v_2, \dots, v_m\}$$

is the set of the video data related to all the items.

2.3.7 Problem Statement

In general, given the interaction graph $\mathcal{IG}$, the knowledge graph $\mathcal{KG}$, and the unstructured side information (text, images, audio, and video) $\mathcal{T}$, $\mathcal{IMG}, \mathcal{A}, \mathcal{V}$, let θ be the parameters of the recommendation model. Then, the rating prediction $y_{u,i}$ provided by the recommendation model can be formalized as follows:

$$\widetilde{Y}(u, i|\mathcal{IG}, \mathcal{KG}, \mathcal{T}, \mathcal{IMG}, \mathcal{A}, \mathcal{V}, \theta) = y_{u,i} \tag{2.11}$$

In all the experiments discussed in this thesis, the *top-k* recommendation setting is used to evaluate the recommendation model. This setting ranks the available items based on the scores predicted by $\widetilde{Y}$ and returns the top-k items.

2.3.8 Datasets

All the experiments carried out during the PhD and reported in this thesis have been executed on one or more of the following datasets[3]. Dataset descriptions are reported here to avoid repetitions in the subsequent chapters.

The considered datasets cover different domains, including movie, music, books, news. It is worth to point out that three of them (MovieLens-1M, Last.fm-2K, DBbook) have been extended with multimedia content during the PhD, as a contribution to the research community. This contribution has been accepted as a Reproducibility Paper and has been presented at the 19th ACM Conference on Recommender Systems [176]: Giuseppe Spillo et al. "See the Movie, Hear the Song, Read the Book: Extending MovieLens-1M, Last. fm-2K, and DBbook with Multimodal Data". In: *Proceedings of the Nineteenth ACM Conference on Recommender Systems*. 2025, pp. 847–856. **Rank A** (Core2023).

For the other considered datasets, Amazon Books and Mind, no additional processing has been performed. While the former is already provided with KG, but lack of multimedia data, the latter is only equipped with CF signals.

MovieLens-1M

MovieLens-1M[4] (*ML1M*) is one of the most commonly used datasets in the recommendation field [204]. Most research has used it in a CF setting, such as [25, 156, 152, 7, 78], or Content-based and Knowledge-Aware recommendation, such as [136, 27, 246, 213]. Only a few studies [226] have used this dataset for multimodal recommendation. However, the lack of publicly released features and data

[3]The only exception to this is represented by Section 5.1, that focuses on large-scale multimodal recommendation systems. For this reason, it is the only chapter that uses other datasets beyond those introduced in this section. Consequently, these additional datasets are presented, described, and discussed in detail within that section, including their sources, characteristics, and relevance to the experimental setup in Section 5.1.

[4]`https://grouplens.org/datasets/movielens/`

ML1M	
Users	6036
Items	3081
Ratings	946120
% pos. Ratings	57.26
Sparsity	0.9491
Rating/user	156.75
Rating/item	307.08
Items in KG	2735
KG entities	17718
KG relations	13
KG triples	70668
Avg. Prop/item	22.94
Textual info - coverage	Plot - 3081
Image info - coverage	Posters - 3079
Video info - coverage	Trailers - 2858
Audio info - coverage	Trailers - 2859

TABLE 2.1: Statistics of the ML1M dataset, including KG, textual and multimedia data.

processing details makes their results unreproducible and limits the dataset's usability in this context.

This dataset provides ratings from 6040 users for 3706 movies, with values ranging from 1 to 5. Moreover, it provides the titles of the movies and the associated genres. Statistics about the *core-5* version of the dataset are provided in Table 2.1. The feature publication [176] is the first to provide, for ML1M, direct links to download raw multimodal data related to movie posters (from which image features can be obtained) and movie trailers (from which both video and acoustic features can be obtained)[5]. Moreover, the plain text related to movie plots (from which textual features can be obtained) is released. In the following, the collection of multimodal data for this dataset is described below (statistics are reported in Table 2.1).

Mapping Data to KGs. For this dataset, DBpedia is considered as KG. To extract

[5]Raw data (images and trailers) is not release to avoid violating copyright, but direct links to download raw data are provided. This holds for the other datasets as well.

information from DBpedia and populate the KG, a mapping already available online[6] has been exploited. Basically, the mapping is carried out by launching a SPARQL query based on some descriptive properties (e.g., the name of the movie or name of the book). After the mapping, a huge set of new entities and new relations is encoded in the graph. Statistics of the KG are reported in Table 2.1.

Textual data. In order to obtain textual data, the *DBpedia* knowledge base [8] was used: following previous works [136, 167], publicly available mappings[7] were exploited to obtain the movie plots of the movies in ML1M, which resulted in high item coverage, as depicted in Table 2.1.

Image data. To collect movie posters, an approach similar to the one used for textual data was employed. Since DBpedia is based on Wikipedia, and Wikipedia pages related to movies report the associated movie posters, the movie posters for many movies in the ML1M catalog could be crawled automatically. Not all Wikipedia pages reported the movie poster, so additional posters were collected manually from IMDb[8]. The resulting coverage is reported in Table 2.1.

Video and Audio data. To collect video and audio data, the trailers related to the movies in the ML1M catalog were used, as trailers provide both video and audio information. To collect movie trailers, a mapping provided by GroupLens in the MovieLens-20M dataset was relied upon, which provides the YouTube ID of the movie trailer, enabling retrieval of the full YouTube link. However, many links have changed since the release of such mapping. Accordingly, the missing links were checked manually and replaced with working ones. Trailers for most of the movies were gathered (see Table 2.1).

From a formal perspective, the MovieLens-1M dataset can be modeled as a tuple $\mathcal{D}_{\text{ML1M}} = (U, I, R, G, \mathcal{M})$. $U = \{u_1, \ldots, u_{|U|}\}$ denotes the finite set of users and $I = \{i_1, \ldots, i_{|I|}\}$ the finite set of items (movies). The interaction data are represented as a sparse rating relation $R \subseteq U \times I \times \mathbb{R}$, where each tuple (u, i, r) indicates that user u rated item i with score $r \in \{1, \ldots, 5\}$; equivalently, R can be seen as a sparse matrix $\mathbf{R} \in \mathbb{R}^{|U| \times |I|}$. The knowledge graph is formalized as a directed labeled graph $KG = (E, \mathcal{R}, T)$, where E is the set of entities, $\mathcal{R}$ the set of relation types, and $T \subseteq E \times \mathcal{R} \times E$ the set of triples; a (partial) mapping $\phi : I \to E$ associates items to KG entities. Finally, the multimodal information is modeled

[6]https://github.com/sisinflab/LODrecsys-datasets
[7]https://github.com/sisinflab/LODrecsys-datasets/tree/master
[8]https://www.imdb.com/

DBbook	
Users	5660
Items	6617
Ratings	129316
% pos. Ratings	52.66
Sparsity	0.9965
Rating/user	22.85
Rating/item	19.54
Items in KG	1963
KG entities	5207
KG relations	39
KG triples	16589
Avg. Prop/item	2.51
Textual info - coverage	Abstract - 6617
Image info - coverage	Covers - 6617

TABLE 2.2: Statistics of the DBbook dataset, including KG, textual and image data.

as a collection of modality-specific raw data associated to text, image, audio, and video: $\mathcal{M} = \{\mathcal{T}, \mathcal{IMG}, \mathcal{A}, \mathcal{V}\}$.

DBbook

DBbook is a recommendation dataset in the book domain, providing ratings for 6181 users on 7672 items in a binarized format (1 or 0). In addition, a mapping linking each book to the related *DBpedia* entity is provided. More detailed statistics about the *core-5* version of the dataset are reported in Table 2.2. It was first proposed for the ESWC-14 Challenge[9], but the resource is now unreachable. One possible workaround is to use Web Archive[10]. However, this solution only partially fixes the problem, as the dataset was provided already split into train and test data, and only the train data is available on Web Archive. To solve this problem, the dataset was reconstructed by exploiting the version used in a previous work and is available on the GitHub repository[11].

[9]Non-working link: `http://challenges.2014.eswc-conferences.org/index.php/RecSys`

[10]`https://web.archive.org/web/20180402195813/http://challenges.2014.eswc-conferences.org/index.php/RecSys`

[11]`https://github.com/swapUniba/Deep_CBRS_Amar/tree/master/datasets/dbbook`

DBbook has mostly been used in CB and KA recommendations, as in [135, 3, 95, 134, 186], and has never been used in multimodal recommendation, with the exception of [177], a work carried out during this PhD. Accordingly, the collection of multimodal data (textual and images) is discussed in the following.

Mapping Data to KGs. For this dataset, DBpedia is considered as KG. To extract information from DBpedia and populate the KG, a mapping already available online[12] has been exploited. Basically, the mapping is carried out by launching a SPARQL query based on some descriptive properties (e.g., the name of the movie or name of the book). After the mapping, a huge set of new entities and new relations is encoded in the graph. Statistics of the KG are reported in Table 2.2.

Textual data. To collect textual data, the *DBpedia* mapping released in the original version of the dataset was used. The abstracts of 4197 books were collected, out of the 7672 total books available in the catalog.

Image data. For image data related to the book domain, book covers were considered, as they affect users' choices [61]. To retrieve such covers, it was observed that most of the books in DBpedia are associated with a Wikipedia webpage, which provides the related book cover. Accordingly, each DBpedia link was converted into a Wikipedia link, then the webpage was scraped to retrieve the book cover. Covers were retrieved for 7588 out of 7672 total books in the catalog.

The DBbook dataset can be formalized as $\mathcal{D}_{\text{DBbook}} = (U, I, R, G, \mathcal{M})$. U and I denote the finite sets of users and items (books), respectively. User–item interactions are represented by a sparse binary relation $R \subseteq U \times I$, where each observed pair (u, i) indicates positive feedback; equivalently, R can be represented as a sparse binary matrix $\mathbf{R} \in \{0, 1\}^{|U| \times |I|}$. The knowledge graph is defined as a directed labeled graph $KG = (E, \mathcal{R}, T)$, with entities and relations extracted from DBpedia and linked to items through a partial mapping $\phi : I \to E$. Finally, the multimodal information is modeled as a collection of modality-specific raw data associated to text and image: $\mathcal{M} = \{\mathcal{T}, \mathcal{IMG}$.

Last.fm-2K

Last.fm-2K[13] (*LFM-2K*) is a popular recommendation dataset in the music domain, used in CF algorithms [26, 222, 139, 214, 17, 245], but also in CB or KA recommendation tasks [227, 253].

[12] `https://github.com/sisinflab/LODrecsys-datasets`
[13] `https://grouplens.org/datasets/hetrec-2011/`

Last.fm-2K	
Users	1881
Items	2828
Ratings	71426
Sparsity	98.66%
% pos.	53.64%
% neg.	46.36%
Rat./user	37.97
Rat./item	25.26
Items in KG	1742
KG entities	9366
KG relations	60
KG triples	15518
Avg. Prop/item	8.91
Textual info - coverage	2813
Image info - coverage	2820
Audio info - coverage	2742

TABLE 2.3: Statistics of the Last.fm dataset, including KG and multimedia data (not available).

This dataset contains information on music artists listened to by 1,892 users and the tags assigned by those users. Table 2.3 provides more detailed statistics about this dataset (*core-5* version). However, this dataset has never been used for multimodal recommendation, mostly due to the absence of available mappings to multimodal data. Accordingly, in the following, the gathering of multimodal data, particularly textual, image, and audio information, is discussed.

Mapping Data to KGs. Also for this dataset DBpedia is considered as KG. To extract information from DBpedia and populate the KG, a mapping already available online[14] has been exploited. Basically, the mapping is carried out by launching a SPARQL query based on some descriptive properties (e.g., the name of the movie or name of the book). After the mapping, a huge set of new entities and new relations is encoded in the graph. Statistics of the KG are reported in Table 2.3.

Textual data. For textual data, the user tags provided with the dataset itself were considered, making it unnecessary to collect additional textual features. Future strategies involving knowledge bases, such as *DBpedia*, will be considered as future work. However, not all artists were assigned tags, resulting in coverage for

[14]`https://github.com/sisinflab/LODrecsys-datasets`

2813 out of the 2823 artists in the *core-5* version of the dataset.

Image data. Since in this dataset the *items* are not songs or albums, but *artists*, image source data were considered as the most popular artist album covers: in particular, the album covers of the *top-5* most popular albums for each artist (when available) were manually collected, as they can reasonably be considered representative of the artist. As a result, 2820 out of the 2823 artists remaining after the core-5 filtering were covered, with an average of 4.9 album covers collected per artist.

Audio data. Similarly to the image data, the top-5 most popular songs for each artist (when available) were manually collected as representative of the artist. As a result, 2742 out of the 2823 artists remaining after the core-5 filtering were covered, with an average of 4.9 songs collected per artist.

From a formal perspective, the Last.fm-2K dataset can be represented as $\mathcal{D}_{\text{LFM}} = (U, I, R, G, \mathcal{M})$. U is the set of users and I the set of items, corresponding to music artists. The interaction data are modeled as a sparse implicit-feedback relation $R \subseteq U \times I$, capturing user listening events and tag assignments, and can be equivalently represented as a sparse binary matrix. The knowledge graph component is formalized as a directed labeled graph $KG = (E, \mathcal{R}, T)$, populated from DBpedia and linked to items through a partial mapping $\phi : I \to E$. Multimodal information is represented by $\mathcal{M} = \{\mathcal{T}, \mathcal{IMG}, \mathcal{A}\}$, where each function maps artists to textual descriptors, visual (album covers), and acoustic (audio signals) data.

Amazon Books

Amazon Books is a well-know datasets used for the task of book recommendations, used in several works [126, 100, 112, 35, 119]. For the experiments carried out in this thesis, no manual work has been performed, as the dataset was provided within the RecBole[15] framework, already equipped with the KG. Statistics of the dataset are reported in Table 2.4. For this dataset, no multimedia content is available.

The Amazon Books dataset is modeled as $\mathcal{D}_{\text{Amazon}} = (U, I, R, G)$. U and I denote the finite sets of users and items (books), respectively. User–item interactions are represented as a sparse implicit-feedback relation $R \subseteq U \times I$, which can be equivalently expressed as a sparse binary matrix $\mathbf{R} \in \{0, 1\}^{|U| \times |I|}$. The dataset includes

[15]The following repository allows to acces a GDrive folder with the dataset: `https://github.com/RUCAIBox/RecSysDatasets/`

Amazon Books	Value
Users	22,155
Items	54,458
Interactions/Ratings	1,465,871
Sparsity	99.88%
Interactions/user	66.16
Interactions/item	26.92
Items in KG	10,878
KG entities	26,316
KG relations	18
KG triples	96,476
Avg. Prop/item	8.87

TABLE 2.4: Statistics of the Amazon Books dataset, including KG (multimedia data is not available).

Mind	Value
Users	23,679
Items	4,414
Interactions/Ratings	1,048,575
Sparsity	98.99%
Interactions/user	44.28
Interactions/item	237.56

TABLE 2.5: Statistics of the Mind dataset, for which KG and multimedia data are not available.

a knowledge graph component $KG = (E, \mathcal{R}, T)$, where items are linked to entities via a partial mapping $\phi : I \rightarrow E$.

Mind

Amazon Books is a well-know datasets used for the task of news recommendations, used in several works [209, 92, 1, 229]. For the experiments carried out in this thesis, no manual work has been performed, as the dataset was provided within the RecBole framework[16]. Statistics of the dataset are reported in Table 2.5. For this dataset, no KG nor multimedia content is available.

[16] https://github.com/RUCAIBox/RecSysDatasets/

The Mind dataset can be formalized as $\mathcal{D}_{\text{MIND}} = (U, I, R)$. U denotes the set of users and I the set of items (news articles). The interaction data are modeled as a sparse implicit-feedback relation $R \subseteq U \times I$, capturing user click behaviors and representable as a sparse binary matrix.

2.3.9 Baselines

All the baseline models considered throughout the experiment presented in this thesis are discussed in this section. Baseline models descriptions are provided here to avoid redundancy in the subsequent chapters and to ensure a consistent reference framework for the comparative analyses that follow. The baselines described below serve as fundamental points of comparison for assessing the performance and relevance of the proposed models and methodologies. Some of these baselines have already been discussed in detail in previous chapters and are therefore reported here in a more concise form. For the baseline implementations, the frameworks Elliot [4], RecBole [254] and MMRec [257] have been considered. Details about data splits, random seeds and hyperparameter tuning are reported in each section in which baseline modals are considered.

- CF linear recommendation models:
  - **ItemKNN/UserKNN** [50]: recommendation models that exploit user (item) neighbors and similarity measures to provide recommendations.
  - **LINE** (Large-scale Information Network Embedding) [189]: a recommendation model suitable for large datasets, which efficiently learns embeddings to provide users with recommendations.
  - **SVD** (Singular Value Decomposition) [193]: a linear matrix factorization model that decomposes the user–item interaction matrix into low-rank latent factors for users and items, in order to predict missing ratings by approximating the original matrix through these latent embeddings.
  - **SLIM** (Sparse Linear Methods) [127]: a linear, item-based CF model that learns a sparse aggregation coefficient matrix to predict a user's preference for an item based on their interactions with other items.
  - **EASE** (Embarrassingly Shallow Autoencoders for Sparse Data with Ridge regression) [183]: a linear, item-based collaborative filtering model that predicts user–item interactions by learning a full item–item weight matrix via regularized linear regression.

- CF deep learning recommendation models:
  - **DMF** (Deep Matrix Factorization) [235]: a model that models learns nonlinear latent representations of users and items through deep neural networks, extending traditional matrix factorization beyond linear embeddings.
  - **NeuMF** (Neural Matrix Factorization) [71] combines generalized matrix factorization and multi-layer perceptron networks to capture both linear and nonlinear user–item interaction patterns, achieving more flexible and expressive recommendation modeling.
  - **MultiDAE**(Variational Autoencoders for CF) [99]: a model that adapts the use of variational autoencoders to the recommendation in a CF scenario.
  - **BPR** (Bayesian Personalized Ranking) [146]: a recommendation model based on Bayesian statistics adapted to perform ranking recommendation tasks.
- CF graph-based models:
  - **CFGAN** (A Generic Collaborative Filtering Framework based on Generative Adversarial Networks) [32]: a generative adversarial network–based collaborative filtering model that learns to generate realistic user–item interactions by training a generator to predict users' implicit preferences and a discriminator to distinguish real from generated interactions, thereby improving recommendation accuracy.
  - **NGCF** (Neural Graph Collaborative Filtering) [221]: a model that propagates embeddings through the graph to capture high-order collaborative signals via neural message passing, enhancing recommendation performance.
  - **DGCF** (Disentangled Graph Collaborative Filtering) [218]: a model that extends NGCF by disentangling user–item representations into multiple latent factors that capture different user preferences or interaction intents, providing more interpretable and fine-grained recommendations.
  - **LightGCN** [70]: a GCN that learns user and item embeddings from the interaction graph by performing node aggregation through the mean operator, resulting in an efficient yet effective recommendation model.

- **SGL** (Self-supervised Graph Learning) [230]: is a self-supervised learning framework designed to enhance graph-based collaborative filtering by introducing auxiliary tasks that improve the robustness and accuracy of recommender systems.
- **Spectral CF** [255]: a model that performs spectral convolution on the interaction graph to capture high-order connectivity patterns, enhancing recommendation by smoothing and propagating signals across similar users and items.

• Content-based and Knowledge-aware recommendation models:

- **VSM** (Vector Space Model) [151]: a model that represents users and items as vectors in a shared vector space based on content features.
- **AttributeItemKNN/AttributeUserKNN** [50]: extension of the original UserKNN/ItemKNN, with the neighbors computed based on user/item attributes.
- **KaHFM** (Knowledge-aware Hybrid Factorization Model) [5]: a knowledge-enhanced recommendation model that integrates structured semantic information from knowledge graphs with traditional collaborative filtering by jointly learning user and item embeddings through both interaction data and knowledge-aware feature propagation, enabling the system to capture richer relationships, improve explainability, and enhance recommendation performance, especially for cold-start or sparse scenarios.
- **CKE** (Collaboratice Knowledge base Emeddings) [246]: a recommendation model that learns heterogeneous embeddings from different knowledge bases and uses them to provide recommendations.
- **CFKG** (Collaborative Filtering over Knowledge Graphs for explainable recommendation) [251]: a recommendation model that exploits KGs to learn item embeddings that encode external knowledge, provide recommendations and provide explanations.
- **KGCN** (Knowledge Graph Convolutional Networks) [211]: a recommendation model that exploits GCNs to learn item similarity starting from a KG and the attributes it encodes, learning item embeddings by combining the relevant attributes for a given node. These embeddings are then used to provide recommendations.

- **KTUP** [27]: a model that exploits TransH [223] to complete a KG and better model user preferences, thus providing more precise recommendations.
- **KGNN-LS** (Knowledge-aware Graph Neural Networks with Label Smoothness Regularization) [210]: a recommendation model that exploits knowledge graph relations to build several user-specific weighted graphs and then learns user-specific item embeddings through GCNs with label smoothness regularization.
- **MKR** [212]: a model that jointly learns user–item interactions and KG embeddings through *cross&compress* units, enabling shared latent feature learning to capture high-order interactions and improve recommendation accuracy, especially under sparse data conditions.
- **KGIN** [220]: a model that captures user intents behind interactions by integrating knowledge graph relations through attentive intent modeling and relational path-aware aggregation, producing intent-informed embeddings for more accurate and interpretable recommendations.
- **KGAT** (Knowledge Graph Attention Networks) [219]: a model that uses GATs on a KG to propagate and weigh relational signals from entities connected to items, producing knowledge-aware embeddings that integrate structural graph information with user–item interactions for more accurate and explainable recommendations.
- **RippleNet** [213]: a recommendation model that uses knowledge graphs as a source of external knowledge. It propagates user preferences over items through the knowledge graph entities to extend the set of potential interests of users.

• Multimodal recommendation models[17]:

- **VBPR** [69]: one of the first multimodal recommendation models. It extends BPR with multimodal features by concatenating the ID item embedding with multimodal features.
- **Lattice** [247]: a model that builds an item-item similarity matrix based on multimodal features. Then, a GCN is used for refining the resulting embeddings and providing recommendations.

[17] Although these models can be included in the previous list, they are reported separately since they all ignore KGs.

- **Freedom** [258]: a model that improves the efficiency of Lattice by denoising the item-item similarity matrix and freezing the multimodal features.
- **MMGCN** [226]: a model that applies separate GCNs to each modality-specific interaction graph, to capture modal-specific user preference, and provide better recommendations.
- **SLMRec** (Self-supervised Learning for Multimedia Recommendation) [190]: a self-supervised graph learning model that uses the available modalities to generate supervised signals, which are then used as components of a contrastive loss, ultimately aligning the various modality spaces.
- **MMGCL** (Multi-Modal Graph Contrastive Learning) [241]: a self-supervised multi-task graph learning model that performs edge dropout and modality masking to better model user preferences, learning the correlation between different modalities. It uses both a self-supervised loss and a BPR loss.
- **LightGCNM**[18]: a model that extends LightGCN by integrating multimodal features.
- **BM3** [259]: a multimodal recommendation model that captures complex interactions between user preferences and multiple item modalities by using a bilinear mixture-of-experts mechanism to weight and combine the complementary signals from each modality.
- **GRCN** (Graph-refined Convolutional Network) [225]: a graph-based multimodal recommendation model that constructs a user–item interaction graph augmented with multimodal item features and employs graph convolutional networks to propagate and integrate these features across the graph, learning rich, high-order embeddings that capture both collaborative and content-based information for more precise recommendations.

[18]This model is available in one of the frameworks used in the experiment, but has never been proposed in the literature.

Chapter 3

Enriching Homogeneous Embeddings for Knowledge-aware Recommendations

This chapter focuses on RSs based on homogeneous item embeddings, exploring how such knowledge sources influence both accuracy and beyond-accuracy performance, while proposing novel approaches to enrich the item embeddings.

The first discussed work (Section 3.1) focuses on KG embedding enriched with First-Order Logic rules are inferred and injected during the embedding process, an approach that was never explored before in the literature. This work falls within Neuro-Symbolic AI research field, scarcely investigated in the RS literature.

Then, another approach to enrich KG data, GAL-KARS, is presented in Section 3.2. This work is motivated by the fact that KGs are often incomplete and do not encode all the relevant information. To this aim, GAL-KARS further enhances KG embeddings by exploiting LLMs (and their background knowledge obtained during the pre-training phase) to infer additional triples describing user preferences and item features, with the resulting embeddings used for recommendations.

A similar approach is then described in Section 3.3, which solely relies on LLM-generated text. LLMs are used to generate textual user profiles, which are then encoded alongside item descriptions to produce embeddings for recommendations. This work is motivated by the simple approaches used in the literature to represent user profiles.

The chapter finally addresses multimedia knowledge sources (Section 3.4), discussing a user study based on one of the extended datasets, ML1M, that has been described in the previous section. It systematically investigates the benefits and limitations of relying on single knowledge sources, considering both standard quantitative metrics and broader qualitative, beyond-accuracy factors, filling a gap in the literature where in-vivo experiments are rarely conducted.

3.1 Knowledge Graph Embeddings with First-order Logic Rules

3.1.1 Introduction

In the featured publications [173] and [174], a novel neuro-symbolic knowledge-aware recommendation model is proposed. It combines Knowledge Graph Embedding (KGE) techniques with First-Order Logic (FOL) rules to enrich item and user representations. An approach is presented that includes a Rule Miner to extract FOL rules from KGs, a Graph Embedding Learner that integrates both explicit KG information and symbolic rules into embeddings, and a Neural Recommender System that uses these enriched embeddings to generate recommendations. The methodology bridges symbolic reasoning and neural learning, aiming to improve recommendation quality by incorporating background knowledge into the embedding process. Experimental results on three benchmark datasets demonstrate that the model not only enhances prediction accuracy but also improves novelty and diversity in recommendations, outperforming several baselines and confirming the benefits of a neuro-symbolic strategy in the RS domain, a topic that is poorly investigated in the current RS literature.

The results discussed in this section have been published at the ACM RecSys 2022 Conference, in the LBR Track (Giuseppe Spillo et al. "Knowledge-aware Recommendations Based on Neuro-Symbolic Graph Embeddings and First-Order Logical Rules". In: *Proceedings of the 16th ACM Conference on Recommender Systems*. 2022, pp. 616–621) and presented as a poster, and subsequently extended and published in the Journal of User Modeling and User-Adapted Interaction (Giuseppe Spillo et al. "Recommender systems based on neuro-symbolic knowledge graph embeddings encoding first-order logic rules". In: *User Modeling and User-Adapted Interaction* 34.5 (2024), pp. 2039–2083) during the PhD.

3.1.2 Methodology

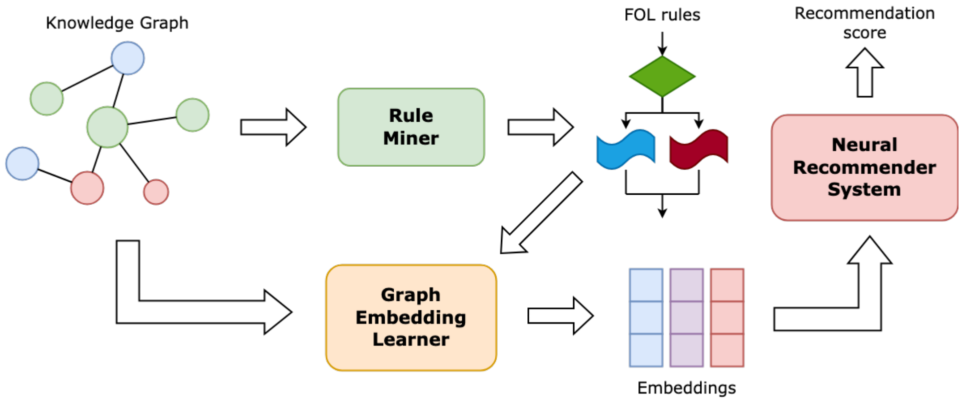


FIGURE 3.1: Workflow carried out by our model.

This section introduces the modules that compose the model for *knowledge-aware recommendations* based on *neuro-symbolic graph embeddings* exploiting FOL rules. Following the formalization of the KG provided in Section 2.3, in this section, the mining of FOL rules from the KG is first discussed; then, how the KGE jointly learns the embeddings by encoding both graph-based and rule-based knowledge is described; finally, how these embeddings are used to feed the neural recommender systems is described. The general workflow of the model is presented in Figure 3.1.

In this section, extensive use is made of terminology borrowed from *first-order languages* and *first-order logic* ([161]). FOL rules exploited in this approach are formally defined as Horn clauses[1], since they are composed of several atoms connected by logical connectives (*e.g.*, $\vee, \wedge, \Rightarrow, \ldots$), and only one atom appears in the *head* of the clauses (the other atoms are in the *body* of the clause). The *arity* of a predicate represents the number of variables it takes as arguments: for example, a predicate taking one argument is said to be *unary*, while a predicate taking two arguments is said to be *binary*. Each atom consists of *variables* (*e.g.*, $x, y, \ldots$) and *predicates* (equivalent to the *relations* previously introduced). Accordingly, the rules exploited in this model are first-order formulas in the form $\forall x, y : (x, r_s, y) \Rightarrow (x, r_t, y)$, meaning that if x and $y \in \mathcal{E}$ are linked by relation

[1]https://en.wikipedia.org/wiki/Horn_clause

$r_s \in \mathcal{R}$, then they are linked by relation r_t too. As an example, a simple FOL rule could be the following: $\forall x, y, z : likes(x, z) \wedge prequel(z, y) \Rightarrow likes(x, y)$. This FOL rule expresses the fact that *'If a user likes a movie and that movie has a prequel, the user will like the prequel too'*. Finally, a rule is said to be *grounded* if every variable has been replaced by a suitable entity $e \in \mathcal{E}$. The *grounded rule* based on the previous logical rule can be:

$$likes(user,\ StarWarsII) \wedge prequel(StarWarsII,\ StarWarsIII) \Rightarrow \\ likes(user,\ StarWarsIII)$$

In this case, the variables y and z have been replaced with the items *StarWarsI* and *StarWarsII*, respectively. Of course, given a FOL rule, it is very likely that the same FOL rule has several grounded rules (or *groundings*); for example, other groundings for the FOL rule presented above might be

$$likes(user, LOTR\_TheTwoTowers) \wedge \\ prequel(LOTR\_TheTwoTowers, LOTR\_TheReturnOfTheKing) \Rightarrow \\ likes(user, LOTR\_TheReturnOfTheKing)$$

or

$$likes(user,\ SpiderMan) \wedge prequel(SpiderMan,\ SpiderMan2) \Rightarrow \\ likes(user,\ SpiderMan2)$$

The concept of *grounded rule* (and *groundings*) is crucial for describing how embeddings are jointly learned from the KG and FOL rules.

Rule Miner Given a knowledge graph $\mathcal{KG}$ (see Figure 2.1), the first task in the mining process is to extract FOL rules that exist in $\mathcal{KG}$; as discussed above, the mined rules are expressed as Horn clauses, i.e., they are clauses in the form $\forall x, y, z : predicate1(x, y) \wedge predicate2(y, z) \Rightarrow predicate3(x, z)$, with at most one atom in the head of the rule.

As shown in [38] and in [158], methodologies for mining logical rules were initially proposed for the task of *knowledge graph completion*, which aims at finding and discovering new and missing triples; subsequently, they have been adopted for other tasks. For example, [84] employs logical rules for ontology alignment and fact prediction. In the featured publications [173] and [174], *background knowledge* encoded in the KG related to patterns that can be useful for improving the embedding learning process was mined. More specifically, the focus of the approach is not to extract *association rules* ([87]) that connect two entities (e.g., users who liked Star Wars II and also liked Star Wars III), but to identify *background knowledge* that characterize several entities and patterns, including users' nodes.

To mine FOL rules, a strategy based on AMIE+ and AMIE 3, proposed in [56, 55, 89], which are specifically designed to mine Horn Clauses on large knowledge bases, was employed in the featured publications [173] and [174]. Before describing the operation of the rules miner, several metrics crucial to the mining phase are presented.

- *Length*: the length of the rule, expressed as the number of atoms composing the rule.
- *Standard Confidence* (or *Confidence*): expresses how often the head of a rule is true given its body in the dataset.
- *Head Coverage*: expresses the number of occurrences of the head of the rule in the data.
- *PCA-Confidence*: expresses how often the head and body of the rule occur together and independently. In other words, it is the sum of the occurrences in which both the head and the body occur, and the occurrences in which only the head occurs.
- *Positive example*: expresses the absolute number of instances for which the FOL rule holds in the dataset (i.e., the occurrences of the rule).

The pseudo-code reported in Algorithm 1, taken from [55], describes the strategy used to mine FOL rules from the KG. It takes as input a knowledge base $\mathcal{KG}$ and the following parameters: the maximum length of rules to be mined *l*; the minimum head coverage *minHC* of the rules; and the minimum confidence *min-conf* of the rules to be mined.

The rule miner uses the *breadth-first search*; in particular, it starts with a queue consisting of all possible rules with an empty body (*e.g.,* rules of size 1, with one atom in the head and an empty body, such as $\emptyset \Rightarrow likes(x, y)$) (line 1).

Next, each rule is dequeued, and its *confidence* and *PCA-confidence* are computed. If the rule is *closed* (which means that its variables appear in at least two atoms), its confidence is greater than or equal to *min-conf*, and its PCA-confidence is greater

Algorithm 1 Rule Mining Algorithm

```
1: $q \leftarrow [\emptyset \Rightarrow r_1(x,y), \emptyset \Rightarrow r_2(x,y), \dots, \emptyset \Rightarrow r_k(x,y)]$
2: $rules \leftarrow \emptyset$
3: while $|q| > 0$ do
4:     $r \leftarrow q$.dequeue()
5:     if closed($r$) $\wedge$ PCAconf($r$) $> minConf$ $\wedge$ betterConf($r, rules$) then
6:         $rules$.add($r$)
7:     end if
8:     if $|r| < l \wedge$ PCAconf($r$) $< 1$ then
9:         for all $R \in$ refine($r$) do
10:            if headCov($R$) $\geq minHeadCov \wedge R \notin q$ then
11:                $q$.enqueue($R$)
12:            end if
13:        end for
14:    end if
15: end while
16: return $rules$
```

than or equal to the PCA-confidence of all previously mined rules with the same head (*line 5*), then the rule is added to the set of the rules to be returned (*line 6*); if the rule is shorter than *l* and its confidence can still be improved (*line 8*), then the rule is refined by adding all possible atoms to the body, and this new rule is added to the queue of the rules to be considered for the mining (*lines 9-11*).

To speed up rule mining, based on [55], several strategies have been applied, including pruning strategies, parallelization, and lazy computation of confidence scores. In this way, the system is able to scale effortlessly to large knowledge bases, while it is still able to compute, for each rule, the exact confidence and support values without approximations.

The output for this step is a *set of logical rules* (*line 16*) with related scores indicating the degree to which the rules hold: these scores were used to partition the rules into several subsets and to identify the most promising ones. An example of the output is provided in Table 3.1, which contains some real FOL rules mined from a movie recommendation dataset. Along with the FOL rules, some of the metrics already mentioned are shown (in particular, the head coverage, the standard confidence, and the number of positive examples).

FOL Rules Selection Next, based on these metrics, selection heuristics were defined to partition them into subsets and inject them into the graph embedding learning process. Given the absence of a proper literature that has defined criteria for identifying relevant rules, the subsets were designed by considering different heuristics. In particular, the heuristics defined were based on both the metrics typically used to assess the validity of FOL rules (*i.e.,* confidence, coverage, positive examples matching the rule) as well as on some background knowledge about the recommendation tasks (*i.e.,* rules having a *like* predicate in the head).

Based on these initial selection heuristics, eight different subsets were defined and evaluated in this work. In particular:

1. ALL RULES (ALL), including all the FOL rules that have been mined by the rule miner;
2. LOW STANDARD CONFIDENCE (LSC), including rules with a standard confidence value greater than 0.2;
3. MEDIUM STANDARD CONFIDENCE (MSC), including rules with a standard confidence value greater than 0.5;
4. MEDIUM-HIGH STANDARD CONFIDENCE (MHSC), including rules with a standard confidence value greater than 0.75;
5. HIGH STANDARD CONFIDENCE (HSC), including rules with a standard confidence value greater than 0.9;
6. LOW HEAD COVERAGE (LHC), including rules with a head coverage value greater than 0.2;
7. HIGH POSITIVE EXAMPLES (HPE), including rules with a high number of positive examples. Since different datasets may have rules with very different values related to this metric, a specific threshold value was identified for each dataset: (i) for Last.FM, 100 is used as threshold; (ii) for DBbook, 250; (iii) for ML1M, 1000.
8. LIKE IN THE HEAD OF THE RULE (LH), including rules that have, in their head, an atom including the relation *like*. This is considered a reasonable choice since the task concerns recommendations, and the *like* relation is crucial.

Although these are general heuristics, the choices can be justified. In particular, rule-based configuration (1) is useful to assess whether all possible FOL rules mined from the graph are useful for improving the model's accuracy; rule-based configurations (2-5) are useful to assess whether a *good-for-all* standard confidence value can be chosen to improve the embeddings; moreover, FOL rules with a high

Rule	**Head Cov.**	**Std Conf.**	**Pos. Ex.**
ML1M			
$producer(a,b) \Rightarrow director(a,b)$	0.06	0.00	12
$writer(m,b) \land producer(a,h) \land basedOn(m,h) \Rightarrow writer(a,b)$	0.01	0.63	5
$like(a,n) \land producer(n,h) \land subject(b,h) \Rightarrow like(a,b)$	0.01	0.33	5255
$producer(a,h) \land language(m,b) \land musicComposer(m,h) \Rightarrow language(a,b)$	0.35	0.34	1306
DBBOOK			
$genre(a,b) \land precededBy(h,b) \Rightarrow genre(h,b)$	0.17	0.62	310
$author(a,b) \land precededBy(h,b) \Rightarrow author(h,b)$	0.19	0.88	328
$like(a,b) \land precededBy(h,b) \Rightarrow like(a,h)$	0.27	0.16	865
LAST.FM			
$genre(a,g) \land subject(a,s) \land subject(b,s) \Rightarrow genre(b,s)$	0.02	0.09	63
$genre(m,b) \land pastMember(a,h) \land currentMember(m,h) \Rightarrow genre(a,b)$	0.01	0.25	30
$genre(a,h) \land genre(m,h) \land instrument(m,b) \Rightarrow instrument(a,b)$	0.27	0.03	124

TABLE 3.1: Some examples of rules mined by our framework. In the tables, we also report the metrics we used for the definition of the heuristics

confidence value should, in theory, contribute to the *graph completion* task, resulting in more paths connecting two nodes, which might lead to more meaningful and precise embeddings. Similarly, configurations (7-8) aim to consider different metrics for the same purpose. Configuration (8) includes all the rules with the *like* relation in the head, and it is the set of greatest interest, since the recommendation task focuses on investigating user preferences; for this reason this set is considered. In Section 4, the effectiveness of the framework on varying subsets of rules selected according to the different heuristics will be assessed.

Graph Embedding Learner Once the rules are extracted, the joint learning based on triples in the KG and FOL rules is carried out. In this work, a model based on KALE ([63]) was used. The *neuro-symbolic nature* of this model lies in the fact that the embeddings for each entity in the KG are learned by exploiting: *(i) explicit* knowledge, expressed in the form of triples (e_i,r_k,e_j) that are encoded in KG; *(ii) background* knowledge, expressed as FOL rules and learned as explained in Section 3.1.2.

The hallmark of the approach lies in the fact that both explicit and background knowledge are represented in a *unified* framework that learns a comprehensive representation based on both information sources. The model is based on KALE, a graph embedding technique that in turn inherits the principles of TransE and

KG triple	**FOL formalism**
$(Alice, like, KillBill)$	$like(Alice, KillBill)$
$(Alice, like, StarWarsI)$	$like(Alice, StarWarsI)$
$(Bob, like, StarWarsI)$	$like(Bob, StarWarsI)$
$(Chloe, like, StarWarsII)$	$like(Chloe, StarWarsII)$
$(KillBill, genre, Thriller)$	$genre(KillBill, Thriller)$
$(KillBill, starring, UmaThurman)$	$starring(KillBill, UmaThurman)$
$(KillBill, directedBy, QuentinTarantino)$	$directedBy(KillBill, QuentinTarantino)$
$(PulpFiction, directedBy, QuentinTarantino)$	$directedBy(PulpFiction, QuentinTarantino)$
$(PulpFiction, starring, UmaThurman)$	$starring(PulpFiction, UmaThurman)$
$(PulpFiction, starring, SamuelLJackson)$	$starring(PulpFiction, SamuelLJackson)$
$(PulpFiction, genre, Gangster)$	$genre(PulpFiction, Gangster)$
$(StarWarsI, starring, SamuelLJackson)$	$starring(StarWarsI, SamuelLJackson)$
$(StarWarsII, starring, SamuelLJackson)$	$starring(StarWarsII, SamuelLJackson)$
$(StarWarsI, prequel, StarWarsII)$	$prequel(StarWarsI, StarWarsII)$
$(StarWarsII, genre, SciFi)$	$genre(StarWarsI, SciFi)$

TABLE 3.2: Mapping of triples in a KG in FOL formalism.

extends it by introducing FOL rules in the learning process. Given the good performance of TransE in recommendation tasks, as shown by several strands of evidence including [128] and [136], the choice to exploit a method that relies on this algorithm can be considered reasonable.

The KGE model implements joint learning based on triangles from the KG and logical rules. Based on previous works by [150, 149], joint training is possible since triples from a KG can be seen as *atoms* in first-order logic (*e.g., likes(Alice,Kill Bill)* or *starring(Uma Thurman,Kill Bill)*). Indeed, each relation that connects two entities in a graph can be seen as a *binary predicate* (that is, a predicate which takes as arguments two variables), and the two entities can be seen as arguments of the predicate.

In this way, the triple of the KG $(Alice, like, KillBill)$ can be written in terms of FOL as follows: $like(Alice, KillBill)$, where $like$ is a binary predicate applied to the variables $Alice$ and $KillBill$. In Table 3.2 we report how the KG in Figure 2.1 can be written as a set of FOL facts.

Given that rules are also expressed in a logical form, *first-order logic* can be exploited as the common framework that allows unifying the representations and carrying out such joint learning. To start the learning process, a set $\mathcal{F}$ consisting of *positive* training elements is built. In particular, $\mathcal{F}^+$ contains: *(i)* all the atomic formulas based on the KG (*i.e.*, triples in the form (e_i, r_k, e_j)); *(ii) grounded* rules based on the logical rules previously extracted.

Given a logical rule in the form $\forall x, y : (x, r_s, y) \Rightarrow (x, r_t, y)$, a grounded rule is obtained by replacing variables x and y with real entities $e_i, e_j \in \mathcal{E}$. By referring again to the toy example in Figure 2.1, if the rule $\forall x, y, z : likes(x, y) \wedge prequel(y, z) \Rightarrow likes(x, z)$ is learned, the following *grounded rule* is generated: $likes(user, StarWars) \wedge prequel(StarWars, StarWarsII) \Rightarrow likes(user, StarWarsII)$. Of course, the replacement is bound only to triples that exist in the graph (*i.e.*, combinations of entities that do not exist in the graph do not constitute a valid grounding). Once the set $\mathcal{F}^+$ is obtained, a *negative* training set $\mathcal{F}^-$ shall be provided as well. To build $\mathcal{F}^-$, for each $(e_i, r_k, e_j) \in \mathcal{F}^+$, a corresponding negative triple is obtained by replacing either e_i or e_j with a random entity $e \in \mathcal{E}$.

Finally, joint learning is performed by minimizing a margin-based ranking loss (see Formula 3.1), enforcing positive training examples to have larger truth values than negative ones:

$$\min_{\{e\},\{r\}} \sum_{f^+ \in \mathcal{F}^+} \sum_{f^- \in \mathcal{F}^-} [\gamma - \mathcal{I}(f^+) + \mathcal{I}(f^-)]_+ \tag{3.1}$$

such that $\|e\|_2 \leq 1$ for all $e \in \mathcal{E}$ and $\|r\|_2 \leq 1$ for all $r \in \mathcal{R}$, where γ denotes a margin separating positive and negative examples, and $[x]_+ \triangleq \max\{0, x\}$.

Even if the embeddings of the entities (nodes) are the only ones of interest, Formula 3.1 is used to learn both entity and relationship embeddings.

To complete the training, it is also necessary to define a function $\mathcal{I} : \mathcal{F} \rightarrow [0,\ 1]$ that assigns to each training example (*i.e., atomic and complex formulas*) a soft truth value, indicating how likely a triple holds or to what degree a ground rule is satisfied.

Given a triple $f^+ \in \mathcal{F}^+$, the computation of $\mathcal{I}(f)$ is based on TransE ([19]), since each triple is modeled such that $e_i + r_k \approx e_j$. Accordingly, each triple is scored as $\|e_i + r_k - e_j\|_1$ using the following equation (see Equation 3.2):

$$\mathcal{I}(f^+) = \mathcal{I}(e_i, r_k, e_j) = 1 - \frac{1}{3\sqrt{d}} \parallel e_i + r_k - e_j \parallel_1, \tag{3.2}$$

where d is the dimension of the embedding space. Similarly, the same holds for negative triples $f^- \in \mathcal{F}^-$.

On the other hand, rules are modeled with t-norm fuzzy logics ([64]): the truth value of a complex formula is given by the composition of the truth values of its constituent formulae through specific t-norm-based logical connectives. In particular, truth values are computed recursively. Supposing f_1 and f_2 are atomic or complex formulae, truth values of complex formulae are computed as follows:

$$\mathcal{I}(f_1 \wedge f_2) = \mathcal{I}(f_1) \cdot \mathcal{I}(f_2)$$
$$\mathcal{I}(f_1 \vee f_2) = \mathcal{I}(f_1) + \mathcal{I}(f_2) - \mathcal{I}(f_1) \cdot \mathcal{I}(f_2)$$
$$\mathcal{I}(\neg f_1) = 1 - \mathcal{I}(f_1)$$
$$\mathcal{I}(f_1 \Rightarrow f_2) = \mathcal{I}(f_1) \cdot \mathcal{I}(f_2) - \mathcal{I}(f_1) + 1$$

where the truth values of the atomic formulas $\mathcal{I}(f_1),\ \mathcal{I}(f_2),\ ...,\ \mathcal{I}(f_n)$ are computed according to Equation 3.2. This study focuses on FOL rules and provides an example related to the last formulas presented, pertaining to the *logical implication*.

Consider the following FOL rule, composed of two atoms:

$$f \triangleq (e_i, r_k, e_j) \Rightarrow (e_k, r_l, e_j)$$

The truth value of f is given by

$$\mathcal{I}(f) = \mathcal{I}(e_i, r_k, e_j) \cdot \mathcal{I}(e_i, r_l, e_j) - \mathcal{I}(e_i, r_k, e_j) + 1 \text{ where}$$
$$\mathcal{I}(e_i, r_k, e_j) = 1 - \frac{1}{3\sqrt{d}} \parallel e_i + r_k - e_j \parallel_1 \text{ and}$$
$$\mathcal{I}(e_i, r_l, e_j) = 1 - \frac{1}{3\sqrt{d}} \parallel e_i + r_l - e_j \parallel_1 \text{ , which refer to Equation 3.2.}$$

The same process can be generalized to formulae with an indefinite number of atoms.

In this way, it is possible to score both triples deriving from the KG and triples derived from the FOL rules; thanks to the score, it is then possible to start the iterative training process, during which neuro-symbolic embeddings will be learned, encoding *explicit* knowledge deriving from the KG, and *background* knowledge deriving from the FOL rules, thus leading to a different representation of users and items.

Neural Recommender System Once the neuro-symbolic embeddings have been learned, the neural recommender system can be trained by feeding it with such embeddings. Given a user $u \in \mathcal{U}$ and an item $i \in \mathcal{I}$, the aim of this RS architecture is to predict the extent to which the user u will be interested in the item i by providing a recommendation score. This score is used to generate a recommendation list by ordering the items in descending order of the recommendation score.

Figure 3.2 shows the architecture of the deep neural network: starting from the embeddings representing the user u and item i (previously learned by the graph embedding learner), the embeddings are fed into three dense layers with decreasing dimensions and ReLU as activation function. Then, the two embeddings are concatenated to obtain a unique representation, and this new embedding is fed again into three dense layers with decreasing dimensions and ReLU as activation function. The last layer has size 1 and uses a *sigmoid* as activation function, to produce a recommendation score $score \in [0,\ 1]$, which estimates the probability that the item i is relevant for user u; this score is finally used to generate the recommendation list. The number of dense layers used in the architecture is based on design choices that are common in the literature (i.e., [167, 60, 102]) and is the result of tuning the architecture. The design choice relies on the intuition that stacking more layers of gradually smaller dimensions allows the model to better capture non-linearities between the input and the output. In turn, this enables better approximation of a function (in the present case, the recommendation function).

Formally, let $\vec{u}$ and $\vec{i}$ be the neuro-symbolic embeddings related to user u and item i, respectively; they are fed into three dense layers with reducing dimensions (with sizes 512, 256, and 128) and ReLU as activation functions:

$$\begin{aligned} u_{dense} &= ReLU\left(W_{u\_dense} * \vec{u} + b_{u\_dense}\right) \\ i_{dense} &= ReLU\left(W_{i\_dense} * \vec{i} + b_{i\_dense}\right) \end{aligned}$$

where u_{dense} and i_{dense} are the reduced embeddings related to u and i, respectively, W_{u_dense} and W_{i_dense} are the learnable parameters of the layers, and b_{u_dense} and b_{i_dense} are the biases. For brevity, only one application of this layer is reported here.

Subsequently, the two embeddings are concatenated, and the resulting embedding is fed into two layers with reduced dimensions (64 and 32) and ReLU activation functions, and one layer with size 1 and a sigmoid activation function. The output is the recommendation score between user u and item i.

$$concat_{dense} = ReLU\left(W_{concat\_dense} * \left(u_{dense} \oplus i_{dense}\right) + b_{concat\_dense}\right)$$
$$score = sigmoid\left(W_{score} * concat_{dense} + b_{score}\right)$$

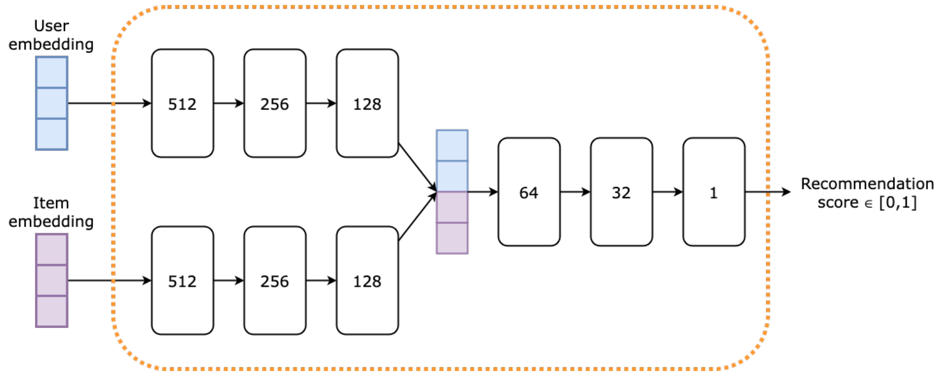


FIGURE 3.2: Recommendation framework architecture

This neural architecture is trained by exploiting the user ratings in the form (u, i) available in the datasets; moreover, since an architecture is trained to identify which elements are *positive* and *negative* for some users, the model has been trained using the *binary cross-entropy* as the loss function, and the *accuracy* as the evaluation metric to be optimized. More details about the training, the size of the layers, the optimizer and the other hyperparameters will be provided in the next section. After the training of the model, it is able to predict the extent to which user u would like unseen items $i \in \mathcal{I}$, from which a recommendation list consisting of the *top-k* items for each user is generated.

3.1.3 Experimental Setting

In the experimental section, experiments were conducted to assess the effectiveness of the methodology in the task of item recommendation. In particular, experiments were designed to answer the following research questions:

- **RQ3.1.1 - Impact of FOL rules on recommendation lists:** Does the introduction of logical rules *change* the elements included in the recommendation lists?
- **RQ3.1.2 - FOL Heuristics Comparisons:** How do the different selection heuristics for picking FOL rules impact the *accuracy, novelty, and diversity* of the recommendations?

- **RQ3.1.3 - Comparisons to Baselines:** How does the approach based on *neuro-symbolic graph embeddings* perform with respect to *competitive baselines*?

Experiments were conducted in a *movie, book and music recommendation* scenario. In particular, the ML1M, DBbook and Last.fm datasets have been considered. More details about this datasets can be found in Section 2.3, where the problem has been formalized.

More details about the protocol and the experimental settings are deeply discussed in the related published papers [170, 174]. Here, the discussion focuses on the most relevant aspect that characterize the work.

First-Order Logic Rules FOL rule mining was performed to mine rules with at most 4 atoms (i.e., the maximum length of the rules), and the minimum confidence was set to 0.001, with 0.1 as the minimum coverage value. This choice followed several runs of the strategy, based on a trade-off analysis between algorithmic complexity and knowledge provided by the FOL rules.

Embeddings Embeddings were learned at three different sizes: $k = 256$, $k = 512$, and $k = 768$ dimensions.

Experimental Configurations Experiments were conducted to compare ten configurations of the recommendation model based on neuro-symbolic graph embeddings combining FOL rules. In particular, two basic configurations, relying only on information from the graph, and eight configurations based on the injection of FOL rules into the KGE process, were considered.

For the first basic configuration, graph embeddings encoding only information derived from user-item interactions were learned: for example, referring to Figure 2.1, the following triples were encoded:

$$\begin{aligned} (Alice, like, KillBill),\\ (Alice, like, StarWarsI),\\ (Bob, like, StarWarsII),\\ (Chloe, like, StarWarsII) \end{aligned}$$

For the second basic configuration, not only user-item interactions but also all the structured properties deriving from the KG were considered: for example, again referring to Figure 2.1, the triples related to the genre of the movies or the actors

Heuristic	**Last.FM**		**DBbook**		**ML1M**	
	Rules	**Groundings**	**Rules**	**Groundings**	**Rules**	**Groundings**
LSC	17	3652	28	2265	505	4M
MSC	4	2371	20	1472	152	75M
MHSC	3	147	12	362	62	3817
HSC	2	14	7	78	39	3029
LHC	9	864	5	2195	63	292K
HPE	9	864	10	17989	70	10M
LH	0	0	2	8189	27	5M

TABLE 3.3: Statistics about the used subsets of FOL rules

starring in them were encoded, such as

$$(KillBill, genre, Thriller),$$
$$(PulpFiction, starring, SamuelLJackson),$$
$$(StarWarsII, starring, SamuelLJackson)$$

Next, eight configurations were evaluated based on the injection of FOL rules into the KGE learning. Table 3.3 reports statistics on the number of rules and groundings found for each dataset; it should be recalled that, given a FOL rule, a *grounding* for that rule is obtained when all variables have been replaced with real entities in the KG. It should be noted that, for LAST.FM, the set of LIKE rules is empty and, consequently, this dataset will not appear in the analysis and discussion of the results.

Evaluation Metrics The evaluation was performed using the Elliot[2] framework [4], to guarantee reproducibility of the experimental protocol. It also has been used to evaluate the recommendation lists. As for accuracy metric, precision, recall, F1, MAP and nDCG were considered. As for beyond-accuracy metrics, Gini index[3] (diversity), EPC and EFD (novelty) were considered.

Baselines Several baselines belonging to different model groups were considered, and are reported in the tables. Their description is discussed in Section 2.3, and is not reported here to avoid repetitions.

[2]`https://github.com/sisinflab/elliot`

[3]This implementation returns the value $1 - Gini$, so that the hiher the better.

3.1.4 Results and Discussion

This section presents the results of the experimental study, beginning with a qualitative analysis of the item representations, and subsequently addressing the research questions.

RQ3.1.1: Impact of FOL rules on recommendation lists In this section, results assessing the impact of FOL rules on the recommendation lists are presented. In other terms, this study aims to investigate whether and how the introduction of FOL rules changes the items recommended to users.

Qualitative results obtained by considering real recommendation lists generated by the recommendation model based on neuro-symbolic graph embeddings with FOL rules are presented. Results pertain to the movie recommendation domain and are reported in Table 3.4; two users were considered and the recommendation lists generated for the two *basic* configurations (*user-item* and *user-item-properties*, shortened as *UI* and *UIP*), and one configuration based on FOL rules (those related to the rules with the relation *like* in the head). It should be noted that different encoded information can have a substantial impact on the recommendation lists: For example, for *User 6039*, only one movie appears in all three recommendation lists (*"North by Northwest"*), while only two movies are shared by the UIP and the UIP + LH recommendation lists (the already mentioned *"North by Northwest"*, and *"Citizen Kane"*). **This corroborates the observation described in the previous section: different encoded knowledge, although deriving from the same source (the KG), leads to different representations.** In addition, further observations can be made by analyzing the recommendation lists for *User 5743*; in this case, the impact of FOL rules can be better assessed: while the two *basic* configurations share *four* movies out of a total of *five*, the lists are almost the same, except for one element and the permutation of the others, whereas the UIP + LH list shares only two movies with the *basic* configurations (*"Wings"* and *"Roman Holiday"*), while the other three movies are totally new. **This illustrates well how injecting FOL rules into the graph embedding process can impact not only the representation of items but also the recommendation task.** Nevertheless, encoding different information does not always lead to differences, as the recommendation lists related to *User 12* show: in this case, the three recommendation lists are very similar to each other, although some differences in ordering can be observed, and movies not appearing in the other lists can also be observed.

User 6039		
UI	***UIP***	***UIP + LH***
Fanny and Alexander	The Usual Suspects	Citizen Kane
Heathers	The Shawshank Redemption	The Treasure of the Sierra Madre
The Pillow Book	North by Northwest	North by Northwest
North by Northwest	Alien	Graduate
Citizen Kane	Monty Python and the Holy Grail	Notorious (1946 film)
User 5743		
UI	***UIP***	***UIP + LH***
Sabrina	Roman Holiday	Wings
Some Like It Hot	Wings	Suspicion
Wings	Sabrina	Winnie the Pooh and the Blustery Day
Roman Holiday	Some Like It Hot	Roman Holiday
Them!	Being John Malkovich	Lady and the Tramp
User 12		
UI	***UIP***	***UIP + LH***
The Silence of the Lambs	Star Wars Episode IV	Star Wars Episode IV
Star Wars Episode IV	The Silence of the Lambs	The Silence of the Lambs
Kolya	Kolya	Jurassic Park
Jurassic Park	Galaxy Quest	Kolya
Men in Black	Men in Black	Men in Black

TABLE 3.4: Top-k real recommendations provided by different configurations of our model in the movie field, for Users *6039*, *5743*, and *12*; for this example, we set *k*=*5*.

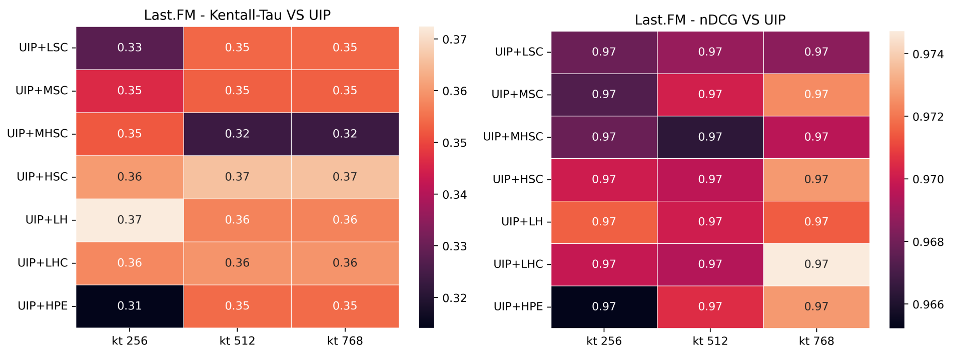


FIGURE 3.3: Values of Kendall-Tau's test and nDCG metric for the LAST.FM dataset

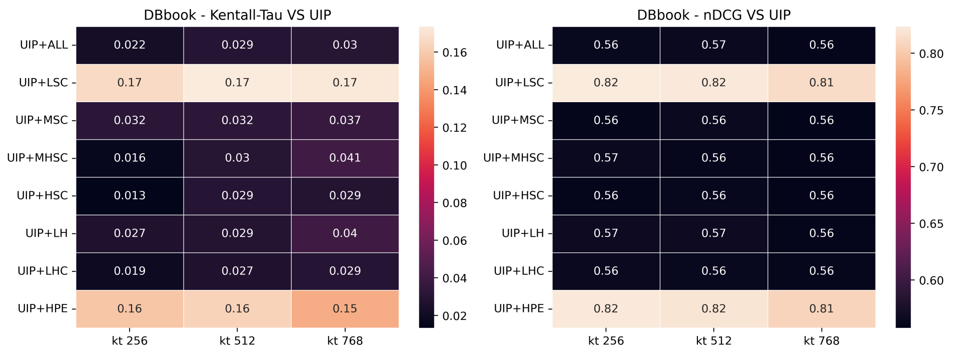
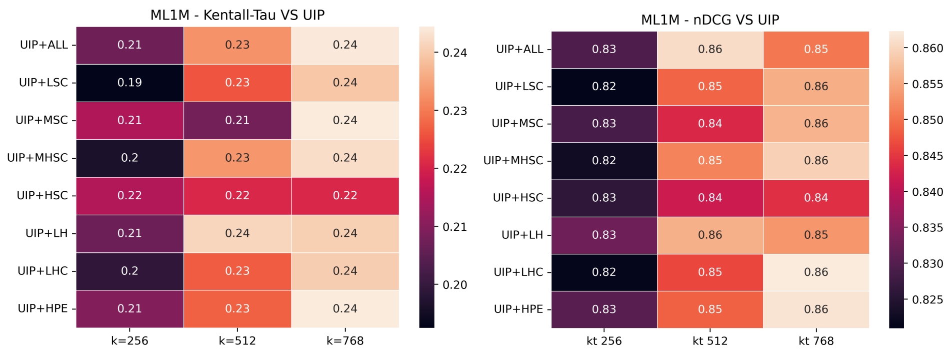


FIGURE 3.4: Values of Kendall-Tau's test and nDCG metric for the DBBOOK dataset

Next, in addition to the above-presented *qualitative* results, this study carried out a more systematic analysis of the variations in the recommendation lists based on Kendall-Tau (KT) coefficient and NDCG. In both cases, the top-5 recommendation lists (*i.e.*, for all the users) were obtained by varying all the different subsets of FOL rules, based on the implementation of KT available in scikit-learn[4]. In both cases, *lower is better*, since KT coefficient measures the overlap in terms of elements in the recommendation lists, while the nDCG score measures how similar a recommendation list is with respect to a *ground truth*. In our case, the ground truth is represented by the recommendation lists obtained by running the simple UIP configuration, that is to say, the one based on the user-item-properties graph without any FOL rule. The results are presented in Figures 3.3, 3.4 and 3.5.

[4]The implementation was also released on our repository: `https://github.com/swapUniba/KARS_NeSy_KGE_with_FOL_rules/blob/main/kendall_tau.py`



	k=256	k=512	k=768
UIP+ALL	0.21	0.23	0.24
UIP+LSC	0.19	0.23	0.24
UIP+MSC	0.21	0.21	0.24
UIP+MHSC	0.2	0.23	0.24
UIP+HSC	0.22	0.22	0.22
UIP+LH	0.21	0.24	0.24
UIP+LHC	0.2	0.23	0.24
UIP+HPE	0.21	0.23	0.24

	kt 256	kt 512	kt 768
UIP+ALL	0.83	0.86	0.85
UIP+LSC	0.82	0.85	0.86
UIP+MSC	0.83	0.84	0.86
UIP+MHSC	0.82	0.85	0.86
UIP+HSC	0.83	0.84	0.84
UIP+LH	0.83	0.86	0.85
UIP+LHC	0.82	0.85	0.86
UIP+HPE	0.83	0.85	0.86

FIGURE 3.5: Values of Kendall-Tau's test and nDCG metric for the ML1M dataset

As for *Last.fm*, the results show that KT scores range between 0.31 and 0.37. This means that all configurations based on FOL change some items in the recommendation lists relative to the basic UIP configuration. However, the analysis of the nDCG scores shows that the change is usually tiny, since these scores range from 0.966 to 0.974. Given that nDCG ranges from 0 to 1, this means that the change is very limited. However, by considering the statistics presented in Table 3.3, this is not surprising since the number of rules (and grounding) mined from the data is very low. Accordingly, it makes sense that this leads to a minimum change in the recommendation lists.

Next, by considering *DBbook* data, we note a more significant change, since nDCG scores range from 0.56 to 0.82. In particular, 6 out of 8 configurations show lower nDCG scores, which means that many different items are modified due to the introduction of FOL rules. **Similar outcomes also emerge from *ML1M* data**, since all the configurations obtain an nDCG score around 0.82. While the score does not differ across the configurations, we note that the lower embedding dimension ($k = 256$) leads to a lower nDCG score (i.e., more changes in the items). In the experimental evaluation, it will be assessed whether the change in the lists leads to an increase in terms of both accuracy and non-accuracy metrics.

RQ3.1.2: FOL Heuristics Comparisons To answer RQ3.1.2, the performance of the eight configurations based on neuro-symbolic knowledge graph embeddings was compared with the two basic configurations based only on *UI* and *UIP* information (shortened, again, as *UI* and *UIP*), respectively. Results for the music domain are reported in Tables 3.5, while those related to the book domain are reported in Table 3.6, and finally, results related to the movie domain are reported in Table 3.7. In these tables, the performance of the model is reported considering not only the different heuristics used to select FOL rules to be injected during the graph embedding learning process, but also the performance as the embedding dimension k varies, as discussed in the methodology section.

In particular, for the music domain (see Table 3.5), it can be observed that in most cases the configurations based on the injection of FOL rules overcome the two *basic* configurations, especially when considering configurations with $k = 256$ and $k = 768$. For the $k = 256$ setting, the best configuration is clearly UIP + LSC, i.e., the configuration obtained by injecting all FOL rules with a Standard Confidence >0.2, the precision, recall, F1 score, Gini index, and EFD score of which exceed the baselines and the other configurations; this indicates that this configuration is not only accurate, achieving the best performance in several accuracy metrics, but also provides more diverse and novel recommendation lists. The configuration UIP + HSC, obtained by injecting FOL rules with a Standard Confidence >0.9, yielded the best results in the other accuracy metrics, MAP, MAR, and nDCG, and achieved the best performance in terms of EPC as well. The setting $k = 512$ shows a more varied scenario: the *basic* configuration based only on *user-item* interactions (UI) obtained the best results in terms of precision, recall, and F1 score, leading the accuracy metrics; best MAP and best nDCG were obtained by the configuration UIP + ALL (configuration injecting *all* the available FOL rules), while the configurations UIP + MHSC and UIP + HSC provided the more diverse recommendation lists; moreover, the latter configuration obtained the best F1 score among the configurations with FOL rules. As for the setting $k = 768$, the best configuration is certainly UIP + LHC (rules with head coverage >0.2), which is not only accurate (best MAP, MAR, and nDCG), but also provides recommendation lists with the best novelty. UIP + MHSC obtained the best results in terms of precision and F1, while UIP + ALL achieved the best F1 score. Finally, the *basic* configuration UIP provided the most diverse recommendation lists.

The book domain is considered next (Table 3.6). For the setting $k = 256$, the overall best configuration is UIP + HSC (rules with standard confidence > 0.75), which achieved the best performance in terms of MAP, MAR, and nDCG for the accuracy metrics; moreover, it proved to be the configuration with the most diverse recommendation lists, and it also achieved good performance in terms of novelty.

The best precision is attained by the configuration UIP + LSC, while the best recall and F1 score are attained by UIP + LH (rules having the relation *like* in their head), which also obtained the highest EPC score. Generally, for the setting

Heuristic	**Accuracy**						**Diversity**	**Novelty**	
	MAP	**Precision**	**MAR**	**Recall**	**F1**	**nDCG**	**Gini**	**EFD**	**EPC**
256									
UI	0.5779	0.4524	0.7227	0.9534	0.5739	0.8460	0.2902	6.87	0.6200
UIP	0.5801	0.4523	0.7276	0.9536	0.5739	0.8507	0.2890	6.87	0.6213
UIP + ALL	0.5809	0.4512	0.7277	0.9532	0.5728	0.8496	0.2889	6.87	0.6202
UIP + LSC	0.5760	**0.4528**	0.7237	**0.9549**	**0.5744**	0.8470	**0.2923**	**6.89**	0.6202
UIP + MSC	0.5816	0.4517	0.7278	0.9512	0.5728	0.8496	0.2897	**6.89**	**0.6215**
UIP + MHSC	0.5802	0.4518	0.7265	0.9532	0.5733	0.8490	0.2881	6.87	0.6203
UIP + HSC	**0.5841**	0.4513	**0.7286**	0.9529	0.5728	**0.8515**	0.2875	6.87	**0.6215**
UIP + LHC	0.5783	0.4514	0.7271	0.9521	0.5727	0.8487	0.2913	6.88	0.6202
UIP + HPE	0.5755	0.4513	0.7244	0.9528	0.5728	0.8462	0.2894	6.86	0.6189
512									
UI	0.5776	**0.4532**	0.7248	**0.9550**	**0.5751**	0.8481	0.2894	6.87	0.6202
UIP	0.5765	0.4515	0.7239	0.9533	0.5732	0.8471	0.2911	6.87	0.6192
UIP + ALL	**0.5799**	0.4510	0.7258	0.9525	0.5726	**0.8502**	0.2889	6.86	0.6204
UIP + LSC	0.5772	0.4497	0.7242	0.9482	0.5705	0.8461	0.2914	6.87	0.6196
UIP + MSC	0.5728	0.4490	0.7210	0.9484	0.5700	0.8394	0.2903	6.84	0.6156
UIP + MHSC	0.5791	0.4528	**0.7279**	0.9523	0.5740	0.8495	0.2901	**6.89**	**0.6210**
UIP + HSC	0.5752	0.4509	0.7242	0.9536	0.5728	0.8463	**0.2915**	6.85	0.6182
UIP + LHC	0.5776	0.4500	0.7241	0.9505	0.5712	0.8455	0.2906	6.86	0.6184
UIP + HPE	0.5770	0.4497	0.7248	0.9506	0.5710	0.8454	0.2892	6.85	0.6182
768									
UI	0.5746	0.4522	0.7228	0.9530	0.5737	0.8442	0.2908	6.86	0.6183
UIP	0.5807	0.4524	0.7262	0.9525	0.5738	0.8489	**0.2911**	6.89	0.6216
UIP + ALL	0.5771	0.4519	0.7238	**0.9547**	0.5737	0.8473	0.2886	6.86	0.6193
UIP + LSC	0.5837	0.4517	0.7285	0.9529	0.5732	0.8530	0.2875	6.90	0.6227
UIP + MSC	0.5755	0.4510	0.7233	0.9518	0.5725	0.8443	0.2900	6.84	0.6180
UIP + MHSC	0.5780	**0.4528**	0.7250	0.9538	**0.5743**	0.8484	0.2902	6.87	0.6203
UIP + HSC	0.5769	0.4524	0.7222	0.9546	0.5742	0.8438	0.2895	6.86	0.6184
UIP + LHC	**0.5849**	0.4524	**0.7310**	0.9536	0.5740	**0.8555**	0.2896	**6.92**	**0.6246**
UIP + HPE	0.5824	0.4521	0.7293	0.9527	0.5733	0.8531	0.2899	6.88	0.6219

TABLE 3.5: Results of our model on the Last.FM dataset. In **bold**, the best overall score, while the best score obtained by a configuration based on FOL rules is underlined.

$k = 256$ in the previous domain, the *basic* configurations have been outperformed by the configurations that include FOL rules.

As for $k = 512$, similarly to the previous case, the *basic* configurations achieved the best performance in almost all metrics: in particular, UIP was the best configuration in terms of MAP (equal score to UIP + HPE, with rules having a high number of positive examples), MAR, recall, F1, nDCG, EFD, and EPC; the highest MAR was obtained by UIP + LHC, while the best diversity of the recommendation lists was achieved by the configurations UIP + HPE and UIP + LH.

Finally, for the setting $k = 768$, it is evident that the configurations with rules outperformed the two baselines, and UIP + LSC was the configuration that achieved the best results. In particular, it achieved the best MAR, NDCG, Novelty, and Diversity, while UIP + HSC achieved the best results in terms of accuracy (precision, recall, and F1).

Finally, the results obtained in the movie domain (Table 3.7) are discussed. For $k = 256$, the configurations using FOL rules surpassed the two baselines. In particular, the overall best configuration is UIP + LH, which achieved the best accuracy in terms of precision, MAR, MAP, and nDCG, while UIP + HSC achieved the highest MAP and UIP + MHSC achieved the highest recall and F1. The highest novelty was obtained by UIP + MHSC for the EFD and UIP + LSC for the EPC, while the basic configuration UI proved to be the configuration providing the most diverse recommendation lists. UIP + MHSC achieved the highest diversity among the configurations based on rules.

In the setting $k = 512$, unlike the previous cases, the configurations with FOL rules surpassed the two *basic* configurations in almost all metrics. In fact, UIP + LHC and UIP + MHSC dominate the accuracy, with the highest precision, recall, F1, and nDCG for UIP + LHC, and the highest MAP and nDCG for UIP + MHSC. The configuration UIP + LSC provided the most novel recommendation lists, while the highest diversity was again obtained by UI, while UIP + MSC provided the highest novelty among the configurations using rules.

The setting $k = 768$ follows the others: the highest accuracy was obtained by UIP + LSC and UIP + MSC (MAP, MAR, and nDCG for UIP + LSC, while precision, recall, and F1 for UIP + MSC); the configuration UIP + HSC provided the most novel recommendation lists; it also achieved the best diversity among the configurations using FOL rules, but the highest diversity was again obtained by UIP. These results suggest that, in the case of the dataset ML1M, which is denser

Heuristic	**Accuracy**						**Diversity**	**Novelty**	
	MAP	**Precision**	**MAR**	**Recall**	**F1**	**nDCG**	**Gini**	**EFD**	**EPC**
					256				
UI	0.6378	0.6362	0.3013	0.4982	0.5145	0.6787	0.2797	7.4064	0.6337
UIP	0.6374	0.6362	0.2989	0.4970	0.5138	0.6770	0.2796	7.4124	0.6337
UIP + ALL	0.6360	0.6328	0.2978	0.4937	0.5108	0.6747	0.2813	7.3670	0.6308
UIP + LSC	0.6380	**0.6385**	0.2986	0.4978	0.5155	0.6777	0.2791	7.4289	0.6349
UIP + MSC	0.6384	0.6367	0.3000	0.4983	0.5148	0.6785	0.2809	7.4132	0.6342
UIP + MHSC	0.6377	0.6351	0.2998	0.4953	0.5128	0.6771	0.2798	7.3972	0.6330
UIP + HSC	**0.6414**	0.6376	**0.3015**	0.4980	0.5150	**0.6806**	**0.2835**	**7.4473**	0.6361
UIP + LHC	0.6380	0.6350	0.2991	0.4979	0.5135	0.6773	0.2806	7.4070	0.6331
UIP + HPE	0.6388	0.6362	0.2995	0.4958	0.5133	0.6780	0.2810	7.4210	0.6343
UIP + LH	0.6413	0.6381	0.3009	**0.4985**	**0.5157**	0.6803	0.2787	7.4381	**0.6362**
					512				
UI	0.6370	0.6362	0.2997	0.4972	0.5141	0.6771	0.2786	7.3964	0.6333
UIP	**0.6397**	**0.6376**	0.2993	**0.4992**	**0.5156**	**0.6791**	0.2796	**7.4234**	**0.6355**
UIP + ALL	0.6378	0.6365	0.2998	0.4979	0.5143	0.6783	0.2819	7.4120	0.6336
UIP + LSC	0.6392	0.6364	0.2987	0.4957	0.5134	0.6779	0.2808	7.4128	0.6348
UIP + MSC	0.6380	0.6358	0.2998	0.4967	0.5137	0.6776	0.2786	7.4116	0.6336
UIP + MHSC	0.6380	0.6352	0.3001	0.4966	0.5135	0.6773	0.2795	7.3928	0.6330
UIP + HSC	0.6372	**0.6379**	0.2997	**0.4993**	0.5155	**0.6795**	0.2796	7.4110	0.6346
UIP + LHC	0.6376	0.6355	**0.3002**	0.4981	0.5138	0.6779	0.2790	7.3970	0.6331
UIP + HPE	**0.6397**	0.6358	0.2991	0.4954	0.5133	0.6774	0.2822	7.4230	0.6344
UIP + LH	0.6387	0.6371	0.2998	0.4973	0.5148	0.6783	**0.2825**	7.4165	0.6345
					768				
UI	0.6351	0.6346	0.2975	0.4947	0.5126	0.6743	0.2807	7.3802	0.6315
UIP	0.6392	0.6372	0.2996	0.4979	0.5148	0.6785	0.2805	7.4192	0.6351
UIP + ALL	0.6375	0.6334	0.2982	0.4926	0.5110	0.6756	0.2800	7.3785	0.6320
UIP + LSC	**0.6404**	0.6375	0.2998	0.4978	0.5150	**0.6795**	**0.2822**	**7.4345**	**0.6358**
UIP + MSC	0.6387	0.6371	0.2992	0.4974	0.5147	0.6784	0.2795	7.4094	0.6345
UIP + MHSC	0.6389	0.6363	0.2991	0.4960	0.5137	0.6781	0.2793	7.4192	0.6345
UIP + HSC	0.6382	**0.6379**	0.2997	**0.4993**	**0.5155**	**0.6795**	0.2796	7.4110	0.6346
UIP + LHC	0.6378	0.6354	0.2992	0.4965	0.5131	0.6778	0.2805	7.4015	0.6333
UIP + HPE	0.6396	0.6356	**0.2999**	0.4963	0.5135	0.6780	0.2812	7.4174	0.6343
UIP + LH	0.6387	0.6371	0.2998	0.4973	0.5148	0.6783	0.2786	7.4165	0.6345

TABLE 3.6: Results of our model on the DBbook dataset. In **bold**, the best overall score, while the best score obtained by a configuration based on FOL rules is underlined.

than the previous two and has many more interactions (about 1M), the properties might negatively influence diversity, while the FOL rules tend to increase it, without bringing it back to the best value obtained by UI.

Generally, the exploitation of FOL rules in the knowledge graph embedding learning process positively influences the performance of the recommender systems in almost all cases. As for Last.fm and DBbook, the injection of FOL rules improves *all* the metrics for $k = 256$ and $k = 768$ and 6 out of 9 metrics (with the exception of Precision, Recall and F1) on $k = 512$. As for MovieLens, the results are even better since the only exception is represented by the Gini index, whose score decreases when FOL rules are injected. Overall, the best-performing configuration is obtained by exploiting a configuration that also encodes rules, with the sole exception of Precision, Recall and F1 on Last.fm. **In all other comparisons, the overall best results exploit the FOL rules mined by the framework.**

As regards the performance of the different heuristics, a very heterogeneous behavior emerges. First, **it is important to note that injecting *all* available rules is not optimal. Indeed, the best rule-based configuration is never the one labeled as UIP+ALL.** Similarly, using only the rules that have a *like* in the head led to satisfactory results. Indeed, while the UIP+LH configuration achieved the best partial results on some comparisons (*i.e.*, Recall and F1 on DBbook with $k = 256$, or some metrics on ML1M with $k = 256$), other subsets of rules tend to obtain better results. Accordingly, this confirms that the injection of FOL can be useful in identifying patterns in the data that do not necessarily identify or clarify user preferences.

In particular, the configurations based on the use of *confidence* scores as a means to select the rules are the most promising ones. As for Last.fm, the UIP+LSC configuration (based on rules with a low confidence threshold) achieved the best results on Precision, Recall, F1 and Gini Index. Good performance was also obtained by UIP+LHC, which achieved the best results on the remaining five metrics. In this case, *coverage* is used as the criterion to select the rules.

Next, considering DBbook data, the best results are obtained with the UIP+HSC configuration. In this case, a lower number of rules is injected since the system only exploits rules having a confidence higher than 0.9. The adoption of this subset allows the results to beat all other configurations on all metrics, with the exception of Precision and F1. Accordingly, this confirms the positive impact of the rules on ranking items better and increasing the novelty and diversity of the recommendations. A similar behavior emerges for ML1M data, since the exploitation

Heuristic	Accuracy						Diversity	Novelty	
	MAP	Precision	MAR	Recall	F1	nDCG	Gini	EFD	EPC
					256				
UI	0.6880	0.6763	0.2548	0.4089	0.4278	0.7242	**0.2618**	7.0021	0.5789
UIP	0.8386	0.7953	0.2990	0.4581	0.4884	0.8648	0.1482	7.9823	0.6621
UIP + ALL	0.8399	0.7955	0.2992	0.4571	0.4878	0.8655	0.1492	8.0102	0.6650
UIP + LSC	0.8373	0.7950	0.2975	0.4572	0.4879	0.8634	0.1568	8.0754	**0.6728**
UIP + MSC	0.8380	0.7958	0.2986	0.4579	0.4884	0.8645	0.1566	8.0463	0.6679
UIP + MHSC	0.8386	0.7976	0.2996	**0.4593**	**0.4901**	0.8659	0.1586	**8.0883**	0.6714
UIP + HSC	**0.8404**	0.7970	0.2995	0.4579	0.4886	0.8661	0.1549	8.0669	0.6700
UIP + LHC	0.8378	0.7971	0.2991	0.4587	0.4892	0.8653	0.1502	8.0352	0.6682
UIP + HPE	0.8391	0.7968	0.2989	0.4581	0.4887	0.8657	0.1504	8.0282	0.6669
UIP + LH	0.8403	**0.7988**	**0.2998**	**0.4593**	**0.4901**	**0.8674**	0.1505	8.0496	0.6690
					512				
UI	0.6426	0.6405	0.2466	0.3994	0.4141	0.6827	**0.2948**	6.6859	0.5527
UIP	0.8361	0.7954	0.2986	0.4581	0.4884	0.8632	0.1587	8.0673	0.6712
UIP + ALL	0.8338	0.7942	0.2981	0.4573	0.4874	0.8619	0.1572	8.0348	0.6684
UIP + LSC	0.8375	0.7960	0.2990	0.4585	0.4889	0.8647	0.1592	**8.0889**	**0.6718**
UIP + MSC	0.8381	0.7951	0.2983	0.4578	0.4884	0.8638	0.1615	8.0756	0.6709
UIP + MHSC	**0.8408**	0.7974	**0.2999**	0.4585	0.4891	0.8669	0.1489	8.0424	0.6682
UIP + HSC	0.8391	0.7947	0.2993	0.4586	0.4886	0.8646	0.1591	8.0432	0.6672
UIP + LHC	0.8404	**0.7994**	0.2996	**0.4601**	**0.4909**	**0.8675**	0.1481	8.0541	0.6704
UIP + HPE	0.8402	0.7965	0.2995	0.4582	0.4889	0.8660	0.1457	8.0154	0.6665
UIP + LH	0.8390	0.7960	0.2997	0.4581	0.4885	0.8657	0.1504	8.0260	0.6686
					768				
UI	0.6937	0.6796	0.2583	0.4121	0.4307	0.7292	**0.2503**	7.0112	0.5815
UIP	0.8383	0.7971	0.2986	0.4574	0.4886	0.8648	0.1508	8.0427	0.6687
UIP + ALL	0.8370	0.7961	0.2986	0.4581	0.4886	0.8642	0.1582	8.0670	0.6698
UIP + LSC	**0.8419**	0.7975	**0.3002**	0.4589	0.4894	**0.8677**	0.1504	8.0540	0.6685
UIP + MSC	0.8386	**0.7977**	0.2998	**0.4598**	**0.4899**	0.8662	0.1504	8.0166	0.6658
UIP + MHSC	0.8377	0.7961	0.2986	0.4582	0.4887	0.8644	0.1545	8.0393	0.6672
UIP + HSC	0.8381	0.7960	0.2985	0.4576	0.4883	0.8645	0.1633	**8.1166**	**0.6750**
UIP + LHC	0.8373	0.7971	0.2991	0.4586	0.4893	0.8647	0.1570	8.0582	0.6689
UIP + HPE	0.8391	0.7972	0.2993	0.4583	0.4892	0.8659	0.1502	8.0239	0.6667
UIP + LH	0.8362	0.7936	0.2975	0.4569	0.4870	0.8619	0.1601	8.0532	0.6703

TABLE 3.7: Results of our model on the ML1M dataset. In **bold**, the best overall score, while the best score obtained by a configuration based on FOL rules is underlined.

of rules with *high confidence* leads to the best results in terms of novelty and diversity, while the adoption of *coverage* as a heuristic improves the accuracy metrics.

Overall, the most promising subsets are those that use *confidence* (LSC and HSC) and *coverage* (LHC) as heuristics to select the rules. Regarding the threshold to cut rule selection, some relationship between the structure of the graph and the number of rules to be injected seems to exist. Indeed, with smaller KGs (such as DBbook), with a lower number of entities and a lower number of properties per item, it is preferable to inject fewer rules. Conversely, when more data are available, more rules can be exploited (i.e., the LSC configuration). For example, this holds for ML1M, where a large number of groundings are injected into the model (see Table 3.3). This can provide a general guideline that can both: *(i)* guide the design of further implementations based on the subsets; *(ii)* provide a deeper understanding of the results of the model.

To conclude, the analyses show that the heuristics designed are able to automatically identify a subset of the rules that can be useful for improving the accuracy of the recommendations. Further qualitative and quantitative analyses (*i.e.*, based on the kinds of properties mentioned in the rules) will be carried out to examine in detail the individual rules injected into the model, in order to better assess the kind of information that is needed or useful to improve the quality of the recommendation process. These analyses are left as future work.

RQ3.1.3: Comparisons to Baselines Experiments were conducted to compare the results obtained by the recommender system based on neuro-symbolic graph embeddings with competitive state-of-the-art recommendation models. The results of the comparison can be found in Table 3.8 for the music domain (LAST.FM), in Table 3.9 for the book domain (DBBOOK), and finally for the movie domain in Table 3.10 (ML1M). In particular, the performance of the best configuration in each domain was compared with all the baselines considered. Generally speaking, it is easy to observe that the model outperformed all baselines for almost all the considered metrics, with the only exception represented by EASE^R. Moreover, most of these improvements are statistically significant ($p < 0.05$).

In the music domain, the configuration of the model that achieved the best overall performance is the one injecting FOL rules with a standard confidence score greater than 0.2 (the UIP + LSC configuration reported in Table 3.5). From Table 3.8, results are observed to be satisfying, since they are comparable with those

Baseline	Accuracy						Diversity	Novelty	
	MAP	Precision	MAR	Recall	F1	nDCG	Gini	EFD	EPC
MultiVAE	0.5531	0.4467	0.6955	0.9406	0.5666	0.8173	**0.3050***	6.7488	0.6079
VSM	0.5617	0.4440	0.7043	0.9381	0.5638	0.8257	0.3043	6.7989	0.6121
CFGAN	0.5669	0.4501	0.7111	0.9484	0.5712	0.8314	0.2819	6.7171	0.6115
LightGCN	0.5665	0.4500	0.7125	0.9483	0.5711	0.8311	0.2792	6.6889	0.6100
NGCF	0.5475	0.4437	0.6921	0.9361	0.5632	0.8080	0.3004	6.6980	0.6026
KaHFM	0.5460	0.3921	0.6509	0.8176	0.4949	0.7653	0.1600	6.3085	0.6099
BPRMF	0.5854	0.4504	0.7277	0.9507	0.5718	0.8519	0.2682	6.6775	0.6175
PureSVD	0.5923	0.4488	0.7292	0.9421	0.5686	0.8555	0.2696	6.7875	0.6273
Slim	0.5118	0.2875	0.5410	0.5866	0.3595	0.6521	0.1366	6.7781	0.6257
ItemKnn	0.5940	0.4463	0.7275	0.9359	0.5654	0.8549	0.2575	6.7666	0.6269
UserKnn	0.5813	0.4230	0.6963	0.8756	0.5332	0.8203	0.2340	6.7974	**0.6340***
Attr.ItemKnn	0.5064	0.3849	0.5977	0.7784	0.4814	0.7097	0.2769	6.5825	0.5953
Attr.UserKnn	0.5681	0.4130	0.6733	0.8524	0.5205	0.7968	0.2218	6.6102	0.6198
EASE^R	**0.6028**	**0.4549**	**0.7451***	**0.9586**	**0.5772**	**0.8726***	0.2714	6.8311	0.6305
Our best	0.5760	0.4528	0.7237	0.9549	0.5744	0.8470	0.2923	**6.8858***	0.6202

TABLE 3.8: Comparisons to baselines in the music field (Last.fm dataset). The best overall value is highlighted in bold, while the best baseline value is underlined. (*) means the differences are statistically significant.

obtained by the best baseline (i.e., EASE^R) in terms of precision, recall and F1 measure. Moreover, the best results in terms of novelty (measured through EFD) were achieved. As for ranking measures, such as NDCG, results are lower than those for EASE^R, but remain in line with the next best-performing techniques such as BPRMF, ItemKNN and PureSVD. To better understand the nature of the results, it is necessary to consult the statistics presented in Table 4 and Table 5. Indeed, both tables show that the number of items mapped to the knowledge graph and the number of FOL rules are generally low, especially when compared to ML1M. Consequently, it is likely that the sparsity of the graph and the low quality of information mined from the graph contributed to these results.

In the book domain, Table 3.9 shows that the presented approach does not overcome all baselines considered. Unfortunately, this holds for all metrics since at least one baseline surpasses the present approach. As noted previously for LastFM, EASE^R had the best results. However, in this case the gap is statistically significant. This behavior can have multiple reasons independent of exploiting FOL in the model (indeed, as shown in Table 3.6, the use of FOL rules overcomes the basic

Baseline	Accuracy						Diversity	Novelty	
	MAP	Precision	MAR	Recall	F1	nDCG	Gini	EFD	EPC
MultiVAE	0.6407	0.6389	0.3022	0.4997	0.5163	0.6803	0.2728	7.3677	0.6331
VSM	0.6388	0.6315	0.3011	0.4951	0.5107	0.6766	**0.2923***	**7.5134***	0.6320
CFGAN	0.6380	0.6341	0.2988	0.4948	0.5122	0.6766	0.2551	7.3709	0.6318
LightGCN	0.6351	0.6352	0.2976	0.4947	0.5127	0.6752	0.2102	7.3602	0.6307
NGCF	0.6298	0.6297	0.2960	0.4915	0.5084	0.6699	0.2295	7.4636	0.6305
KaHFM	0.6719	0.6424	0.3067	0.4864	0.5127	0.6943	0.0791	6.7468	0.6673
BPRMF	0.6711	0.6593	0.3137	0.5141	0.5333	0.7071	0.1879	6.9947	0.6547
PureSVD	0.6638	0.6364	0.3062	0.4878	0.5097	0.6894	0.1432	7.0166	0.6576
Slim	0.6352	0.5604	0.2724	0.4053	0.4360	0.6283	0.0868	7.1835	0.6620
ItemKnn	0.6211	0.5177	0.2741	0.3936	0.4111	0.6061	0.1028	7.3233	0.6911
UserKnn	0.6312	0.5270	0.2738	0.3906	0.4138	0.6141	0.0930	7.2967	**0.6998**
Attr.ItemKnn	0.5107	0.3731	0.2171	0.2887	0.2984	0.4720	0.0995	7.1571	0.6495
Attr.UserKnn	0.5963	0.4880	0.2465	0.3474	0.3759	0.5700	0.0853	6.9958	0.6799
EASE^R	**0.7096***	**0.6781**	**0.3351***	**0.5339***	**0.5502***	**0.7415***	0.1887	7.4077	0.6841
Our best	0.6413	0.6381	0.3009	0.4985	0.5157	0.6803	0.2787	7.4381	0.6362

TABLE 3.9: Comparisons to baselines in the book field (DBbook dataset). The best overall value is highlighted in bold, while the best baseline value is underlined. (*) means the differences are statistically significant.

configurations, whose performance would have been even worse). In the present study, it is very likely that the characteristics of the dataset, as shown in Table 2.2, show that only 1931 out of 6698 items in the dataset are mapped to the knowledge graph. Accordingly, most of the users and most of the items do not exploit the information coming either from descriptive properties or logical rules. Accordingly, it is not surprising that a knowledge-aware method such as the proposed method does not beat advanced data-driven approaches such as BPRMF, which does not exploit exogenous knowledge and only relies on the available items. However, further investigations will be conducted to better understand the behavior of the model on this dataset.

Finally, the results obtained in the movie domain (ML1M) are reported in Table 3.10. In this case, these results better illustrate the effectiveness of the approach, since the best configuration (the one injecting FOL rules with head coverage scores greater than 0.2, labeled as UIP + LHC, as reported in Table 3.7) achieves the best results for all accuracy metrics. The model also yields the best results in terms of novelty, with substantial gaps with respect to the baselines. All these differences are statistically significant with $p < 0.01$ (with the only exception of the MAR). Also in this case, EASE^R emerges as the best baseline, since only the best diversity is obtained by CFGAN. However, these results are generally not surprising. ML1M is a denser and larger dataset than the previous two, where a larger number of FOL rules are mined and exploited by the model. Accordingly, this naturally improves performance with respect to other methods; moreover, for the same reason, the impact of external knowledge is less pronounced, which improves methods based on matrix factorization and, more generally, collaborative filtering. **To conclude, the comparison with the state of the art indicates that the approach is particularly suitable and effective when the graph is sufficiently large and when a large amount of rules can be mined and extracted from the triples. In these scenarios, current state-of-the-art approaches can even be surpassed.**

Overall, the model outperformed almost all baselines across almost all accuracy metrics, with statistically significant improvements in most cases; good and significant improvements were also observed for non-accuracy metrics, such as diversity (Gini index) and novelty (EFD, EPC).

Discussion and Limitations Although these aspects of the model are favorable, some limitations are acknowledged and will be addressed in future work:

Baseline	Accuracy						Diversity	Novelty	
	MAP	Precision	MAR	Recall	F1	nDCG	Gini	EFD	EPC
MultiVAE	0.7586	0.7245	0.2743	0.4275	0.4515	0.7861	0.0713	6.6728	0.5385
VSM	0.7036	0.6729	0.2559	0.4041	0.4242	0.7294	0.1560	6.7085	0.6433
CFGAN	0.6311	0.6229	0.2404	0.3883	0.4024	0.6670	**0.2801***	6.3929	0.5751
LightGCN	0.5997	0.6016	0.2349	0.3835	0.3942	0.6412	0.1781	6.1745	0.5141
NGCF	0.6573	0.6377	0.2413	0.3895	0.4051	0.6873	0.2133	6.4227	0.5802
KaHFM	0.7616	0.7259	0.2717	0.4244	0.4492	0.7877	0.1237	6.9823	0.5356
BPRMF	0.7622	0.7291	0.2729	0.4264	0.4512	0.7894	0.1038	6.8912	0.6120
PureSVD	0.7683	0.7280	0.2750	0.4275	0.4518	0.7927	0.1040	6.9338	0.5677
Slim	0.7569	0.7241	0.2717	0.4256	0.4504	0.7842	0.2563	7.6629	0.5968
ItemKnn	0.8239	0.7767	0.2929	0.4487	0.4773	0.8472	0.0957	7.3958	0.5141
UserKnn	0.8182	0.7750	0.2888	0.4467	0.4758	0.8425	0.0959	7.3426	0.6047
Attr.ItemKnn	0.7322	0.6982	0.2630	0.4117	0.4341	0.7581	0.1693	7.0742	0.5807
Attr.UserKnn	0.7802	0.7456	0.2741	0.4313	0.4587	0.8056	0.1069	7.0611	0.5651
EASE^R	0.8328	0.7865	0.2936	0.4508	0.4811	0.8563	0.0995	7.5592	0.626
Our best	**0.8404***	**0.7994***	**0.2996**	**0.4601***	**0.4909***	**0.8675***	0.1481	**8.0541***	**0.6704***

TABLE 3.10: Comparisons to baselines in the movie field (ML1M dataset). The best overall value is highlighted in bold, while the best baseline value is underlined. (*) means the differences are statistically significant with $p < 0.05$.

- *Choice of the embedding learning model*: Although TransE obtained good performance in the task of graph embedding for the recommendation, there are more recent and sophisticated methods to embed graph-derived information, including GNNs and GCNs; exploiting again the First-Order Logic as a unified framework for both triples and FOL rules could be a good strategy to combine more effective graph embedding models and maintaining the injection of FOL rules, to obtain again *neuro-symbolic graph embeddings*.
- *Identification of the FOL rules*: in this work, FOL rules were mined from the KG in this work. Another option could be adding rules provided by domain experts to discover new knowledge that is not encoded in the original KG as *background knowledge*.
- *A weighting mechanism for FOL rules*: in this work, all FOL rules were treated with the same importance, even when they had different confidence values. An extension of this work could be a *weighting mechanism* for the rules, aiming to lower the impact of rules with low confidence scores while amplifying the impact of rules with higher confidence scores.
- *An automated FOL rules selection strategy*: in this work, eight different subsets of FOL rules were considered, related to eight different heuristics. Although standard confidence appears to be the best metric for selecting the most prominent heuristic, an automated mechanism to select the best threshold value is currently lacking and will be addressed in future work.
- *A user study*: in this work, *in-vitro* experiments were performed; conducting *in-vivo* experiments to evaluate how the model affects the recommendation lists for real users is also necessary.
- *More sophisticated methodologies for providing recommendations*: simple concatenation was used to combine the embeddings related to users and items; more sophisticated methods could be exploited, such as the *attention mechanisms*. In addition, although beyond the scope of this work, information from other sources (e.g., plain text) could further improve the performance of the recommender system.

3.1.5 Take-Home Messages

The featured publications [173] and [174] present a model aiming at *(i):* mining FOL rules from a KG; *(ii):* learning graph embedding by injecting the knowledge provided by the FOL rules; *(iii):* providing users with effective recommendations.

The methodology was tested on three datasets related to different domains (music, books, and movies), and extensive experiments were carried out to assess the effectiveness of the model.

This work represents a first step in the adoption of neuro-symbolic AI in the field of graph embedding for recommendations, which is poorly investigated in the KARS literature. More research should be conducted in this field.

Main Findings

- **RQ3.1.1**: The introduction of FOL rules has a decisive impact on both item representation and the generation of recommendation lists
- **RQ3.1.2**: The most promising subsets of FOL rules to be injected are based on the standard confidence and coverage scores.
- **RQ3.1.3**: The approach outperformed state-of-the-art baselines, in particular in scenarios in which a large amount of rules can be mined and injected.

The results discussed in this section have been published at the ACM RecSys 2022 Conference, in the LBR Track (Giuseppe Spillo et al. "Knowledge-aware Recommendations Based on Neuro-Symbolic Graph Embeddings and First-Order Logical Rules". In: *Proceedings of the 16th ACM Conference on Recommender Systems*. 2022, pp. 616–621) and presented as a poster, and subsequently extended and published in the Journal of User Modeling and User-Adapted Interaction (Giuseppe Spillo et al. "Recommender systems based on neuro-symbolic knowledge graph embeddings encoding first-order logic rules". In: *User Modeling and User-Adapted Interaction* 34.5 (2024), pp. 2039–2083) during the PhD.

3.2 *GAL-KARS*: Graph Augmentation through LLMs for Knowledge-aware Recommendations

3.2.1 Introduction

The previous section discussed a methodology for integrating FOL rules into the KG embedding processes to improve the representations. In this Section, an alternative approach to improve the embeddings is discuss, this time by exploiting LLMs.

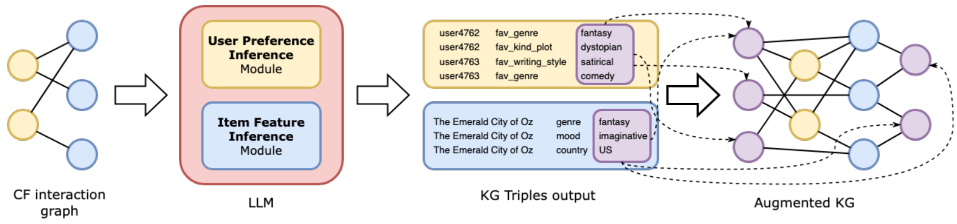
In particular, the featured publication [171] introduces **GAL-KARS** (Graph Augmentation with LLMs for Knowledge-Aware Recommender Systems), a novel recommendation framework that addresses key limitations in current KARS literature by leveraging the reasoning and generative capabilities of LLMs to perform graph augmentation.

There are two main problems in the current KARS literature relying on KGs. First, KGs are often incomplete or noisy, failing to capture nuanced and descriptive features of items such as emotions, themes, or stylistic elements This issue is known as the knowledge scarcity problem [37]. Second, they typically limit user preferences to item-level interactions, which restricts the model's ability to reflect higher-level or more abstract interests (e.g., a preference for a particular genre or creative style). These issues lead to suboptimal representations of users and items, and thus to lower recommendation quality.

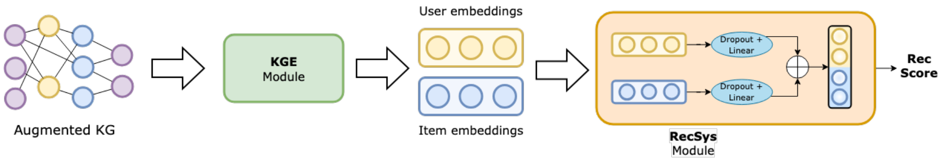
To overcome these challenges, the GAL-KARS framework proposes a twofold augmentation strategy using LLMs: (1) item representations are enriched by generating additional semantic features that are not originally present in the KG, and (2) user representations are enhanced by reasoning over the items liked by users and by inferring more abstract or generalized preferences. The core idea is that *LLM prompting* can be used to query the large body of (textual) knowledge that is used to train LLMs and extract new triples that can augment the original KG to better describe both the users and the items. The augmented KG is then encoded using a knowledge graph embedding model, which learns vector representations of users and items. These embeddings serve as input to a neural recommendation module, which produces personalized recommendation lists.

The key contribution of GAL-KARS lies in its ability to integrate further features describing the items and further nodes describing the preferences of the users, obtained by reasoning over the items they like, thereby producing a more complete and semantically rich representations. Through extensive experiments on benchmark datasets, including ablation studies and comparisons with state-of-the-art KARS models, GAL-KARS graph augmentation via LLMs leads to significant improvements not only in predictive accuracy but also in the diversity and novelty of the recommended items.

The results discussed in this section have been published at the ACM UMAP 2025 Conference, at the Main Track as a full paper (Giuseppe Spillo et al. "GAL-KARS: Exploiting LLMs for Graph Augmentation in Knowledge-Aware Recommender


CF interaction graph
User Preference Inference Module
Item Feature Inference Module
LLM
user4762 fav_genre fantasy
user4762 fav_kind_plot dystopian
user4763 fav_writing_style satirical
user4763 fav_genre comedy
The Emerald City of Oz genre fantasy
The Emerald City of Oz mood imaginative
The Emerald City of Oz country US
KG Triples output
Augmented KG

(A) Starting from the CF graph, we infer new triples describing users and items, and we incorporate them to obtain our augmented KG.



(B) Based on the augmented graph, user and item embeddings are learned, and they are used to provide users with recommendations.

FIGURE 3.6: Our framework workflow, in which we first *(a)* infer new triples to build an augmented KG, from which we *(b)* learn the embeddings and provide recommendations.

Systems". In: *Proceedings of the 33rd ACM Conference on User Modeling, Adaptation and Personalization*. 2025, pp. 73–82).

3.2.2 Methodology

The overall workflow of this methodology is depicted in Figures 3.6a and 3.6b. In particular, starting from a graph $\mathcal{G}$ encoding the preferences of the users in $\mathcal{U}$ over the items in $\mathcal{I}$ (see Section 3), LLMs are first exploited to implement the *Item Feature Inference* and the *User Preference Inference* modules. The former aims at obtaining a set of triples that describe the items, and the latter aims at encoding more detailed preferences of the users by reasoning over the list of the items they like. In both cases, the output is represented by a set of KG triples in the form *(user/item, relation, entity)*. The merge of all these triples constitutes the augmented KG used in KARS.

After the augmentation, a *Knowledge Graph Encoder* module is employed to learn user and item embeddings starting from the augmented KG. Finally, the resulting embeddings are passed to a *RecSys*, which provides recommendations to users. The framework modules are described in detail in the following.

LLM Input	
System Prompt	Considering the book name in the user prompt, write a list of properties for each of these elements: the **name of the author**. The **subject** of the book. The **mood** of the book; the **topic** of the book. The **genre** of the book. The **writing style** of the book; the **kind of plot** of the book. The **kind of book**. The **country** in which the book has been written. Infer this information for the given book. Furthermore, strictly follow this format in the output without adding anything else: <genres: <genre_1>, ..., <genre_n>. plot: <plot_1>, ..., <plot_n>.>and so on, for each attribute. Make sure that every <genre_1>, ..., <genre_n>, <plot_1>, ..., <plot_n>etc does not contain multiple values (like "polar and thriller"), but rather separates them with commas ("polar, thriller"). If you are unsure about something, avoid providing an answer.
User Prompt	The book you have to infer information is: **Batman: The Killing Joke**
LLM Output	
Model output	**plot**: [murder, mystery, suspance]. **author**: [Alan Moore]. **subject**: [Batman]. **mood**: [dark, grim]. **topic**: [crime, justice]. **genre**: [horror, triller, detective fiction]. **writing style**: [dark, grim]. **kind of plot**: [suspance]. **kind of book**: [graphic novel]. **country**: [U.S].
KG triples	
	(Batman: The Killing Joke, subject, Batman), (Batman: The Killing Joke, author, Alan Moore), (Batman: The Killing Joke, plot, murder), (Batman: The Killing Joke, genre, thriller), (Batman: The Killing Joke, topic, crime), (Batman: The Killing Joke, kind of book, graphic novel)

TABLE 3.11: Real example of *zero-shot* prompting to the Item Feature Inference Module, with a subset of the obtained KG triples

Item Feature Inference Module To infer item features in the form of KG triples, we prompt an LLM through the *zero-shot* prompting paradigm. In particular, we design a specific prompt that, given the name of an item, returns a set of triples that describe its characteristics. An example of such a prompt, specifically designed for the *book* domain, is reported in Table 3.11.

Generally speaking, the prompt is composed of three parts: the *System Prompt*, designed to provide the LLM with information about the task to accomplish. The *User Prompt*, which provides the LLM information about the specific instance the task has to be performed on, and the *Model Output*, which represents the answer of the LLM.

In this case, the System Prompt asks the LLM to generate a list of relevant features that characterize the item in a particular domain. Moreover, the format of the output is specified to the LLM. Regarding the choice of the item features, it is worth emphasizing that the LLM is asked to generate both features that are typically encoded in popular KGs (*e.g.*, the name of the author, the topic or the genre of the book), as well as novel features, such as the *writing style* or the *mood*, which are typically not encoded.

In our vision, such a detailed prompt has a twofold advantage: on the one hand, it can exploit the LLM to complete the knowledge encoded in the original KG, thus tackling data sparsity and missing feature issues. On the other hand, it can incorporate new features, based on the information that is pre-trained in the LLM, that enrich the knowledge available in the data model. Table 3.11 reports the real output of the graph augmentation for the book *Batman: The Killing Joke* based on the prompt we designed. Based on the output, through simple transformation rules, we obtain a set of KG triples. As previously explained, this process is repeated over all the items $i \in \mathcal{I}$, and the merge of all the triples leads to our graph $\mathcal{KG}_{\mathcal{I}}$, which encodes all the item features generated by the LLM in the graph augmentation process.

User Preference Inference Module Next, a similar process is designed to infer user preferences in the form of KG triples. To this end, the LLM is prompted in a *zero-shot* fashion such that, given the list of items the user likes, the LLM returns a set of triples encoding their preferences. An example of the prompt applied to the book domain is reported in Table 3.12.

While the structure of the prompt is very similar, it is worth emphasizing that it accomplishes a different task. Indeed, rather than incorporating further knowledge about the items, the goal of this part of the augmentation process is to introduce new *edges* in the original KG. In this framework, these edges - connecting users' nodes with the entities describing their preferences - may improve the quality of the underlying data model, thus improving the performance in a downstream recommendation task as well.

Regarding the choice of elements included in the prompt, it is noted that features encoded in the original KG (*i.e.,* preferred genre or authors) were mixed with more fine-grained characteristics, such as the *mood* of movies liked by the user. Also in this case, the prompt is generated for all the users $u \in U$ based on the items they like, and the triples returned by the LLM are merged in the graph $\mathcal{KG}_{\mathcal{U}}$, which encodes all the user features obtained through the graph augmentation process.

Knowledge Graph Encoder Module After querying the LLMs, an augmented graph is built that merges $\mathcal{G}$ and $\mathcal{KG}$ with either $\mathcal{KG}_{\mathcal{I}}$ or $\mathcal{KG}_{\mathcal{U}}$, or both[5]. Based on this data model, a Knowledge Graph Encoder is employed to learn embeddings representing users and items in the augmented graph. CompGCN [202] is used as a Graph Encoder, as discussed in Chapter 2. This choice is justified by the competitive performance shown by other models exploiting GCNs for recommendation tasks [70, 167, 211, 169].

[5]Post-processing has been applied as well to unify the different labels associated with the same concepts (e.g., the labels *genres* and *genre* were unified as *genre*).

LLM Input	
System Prompt	Considering the books given in the user prompt, write a list of the user's favorite: **kind of plot**; **genre**; **writing style**; **subjects**; **topics**; **kind of books** (novel, essay, poetry). Infer this information from the books the user likes. Furthermore, strictly follow this format in the output without adding anything else: <Favourite genre: <genre_1>, <genre_2>. Favourite kind of plot: <plot_1>, ..., <plot_n>.>and so on. Make sure that every <genre_1>, ..., <genre_n>, ..., <plot_1>, <plot_n>etc. does not contain multiple values (like "polar and thriller"), but rather separates them with commas ("polar, thriller"). If you are unsure about something, avoid providing an answer.
User Prompt	My favorite books are The Language of God: A Scientist Presents Evidence for Belief; The Magician's Nephew; A Morbid Taste for Bones; An Excellent Mystery; The Hunt for Red October; Red Storm Rising; Executive Orders.
LLM Output	
Model output	**genre**: [fiction, mystery, science fiction, thriller, war novel], **kind of plot**: [detective story, political thriller, religious thriller, spy story, war story], **writing style**: [clear, concise, descriptive, narrative, simple], **kind of books**: [essay, novel] **subjects**: [evolution, creationism, espionage, history, magic, medieval times, politics, religion, war], **topics**: [creationism, espionage, evolution, history, magic, medieval times, politics, religion, war],
KG triples	
(user_1, fav_genre, fiction), (user_1, fav_genre, mystery), (user_1, fav_genre, science_fiction) (user_1, fav_plot, detective_story), (user_1, fav_plot, politica_thriller), (user_1, fav_plot, spy_story)	

TABLE 3.12: Real example of *zero-shot* prompting to the User Preference Inference Module, with a subset of the obtained KG triples

Following the procedure already discussed, at the end of the process, an embedding $\vec{e_u}$ for each user $u \in \mathcal{U}$ and an embedding $\vec{e_i}$ for each item $i \in \mathcal{I}$ are learned.

RecSys Module Based on $\vec{e_u}$, $\vec{e_i}$, which represent the embeddings of user u and item i learned in the previous step, the goal of the recommendation module is to predict $rec_{u,i}$, that is the predicted interest of user u in item i. It corresponds to $\widetilde{Y}$ introduced previously.

To this end, dropout (with keep probability set to p) is applied to $\vec{e_u}$ and $\vec{e_i}$. Next, the embeddings are projected onto different smaller layers using the ReLU activation function[6]:

$$\vec{h_u} = ReLU\left(W_u * Dropout\left(\vec{e_u}, p\right) + b_u\right) \tag{3.3}$$

$$\vec{h_i} = ReLU\left(W_i * Dropout\left(\vec{e_i}, p\right) + b_i\right) \tag{3.4}$$

W_u and W_i are trainable parameters, b_u and b_i are the biases. Then, h_u and h_i are fused through concatenation, and the resulting fused embedding is then projected onto smaller linear layers with ReLU (analogously to Equations 3.3 and 3.4). This aims at obtaining a unique embedding $\vec{h}$:

$$\vec{h} = ReLU\left(W_{cat} * \left[\vec{h_u} \mid\mid \vec{h_i}\right] + b_{cat}\right) \tag{3.5}$$

[6]To avoid repetitions, we report the formulae of a single layer since the other layers follow the same methodology.

where W_{cat} is a trainable parameter, b_{cat} is a bias, and $||$ indicates the concatenation between the embeddings.

Lastly, the final embedding is used $\vec{h}$ to compute the recommendation score as follows:

$$rec_{score} = \sigma\left(W_h * \vec{h} + b_h\right) \tag{3.6}$$

where W_h is a trainable parameter, b_h is a bias, and σ is the sigmoid activation function that returns a recommendation score $score_{u,i} \in [0, 1]$.

The model is trained on a subset of the ratings encoded in $\mathcal{Y}$ (the interaction matrix introduced in Section 3.2.2), and the *binary cross-entropy* is used as the loss function. At inference, the scores for all items in the test set are computed. Items are ranked according to the predicted scores, and the *top-k* are included in the recommendation list.

Design Discussion Before detailing the experiments, the choices and potential limitations of the approach are discussed.

First, potential issues related to the well-known hallucination problem [76], typical of LLMs, are introduced. In this work, hallucination may lead to the generation of triples that are not real or not correct, potentially introducing noise in the graph augmentation process. This risk is inherent to LLMs in most downstream applications and can be managed by applying several strategies [76]. To mitigate this risk, prompts were designed by incorporating the sentence: *"If you are unsure about something, avoid providing an answer"*, following [252, 194]. Although this approach is simple, as shown in the quantitative analysis, this prompt design led the LLM to avoid generating triples for some items in the datasets (see Table 3.13, discussed later). Extensive manual checks on dozens of randomly sampled subsets of user/item triples showed no hallucinated knowledge. More sophisticated strategies to tackle hallucinations, including retrieval-augmented generation [57], will be considered as future work.

A second potential concern regards the opposite scenario, *i.e.*, the LLM generates knowledge that is *already* encoded in the original KG. This may happen for very common triples, such as those describing the author of a book or the director of a movie. Generally, this is not problematic in itself, since it merely leads to the injection of unnecessary knowledge. This choice also has a potential positive impact, since it allows evaluation of the extent to which LLMs can be used to generate from scratch KGs that support a recommendation task. As argued in [74], building a KG is a time-intensive task that requires substantial (manual) effort, and the use of LLMs represents a suitable alternative to speed up this task [261]. As shown in Table 3.13, the KGs constructed through LLMs exhibit better item coverage relative to SOTA KGs (e.g., DBpedia), confirming the potential of this strategy.

The design of the prompt is a distinctive trait of this work. Indeed, together with properties available in the original KGs, this work aims to generate additional characteristics generally not available, such as the *writing style* of an author or the main *mood* of a movie. Further extensions, *i.e.,* encoding the emotions triggered by a movie or other subjective features, will be investigated as future work.

Moreover, although running these prompts for each user and item may be computationally expensive in large datasets, this process is performed only once, in advance, before the Knowledge Graph Encoder or the RecSys Module are used. Additionally, if new users or items are introduced, the same prompts may be run only on the newly added nodes, rather than reprocessing the entire dataset.

Since the recommendation task is not performed via an LLM but the LLM is used to augment the KG connected to the dataset catalog, the problem of data leakage [260] (which occurs when an LLM is asked to perform a task on a benchmark dataset it may have already *seen* during training) is mitigated if not entirely absent.

Finally, this study highlights a modular design in which the graph augmentation process is independent of the knowledge graph embedding and the recommendation. Indeed, unlike an end-to-end model in which graph augmentation is jointly carried out together with the recommendation, this design enables the evaluation of the augmentation with different recommendation algorithms and embedding techniques, as well. This analysis is left as future work.

3.2.3 Problem Formulation

In this section, the problem formulation introduced in Section 2.3 is extended only with the information necessary to better frame this work, while the remainder of the formalization remains unchanged.

Graph Augmentation LLMs are employed for graph augmentation, *i.e.,* to infer triples encoding item features and user preferences. To obtain *item features*, for each item $i \in \mathcal{I}$, a prompt p_i is designed and used to query an LLM based on the item's name. The output of the prompt is a new set of triples $\mathcal{T}_i$ describing item i, such as *(Nixon (movie), theme, political corruption)*. Formally: $\mathcal{T}_i = LLM(p_i)$. These triples may overlap with those in $\mathcal{KG}$ or be completely novel, *i.e.*, generated based on the knowledge encoded in the LLM. At the end of the process, for each $i \in \mathcal{I}$, all sets of triples $\mathcal{T}_i$ are merged to form a new KG describing items that are completely generated by an LLM. Let $\mathcal{KG}_{\mathcal{I}}$ be this graph.

Next, a similar process is followed to create an augmented KG describing the preferences of the users. In this case, for each user $u \in \mathcal{U}$, a prompt p_u including a list of the items liked by the user is designed, and an LLM is queried to obtain a new set of triples describing the user's preferences. For example, *(user83, fav_setting, post-apocalyptic)*. Similarly, the triples are merged to obtain a new KG describing the users. Let $\mathcal{KG}_{\mathcal{U}}$ be this graph.

Note that $\mathcal{KG}$ and $\mathcal{KG}_{\mathcal{I}}$ share the same item nodes with $\mathcal{G}$, while $\mathcal{KG}_{\mathcal{U}}$ shares the user nodes with $\mathcal{G}$.

Finally, a final augmented KG is created by merging the original graphs $\mathcal{G}$ and $\mathcal{KG}$ with either $\mathcal{KG}_{\mathcal{I}}$, or $\mathcal{KG}_{\mathcal{U}}$, or both. This graph is the one obtained at the end of the process.

3.2.4 Experimental Evaluation

This study aims to address the following research questions (RQs):

- **RQ3.2.1 - Quantitative analysis:** How does the graph augmentation affect the number of nodes and triples encoded in the KGs?
- **RQ3.2.2 - Ablation studies:** How does each KG configuration contribute to the overall performance of the model?
- **RQ3.3.3 - Comparison to SOTA:** How does the approach perform with respect to baselines?

Datasets and Knowledge Graphs Experiments were conducted in a *movie and book recommendation* scenario. In particular, the ML1M and DBbook datasets have been considered. More details about these datasets can be found in Section 2.3, where the problem has been formalized. Statistics about the KGs are reported in Table 3.13. The datasets used in this work, together with all augmented variants of the KGs[7], are released to guarantee the reproducibility of the results.

[7] `https://github.com/swapUniba/gal-kars`

KG	Users	Items	Entities	Rels.	Triples	Trip./user	Trip./items
			DBBOOK				
$\mathcal{KG}$	0	1,970	5,200	37	16,589	-	8.4
$\mathcal{KG}_\mathcal{I}$	0	6,392	7,462	11	72,905	-	11.4
$\mathcal{KG}_\mathcal{U}$	4,981	0	8,500	7	92,924	18.7	-
$\mathcal{KG}^+$	4,981	6,392	19,396	54	182,418	36.6	28.5
			ML1M				
$\mathcal{KG}$	0	2,823	13,738	11	44,682	-	15.8
$\mathcal{KG}_\mathcal{I}$	0	2,799	2,296	10	47,027	-	16.8
$\mathcal{KG}_\mathcal{U}$	5,463	0	1,536	13	162,167	29.7	-
$\mathcal{KG}^+$	5,463	3,133	16,766	25	253,876	46.5	81.1

TABLE 3.13: Knowledge graph statistics for DBBOOK and ML1M.

Protocol To run graph augmentation, prompts were populated based on the data available in the training set. In particular, positive ratings were used to infer user preferences and obtain $\mathcal{KG}_\mathcal{U}$. Similarly, items occurring in the training set were used to infer item features and obtain $\mathcal{KG}_\mathcal{I}$. User preferences in the training set are used to learn the embeddings and train the recommendation models. Finally, to obtain the *top-k* items, the interest score for all user-item pairs in the test set is predicted, the items are ranked according to the predicted scores, and the first k are returned. The values of k are set to 5 and 10. Due to space constraints, results for $k = 5$ are shown only. Full results are available in the repository.

Representation Learning Given the augmented graph in one of its variants, a Knowledge Graph Encoder is employed to learn latent representations (embeddings) for users and items. Let $\vec{e_u}$ and $\vec{e_i}$ be the embeddings learned from the augmented KG related to user u and item i, respectively. These embeddings are then used to train a neural network that carries out the recommendation task, i.e., the extent to which the user u will interact with item i is predicted. Let $\mathcal{KG}_{\text{aug}}$ denote any augmented KG and θ denote the model's parameters.

Ablation studies To assess the effectiveness of the methodology, the framework was evaluated on varying underlying graphs. In particular, eight different data models were considered for the performance of the recommendations: *(1)* Collaborative graph, obtained by learning user and item embeddings relying solely on the information encoded in $\mathcal{G}$; *(2)* Original $\mathcal{KG}$ without augmentation, merging

$\mathcal{G}$ and the information available in DBpedia; *(3)* LLM-augmented graph for the items, corresponding to the graph $\mathcal{KG}_\mathcal{I}$ introduced earlier; *(4)* LLM-augmented graph of the users, corresponding to the graph $\mathcal{KG}_\mathcal{U}$ introduced earlier; *(5)* LLM graph, corresponding to the merge of $\mathcal{KG}_\mathcal{I}$ and $\mathcal{KG}_\mathcal{U}$, evaluating the effectiveness of a KG built from scratch through LLMs; *(6)* item-augmented graph $\mathcal{KG}_\mathcal{I}^+$, obtained as the merge of $\mathcal{KG}$ and $\mathcal{KG}_\mathcal{I}$; *(7)* user-augmented graph $\mathcal{KG}_\mathcal{U}^+$, obtained as the merge of $\mathcal{KG}$ and $\mathcal{KG}_\mathcal{U}$; *(8)* complete augmented graph, labeled as $\mathcal{KG}^+$, merging all data sources. A comparative analysis among these sources will be discussed next.

Baselines Several baselines belonging to different model groups were considered, and are reported in the tables. Their description is discussed in Section 2.3, and is not reported here to avoid repetitions.

Evaluation Metrics To evaluate this approach, Precision, Recall, F1, nDCG as *accuracy* metrics, the Gini Index and the EPC for *diversity* and *novelty*, respectively, and the Average Percentage of Long Tail items (APLT) to assess the bias of recommendations have been considered.

DATA	Users	Items	Entities	Rels.	Triples	Trip./user	Trip./items
			DBBOOK				
$\mathcal{KG}$	0	1,970	5,200	37	16,589	-	8.4
$\mathcal{KG}_\mathcal{I}$	0	6,392	7,462	11	72,905	-	11.4
$\mathcal{KG}_\mathcal{U}$	4,981	0	8,500	7	92,924	18.7	-
$\mathcal{KG}^+$	4,981	6,392	19,396	54	182,418	36.6	28.5
			ML1M				
$\mathcal{KG}$	0	2,823	13,738	11	44,682	-	15.8
$\mathcal{KG}_\mathcal{I}$	0	2,799	2,296	10	47,027	-	16.8
$\mathcal{KG}_\mathcal{U}$	5,463	0	1,536	13	162,167	29.7	-
$\mathcal{KG}^+$	5,463	3,133	16,766	25	253,876	46.5	81.1

TABLE 3.14: Statistics about side information in different KGs for DBBOOK and for ML1M.

3.2.5 Results Discussion

RQ3.2.1: Quantitative Analysis. Table 3.14 reports statistics regarding the different variants of KGs introduced previously. In particular, the original $\mathcal{KG}$, i.e., DBpedia, is compared to the augmented graphs $\mathcal{KG}_\mathcal{I}$, $\mathcal{KG}_\mathcal{U}$ and $\mathcal{KG}^+$ in terms of covered items, number of entities, and relations. The numbers in Table 3.14 do not include collaborative data, i.e., user-item interactions. This is done to simplify the analyses. The goal is to compare the amount of exogenous knowledge encoded in the original graph and in the augmented variants.

First, the original $\mathcal{KG}$ does not cover all items in the catalog. This is particularly evident for DBBOOK, where only 30% (1,970 out of 6,617) of the items are covered. In this scenario, the results indicate that LLMs can fill this gap since the item coverage of $\mathcal{KG}_{\mathcal{I}}$ is significantly higher and reaches 96.5%. This constitutes the first important finding, indicating that LLMs can be useful in providing KARS with more knowledge about the items. This also holds for the number of overall triples, which is more than fourfold. An analysis of the same data for ML1M reveals that the number of covered items and total triples are comparable, while the number of encoded entities is significantly lower. Table 5 provides a quantitative comparison of the triples available in the original $\mathcal{KG}$ to those generated by the item augmentation process for an example item[8]. Although some features overlap, the LLM is able to generate some triples that are not encoded in the original data model. The next experiment will investigate whether the different information encoded in $\mathcal{KG}$ and $\mathcal{KG}_{\mathcal{I}}$ leads to different performances or not.

Next, for $\mathcal{KG}_{\mathcal{U}}$, a significant number of triples for the users can also be inferred by LLMs. In this case, coverage ranges between 88% for DBBOOK and 90% for ML1M. This indicates that the graph augmentation strategy generates and encodes a relevant number of triples not available in the original data model. Indeed, as indicated in Table 4, no side information about the users is encoded in $\mathcal{KG}$.

Finally, the augmented graph $\mathcal{KG}^{+}$ is the largest, since it includes augmented triples for both users and items. The overall number of triples has increased significantly (more than 10x for DBBOOK and more than 6x for ML1M). The next experiment will assess whether such an increase in size also leads to an improvement in performance.

Regarding item coverage, an interesting observation can be made for ML1M. Indeed, the item coverage of $\mathcal{KG}^{+}$ is higher than that of both $\mathcal{KG}$ and $\mathcal{KG}_{\mathcal{I}}$. This means that, while some items described in $\mathcal{KG}$ are not covered by $\mathcal{KG}_{\mathcal{I}}$ (and vice versa), $\mathcal{KG}^{+}$ covers items described by both sources. In other words, the graph augmentation process allows incorporation of further knowledge and coverage

[8]Due to space reasons, we only report one item. The complete set of triples is available in our repository.

DBPEDIA KG (ORIGINAL GRAPH)		LLM ITEM AUGMENTATION	
relation	*entity*	*relation*	*entity*
publisher	DC Comics	author	frank miller
writers	Frank Miller (comics)	topic	batman
subject	Elseworlds titles	writing style	comic book
subject	Comics by Frank Miller	genre	polar
subject	Justice League storylines	genre	thriller
subject	Sequel comics	plot	action
subject	DC Comics	plot	adventure
subject	Batman	plot	science fiction
subject	American comics	subject	batman
subject	Batman titles	mood	dark
		country	united states
		kind book	fiction

TABLE 3.15: A comparison between the triples encoded for the book 'The Dark Knight Strikes Again'. We omit the head entity, as it is the same for all the triples (the item itself).

of a higher number of items. This further confirms the usefulness of employing LLMs for graph augmentation goals. **To conclude, RQ3.2.1 is answered by noting that the graph augmentation increases item coverage and provides the data model with new and interesting features that were not included in the original data model. The effectiveness of the augmented KGs in a downstream recommendation task will be evaluated next.**

RQ3.2.2: Ablation Study. To answer RQ3.2.2, Tables 3.16 and 3.17 report the results of the ablation study conducted on the information encoded in the KG for DBbook and ML1M, respectively. As stated in the previous sections, all possible combinations obtainable by merging $\mathcal{KG}$, $\mathcal{KG}_{\mathcal{I}}$, and $\mathcal{KG}_{\mathcal{U}}$ were considered. In the table, the configuration that exploits only user-item interactions without any side information is indicated as $\emptyset$. First, the best overall results on DBBOOK are achieved by $\mathcal{KG}_{\mathcal{U}}^{+}$, which surpasses the performance of the original $\mathcal{KG}$ by a statistically significant margin. This indicates that the use of LLMs to augment the graph and inject edges modeling user preferences leads to the best result. This is probably due to the high sparsity of the DBBOOK dataset, which makes encoding more detailed user preferences and increasing the connectivity in the graph useful. Conversely, analysis of the results on ML1M shows that $\mathcal{KG}_{\mathcal{I}}^{+}$ yields the best accuracy, with a statistically significant gap again. In this case, the lower sparsity suggests that encoding more user preferences is less necessary. This is also

KG	Precision	Recall	F1	NDCG	Gini	EPC	APLT
$\emptyset$	0.6858	0.5450	0.5590	0.8781	**0.6458**	0.6162	**0.2279***
$\mathcal{KG}$	0.7006	0.5575	0.5720	0.8912	0.6641	0.6318	0.2091
$\mathcal{KG}_{\mathcal{I}}$	0.6976	0.5547	0.5688	**0.8958**	0.6815	0.6313	0.1877
$\mathcal{KG}_{\mathcal{U}}$	0.6996	0.5557	0.5704	0.8927	0.6792	0.6315	0.1892
$\mathcal{KG}_{\mathcal{IU}}$	0.6992	0.5547	0.5699	0.8932	0.6797	0.6307	0.1902
$\mathcal{KG}^+_{\mathcal{I}}$	*0.7028*	*0.5617*	*0.5745*	0.8905	0.6755	*0.6342*	0.1958
$\mathcal{KG}^+_{\mathcal{U}}$	**0.7043***	**0.5619**	**0.5755**	*0.8950*	*0.6640*	**0.6364***	*0.2108*
$\mathcal{KG}^+$	0.6931	0.5498	0.5646	0.8881	0.6844	0.6246	0.1861

TABLE 3.16: Ablation results on DBBOOK. Best results are in **bold**, second-best in *italic*; * denotes significance ($p = 0.05$) vs. $\mathcal{KG}$.

confirmed by the good results obtained by using the user-item interactions alone (labeled as $\emptyset$), which yielded the second-best result. However, it is interesting to note that, even in this scenario, the generation of further features describing the items can provide the model with useful knowledge. Regarding the performance of $\mathcal{KG}^+$, it should be noted that the injection of both kinds of information simultaneously rarely increases the model's performance. This is likely due to the large number of entities and relations added in this variant of the data model, which are likely to introduce some noise in the encoding process. Further analyses will be carried out to analyze this behavior. Regarding beyond-accuracy metrics, the use of graph augmentation also increases the novelty of the recommendations, particularly for DBBOOK. Moreover, the effectiveness of the KGs generated from scratch by the LLM (i.e., $\mathcal{KG}_{\mathcal{I}}$ and $\mathcal{KG}_{\mathcal{U}}$) is comparable to that of the original KG for both datasets, since no statistical gaps emerge from the experiments. This is another interesting outcome, confirming the usefulness of exploiting LLMs for knowledge generation. **To conclude, the results confirm the effectiveness of the graph augmentation pipeline. While the best-performing configuration appears to depend on dataset characteristics, the injection of further knowledge always increases the model's performance.**

RQ3.2.3: Comparison to SOTA In Table 3.18 and 3.19, the comparison between the best results and the baselines introduced previously for DBbook and ML1M, respectively, is reported. For fairness, all KARS algorithms were trained using the

KG	Precision	Recall	F1	NDCG	Gini	EPC	APLT
$\emptyset$	0.8022	0.4574	0.4938	*0.9438*	0.7504	*0.6642*	*0.0803*
$\mathcal{KG}$	0.7965	0.4550	0.4907	0.9400	0.7576	0.6624	0.0782
$\mathcal{KG}_{\mathcal{I}}$	0.7995	0.4569	0.4926	0.9421	**0.7261**	**0.6761***	**0.0897***
$\mathcal{KG}_{\mathcal{U}}$	*0.8029*	*0.4578*	*0.4942*	0.9435	0.7725	0.6539	0.0707
$\mathcal{KG}_{\mathcal{IU}}$	0.7919	0.4534	0.4882	0.9364	0.7514	0.6558	0.0734
$\mathcal{KG}_{\mathcal{I}}^{+}$	**0.8035***	**0.4585***	**0.4947***	**0.9440***	*0.7489*	0.6629	0.0701
$\mathcal{KG}_{\mathcal{U}}^{+}$	0.7990	0.4557	0.4919	0.9421	0.7725	0.6512	0.0605
$\mathcal{KG}^{+}$	0.8007	0.4570	0.4931	0.9423	0.7795	0.6511	0.0663

TABLE 3.17: Ablation results on ML1M. Best results are in **bold**, second-best in *italic*; * denotes significance ($p = 0.05$) vs. $\mathcal{KG}$.

augmented graph $\mathcal{KG}_{\mathcal{I}}^{+}$ to ensure the same data model as that used by the framework. For both datasets, the proposed approach surpasses the baselines in terms of accuracy and novelty metrics, with statistically significant differences. For DBBOOK, CF methods obtain the best results, in particular LightGCN and ItemKNN. The latter also achieves the best results in terms of diversity and the percentage of long-tail items. For ML1M, EASE and MultiVAE achieved the best accuracy, while NeuMF achieved the best diversity and long-tail item coverage. Regardless, the results show consistent behavior across the datasets, indicating that the proposed approach can significantly outperform state-of-the-art baselines. **To conclude, the results indicate that the framework exploiting graph augmentation to improve representations of users and items in KARS improves the current SOTA in terms of accuracy and novelty of the recommendations.**

3.2.6 Take-Home Messages

In this work [171], a general methodology for graph augmentation by using LLM prompting to infer missing and additional knowledge to improve KARS's performance is proposed. In particular, the approach focused on popular KGs (DBpedia) augmented with item features and user preferences inferred by LLMs. Extensive experiments, including ablation studies and baseline comparisons, confirmed the intuitions and the validity of the proposed approach.

Limitations related to the approach are acknowledged, particularly those arising from the intrinsic characteristics of LLMs, such as hallucinations. Accordingly, prompts were designed to mitigate such problems, and it is believed that this has

Model	Prec@5	Rec@5	F1@5	NDCG@5	Gini@5	EPC@5	APLT@5
ItemKNN	0.6742	*0.5359*	*0.5503*	0.8612	**0.6332**	0.5974	**0.2285***
BPR	0.6726	0.5231	0.5436	0.8699	0.7274	0.5989	0.0955
EASE	0.6548	0.5099	0.5290	0.8614	*0.6450*	0.5861	0.2009
MultiVAE	0.6738	0.5224	0.5443	0.8724	0.7515	0.5984	0.0753
NeuMF	0.6608	0.5137	0.5335	0.8626	0.6921	0.5890	0.1366
SLIMElastic	0.6718	0.5274	0.5449	0.8675	0.6874	0.5986	0.1426
SGL	0.6700	0.5188	0.5407	0.8726	0.7038	0.5993	0.1215
NGCF	0.6735	0.5249	0.5450	0.8696	0.7031	0.6002	0.1145
LightGCN	*0.6744*	0.5225	0.5448	*0.8734*	0.7233	*0.6015*	0.0995
CKE	0.6719	0.5226	0.5431	0.8723	0.7274	0.5994	0.0958
CFKG	0.6733	0.5224	0.5439	0.8732	0.7440	0.6003	0.0712
KGCN	0.6715	0.5202	0.5422	0.8683	0.7245	0.5969	0.1040
KTUP	0.6694	0.5197	0.5409	0.8693	0.7463	0.5949	0.0958
KGNNLS	0.6723	0.5222	0.5430	0.8710	0.7132	0.5998	0.1122
MKR	0.6651	0.5173	0.5377	0.8653	0.7252	0.5913	0.1042
KGIN	0.6702	0.5233	0.5424	0.8687	0.7005	0.5988	0.1228
KGAT	0.6716	0.5194	0.5423	0.8705	0.7285	0.5975	0.0840
Our best	**0.7043***	**0.5619***	**0.5755***	**0.8950***	0.6640	**0.6364***	*0.2108*

TABLE 3.18: Comparison to baselines on DBBOOK. Best results are in **bold**, second-best in *italic*; * means statistically significant ($p = 0.05$).

Model	Prec@5	Rec@5	F1@5	NDCG@5	Gini@5	EPC@5	APLT@5
ItemKNN	0.7237	0.4241	0.4531	0.9024	0.7935	0.5679	0.0295
BPR	0.7206	0.4218	0.4509	0.9021	0.7226	*0.5878*	0.0559
EASE	*0.7337*	0.4249	0.4558	*0.9098*	0.7709	0.5824	0.0331
MultiVAE	0.7335	*0.4278*	*0.4582*	0.9051	0.8408	0.5436	0.0236
NeuMF	0.7026	0.4147	0.4424	0.8837	**0.6883**	0.5835	**0.0776***
SLIMElastic	0.7162	0.4194	0.4487	0.8980	0.7307	0.5809	0.0436
SGL	0.7238	0.4222	0.4517	0.9015	0.7338	0.5864	0.0454
NGCF	0.7168	0.4207	0.4496	0.8963	0.7115	0.5873	0.0555
LightGCN	0.7318	0.4257	0.4554	0.9087	0.7688	0.5797	0.0385
CKE	0.7197	0.4214	0.4505	0.8990	0.7203	0.5873	0.0572
CFKG	0.7234	0.4237	0.4526	0.9015	0.7334	0.5861	0.0505
KGCN	0.7269	0.4244	0.4538	0.9052	0.7482	0.5833	0.0473
KTUP	0.7276	0.4247	0.4539	0.9051	0.7441	0.5842	0.0409
KGNNLS	0.7296	0.4260	0.4555	0.9067	0.7615	0.5830	0.0427
MKR	0.7253	0.4233	0.4528	0.9024	0.7521	0.5796	0.0463
KGIN	0.7110	0.4189	0.4475	0.8910	*0.6999*	0.5864	0.0650
KGAT	0.7149	0.4207	0.4490	0.8925	0.7125	0.5848	0.0573
Our best	**0.8035***	**0.4585***	**0.4947***	**0.9440***	0.7489	**0.6629***	*0.0701*

TABLE 3.19: Comparison to baselines on ML1M. Best results are in **bold**, second-best in *italic*; * means statistically significant ($p = 0.05$). For fairness, KARS models use the augmented graph $\mathcal{KG}_{\mathcal{I}}^{+}$.

been achieved, as confirmed by the quantitative analysis conducted. In future works, the LLM will be prompted with more context than only ratings or item names (*e.g.*, the abstract of the book or the plot of the movie - if available).

Main Findings

- **RQ3.2.1**: Graph augmentation successfully enriches the data model by increasing item coverage and adding previously unavailable features.
- **RQ3.2.2**: Incorporating additional knowledge through graph augmentation consistently improves model performance.
- **RQ3.2.3**: Exploiting graph augmentation in KARS enhances recommendation quality, achieving higher accuracy and more novel suggestions than existing methods.

The results discussed in this section have been published at the ACM UMAP 2025 Conference, at the Main Track as a full paper (Giuseppe Spillo et al. "GAL-KARS: Exploiting LLMs for Graph Augmentation in Knowledge-Aware Recommender Systems". In: *Proceedings of the 33rd ACM Conference on User Modeling, Adaptation and Personalization*. 2025, pp. 73–82).

3.3 Sentence Embeddings Based on LLM-generated User Profiles and Item Descriptions

3.3.1 Introduction

In this section, a third approach to improving homogeneous user embeddings is introduced, motivated by theoretical limitations of interaction- and graph-based user representations in sparse and semantically rich domains. Traditional methods, such as GCNs or aggregation-based embeddings, primarily rely on structural signals and assume that user preferences can be inferred from proximity or linear combinations of interacted items. However, these assumptions might fail to capture the semantic and contextual nature of user interests, especially when interaction data are sparse.

To address this issue, LLMs are exploited to generate textual user profiles that semantically summarize user preferences. Unlike the previous approach, where LLMs were used to generate structured knowledge graph triples, this method focuses exclusively on unstructured textual profiles. This allows LLMs to act as semantic abstraction mechanisms, compressing user interaction histories into dense representations that preserve meaningful preference signals while filtering out noise. These textual user profiles can then be encoded using state-of-the-art transformer-based sentence encoders to obtain homogeneous user embeddings. To further enrich the semantic content of the profiles, LLMs are provided not only with the items a user likes, but also with textual descriptions of those items (item abstracts), enabling the model to capture latent features of interest that are not directly observable from interaction data alone. The resulting user embeddings are subsequently combined with item embeddings within a recommendation architecture to predict user preferences.

In the experimental evaluation, the proposed methodology is compared against two baselines: (i) a graph-based approach, where user and item embeddings are learned via GCNs applied to the user–item interaction graph, and (ii) a centroid-based approach, where user embeddings are computed as the average of the embeddings of liked items, with item embeddings obtained from textual descriptions using sentence encoders.

Experiments conducted in the movie and book domains show that LLM-generated textual user profiles can significantly improve recommendation effectiveness, particularly in scenarios characterized by high sparsity, with performance gains varying across domains.

3.3.2 Methodology

The methodology comprises four main steps, each contributing to the construction, training, and evaluation of a recommender system relying on LLMs-generated user profiles. The methodological process can be divided into the following main steps:

- Preprocessing of abstracts: The raw data related to the items (films and books) were first subjected to a cleaning and normalization phase. This step is necessary to make the abstracts tractable by LLMs, due to their limited window size; moreover, this step emphasizes the relevant aspect that characterizes the items.
- Construction of textual user profiles: Semantic profiles were constructed using two distinct approaches: an LLM-based profile, represented by a textual

description of user preferences, from which an embedding was obtained using sentence transformers, and a centroid profile, obtained as the average of the abstract embeddings.

- Generation of baseline embeddings: In addition to textual profiles, structural embeddings were extracted from a custom-built knowledge graph modeling semantic relations among entities. The various embeddings (textual and structural) were employed either individually or combined using concatenation techniques or fusion mechanisms with attention.
- Training of the recommender system architecture: All experiments were conducted using neural network–based recommendation models capable of processing different types and numbers of sources.

The following sections detail each step.

Abstract Processing In the preliminary phase of data preparation, textual preprocessing was performed to standardize and reduce the length of the abstracts associated with the items (books and films) in order to make them more suitable for analysis and for the extraction of semantic representations with LLMs.

To this aim, an LLM system prompt was designed to transform each abstract into a short two-sentence text capturing its main themes, excluding redundancies or unnecessary stylistic elements. The resulting summaries were saved in textual format and linked to the identifiers of the original items, making them readily available for the subsequent embedding and modeling phases. Automatic checks on the model's output were implemented to ensure the quality and consistency of the generated summaries. In cases where the model failed to meet the required standards (for example, producing a text that was too long, incomplete, or formally incorrect), the result was discarded and the process was repeated. This validation maintained high quality standards, ensuring the homogeneity of the synthesized dataset and its reliability for the following stages of analysis.

This way, it was possible to: *i)* eliminate redundancies and non-essential details, while preserving key information related to the plot, characters, and context of the work; *ii)* standardize the length and structure of the texts, thus producing more homogeneous inputs for subsequent stages (such as embedding generation); *iii)* improve the semantic quality of the textual content, reducing noise and ambiguities introduced by verbose or informal descriptions.

The system prompt designed to carry out this step is the following:

Abstract Summarization System Prompt

Role: You are a professional book editor.
Task: Summarize a book abstract.
Length Constraint: The summary must consist of exactly 2 sentences.
Output Requirements:

- Do not include any introductory phrases (e.g., "Here is a summary", "This is a two-sentence summary", or similar).
- Do not mention the book title or refer to the summary itself.

Objectives:

- Emphasize central themes and key ideas.
- Omit redundant or non-essential details.
- Ensure the summary is shorter and more direct than the original.

Tone: Neutral, informative, and suitable for catalogues or recommendation systems. **Style**: Clear, concise, and coherent.

This system prompt is designed to ensure consistent, high-quality abstract summarization by tightly constraining both form and intent, thereby reducing variability in model outputs. By assigning the role of a professional book editor and explicitly defining objectives, tone, and style, the prompt encourages summaries that foreground core themes while eliminating superfluous detail. The fixed two-sentence length and strict output requirements allows short and meaningful summarizations, which can be subsequently included in the user profile construction prompt in an efficient manner. Overall, these constraints balance brevity with semantic fidelity, providing concise, comparable summaries across items.

In the user prompt, the abstract of the item is provided, and the output of this step is a set of summaries associated with each item.

Textual User Profile Construction Once the summarized item abstracts were obtained, textual user profiles were constructed, intended as a description of the item features preferred by each user. This step represents a fundamental component in modeling user behavior, as it enables the capture, at a semantic level, of individual preferences based on content. For each user, the set of items liked by that user was identified; the summarized abstracts associated with such items were gathered, and an LLM was prompted with this information. The decision to use abstract summaries allowed the overall text length to be reduced and redundancies to be avoided while maintaining high informational density.

To standardize the model's behavior during user profile inference, the following system prompts were designed for the book and movie domains, respectively:

User Profile Generation System Prompt - Book Domain

Considering that the user likes the books given in the user prompt, infer information about these features:

- the user's favourite genre;
- the user's favourite kind of plot;
- the user's favourite kind of musical score;
- the user's favourite mood for the musical score;
- the user's favourite setting;
- the user's favourite visual style;
- the user's favourite writing style;
- the user's favourite runtime;
- the user's favourite themes.

Write a summary of this information and any other related likely information that you can infer about the user's taste.

User Profile Generation System Prompt - Movie Domain

Considering that the user enjoys the movies listed in the user prompt, infer information about the following features of their cinematic preferences:

- the user's favourite film genres;
- the user's preferred types of plot;
- the user's favourite types of musical score or soundtrack;
- the user's preferred emotional tone or mood in music;
- the user's favourite settings (e.g., historical, futuristic, urban, rural);
- the user's preferred visual or cinematographic style (e.g., color palette, camera work);
- the user's favourite directing or storytelling style;
- the user's preferred movie runtime or pacing;
- the user's most appreciated recurring themes or messages in films.

Write a summary of these insights and include any other relevant information about the user's likely movie taste that you can infer.

These prompts are designed to guide the LLM toward the generation of rich, semantically grounded user profiles that capture latent preferences rather than surface-level item similarities. By explicitly requiring a diverse yet structured set of features, ranging from genre and plot to mood, style, and thematic elements, the prompts encourage comprehensive inference across multiple dimensions of taste while maintaining consistency across users. The use of summarized abstracts as input, produced at the previous step, ensures that the model operates on dense, non-redundant representations of liked items, improving signal quality and scalability.

The list of user preferences is provided in the user prompt. Due to space limitations, only an example in the book domain is shown; however, an analogous output has been obtained in the movie domain.

User Profile Generation User Prompt - Movie Domain

User 0 Prompt (Dbbook) User 0 has the following favourite books:

- Contingency, Irony, and Solidarity Description: Richard Rorty presents a philosophical framework that rejects traditional notions of truth and reality, instead emphasizing the contingency of language and the importance of individual vocabularies.
- Wise Blood Description: Hazel Motes, a World War II veteran, returns to his hometown and forms an anti-religious ministry, grappling with existential crises and theological doubts. O'Connor explores themes of faith, redemption, and the search for meaning in a chaotic world.
- A Heartbreaking Work of Staggering Genius Description: Dave Eggers recounts his experience as a young adult raising his younger brother after their parents' death, exploring themes of grief, responsibility, and identity.

Please analyze the books above to infer the user's preferences.

The related LLM output, representing the user textual profile, is the following:

User Profile Generated for a real user (User0)

Based on the provided book list, it appears that User 0 has a strong affinity for philosophical, literary, and thought-provoking works. Here's a breakdown of their likely preferences:
Genre: The user seems to enjoy a mix of philosophical non-fiction (Contingency, Irony, and Solidarity), literary fiction (Wise Blood), and memoirs (A Heartbreaking Work of Staggering Genius). This suggests an appreciation for diverse genres, with a focus on intellectually stimulating content.
Plot: The user likely prefers complex, character-driven narratives that explore themes of identity, morality, and existential crises. Plots that challenge traditional notions and offer nuanced, open-ended conclusions may appeal to them.
Setting: The settings in the preferred books are varied, ranging from philosophical discussions to rural Tennessee and California. This suggests that the user is open to different environments, as long as they serve the story and its themes.
Writing Style: The user probably appreciates lyrical, descriptive, and introspective writing styles, as seen in the works of Flannery O'Connor and Dave Eggers. They may also enjoy philosophical and essayistic writing, as exemplified by Richard Rorty.
Themes: The user's preferred themes appear to include: existentialism and the search for meaning; identity, morality, and ethics; social critique and commentary; personal growth, responsibility, and coming-of-age; and the human condition, with its complexities and flaws.
Tone and Mood: The user may prefer works with a contemplative, introspective tone, often accompanied by a sense of melancholy, irony, or satire.
Other Preferences: While the provided information focuses on books, it's possible to make some inferences about the user's tastes in other media: they might appreciate music with a similar tone and mood; cinematic, dialogue, and thematic exploration; and longer, more immersive experiences such as novels, films, or series with multiple episodes.
Overall, User0 appears to be a thoughtful and intellectually curious individual who values complex, nuanced, and emotionally resonant stories. They likely appreciate works that challenge their perspectives, explore the human condition, and offer insights into the complexities of existence.

The final output of this step is a set of textual user profiles, one per user in the catalog, which emphasizes the characteristics of the items users are interested in.

Embedding Learning Once the user profiles have been obtained, they have been encoded using sentence-encoder embedding models; in particular, MiniLM [217] was considered, as it showed good performance in previous works [167].

Sentence encoders were applied to obtain two sets of embeddings:

- User profiles: user profiles obtained at the previous step were encoded to obtain meaningful user embeddings
- Item abstracts: item abstracts were encoded to obtain meaningful item embeddings

In addition, another set of user embeddings was obtained as the centroids of the item embeddings associated with the user preferences, a common strategy in the field [167, 169]. This methodology for obtaining user embeddings is also described in Chapter 2.

Formally, for user u, let $L = \{i \in I \mid r_{u,i} = 1\}$ denote the set of items liked; an embedding for user u can be obtained as the centroid of the embeddings of the items liked by the user:

$$\vec{u_w} = \frac{\sum_{i \in L} \vec{i_w}}{|L|} \tag{3.7}$$

Finally, user-item preferences were encoded based on the interaction graph by exploiting GCNs; in particular, CompGCN [202] was considered, as it showed good results in the field [167, 169, 171]. A more detailed description of this step is reported in Section 3.2.2, where the same Graph Encoding Model is considered.

The output of this step is represented by the following sets of user and item embeddings:

- User Embeddings:
  - **User Profile Embeddings**, obtained by a sentence encoder starting from textual user profiles obtained by the LLM with our strategy.
  - **User Centroid Embeddings**, obtained as centroids of the item embeddings, as described above.
  - **Graph User Embeddings**, obtained by a GCN starting from the interaction graph.

- Item Embeddings:
  - **Textual Item Embeddings**, obtained by a sentence encoder starting from item abstracts.
  - **Graph Item Embeddings**, obtained by a GCN starting from the interaction graph.

These embeddings are finally used in a deep recommendation architecture to predict user preferences.

Recommendation Model Training Once the embeddings have been obtained, a recommendation model was trained to predict user interests in items. To this end, a recommendation architecture consisting of three variants is considered:

- **Single sources**, in which each set of user-item embedding pair is considered apart from the others.
- **Double sources**, in which we combine two user-item embedding pairs to predict user preferences.
- **Three sources**, in which we combine three user-item embedding pairs.

In addition, the employed architecture supports *dropout* applied to all embeddings.

For the single source architecture, first, dropout is applied to each user and item embedding; then, such embeddings are projected onto linear layers (with ReLU activation functions) to reduce their dimensionality. Finally, the resulting user and item embeddings are concatenated together, and this representation is projected onto a 1-dimensional linear layer with sigmoid activation function, which provides a recommendation score in the range $[0, 1]$.

Formally, let $\vec{u}$ and $\vec{i}$ denote the user and item embeddings, respectively. Dropout with a probability p is applied to these embeddings, and they are projected onto linear layers:

$$\vec{h_u} = ReLU\left(W_u * dropout(\vec{u}, p) + b_u\right) \tag{3.8}$$

$$\vec{h_i} = ReLU\left(W_i * dropout(\vec{i}, p) + b_i\right) \tag{3.9}$$

with W_u and W_i being learnable parameters, b_u and b_i being biases for the two layers.

Then, the vectors h_u and h_i are concatenated and projected onto a linear layer, with ReLU activation function:

$$\vec{h} = ReLU\left(W_{cat} * [h_u || h_i] + b_{cat}\right) \tag{3.10}$$

with W_{cat} being a learnable parameter, b_{cat} the bias for the layer, and $||$ the concatenation operator.

Finally, the resulting embedding is used to obtain the recommendation score:

$$rec_{score} = \sigma\left(W_h * \vec{h} + b_h\right) \tag{3.11}$$

The double-source architecture follows a very similar flow: given two different user-item embedding pairs (e.g., embeddings coming from the LLM-generated text and embeddings coming from the centroids of the abstracts), the two embeddings associated with the same user are projected onto linear layers (with ReLU) and concatenated, and the same happens for the two embeddings associated with the same item; the resulting embeddings (one for the user, one for the item) are concatenated and projected onto a 1-dimensional linear layer with sigmoid as activation function, which returns the recommendation score.

Formally, given a user u and an item i, and two different sources a and b, the embeddings $\vec{u_a}$, $\vec{i_a}$ for the user and the item coming from source a, and $\vec{u_b}$, $\vec{i_b}$ for the user and the item coming from source b, first we apply dropout and project each embedding onto linear layers[9]:

$$\vec{h_{ua}} = ReLU\left(W_{ua} * dropout(\vec{u_a}, p) + b_{ua}\right) \tag{3.12}$$

$$\vec{h_{ia}} = ReLU\left(W_{ia} * dropout(\vec{i_a}, p) + b_{ia}\right) \tag{3.13}$$

$$\vec{h_{ub}} = ReLU\left(W_{ub} * dropout(\vec{u_b}, p) + b_{ub}\right) \tag{3.14}$$

$$\vec{h_{ib}} = ReLU\left(W_{ib} * dropout(\vec{i_b}, p) + b_{ib}\right) \tag{3.15}$$

Then, h_{ua} with h_{ub} are concatenated and projected onto linear layers, and the same happens for h_{ia} and h_{ib}:

$$\vec{h_u} = ReLU\left(W_u * \left[\vec{h_{ua}} || \vec{h_{ub}}\right] + b_u\right) \tag{3.16}$$

[9]For the sake of readability, some of the elements already discussed (e.g., the dropout probability p or trainable parameters and biases) are not discussed here

$$\vec{h_i} = ReLU\left(W_i * \left[\vec{h_{ia}}||\vec{h_{ib}}\right] + b_i\right) \tag{3.17}$$

Finally, the resulting embeddings are concatenated, and a recommendation score is computed:

$$\vec{h} = ReLU\left(W_{cat} * [h_u||h_i] + b_{cat}\right) \tag{3.18}$$

$$rec_{score} = \sigma\left(W_h * \vec{h} + b_h\right) \tag{3.19}$$

Finally, the architecture with three sources follows the same principles as the previous two. To preserve readability, a brief description is provided without a formal definition.

In this case, three user-item embedding pairs are used, one per source. As described previously, first, each user and item embedding is projected through linear layers with ReLU activations; then the user embeddings associated with the three sources are concatenated on one side, and the item embeddings associated with the three sources on the other; these concatenations are projected through linear layers with ReLU to reduce dimensionality. At the end of this process, two embeddings are obtained, one for the user and one for the item; these embeddings are then concatenated and projected onto a 1-dimensional linear layer with a sigmoid activation function, which yields the recommendation score.

These models are trained on a subset of the ratings encoded in $\mathcal{Y}$ (the interaction matrix introduced in Section 3.2.2), and binary cross-entropy is used as the loss function. During inference, the score is computed for all items in the test set, items are ranked by the predicted score, and the *top-k* items are included in the recommendation list.

3.3.3 Problem Formulation

In this section, the problem formulation introduced in Section 2.3 is extended only with the information necessary to better frame this work, while the remainder of the formalization remains unchanged.

Abstract Summarization Each item $i \in \mathcal{I}$ is associated with a textual description t_i; in this methodology, an LLM is prompted to summarize such description and highlight the most relevant features that characterize the item. Let t_{sum-i} denote the summarization of the textual description t_i.

User Profile Generation To obtain textual user profiles, both CF data and the summarized abstracts are exploited: given a user $u \in \mathcal{U}$, let L_u be the set of items liked by the user; then, the abstract summarizations of the items in L_u are gathered and concatenated into a single text t_{sum-u}. The text is used to prompt an LLM to obtain a textual user profile, denoted as t_u.

Representation Learning Given the above information, a sentence encoder is used to encode both user profiles and item abstracts; these are denoted as t_u and t_i, respectively. In addition, starting from the embeddings derived from the item abstracts, user embeddings are obtained as the centroids of the items liked by the user, as described in the Methodology section, and these are denoted as t_{uc}. Finally, starting from the interaction graph $\mathcal{IG}$, a Graph Encoder is employed to learn graph-based user and item embeddings, denoted as g_u and g_i.

3.3.4 Experimental Evaluation

This study aims to address the following research questions:

- **RQ3.3.1 - Quantitative Analysis:** Do LLM-generated user profiles improve recommendation performance?
- **RQ3.3.2 - Ablation Tests:** How does each embedding combination contribute to the overall performance of the model?

Datasets Experiments were conducted in a *movie, book and music recommendation* scenario. In particular, the ML1M, DBbook and Last.fm datasets have been considered. More details about this datasets can be found in Section 2.3, where the problem has been formalized.

The prompts are populated based on the data available in the training set. In particular, users' preferences in the training set are used to infer user profiles first, and then to learn the various embeddings and train the recommendation models.

Evaluation Metrics To evaluate this approach, Precision, Recall, F1, nDCG as *accuracy* metrics, the Gini Index and the EPC for *diversity* and *novelty*, respectively, and the Average Percentage of Long Tail items (APLT) to assess the bias of recommendations have been considered.

Sources	Precision	Recall	F1	NDCG	Gini	EPC	APLT
Graph (G)	**0.8079**	**0.4586**	**0.4957**	**0.9445**	**0.7562**	**0.6643**	**0.0778**
Centroid (C)	0.7856	0.4514	0.4859	0.9351	0.8011	0.6348	0.0638
LLM-profiles (LLM)	0.7984	0.4577	0.4933	0.9399	0.7756	0.6580	0.0745

TABLE 3.20: Single source configuration performance on the ML1M dataset.

Sources	Precision	Recall	F1	NDCG	Gini	EPC	APLT
Graph (G)	0.6776	0.5372	0.5518	0.8643	**0.6125**	0.6039	**0.2663**
Centroid (C)	**0.7004**	**0.5566**	**0.5714**	**0.8874**	0.6813	**0.6308**	0.2170
LLM-profiles (LLM)	0.6948	0.5541	0.5668	0.8801	0.6388	0.6239	0.2390

TABLE 3.21: Single source configuration performance on the DB-book dataset.

3.3.5 Results and Discussion

To address the first RQ, Tables 3.20 and 3.21 report the recommendation performance of the three approaches under comparison; in particular, the comparison is between an approach based solely on graph embeddings (G in the tables) and two approaches based on text embeddings: one where the user embedding is represented by centroids (C in the tables), and another based on the LLM-generated profile (LLM in the tables).

On ML1M (Table 3.20), the best performing approach, in terms of all the metrics, is the one based on the graph, followed by the approach based on the LLM user profile, and then the approach that represents the user as the centroid of the liked item embeddings. This result is not surprising, as ML1M is a very dense dataset, so the collaborative signals provide sufficient information to perform the recommendation task effectively. However, it is observed that the LLM-based approach outperforms the centroid approach: this indicates that the textual description generated by the LLM is sufficiently precise to guarantee a more effective user embedding than the simple centroid strategy.

On the other hand, DBbook (Table 3.21) is a much sparser dataset, and here the graph-based configuration fails relative to the text-based approaches (with the only exceptions represented by Gini index and APLT), thus confirming the initial intuition; here, the centroid-based configuration performs slightly better than the LLM configuration for all accuracy and diversity metrics.

To answer RQ3.3.1, it can be stated that the LLM-generated profiles can improve

Sources		Precision	Recall	F1	NDCG	Gini	EPC	APLT
Double	G + C	0.8069	0.4600	0.4963	0.9440	**0.7549**	**0.6658**	**0.0815**
	G + LLM	**0.8081**	**0.4602**	**0.4968**	**0.9451**	0.7607	0.6646	0.0756
	C + LLM	0.7971	0.4561	0.4919	0.9381	0.7822	0.6505	0.0676
Triple	G + C + LLM	0.8070	0.4600	0.4965	0.9434	0.7562	0.6656	0.0925

TABLE 3.22: Multiple sources configuration performance on the ML1M dataset.

Sources		Precision	Recall	F1	NDCG	Gini	EPC	APLT
Double	G + C	0.6922	0.5494	0.5642	0.8790	0.6391	0.6207	0.2268
	G + LLM	0.6921	0.5488	0.5639	0.8800	0.6381	0.6208	0.2277
	C + LLM	**0.6959**	**0.5518**	**0.5665**	**0.8834**	**0.6357**	**0.6261**	**0.2416**
Triple	G + C + LLM	0.6870	0.5467	0.5601	0.8704	0.6247	0.6131	0.2434

TABLE 3.23: Multiple sources configuration performance on the DBbook dataset.

recommendation performance relative to simpler graph-based strategies, but this improvement depends on the domain considered, and, in general, is more evident in highly sparse contexts.

The results obtained by fusing the various sources are presented in Tables 3.22 and 3.23 for the ML1M and DBbook datasets, respectively.

On ML1M, the best overall configuration among all is the one that fuses graph embeddings with text embeddings whose user profiles are LLM-generated; this result is important, as it shows that LLM-generated user profiles can yield a general improvement in accuracy when combined with effective graph embeddings (as discussed above). The combination of all three sources, however, shows an improvement over configurations in which the sources were considered separately, but is outperformed by other configurations that fuse two sources (in particular, G + LLM and G + C). On DBbook, the best overall configuration is the one that combines centroid embeddings with LLM-generated embeddings. This result, together with observations on ML1M, shows that LLM-generated user profile embeddings can improve the best-performing single-source configuration. This demonstrates the effectiveness of this strategy. Furthermore, on DBbook, a configuration that combines three sources does not yield a meaningful improvement.

To answer RQ3.3.2, it is found that combining all three sources does not yield a meaningful improvement. In contrast, configurations that combine the best-performing single source with the LLM-generated user profile embeddings achieve

a substantial gain in recommendation accuracy, demonstrating the effectiveness of this approach.

3.3.6 Take-Home Messages

In conclusion, the results show that LLM-generated user profile embeddings can provide a meaningful improvement in recommendation performance, both as a single source and when fused with other embeddings. On ML1M, where collaborative signals are strong due to the density of the dataset, graph embeddings remain the most effective configuration, but LLM-based profiles still outperform the centroid-based strategy, demonstrating the added value of textual descriptions in constructing user embeddings. On DBbook, where sparsity reduces the effectiveness of collaborative information, the contribution of LLM-generated profiles is even more evident, especially when combined with centroid embeddings, which lead to the best overall performance. Moreover, it is observed that fusing all three sources does not yield additional benefits, while targeted combinations of graph or centroid embeddings with LLM-based profiles consistently achieve higher accuracy and diversity. These findings confirm that the improvement provided by LLM-generated profiles is dependent on the domain, but also highlight their robustness as a strategy that can enhance both single-source and fused configurations, thus demonstrating the effectiveness of this approach.

Main Findings

- **RQ3.3.1**: In highly sparsity context, LLM-generated profile embeddings show better performance than graph-based approaches.
- **RQ3.3.2**: Combining LLM-generated profile embeddings with graph-based embeddings further improves the performance of the recommendation.

3.4 Evaluating Multimodal Item Embeddings through User Feedback: An Empirical Evaluation

3.4.1 Introduction

This chapter presents a user study conducted to assess how different modalities affect the perceived suggestions in real-world scenarios. To this end, a web application, named *Movie4You*, was developed, which allows users to build their own profile, made up of past liked movies, and receive suggestions for movies to watch, as a recommendation list composed of 5 movies. After the suggestions have been provided, the ResQue (Recommendation Quality and Usability Evaluation) [137] questionnaire is used to evaluate the overall interaction quality.

For the purposes of this study, different recommendation lists (one per modality) are provided to users and evaluated separately through the ResQue questionnaire to assess how the different modalities affect several aspects of the recommendation lists.

To conduct this user study, the movie domain was chosen for two reasons. First, it is a very popular domain, making it likely that users can effectively interact with the platform, build realistic user profiles, and obtain realistic movie suggestions. Second, it is the domain for which the highest number of modalities is available (as described in Section 2.3), enabling a wider comparison between the modalities.

3.4.2 The *Movie4Rec* Web-app

This section presents the *Movie4Rec* web application and its flow, illustrating how a profile can be built, how suggestions are obtained, and how they are evaluated.

An example of this page is shown in Figure 3.7.

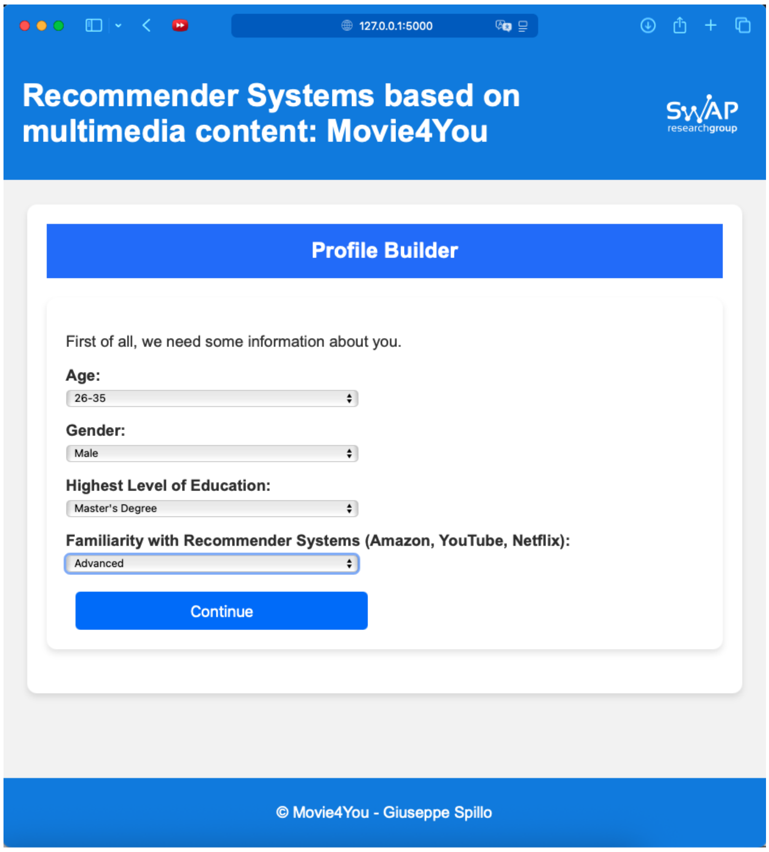
Then, the app enables the user to search for movies in the catalog and to insert them as part of the profile (Figure 3.8), which is incrementally built.

Once all liked movies have been added and the profile has been completed, the system can generate movie suggestions. An example of a full profile is shown in Figure 3.9.

The system provides a list of 5 suggested movies (see Figure 3.10). Optionally, any suggested movie can be selected, and the application will direct to the DBpedia webpage associated with that movie. Thereafter, evaluation of the received suggestions can be performed.

If the user decides to answer the survey, the instructions for the survey (Figure 3.11) and the survey form are displayed and can be submitted.

At the end of the process, the system generates other recommendation lists, which are all evaluated with the same survey. Figure 3.13 shows an example of another recommendation list obtained starting from the same user profile.


127.0.0.1:5000

Recommender Systems based on multimedia content: Movie4You

Profile Builder

First of all, we need some information about you.

Age: 26-35

Gender: Male

Highest Level of Education: Master's Degree

Familiarity with Recommender Systems (Amazon, YouTube, Netflix): Advanced

Continue

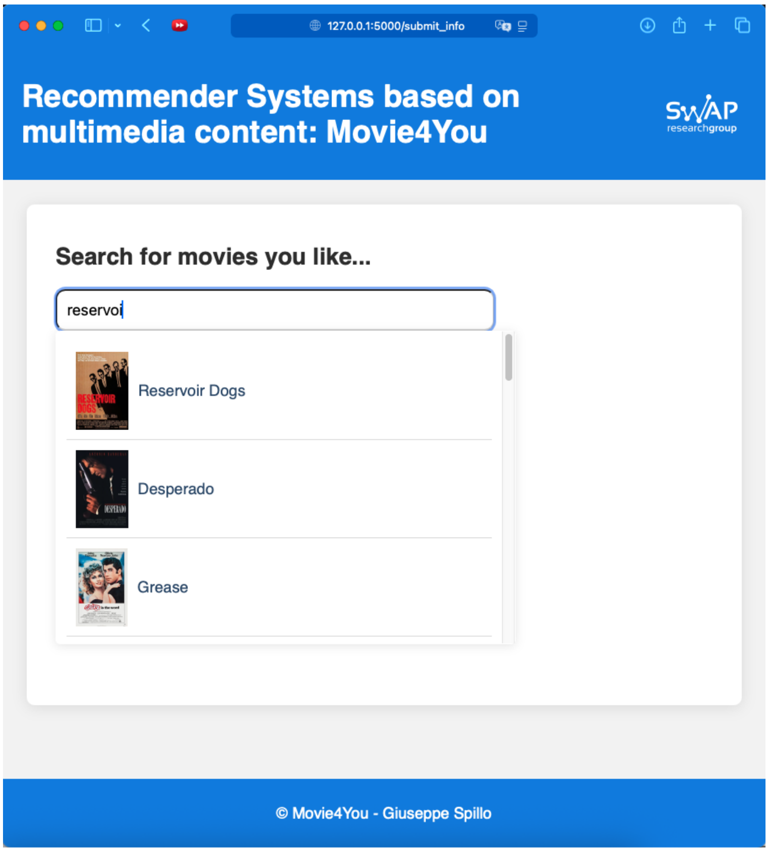
© Movie4You - Giuseppe Spillo

FIGURE 3.7: Demographic information form.

3.4.3 Experimental Evaluation

Movie Catalog. The catalog of movies considered for the *Movie4You* web application, from which users can build their profile and obtain recommendations, is based on MovieLens-1M. In particular, the items in the MovieLens-1M version extended with multimodal item features presented in [176], which represent one of the contributions of this thesis.

Modalities Considered. The above-mentioned datasets, as described in Section 2.3, come with different modalities, including plain text, images, audio and videos. Moreover, the datasets come with both a user-item interaction matrix and KGs. These heterogeneous sources can be encoded with state-of-the-art models (as discussed in the previous section of this manuscript) to obtain distinct sets of item embeddings derived from each modality.



Recommender Systems based on multimedia content: Movie4You

Search for movies you like...

reservoi

Reservoir Dogs

Desperado

Grease

© Movie4You - Giuseppe Spillo

FIGURE 3.8: Search for a movie.

In particular, based on the results obtained in a previous contribution of this thesis [177], the following encoders were considered:

- Interaction Graph (IG): CompGCN
- Knowledge Graph (KG): CompGCN
- Text: Sentence-BERT (MiniLM)
- Images: Visual Transformers (ViT)
- Audio: VGGish
- Video: R(2+1)D

For each item in the catalog, a total of 6 representations from different modalities is obtained, and their comparison constitutes the focus of this study.

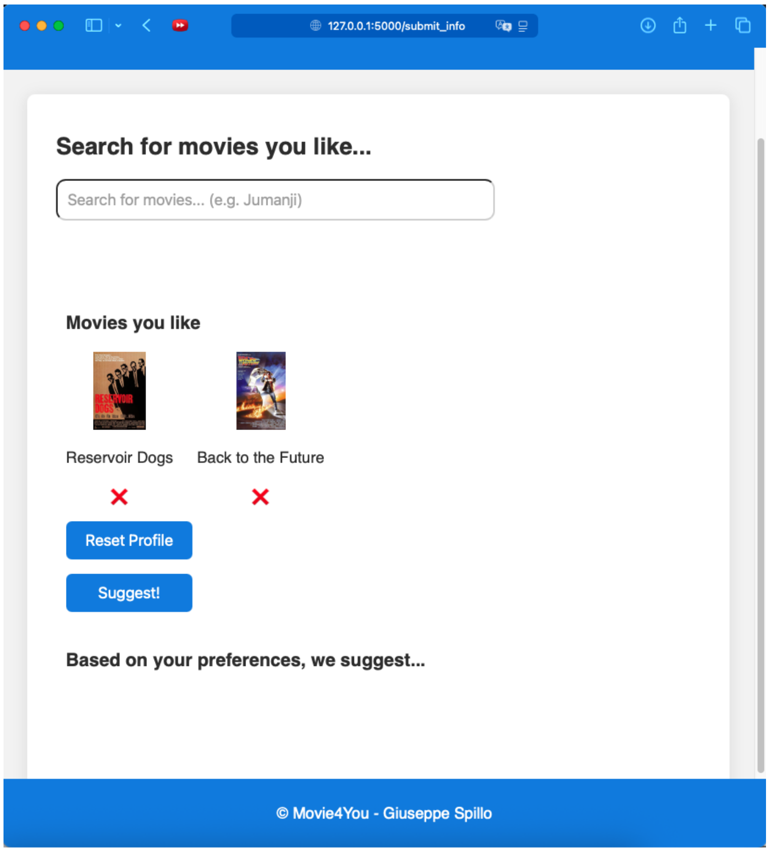

127.0.0.1:5000/submit_info
Search for movies you like...
Search for movies... (e.g. Jumanji)
Movies you like
Reservoir Dogs
Back to the Future
Reset Profile
Suggest!
Based on your preferences, we suggest...
© Movie4You - Giuseppe Spillo

FIGURE 3.9: Profile building.

User Profile Learning. Once the item catalog embeddings are learnt, a user representation is obtained from the item preferences expressed during interaction with the platform. Then, for each modality, the user is represented as the centroid of the embeddings of the liked items associated with that modality.

Thus, each user interacting with the web application can be represented with a modality-specific embedding.

Movie Suggestion List Generation. Finally, with each item associated with an embedding (per modality) and with each user represented in the same way, recommendations can be generated. For each modality, the corresponding item embeddings with the highest cosine similarity to the user embedding are selected, excluding items that have already been liked by the user.

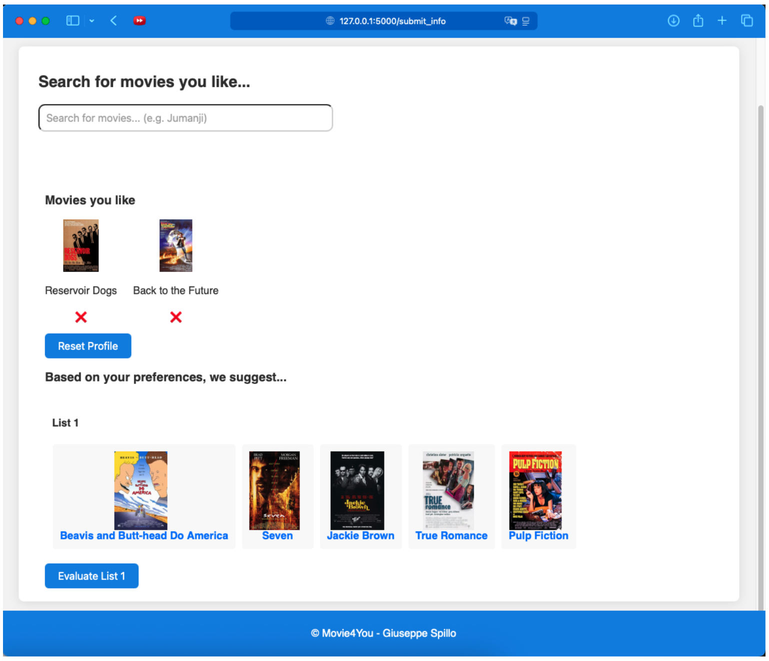

127.0.0.1:5000/submit_info
Search for movies you like...
Search for movies... (e.g. Jumanji)
Movies you like
Reservoir Dogs
Back to the Future
Reset Profile
Based on your preferences, we suggest...
List 1
Beavis and Butt-head Do America
Seven
Jackie Brown
True Romance
Pulp Fiction
Evaluate List 1
© Movie4You - Giuseppe Spillo

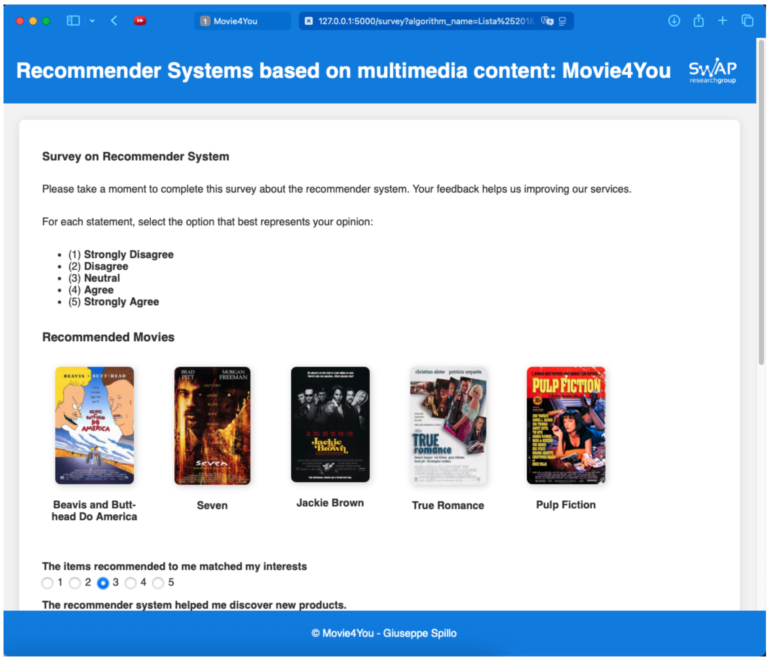
FIGURE 3.10: First displayed list of suggested movies.

Thus, for each of the 6 modalities, 6 total recommendation lists are generated, which are displayed to users in a random order and evaluated through the ResQue questionnaire, discussed in the next paragraph.

ResQue Questionnaire. The ResQue questionnaire [137] is a standardized evaluation tool designed to assess users' subjective perceptions of RSs. It captures multiple dimensions of user experience, including perceived accuracy, usefulness, diversity, novelty, transparency, and trust, as well as overall satisfaction and intention to reuse the system. By covering both system-related qualities and user-centric outcomes, ResQue provides a comprehensive framework for measuring the effectiveness and acceptance of RSs from the end-user's perspective.

From the original set of questions described in Pu et al. [137], 14 items were selected, and participants were invited to complete the questionnaire. Each item was rated on a 5-point Likert scale to indicate the level of agreement, from the lowest (1) to the highest (5).

The following is the set of questions presented to users:



Recommender Systems based on multimedia content: Movie4You

Survey on Recommender System

Please take a moment to complete this survey about the recommender system. Your feedback helps us improving our services.

For each statement, select the option that best represents your opinion:

- (1) **Strongly Disagree**
- (2) **Disagree**
- (3) **Neutral**
- (4) **Agree**
- (5) **Strongly Agree**

Recommended Movies

Beavis and Butt-head Do America · Seven · Jackie Brown · True Romance · Pulp Fiction

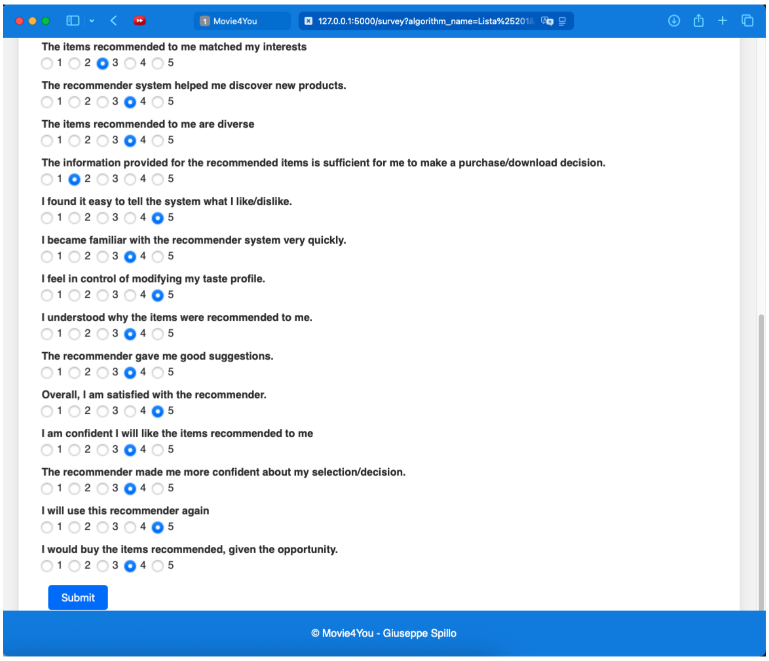
The items recommended to me matched my interests
1 2 [✓ 3] 4 5

The recommender system helped me discover new products.

© Movie4You - Giuseppe Spillo

FIGURE 3.11: Survey instructions.

- **Q1 - Recommendation Accuracy:** The items recommended to me matched my interests.
- **Q2 - Recommendation Originality:** The recommender system helped me discover new products.
- **Q3 - Recommendation Variety:** The items recommended to me are diverse.
- **Q4 - Information Sufficiency:** The information provided for the recommended items is sufficient for me to make a purchase/download decision.
- **Q5 - Interaction Adequacy:** I found it easy to tell the system what I like/dislike.
- **Q6 - Interaction Adequacy:** I became familiar with the recommender system very quickly.
- **Q7 - Control:** I feel in control of modifying my taste profile.
- **Q8 - Transparency:** I understood why the items were recommended to me.
- **Q9 - Perceived Usefulness:** The recommender gave me good suggestions.
- **Q10 - General Satisfaction:** Overall, I am satisfied with the recommender.


Movie4You

127.0.0.1:5000/survey?algorithm_name=Lista%25201

The items recommended to me matched my interests
1 2 [✓ 3] 4 5

The recommender system helped me discover new products.
1 2 3 [✓ 4] 5

The items recommended to me are diverse
1 2 3 [✓ 4] 5

The information provided for the recommended items is sufficient for me to make a purchase/download decision.
1 [✓ 2] 3 4 5

I found it easy to tell the system what I like/dislike.
1 2 3 4 [✓ 5]

I became familiar with the recommender system very quickly.
1 2 3 [✓ 4] 5

I feel in control of modifying my taste profile.
1 2 3 4 [✓ 5]

I understood why the items were recommended to me.
1 2 3 [✓ 4] 5

The recommender gave me good suggestions.
1 2 3 [✓ 4] 5

Overall, I am satisfied with the recommender.
1 2 3 4 [✓ 5]

I am confident I will like the items recommended to me
1 2 3 [✓ 4] 5

The recommender made me more confident about my selection/decision.
1 2 3 [✓ 4] 5

I will use this recommender again
1 2 3 4 [✓ 5]

I would buy the items recommended, given the opportunity.
1 2 3 [✓ 4] 5

Submit

© Movie4You - Giuseppe Spillo

FIGURE 3.12: ResQue questionnaire.

- **Q11 - Trust:** I am confident I will like the items recommended to me.
- **Q12 - Confidence:** The recommender made me more confident about my selection/decision.
- **Q13 - Use Intention:** I will use this recommender again.
- **Q14 - Purchase Intention:** I would buy the items recommended, given the opportunity.

3.4.4 Results and Discussion

Demographic Information. Table 3.24 reports the statistics of the demographic data associated with participants involved in the user study. The sample consisted of 53 users, with 34 males (64%) and 19 females (36%). Participants spanned various age groups, with the majority (31 users, 57%) aged 26–35, indicating a predominantly young-adult sample likely familiar with digital recommender systems. The sample included a range of educational backgrounds, with nearly half holding a high school diploma (26 participants, 47%) and a balanced representation of bachelor's and master's degrees (20% each), providing a diverse and balanced mix. Additionally, 23 participants (42%) reported advanced knowledge

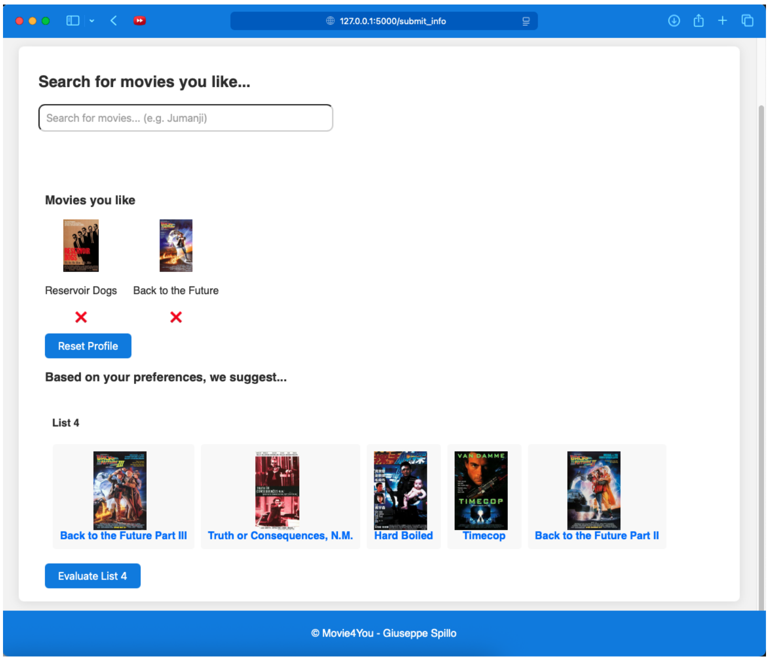

127.0.0.1:5000/submit_info
Search for movies you like...
Search for movies... (e.g. Jumanji)
Movies you like
Reservoir Dogs
Back to the Future
Reset Profile
Based on your preferences, we suggest...
List 4
Back to the Future Part III
Truth or Consequences, N.M.
Hard Boiled
Timecop
Back to the Future Part II
Evaluate List 4
© Movie4You - Giuseppe Spillo

FIGURE 3.13: Second displayed list of suggested movies.

of recommender systems, suggesting that a significant portion of the sample may evaluate recommendations more critically due to higher expectations for accuracy and personalization.

ResQue Results. Table 3.25 reports the average ratings on a 1–5 scale (as discussed previously) for the 14 evaluation questions (Q1–Q14) across the information sources IG, KG, Text, Image, Audio, and Video.

The evaluation results show that graph-based approaches, particularly KG and IG (with average ratings of 3.67 and 3.71, respectively), outperformed the content-based sources across most dimensions of user experience (3.37 for text, 3.6 for audio, 3.65 for video).

KG and IG were especially strong in helping users match their interests (Q1), discover new products (Q2), and quickly become familiar with the system (Q6, the highest overall rating at 4.34), while providing a stronger sense of control (Q7) and confidence in recommendations (Q11–Q12).

Multimedia sources, in contrast, contributed more to perceived diversity (Q3),

Variable	Category	Count	Percentage
Gender	Male	34	64.2 %
	Female	19	35.8 %
Age	<18	3	6.0 %
	18-25	7	14.0 %
	26-35	31	62.0 %
	36-40	5	10.0 %
	>50	7	14.0 %
Education	High school	26	52.0 %
	Bachelor's degree	11	22.0 %
	Master's degree	10	20.0 %
	Other	6	12.0 %
Familiarity with RSs	None	8	16.0 %
	Low	8	16.0 %
	Medium	14	28.0 %
	High	23	46.0 %

TABLE 3.24: Demographic Information of the participants involved in the study.

but failed to provide sufficient information (Q4), transparency (Q8), and decision-making support.

Overall satisfaction (Q10) and willingness to reuse the system (Q13) were highest for IG and KG, reflecting the reliability and trust associated with structured data.

However, purchase intention (Q14) received lower scores across all sources, indicating a gap between positive perceptions of the system and actual behavioral intent. **These results suggest that while graph-based approaches provide the most effective and trustworthy recommendations, combining them with multimedia signals could further enhance diversity and engagement, leading to a more balanced recommender system.**

3.4.5 Take-Home Messages

The main findings of the user study show that graph-based approaches provide the most effective, usable, and trustworthy recommendations, while multimedia

	IG	**KG**	**Text**	**Image**	**Audio**	**Video**
Q1	**3.66**	**3.66**	3.17	3.00	3.60	3.60
Q2	3.45	3.58	3.62	3.60	3.56	**3.66**
Q3	3.17	3.36	3.51	**3.58**	**3.58**	3.33
Q4	3.40	3.45	3.15	3.15	3.30	**3.49**
Q5	3.89	**3.94**	3.81	3.62	3.72	3.87
Q6	4.15	**4.34**	4.03	3.90	3.96	4.03
Q7	**3.96**	**3.96**	3.70	3.62	3.83	3.79
Q8	**3.90**	3.79	3.27	3.40	3.58	3.58
Q9	3.62	3.66	3.30	3.17	3.55	**3.72**
Q10	**3.75**	3.68	3.55	3.26	3.64	**3.75**
Q11	**3.75**	3.70	3.24	3.15	3.53	3.64
Q12	3.47	**3.68**	3.26	3.36	3.62	3.66
Q13	3.73	**3.77**	3.60	3.40	3.75	3.62
Q14	**3.53**	3.37	3.11	2.98	3.22	3.45
AVG	3.67	**3.71**	3.45	3.37	3.60	3.65

TABLE 3.25: Average ratings for each question across different information sources. The bold values indicate the highest score for each row.

sources offer added diversity but limited transparency and decision-making support.

A hybrid strategy that integrates the strengths of structured graphs with the richness of multimedia data could enhance both recommendation quality and user trust, thus leading to a more engaging and fair RSs.

Main Findings

- Graph-based recommendations are perceived as more accurate and trustworthy.
- Multimedia content based recommendation are perceived as more diverse.
- Fusing the sources is necessary to get the best from the two worlds: accuracy and diversity.

3.5 Conclusion

This section concludes the chapter and draws some conclusions to answer RQ1, introduced in Chapter 1.

First, this chapter showed that injecting FOL rules during the process of graph embedding generally increases the performance of KARS, so it can be considered an effective strategy to enrich KG embeddings. Moreover, it is also possible to identify a group of heuristics for the rule selection that is more effective than others. This is a relevant contribution to the RSs literature, as neuro-symbolic AI is poorly investigated in this field, although shows promising results.

Then, the chapter showed that also LLMs can improve the representation coming from common KGs, due to the absence of user profile modeling on one side, and missing facts in KGs on the other. To this aim, proper Item Feature Inference and User Modeling prompts have been designed to obtain additional KG triples to be integrated in the original KG. Results showed that LLM-generated triples can improve the performance of RSs with respect to models solely relying on popular KGs, thus enriching KG embeddings as well.

The subsequent section focused on LLMs from another perspective: instead of encoding the LLM output in the form of KG triples, LLMs were exploited to infer user profiles in textual form, based on their preferences. Together with item textual descriptions, this data has been encoded in the form of embeddings, and a RS architecture has been exploited to provide recommendations. Results showed that in highly sparsity context, LLM-generated profile embeddings show better performance than graph-based approaches, and the performance further increased when graph embeddings and LLM-generated embeddings are fused together.

Finally, a user study seeking how the different modalities affect *in-vivo* recommendation in the movie domain has been discussed. Results showed that while graph-based recommendation is more accurate and trustworthy, unstructured knowledge sources provide the more diverse recommendations. Assessing how different modalities affect *in-vivo* recommendation is a crucial aspect that research often overlooks, since it generally focuses on *in-vitro* experiments. Under this perspective, although preliminary, this is a relevant contribution for the research community.

This discussion allows to answer the first RQ of this thesis, investigating the contribution of each individual knowledge source, and how they can be enriched.

Answer to RQ1: Contribution of Individual Knowledge Sources

- Mining FOL rules from KGs and injecting them into the KG embedding process can be considered an effective approach to enrich KG embeddings for recommendation task.
- KGs can be augmented through LLMs inference capabilities, and the resulting embeddings provide richer semantics than common KGs, thus improving recommendation performance.
- Similarly, LLMs can also be exploited to reason over user preferences and generate textual user profiles, which show better performance than only graph-based approaches.
- Finally, a user study assessed the recommendation performance provided by each individual modality, showing that graph-based recommendations are more accurate and trustworthy, while multimedia based recommendation are more diverse.

Chapter 4

Recommendations based on Heterogeneous Item Embeddings

This chapter addresses the problem of recommendation when multiple knowledge sources are considered. This is a relevant topic in the literature, since fusing heterogeneous knowledge sources requires specific mechanisms to be effective. To this aim, this chapter advances the state-of-the-art by proposing different fusion mechanisms, typically overlooked in previous works, and by analyzing how they affect the recommendation performance.

The first work, presented in Section 4.1, proposes a novel fusion strategy to combine KG embeddings and textual embeddings into a unified representation. The approach exploits deep neural architectures and attention mechanisms to effectively integrate heterogeneous information, and the resulting embeddings are used to generate recommendations. The key novelty lies in introducing a more sophisticated fusion mechanism than those typically explored in the literature, which have been limited to simple concatenation or sum of embeddings, thus leading to a sub-optimal solution.

A second approach to fusing heterogeneous embeddings is presented in Section 4.2, where pre-training strategies are explored for the first time in this context. Instead of the common practice of randomly initializing the embeddings of the graph encoder representing users and items, this work proposes initializing them with pre-trained text-based embeddings. In this way, textual knowledge is further refined through the message passing process of GCNs and the graph structure itself. This represents a novel application of pre-training techniques for knowledge-source fusion, moving beyond traditional methods that rely merely on concatenation or summation of heterogeneous embeddings, as discussed above.

The third work (Section 4.3) provides a comprehensive benchmark study to investigate how multimedia embeddings influence recommendation performance. In particular, the benchmark focuses on the extended datasets ML1M, DBbook, and Last.fm-2K, introduced in Section 2.3. The benchmark first examines each modality individually, and then assesses performance when modalities are fused, enabling a detailed understanding of multimodal fusion effects. This constitutes the first systematic evaluation of multimedia embedding fusion on these extended datasets, which were never previously released with the associated multimedia sources, thus filling an important gap in the current literature.

Finally, Section 4.4 introduces a novel recommendation architecture, Gotta Embed Them All! (GETALL!), designed to fuse all available heterogeneous embeddings, including those derived from KGs, text, images, audio, and video, into a single, unified representation. The architecture is based on deep attention mechanisms that allow the model to effectively integrate and balance diverse information sources. This work is the first to combine all these heterogeneous modalities, including KGs, within a single comprehensive architecture, extending beyond existing multimodal RSs that typically consider only subsets of such data.

4.1 Fusion of Graph and Text Embeddings through Deep Architectures

4.1.1 Introduction

In the featured publication [167], a knowledge-aware recommendation framework that leverages the complementary capabilities of graph neural networks (GNNs) and sentence encoders to provide effective personalized recommendations is introduced. The core idea is that structured data from KGs and unstructured textual descriptions capture unique aspects of items and user preferences. Instead of relying on a single data modality, this study aims to integrate both sources into a unified and expressive representation learning pipeline. To accomplish this, GNNs are first employed to encode collaborative information, such as user-item interactions, and structured item features modeled within a KG. This step captures the relational dependencies and contextual semantics inherent in the graph structure. In parallel, a transformer-based sentence encoder is applied to extract dense semantic embeddings from textual descriptions associated with items, such as product summaries, reviews, or metadata. These two distinct representations are then fused through a deep neural architecture that exploits attention mechanisms, allowing the model to dynamically refine and align the embeddings based on their interdependencies. The final output of this architecture is used to estimate the likelihood of user interest in items, enabling the generation of a top-k recommendation list tailored to each user.

The featured publication [167] falls into the broader context of KARS, which seek to improve recommendation performance by integrating several sources of side information. While prior models have explored either graph-based or content-based representations in isolation, this approach demonstrates how combining these sources leads to richer and more informative representations, thus improving recommendation performance. Extensive experiments on two real-world benchmark datasets show that the model significantly outperforms several state-of-the-art baselines across standard evaluation metrics, confirming the effectiveness of the hybrid design. Moreover, a detailed sensitivity analysis is conducted, evaluating different configurations of the GNN and sentence encoder components, as well as varying key hyperparameters to assess the robustness and flexibility of the method. Overall, this work contributes a novel strategy for multi-modal representation learning in recommender systems, highlighting the benefits of combining structured and unstructured data through advanced neural attention mechanisms to deliver more accurate, context-aware recommendations.

This section discusses a paper produced during the PhD and published at ACM UMAP 2023 in the Main Track (Giuseppe Spillo et al. “Combining graph neural networks and sentence encoders for knowledge-aware recommendations”. In: *Proceedings of the 31st ACM Conference on User Modeling, Adaptation and Personalization*. 2023, pp. 1–12), which was presented at the main session as a full paper. In addition, this paper has been awarded with the **James Chen Best Student Paper Award**.

4.1.2 Methodology

In this section, the strategies for graph encoding and sentence encoding, based on GCNs and Transformers, respectively, are presented. Then, a deep architecture is introduced to further refine the representation of users and items and to generate the recommendation list. Figure 4.1 shows the overall workflow of this approach.

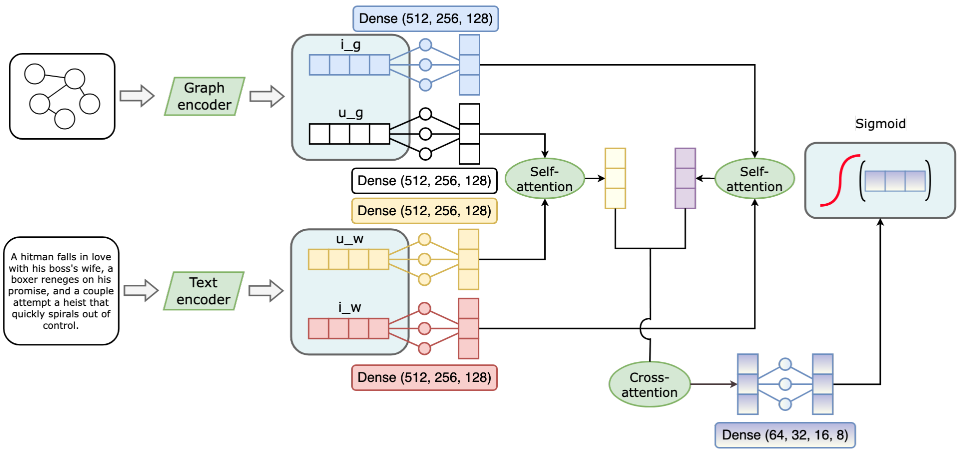


FIGURE 4.1: Overview of the workflow

Graph Encoding Graph embeddings are learned using GCNs. In particular, a GCN based on CompGCN [202], which has been discussed in Chapter 2, was employed in this study and in other sections of this manuscript (see Section 3.2).

The process was applied to the KG encoding of both user-item interactions and item features; at the end of the graph embedding process, embeddings u_g for each user and i_g for each item in $\mathcal{G}$ were learned.

Sentence Encoding An item representation $\vec{i_w}$ for each item i and a representation $\vec{u_w}$ for each user u in $\mathbb{R}^d$ are learned, so a d-dimensional embedding is obtained at the end of the process.

As stated above, the sentence encoding strategy is based on Transformers. The same strategy has been applied in other works described in this manuscript (see Sections 3.2 and 3.3); a more detailed discussion of how embeddings for users and items derived from textual knowledge sources can be learned is provided in Chapter 2.

At prediction time, given a description t_i of the item i, an embedding based on SBERT is returned by the sentence encoder. Formally, $\vec{i_t} = SBERT(t_i)$.

Next, a similar process is carried out to learn the embedding of the user $\vec{u_t}$. In this case, the interaction matrix Y (see Chapter 2) is exploited, and the items liked by user u are collected. Let $L = \{i \in I \mid r_{u,i} = 1\}$ denote the set of items liked; the embedding of the user u is obtained as the centroid of the embeddings of the items liked by the user. Formally:

$$\vec{u_t} = \frac{\sum_{i \in L} \vec{i_t}}{|L|} \tag{4.1}$$

Deep Recommendation Architecture The deep neural architecture used to merge graph and sentence embeddings and to provide users with recommendations is introduced. Figure 4.1 provides details.

As previously stated, the neural network takes as input both graph and sentence embeddings of users and items. Let u_g and u_t denote, respectively, the graph embedding and the word embedding of a user[1]. First, dropout is applied to both of them:

$$\begin{aligned} u_{g\_drop} &= dropout\left(W_{g\_drop} * u_g + b_{g\_drop}\right) \\ u_{t\_drop} &= dropout\left(W_{t\_drop} * u_t + b_{t\_drop}\right) \end{aligned} \tag{4.2}$$

Then, several dense layers with decreasing dimensions and ReLU activation functions are introduced[2]:

$$\begin{aligned} u_{g\_dense} &= ReLU\left(W_{g\_dense} * u_{g\_drop} + b_{g\_dense}\right) \\ u_{t\_dense} &= ReLU\left(W_{t\_dense} * u_{t\_drop} + b_{t\_dense}\right) \end{aligned} \tag{4.3}$$

Next, embeddings learned from heterogeneous sources (graph and text) are combined into a single embedding using *self-attention* [203]. The intuition is to use self-attention to improve the representation by identifying the most relevant portions of the original embeddings. To this end, the embeddings are concatenated and a softmax activation function is applied (via a dense layer) to obtain attention scores α_u; these scores are then used to weight the embeddings, which are finally added to obtain the embedding u representing the user. Formally:

$$\alpha_u = softmax\left(W_{att\_user} * \left(u_{g\_dense} \oplus u_{t\_dense}\right) + b_{att\_user}\right) \tag{4.4}$$

$$u = \alpha_u * u_{g\_dense} + (1 - \alpha_u) * u_{t\_dense} \tag{4.5}$$

[1]For readability, arrows from embeddings have been removed. Nevertheless, all elements mentioned in the formulas are vectors.

[2]For brevity, a single application of this layer is shown here. For the complete description of the architecture, see Figure 4.1.

Next, processing of item embeddings i_g and i_w follows the same layers. For brevity, the complete steps are not reported, since it suffices to replace i_g and i_w with u_g and u_w in Formulas 4.2, 4.3 and 4.4. In this case, the final embedding representing item i is obtained by merging again the vectors using *self-attention*:

$$i = \alpha_i * i_{g\_dense} + (1 - \alpha_i) * i_{t\_dense} \tag{4.6}$$

Once the user and item embeddings are learnt, they are combined using *cross-attention* [94]. The aim is to refine the representation by encoding the relations between user and item embeddings. Finally, the resulting vectors are passed through additional dense layers and a sigmoid activation function.

$$\begin{aligned} \alpha_{cross} &= \delta\left(W_{att\_cross} * (u \bigotimes i) + b_{att\_cross}\right) \\ merged &= ReLU\left(W_{merge} * (\alpha_{cross} * u + (1 - \alpha_{cross}) * i) + b_{merge}\right) \\ score &= sigmoid\left(W_{score} * merged + b_{score}\right) \end{aligned} \tag{4.7}$$

Thus, at prediction time, a recommendation score $score \in [0, 1]$ can be provided for a user u and an item i.

This work follows the problem formulation introduced in Section 2.3, which is not discussed here to avoid repetitions. More details can be found in the published paper [167].

4.1.3 Experimental Evaluation

This study aims to answer the following research questions:

- **RQ4.1.1 - Comparison to State of the Art:** How does the proposed approach perform with respect to baselines?
- **RQ4.1.2 - Ablation Tests:** How do embeddings (i.e., graph, text) and attention mechanisms contribute to the overall performance of the method?
- **RQ4.1.3 - Sensitivity Analysis:** How do different hyper-parameter settings (size of the embeddings, information encoded in the graph, dropout) affect the overall performance of the method?

Datasets Experiments were conducted in a *movie, book and music recommendation* scenario. In particular, the ML1M, DBbook and Last.fm datasets have been considered. More details about this datasets can be found in Section 2.3, where the problem has been formalized.

Sensitivity Analysis Experiments were conducted varying several parameters. Training epochs in the range $\{5, 10, 15, 20 \ldots 50\}$, while dropout values varies in the range $\{0, 0.25, 0.4, 0.5 \ldots 0.7, 0.8\}$. Regarding graph embeddings, the dimension was varied over $\{256, 384, 512, 768\}$. Finally, different implementations of Transformers were evaluated, based on MPNet, DistilBERT, RoBERTa and ALBERT.

Evaluation Metrics For this work, precision, recall, F1, NDCG, and MAP were considered as accuracy metrics. Diversity was assessed through the Gini index, while novelty was assessed through the EPC metric. Statistical significance was assessed using a t-test (P-value = 0.05).

Baselines Several baselines belonging to different model groups were considered, and are reported in the tables. Their description is discussed in Section 2.3, and is not reported here to avoid repetitions.

More details about the protocol and the experimental settings are deeply discussed in the related published papers [170, 174].

RQ4.1.1: Comparison to Baselines In order to answer RQ4.1.1, Table 4.1 and Table 4.2 compare the performance of the proposed approach with respect to the baselines introduced earlier. For readability, the baselines have been split according to their recommendation paradigm.

As shown in the tables, results are consistent across the datasets, since the proposed strategy outperforms all baselines for all the metrics considered. In particular, results show that matrix factorization techniques obtain the best results on ML1M, while KARS are more competitive on DBbook. This is somewhat expected, since ML1M is denser, making it unsurprising that latent connections among users and items learned by deep learning techniques and matrix factorization enable good predictive accuracy. On the contrary, in a sparser setting such as DBbook, the introduction of exogenous *knowledge* increases the performance of the methods. Indeed, approaches exploiting side information, such as KARS, and methods based on GNNs and GCNs over heterogeneous graphs achieve the best results. Overall, SLIM is the best-performing baseline on ML1M while KGAT is the most competitive approach on DBbook. However, irrespective of this analysis, the proposed approach outperforms all baselines: gaps are statistically significant ($p < 0.01$) for all accuracy metrics except the F1 measure on ML1M. In all

Baseline	**Accuracy**					**Diversity**	**Novelty**
	MAP	**Precision**	**Recall**	**F1**	**nDCG**	**Gini**	**EPC**
Matrix Factorization and Deep Learning Techniques (no KARS)							
BPRMF	0.7486	0.7119	0.4554	0.4636	0.7861	0.0779	0.558
PureSVD	0.7553	0.7121	0.453	0.4615	0.7896	0.113	0.5856
Slim	*0.7852*	*0.7419*	***0.4612***	*0.4743*	*0.8189*	0.1897	*0.6521*
MultiVAE	0.7358	0.6958	0.4483	0.4552	0.7714	0.1445	0.5845
CFGAN	0.6229	0.6127	0.4151	0.414	0.6677	***0.2842***	0.5396
NGCF	0.5946	0.5929	0.4095	0.4057	0.6442	0.192	0.5229
LightGCN	0.5946	0.5929	0.4095	0.4057	0.6442	0.192	0.5229
KARS based on Graph Embeddings and Matrix Factorization Techniques							
AttributeItemKnn	0.7302	0.6911	0.4422	0.449	0.765	0.2257	0.6127
AttributeUserKnn	0.7499	0.714	0.4454	0.4557	0.784	0.1556	0.5957
KaHFM	0.7403	0.7018	0.449	0.4565	0.7763	0.1468	0.5865
CKE	0.7222	0.7057	0.4179	0.4455	0.7535	0.1834	0.5817
CFKG	0.7318	0.7102	0.4193	0.447	0.7604	0.1732	0.5857
KTUP	0.7093	0.6926	0.4117	0.4381	0.7397	0.1743	0.5729
KARS based on Graph Neural Networks and Graph Convolutional Networks							
KGAT	0.7263	0.7064	0.4172	0.445	0.7559	0.1765	0.5812
KGCN	0.7164	0.701	0.4171	0.4439	0.7477	0.1899	0.5814
KGNN-LS	0.7152	0.6998	0.4157	0.4427	0.7462	0.1884	0.5796
Our best	**0.8454****	**0.8082****	0.4602	**0.5432****	**0.8712****	0.1307	**0.6657****

TABLE 4.1: Comparison to baselines on ML1M data. Best-performing baseline is reported in *italics*. Overall best performing configuration is highlighted in **bold**. Significant improvements are emphasized with * (for $p < 0.05$) and ** (for $p < 0.01$).

other cases, a significant improvement is obtained. Regarding novelty and diversity, other interesting findings emerge: for DBBOOK, a significant improvement ($p < 0.05$) is obtained for diversity and a non-significant improvement in the novelty of the recommendations, while an opposite behavior is observed on ML1M, where the proposed approach significantly increases the novelty of the recommendations at the cost of lower diversity. Further investigations will be conducted to better understand this behavior. **Overall, we can answer RQ4.1.1 by stating that our approach significantly outperforms all families of baselines in nearly all experimental settings.**

RQ4.1.2: Ablation Tests To answer RQ4.1.2, ablation tests were conducted on several variants of the best-performing configuration. In particular, four different architectures were evaluated: this was achieved by removing, one at a time, the graph encoding module, the text encoding module, the attention mechanisms,

Baseline	**Accuracy**					**Diversity**	**Novelty**
	MAP	**Precision**	**Recall**	**F1**	**nDCG**	**Gini**	**EPC**
Matrix Factorization and Deep Learning Techniques (no KARS)							
BPRMF	0.6392	0.5721	0.4182	0.4472	0.6372	0.0360	0.6626
PureSVD	0.6365	0.5714	0.4182	0.4468	0.6349	0.0399	0.6622
Slim	0.6240	0.5661	0.4176	0.4437	0.6285	0.0482	0.6556
MultiVAE	0.6222	0.5631	0.4126	0.4400	0.6245	0.0451	0.6519
CFGAN	0.6104	0.5609	0.4124	0.4391	0.6166	0.0507	0.6458
NGCF	0.6074	0.5591	0.4114	0.4377	0.6150	0.0511	0.6448
LightGCN	0.6147	0.5643	0.4156	0.4420	0.6212	0.0462	0.6490
KARS based on Graph Embeddings and Matrix Factorization Techniques							
AttributeItemKnn	0.6316	0.5674	0.4176	0.4443	0.6333	0.0496	0.6602
AttributeUserKnn	0.6371	0.5700	0.4191	0.4463	0.6363	0.0452	0.6629
KaHFM	0.6384	0.5745	0.4207	0.4493	0.6381	0.0354	0.6633
CKE	0.684	0.6718	0.5203	0.5418	0.7189	0.1309	0.6645
CFKG	0.6844	0.6713	0.5201	0.5421	0.7173	0.1235	0.664
KTUP	*0.6845*	0.6712	*0.5218*	0.5424	*0.719*	0.1041	*0.6646*
KARS based on Graph Neural Networks and Graph Convolutional Networks							
KGAT	0.682	*0.6723*	0.5206	*0.5427*	0.7182	0.1361	0.6661
KGCN	0.6776	0.6664	0.5171	0.5378	0.7131	*0.1562*	0.6614
KGNN-LS	0.6743	0.668	0.518	0.5389	0.7116	0.1548	0.6604
Our best	**0.7291****	**0.7024****	**0.5579****	**0.6335****	**0.7667****	**0.1668***	0.7058

TABLE 4.2: Comparison to baselines on DBBOOK data. Best-performing baseline is reported in *italics*. Overall best performing configuration is highlighted in **bold**. Significant improvements are emphasized with * (for $p < 0.05$) and ** (for $p < 0.01$).

Configuration	**Accuracy**					**Diversity**	**Novelty**
	MAP	**Precision**	**Recall**	**F1**	**nDCG**	**Gini**	**EPC**
Our best	**0.8454**	**0.8082**	0.4602	**0.5432**	**0.8712**	0.1307	**0.6657**
Architecture							
w/o graph	0.8397 (*)	0.8031 (**)	0.4587	0.541	0.865 (*)	0.1437	0.6671 (**)
w/o text	0.8415	0.8058	0.4587	0.5415	0.8675	0.1474	0.6675
w/o attention	0.8334 (*)	0.7996 (**)	0.4576	0.5395	0.8605 (*)	0.1333	0.6595
w/o dropout	0.8394 (*)	0.805	0.4582	0.5408	0.8661	0.1447	0.6687
Features							
w/o properties	0.8429	0.8048 (*)	0.4587	0.5413	0.8682	0.1265	0.6632
w/o prop & text	0.8368	0.8043 (*)	0.4583	0.541	0.8639	**0.1600** (*)	0.6715 (*)

TABLE 4.3: Results of the ablation test on ML1M data. Best-performing configuration is reported in **bold**. Each line indicates the module which is removed from the architecture. Significant improvements are emphasized with (*) (for $p < 0.05$) and (**) (for $p < 0.01$).

and the dropouts. In the first two cases, the graph encoding module or the text encoding module was removed from the architecture presented in Figure 4.1. In the third case, Equation 4.2 was not considered, and in the fourth case, self-attention and cross-attention were replaced with a simple concatenation of the embeddings. Next, performance was evaluated by removing some of the side information encoded in the model. In particular, structured properties were removed first, and then textual content was removed. In the first case, this was obtained by exploiting the interaction graph $\mathcal{IG}$ introduced in Section 2.3, without considering the descriptive properties coming from the knowledge graph $\mathcal{KG}$. In the latter case, the text encoder was also removed from the architecture presented in Figure 4.1. Results are presented in Tables 4.3 and 4.4.

As regards ML1M, the best-performing configuration achieves the overall performance in the ablation tests as well. Indeed, removing the different components of the architecture or varying the group of content-based features encoded in the model leads to a decrease in performance. In particular, graph-based information emerges as the most relevant source for this dataset. By removing graph encoding based on GCNs, a significant decrease in MAP ($p < 0.05$) and in Precision ($p < 0.01$) is observed. A significant decrease also emerges for novelty. Conversely, removing the text encoder yields a decrease in performance, but it is not statistically significant. Based on the analysis presented to answer **RQ4.1.2**, this is not surprising, since it has been shown that methods that do not use content-based features (such as those based on matrix factorization and deep learning) perform

Configuration	Accuracy					Diversity	Novelty
	MAP	Precision	Recall	F1	nDCG	Gini	EPC
Our best	**0.7291**	**0.7024**	**0.5579**	**0.6335**	**0.7667**	**0.1668**	**0.7058**
				Architecture			
w/o graph	0.6968 (**)	0.6842 (**)	0.5416 (**)	0.6151 (**)	0.7373 (**)	0.2481	0.6822 (**)
w/o text	0.7022 (**)	0.6883 (**)	0.5463 (**)	0.6204 (**)	0.7418 (**)	0.2261	0.6867 (**)
w/o attention	0.7052 (**)	0.6909 (**)	0.5479 (**)	0.6239 (**)	0.7459 (**)	0.2293	0.6900 (**)
w/o dropout	0.7042 (**)	0.6916 (**)	0.5502 (*)	0.6242 (*)	0.7464 (**)	0.2306	0.6900 (**)
				Features			
w/o properties	0.7231 (**)	0.6975 (**)	0.5565 (*)	0.6314 (*)	0.7619	0.1935	0.7019 (**)
w/o prop & text	0.6875 (**)	0.6814 (**)	0.5409 (**)	0.6145 (**)	0.7325 (**)	0.2410	0.6767 (**)

TABLE 4.4: Results of the ablation test on DBBOOK. Best-performing configuration is reported in **bold**. Each line indicates the module which is removed from the architecture. Significant improvements are emphasized with (*) (for $p < 0.05$) and (**) (for $p < 0.01$).

better than KARS on this data. Next, an analysis of the network elements shows that *attention mechanisms* significantly improve nDCG, Precision, and MAP. This is a very interesting finding that confirms one of the underlying intuitions of the strategy. Attention mechanisms were used to refine the representations learned by the graph and sentence encodings and to combine both information sources into a single representation. These results provide evidence of the effectiveness of the strategy. **The introduction of dropout significantly increases the MAP of the recommendations.**

As regards the contribution of the different families of features, **results show that removing descriptive features and textual content leads to a significant decrease in the precision of the recommendations.** As shown in Table 4.1, this comes at the cost of an increase in non-accuracy metrics such as novelty and diversity. Indeed, in the absence of textual content, the scores of these properties are significantly higher. **Accordingly, the introduction of content-based features has a positive impact on accuracy metrics and a negative impact on diversity and novelty.**

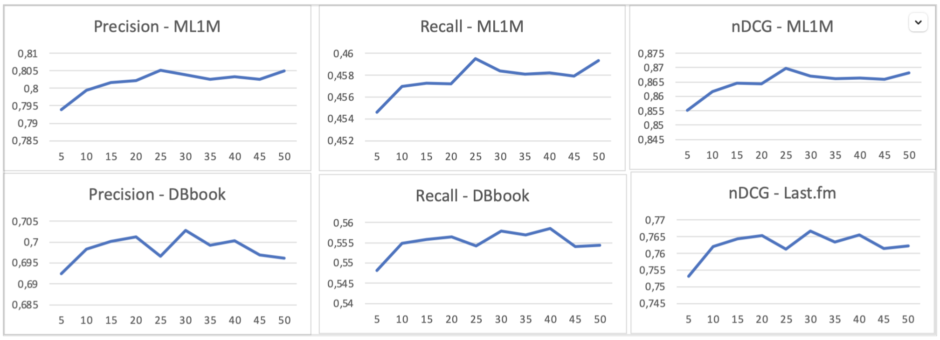
Finally, the results collected for DBBOOK can also be analyzed. Here, the discussion can be shortened since the ablation tests showed that each of the components included in the architecture has a significant impact on the overall performance of the systems. This holds for all families of features and for all elements of the model. This finding confirms the design choices. In particular, it is shown that combining heterogeneous embeddings and exploiting attention mechanisms to improve the representations can provide very good performances.

RQ4.1.3: Sensitivity Analysis To answer RQ4.1.3, the results of the sensitivity analysis based on the parameters introduced in Section 4.1.3 are presented. Due to space constraints, the results for Precision, Recall and nDCG are reported, but the outcomes are valid for the other metrics as well. As for the number of epochs (see Figure 4.2a), the top performance is obtained at 25 epochs for ML1M and 30 epochs for DBBOOK. In particular, for ML1M, the small increase observed at 50 epochs is not justified by the increase in training time needed to train the model. These values were used as parameters in the configurations previously presented. Next, the analysis of the dropout ratio, presented in Figure 4.2b, shows a different behavior on the datasets: for ML1M, best performances are obtained at 0.4 and a swinging behavior is noted for higher values. Conversely, for DBBOOK, performance peaks at 0.7, which was used as the dropout ratio in the best-performing configuration. The performance of the model for varying graph embedding sizes is presented in Figure 4.2c. In this case, the peak for DBBOOK is obtained at a dimension of 384 for all metrics, while a dimension of 768 yields the best results for ML1M. However, such an improvement is not statistically significant and is not justified by the increase in training time. Accordingly, 384 is chosen as the dimension of graph embeddings for ML1M as well.

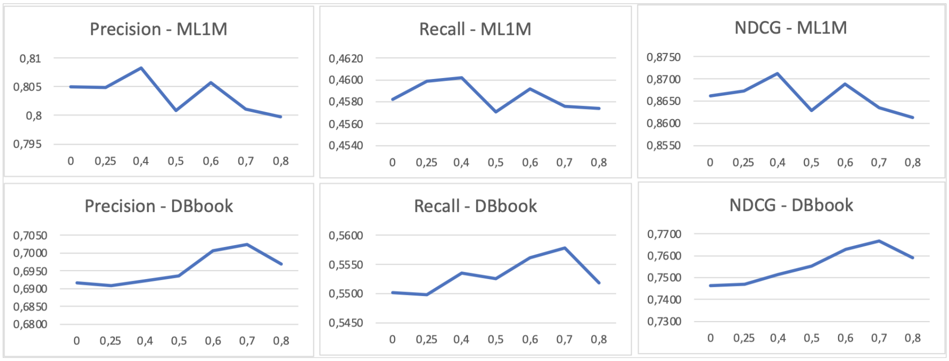
Finally, in Figure 4.2d the performance of the strategy across varying sentence embedding techniques is compared. In particular, five different strategies based on TRANSFORMERS are compared to feed SBERT. As shown in the figure, none of the techniques significantly outperforms the others, with the exception of small gaps in favor of Distillbert for some of the metrics. Accordingly, the choice of adopting MiniLM as embedding technique is justified. Indeed, thanks to the smaller dimension of the embeddings (384, instead of 768 for Albert, MPNet and Distillbert and 1024 for RoBERTa), these embeddings are easier to learn and exploit in the model.

4.1.4 Take-Home Messages

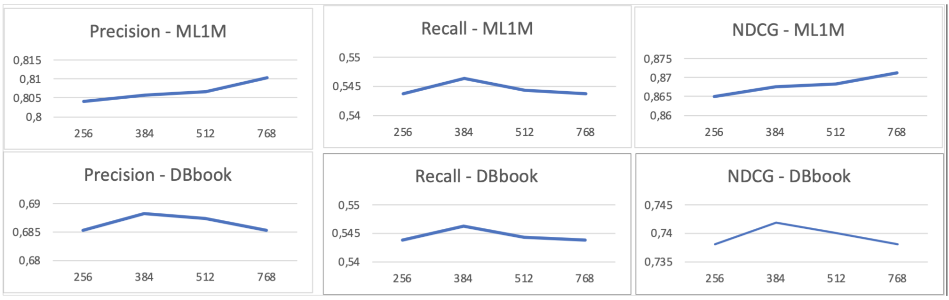
The featured publication [168] presents a KARS based on GNNs and Transformers. In particular, GCNs are used to encode collaborative information and structured features of the items, and a sentence encoding strategy based on SBERT is employed to learn a content-based representation. Such embeddings are then combined within a deep neural network by employing self-attention and cross-attention mechanisms, and the resulting representation is then used to predict a user's interest in unrated items. In the experimental evaluation, the performance of the approach is compared with competitive baselines, and the results indicate



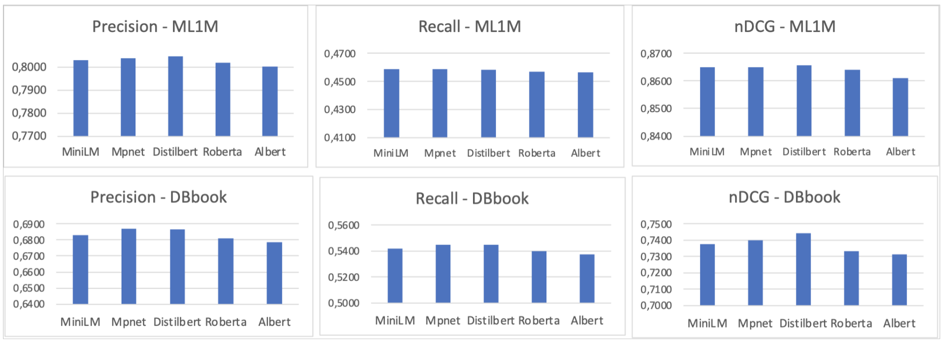
(A) Number of Training Epochs



(B) Dropout Ratio



(C) Dimension of Graph Embeddings



(D) Word Embedding Techniques

FIGURE 4.2: Sensitivity Analysis - Performances of Precision, Recall and NDCG on the datasets on varying of different hyper-parameters.

a significant improvement in accuracy that is consistent across different datasets. Moreover, ablation tests show that the different encoding strategies, as well as the exploitation of attention mechanisms, contribute equally to the overall performance, especially in recommendation settings where the data are more sparse.

Main Findings

- **RQ4.1.1**: Fusing KG embeddings with sentence encoders showed better recommendation performance with respect state-of-the-art baselines.
- **RQ4.1.2**: Removing KG embeddings and textual embeddings leads to a significant reduction of the performance. The application of the dropout leads to performance improvements as well.
- **RQ4.1.3**: The sensitivity analysis confirmed the choices in terms of KG embedding dimension, textual encoders and dropout.

The results discussed in this section have presented in a paper produced during the PhD and published at ACM UMAP 2023 in the Main Track (Giuseppe Spillo et al. "Combining graph neural networks and sentence encoders for knowledge-aware recommendations". In: *Proceedings of the 31st ACM Conference on User Modeling, Adaptation and Personalization*. 2023, pp. 1–12), which was presented at the main session as a full paper, and was awarded with the **James Chen Best Student Paper Award**.

4.2 Content-based Pre-training Strategies for KARS

4.2.1 Introduction

In the featured publication [169], a novel methodology for knowledge-aware recommendation is proposed that integrates content-based and graph-based representations in a unified learning process. Since the current KARS literature typically relies exclusively on either KGs or unstructured textual content, or combines them only after separate training, an alternative strategy is introduced that allows these heterogeneous data sources to interact more meaningfully during the learning phase. The approach begins by generating Pre-Trained Content-Based (PTCB) embeddings for users and items using textual descriptions sourced from

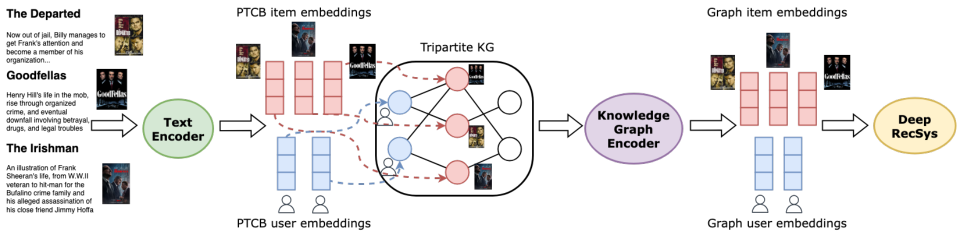

The Departed
Now out of jail, Billy manages to get Frank's attention and become a member of his organization...
Goodfellas
Henry Hill's life in the mob, rise through organized crime, and eventual downfall involving betrayal, drugs, and legal troubles
The Irishman
An illustration of Frank Sheeran's life, from W.W.II veteran to hit-man for the Bufalino crime family and his alleged assassination of his close friend Jimmy Hoffa
Text Encoder
PTCB item embeddings
PTCB user embeddings
Tripartite KG
Knowledge Graph Encoder
Graph item embeddings
Graph user embeddings
Deep RecSys

FIGURE 4.3: The workflow of our work

Wikipedia, which serve as semantically rich initializations for graph-based learning. Unlike conventional GCN-based models that initialize node embeddings randomly and ignore textual information during message passing, the PTCB embeddings are injected directly into the Graph Convolutional Network (GCN) as the starting point. This enables the GCN to refine the initial textual semantics by propagating them over the structure of the knowledge graph, allowing both sources of information—text and structured links—to be jointly modeled from the beginning of the learning process.

Accordingly, the approach aims to overcome the limitations of previous KARS that treat content and graph signals independently, ensuring that the knowledge embedded in textual content is preserved and propagated through the graph in a meaningful way. The resulting graph-aware, text-enhanced embeddings are used to train a deep recommendation model capable of accurately predicting user preferences and generating personalized top-k recommendation lists. To evaluate the effectiveness of the approach, extensive experiments are conducted on two real-world benchmark datasets, comparing the method to several strong baselines and conventional integration strategies. The results demonstrate that the model consistently achieves superior performance across all tested settings, validating the core hypothesis that a deep integration of textual and graph-based information leads to more effective representations and improved recommendation quality.

These results have been carried out during the PhD and published at ACM UMAP 2024 in the Main Track (Giuseppe Spillo et al. "Evaluating Content-based Pre-Training Strategies for a Knowledge-aware Recommender System based on Graph Neural Networks". In: *Proceedings of the 32nd ACM Conference on User Modeling, Adaptation and Personalization*. 2024, pp. 165–171) as a short paper, presented during the poster session.

4.2.2 Methodology

In this Section, the modules that composes the recommendation architecture are discussed, that is depicted in Figure 4.3.

Content-based Text Encoder In this work, PTCB embeddings are learned by exploiting WIKI2VEC [236], an extension of WORD2VEC [117] adapted to Wikipedia. In particular, instead of learning a *word* embedding as in WORD2VEC, WIKI2VEC learns an embedding for each Wikipedia *page*.

This is done by designing a model that combines three knowledge sources: the content of the page, the text of the anchors (hyperlinks) and the links that connect Wikipedia pages. All these elements are jointly optimized through word-based skip-gram of the content of each page, the anchor context (aiming to capture in which contexts other pages are relevant), and a link graph model (similar to graph embeddings) applied to the graph built upon the hyperlinks between pages.

In particular, the first task performed by *wiki2vec* is the well-known skip-gram model, which learns word embeddings by predicting neighboring words for a specific term, and this is performed considering the textual content of each Wikipedia page. The second task consists of predicting the surrounding words of a hyperlink pointing to entities, so as to capture in which context (the surrounding words) other entities are mentioned. This way, the model is able to capture the context in which connected entities (other Wikipedia pages) are mentioned. The third task consists of learning entity embeddings from a graph built by exploiting hyperlinks pointing between Wikipedia pages. To do this, first, a Wikipedia hyperlink graph is built: here, nodes are Wikipedia entities (pages) and edges represent the presence of hyperlinks from the source page to the target page. Then, entity embeddings are learned by predicting the edges in the graph, following a similar graph embedding process [19]. Following this methodology, *wiki2vec* is able to learn both content-based Wikipedia word embeddings $\vec{word}_t$ and content-based Wikipedia entity embeddings $\vec{ent}_t$.

The resulting representation of a Wikipedia page encodes both content-based information coming from the page itself, as well as some information emerging from the link structure of Wikipedia. For the purposes of this study, all $i \in \mathcal{I}$ can be mapped to a Wikipedia page, since it is reasonable to state that a Wikipedia page describing the item exists[3]. Accordingly, the content-based embedding based on Wikipedia can be learned for each item[4].

Formally: $\vec{i_t} = wiki2Vec(i)$; Next, to learn user embeddings $\vec{u_t}$, the centroid of the PTCB item embeddings liked by user u is used, exploiting the information encoded in the interaction matrix $\mathcal{Y}$ (see Section 2.3 for more details about this process):

$$\vec{u_t} = \frac{\sum_{i \in L} \vec{i_t}}{|L|}$$

where $L = \{i \in I | r_{u,i} = 1\}$ is the set of the items liked by the users.

Knowledge Graph Encoder. For this work, a GCN was used as the Knowledge Graph Encoder to learn user and item graph embeddings $\vec{u_g}$ and $\vec{i_g}$ starting from the tripartite knowledge graph $\mathcal{G}$.

Although the encoding model described in Chapter 2 was adopted, recalling some underlying assumptions is worth to better frame this work. See Section 2.3 for the problem formalization.

Given a node $v \in V$, let $N(v)$ be the set of its one-hop nodes and let $r \in R$ be the relation connecting v to a neighbor $u \in N(v)$. Then, the d-dimensional embedding $\vec{v}^{k+1} \in R^d$ of the node v at the $k+1$-th layer is given by:

$$\vec{v}^{k+1} = \sum_{(u,r) \in N(v)} W^k_{\lambda(r)} \Phi \left(\vec{u}^k, \vec{r}^k \right) \tag{4.8}$$

where $\vec{u}$ and $\vec{r}$ are the representations of node u and relation r, respectively, Φ is a composition operator, $W_{\lambda(r)}$ is a (learnable) weight matrix, and $\lambda(r)$ is a function that returns the direction of the relation r.

While in conventional approaches $\vec{v}^0 = \text{random}()\ \forall v \in V$, in this work the graph embeddings are initialized as follows:

$$\vec{v}^0 = \begin{cases} \vec{u_t} & \text{if } v \text{ is a user node} \\ \vec{i_t} & \text{if } v \text{ is an item node} \\ random() & \text{otherwise} \end{cases} \tag{4.9}$$

[3]For example, the movie *The Departed* is mapped to the Wikipedia page `https://en.wikipedia.org/wiki/The_Departed`

[4]The label 'content-based' embedding is more intuitive and is used.

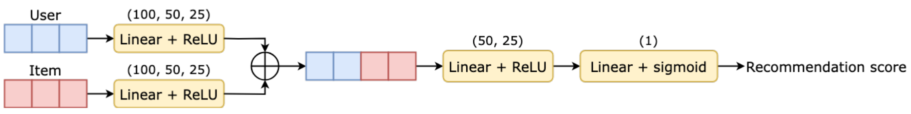


FIGURE 4.4: The Deep RS architecture. User and item embeddings are projected onto separate linear layers with ReLU as the activation function, then concatenated. Subsequently, a linear layer with a sigmoid activation predicts the recommendation score, which lies in [0,1].

that is, if a node is associated with a user or item, a PTCB embedding learned by the Text Encoder is used as the initial value.

Following this strategy, the content-based information (encoded in the PTCB embeddings) is combined with the complementary information encoded in the KG, learned by the GCN. In other words, the graph embeddings are *aware* of the knowledge provided by the textual content and encoded in the PTCB embeddings. The output of the Knowledge Graph Encoder is represented by user and item graph embeddings $\vec{u_g}$ and $\vec{i_g}$.

Deep Recommender System. The resulting graph embeddings, aware of content-based information $\vec{u_g}$ and $\vec{i_g}$, are used to predict users' interest in items through a deep neural network. The RS architecture is shown in Figure 4.4.

First, both embeddings are projected onto separate smaller layers with the ReLU activation function. Due to space constraints, the formalization of only one is reported:

$$\begin{aligned} \vec{u} &= ReLU\left(W_{u_p} * \vec{u_g} + b_u\right), \\ \vec{i} &= ReLU\left(W_{i_p} * \vec{i_g} + b_i\right) \end{aligned} \tag{4.10}$$

where W_{u_p} and W_{i_p} are trainable parameters, and b_u and b_i are biases. Then, the resulting embeddings are concatenated and projected onto smaller layers with ReLU as the activation function (similarly to Equation 4.10), to obtain a single embedding $\vec{e}$, which is finally used to predict a recommendation score $score_{rec}$ through a sigmoid function:

$$\begin{aligned} \vec{e} &= ReLU\left(W_{concat} * \left(\vec{u} \oplus \vec{i}\right) + b_{concat}\right) \\ score_{rec} &= \sigma\left(W_{rec} * \vec{e} + b_{rec}\right) \end{aligned} \tag{4.11}$$

where W_{concat} and W_{rec} are trainable parameters, b_{concat} and b_{rec} are biases, $\oplus$ is the concatenation operator, and σ is the sigmoid activation function that returns a recommendation score $score_{rec} \in [0, 1]$. This architecture is trained on a subset of the ratings in $\mathcal{Y}$ using binary cross-entropy as the loss function. Note that $score_{rec}$ corresponds to $\widetilde{Y}$ introduced previously, which predicts the interest of user u in item i. In this setting, the score is computed for all items in the test set. The items are thereafter ranked according to this score, and the top-k are returned as recommendations.

4.2.3 Problem Statement

This section extends the problem formalization introduced in Section 2.3 with the only information that characterize this work. Please refer to Section 2.3 and to the published paper [169] for more deatils.

Embedding Learning In this work, a Text Encoder is first exploited to learn PTCB embeddings for each user u and for each item i from the available content-based information. Let these embeddings be $\vec{u_t}$ and $\vec{i_t}$. Then, the interaction graph $\mathcal{IG}$ is merged with $\mathcal{KG}$, exploiting the overlapping nodes (the items) to obtain a tripartite KG $\mathcal{G}$ that encodes both user interactions and item properties.

Next, a Knowledge Graph Encoder is used to learn user and item embeddings starting from $\mathcal{G}$. A key point of the approach is that, in this phase, user and item embeddings are not initialized with random values (as in conventional approaches), but the PTCB embeddings learnt by the Text Encoder, $\vec{u_t}$ and $\vec{i_t}$, are exploited to combine the complementary information provided by the text and the KG.

Let $\left(\vec{u_g}, \vec{i_g}\right) = KGE\left(\mathcal{G}, \vec{u_t}, \vec{i_t}, W_g\right)$ denote the embeddings learnt by the Knowledge Graph Encoder KGE, whose parameters are the tripartite KG $\mathcal{G}$, the initial embeddings $\vec{u_t}, \vec{i_t}$ learnt by the Text Encoder, and W_g that is a learnable parameter.

Finally, these embeddings are used to feed a neural network (the Deep Recommender System), which uses them in order to predict whether a user u likes an item i. Let θ denote the parameters of the network.

4.2.4 Experimental Evaluation

Experiments were designed to answer the following Research Questions (RQs):

- **RQ4.2.1:** Does content-based pre-training improve the accuracy of KARS based on GNNs?

- **RQ4.2.2:** How does the approach perform with respect to popular baselines?
- **RQ4.2.3:** How do different hyper-parameter values affect the performance of the approach?

Experiments were conducted in a *movie, book and music recommendation* scenario. In particular, the ML1M, DBbook and Last.fm datasets have been considered. More details about this datasets can be found in Section 2.3, where the problem has been formalized. Table 4.5 reports the coverage of the PTCB item embeddings, which was missing in the Section 2.3.

Dataset	Users	Items	Ratings	KG items	KG entities	KG rels	KG triples	PTCB item coverage
ML1M	6036	3081	946120	2735	17718	13	70668	3008 (93.7%)
DBBOOK	5660	6617	129316	1963	5207	39	16589	5081 (75.8%)

TABLE 4.5: Statistics of the datasets.

Evaluation Metrics For this work, precision, recall, F1, and NDCG, and MAP were considered as evaluation metrics, computed by using the ClayRS framework [109]. Statistical significance was assessed using a t-test (P-value = 0.05).

Baselines Several baselines belonging to different model groups were considered, and are reported in the tables. Their description is discussed in Section 2.3, and is not reported here to avoid repetitions.

4.2.5 Results and Discussion

RQ4.2.1: Comparison between different pre-training approaches To address RQ4.2.1, three configurations were compared: (i) both user and item embeddings are initialized with PTCB embeddings (full model); (ii) only item embeddings are initialized with PTCB embeddings (only items pre-trained). And (iii) the conventional configuration without pre-training (no pre-training). Results are presented in Table 4.6, and show improvements across all considered metrics for both datasets. **The gaps in performance between the conventional model and the proposed approach are larger for DBbook than for ML1M, which may be due to DBbook having a lower number of user-item interactions (Table 4.5). Hence, it could benefit more from the use of PTCE embedding, which already encodes meaningful semantics, rather than ML1M, which can rely on a higher number of interactions and a richer KG.** In addition, the improvements observed in DBbook are statistically significant in almost all cases.

Configuration	**Precision**	**Recall**	**F1**	**nDCG**
		ML1M		
users and items pre-trained (full model)	**0.7958**	**0.4547**	**0.5363**	**0.8571**
only items pre-trained	0.7941	0.4538	0.5352	0.8541*
no pre-training	0.7941	0.4542	0.5357	0.8550
		DBbook		
users and items pre-trained (full model)	**0.7003**	**0.5562**	**0.6315**	**0.7601**
only items pre-trained	0.6989	0.5529*	0.6283*	0.7595
no pre-training	0.6962**	0.5521**	0.6272*	0.7569

TABLE 4.6: Comparison between the full model, in which both user and item embeddings are initialized with PTCB embeddings, a variant in which only item embeddings are initialized with PTCB embeddings, and the conventional setting (no pre-training). Best results are in **bold**; (*) and (**) indicate statistically significant differences with p-value < 0.05 and < 0.01, respectively.

Baseline	**Precision**	**Recall**	**F1**	**nDCG**
		General CF models		
BPR	0.7205	0.4214	0.4944	0.7741
ItemKNN	0.7239	0.4242	0.4974	0.7773
		Graph-based models		
LightGCN	0.7289	0.4238	0.4977	0.7838
NGCF	0.7157	0.4209	0.4932	0.7665
SGL	0.7175	0.4199	0.4926	0.7666
		Knowledge-aware models		
CFKG	0.7326	0.4268	0.5010	0.7872
CKE	0.7120	0.4188	0.4908	0.7634
KGAT	0.7134	0.4195	0.4917	0.7642
KGCN	0.7343	0.4279	0.5024	0.7859
KGNNLS	0.7364	0.4281	0.5029	0.7888
Full model	**0.7958****	**0.4547****	**0.5363****	**0.8571****

TABLE 4.7: Baseline comparison on ML1M. Best results in **bold**; (**) means $p < 0.01$.

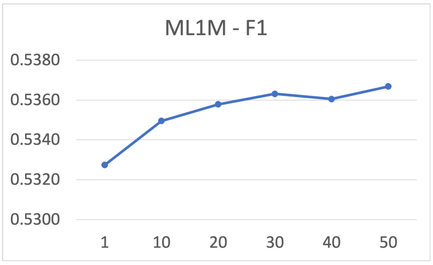
RQ4.2.2: Comparison to baselines. To answer RQ4.2.2, the full model was compared with several baselines described in the previous section. The results of the comparison are reported in Tables 4.7 and 4.8. As can be observed, the full

Baseline	**Precision**	**Recall**	**F1**	**nDCG**
General CF models				
BPR	0.6720	0.5239	0.5971	0.7204
ItemKNN	0.6637	0.4819	0.5534	0.6835
Graph-based models				
LightGCN	0.6754	0.5255	0.5992	0.7225
NGCF	0.6738	0.5254	0.5989	0.7205
SGL	0.6711	0.5210	0.5944	0.7183
Knowledge-aware models				
CFKG	0.6742	0.5252	0.5986	0.7231
CKE	0.6721	0.5242	0.5974	0.7183
KGAT	0.6722	0.5231	0.5965	0.7199
KGCN	0.6719	0.5225	0.5958	0.7171
KGNNLS	0.6723	0.5238	0.5971	0.7188
Full model	**0.7003****	**0.5562****	**0.6315****	**0.7601****

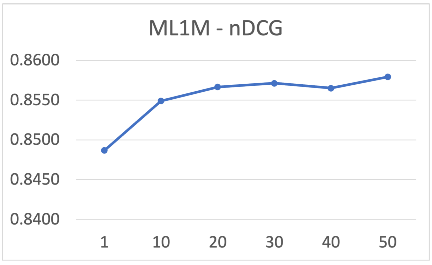
TABLE 4.8: Baseline comparison on DBBOOK. Best results in **bold**; (**) means $p < 0.01$.

model significantly outperforms all the considered baselines, and the improvements are also statistically significant (with $\alpha = 0.01$). An interesting observation can be made for DBbook, where the best overall baseline is LightGCN: indeed, this graph-based model relies only on CF information, and it outperforms all the knowledge-aware baselines for almost all metrics; **This confirms the relevance of user-item interactions in DBbook, bringing further evidence that, in such case, the semantics provided by PTCB embeddings could be useful and should be further investigated, as previously discussed.**

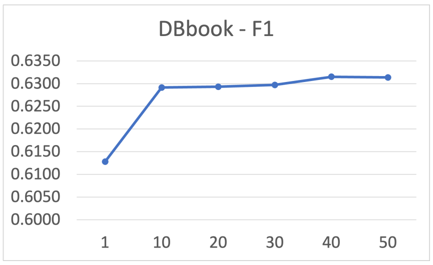
RQ4.2.3: Sensitivity Analysis To answer RQ4.2.3, the performance of the RS models was assessed in terms of F1 (summarizing precision and recall) and nDCG at varying numbers of epochs (from 1 to 50, with a step 10). Results are shown in Figure 4.5. The best performance on ML1M was obtained with 50 epochs, and very similar results were observed when the model was trained for 30 epochs (this setting). For DBbook, the optimal number of epochs is 40, confirming the chosen settings. Moreover, it is observed that, in all cases, the full model overcomes both the configuration in which only item embeddings are initialized with PTCB and the configuration without pre-training.



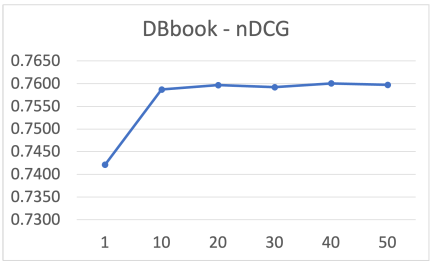
(A) F1 on ML1M



(B) nDCG on ML1M



(C) F1 on DBbook



(D) nDCG on DBbook

FIGURE 4.5: Sensitivity analysis on the F1 and nDCG metrics at the varying number of training epochs.

Discussion The adoption of PTCB embeddings improves the accuracy of a KARS based on GNNs, with performance surpassing several state-of-the-art techniques. This confirms the conjecture that the injection of content-based knowledge available in Wikipedia (encoded in PTCB) provides a more accurate representation of users and items, especially when the information encoded in KGs is scarce, as in DBbook. In particular, knowledge sources can be combined through pre-training, which is a poorly investigated direction. Further analyses need to be carried out, *i.e.*, by evaluating newer and more sophisticated text encoding strategies, such as those based on Transformers, or by injecting PTCB into other KARS currently existing in the literature.

4.2.6 Take-Home Messages

The featured publication [169] proposes a new approach to combine textual information with KGs to provide users with recommendations. To this end, PTCB embeddings are learned and used to initialize graph embeddings of a GCN. The resulting graph embeddings are aware of the textual content and encode the structured properties of items, and are used to feed a deep RS. The experiments on two datasets show the effectiveness of this approach, which appears to be particularly effective in scenarios with sparser interaction matrices and scarce KGs. Further investigations will be conducted: first, the coverage of textual content for all items will be improved, for example by exploiting knowledge sources providing this kind of data, such as DBpedia. Then, other strategies to learn PTCB embeddings (e.g., transformers) will be analyzed. Next, the effectiveness of this methodology in scenarios where KGs are not available at all will be assessed, since the results suggested useful insights in this direction. Finally, the impact of pre-trained embeddings on the overall sustainability [179] of the framework will also be studied.

Main Findings

- **RQ4.2.1**: Pre-training graph embeddings with content-based embedding is an effective strategy to improve recommendation performance in highly sparsity datasets.
- **RQ4.2.2**: This approach significantly outperformed several state-of-the-art baselines.
- **RQ4.2.3**: The sensitivity analysis confirmed the findings and the choices taken in the experimental design.

4.3 Recommendation Dataset with Multimodal Fusion: a Comprehensive Benchmark

4.3.1 Introduction

In recent years, research in RSs has seen a growing interest in multimodal recommendation [49, 48, 114]. Multimodal recommendation exploits multimodal side information about items to enrich the interaction matrix, alleviating its sparsity [46], and to better understand the content to be recommended [104]. These advantages, in general, lead to more accurate and precise recommendations [49, 48]. However, a common problem in this research line is the lack of multimodal side information associated with catalog items. Current research [114, 239, 259, 247, 258] typically adopts the Amazon Reviews datasets [75], that come with textual and visual raw features (the textual description of products and their related pictures). However, besides text and images, these datasets do not include other multimodal features. Conversely, in particular domains, different features might be more useful (e.g., audio signals in music recommendation or video signals in movie recommendation). Other datasets that have been adopted in the field are

TikTok and Kwai, for the micro-video recommendation task [240, 226, 190], which provide visual and textual modalities too. MovieLens-10M has also been used for the multimodal recommendation task [226], but the paper does not provide the raw multimodal features, making reproduction of the results or the use of this version of the dataset difficult[5].

However, several popular datasets commonly used in CF research, such as MovieLens-1M, DBbook, and Last.fm-2K, have received limited attention in multimodal recommendation. DBbook and Last.fm-2K have not been employed in this setting, despite the fact that their items (i.e., books and music) are well-suited to multimodal representation through side information such as text descriptions, cover images, or audio features. MovieLens-1M has been used in prior multimodal recommendation works [226], but the modalities considered were limited (e.g., few types of side information and encoders), and the multimodal features employed were never made publicly available. This represents a significant limitation, as MovieLens-1M is one of the most widely used datasets in recommender systems research [204]. Releasing a multimodal version of this dataset would not only support more consistent and reproducible evaluations but also build on MovieLens-1M's long-standing role as a benchmark. This would make it possible to trace the development of recommender models over time and directly compare traditional CF approaches with newer multimodal methods.

In this context, the following contributions are provided by this study: *i)* This study extends three popular datasets — *MovieLens-1M*, *Last.fm-2K*, and *DBbook* — from the movie, music, and book recommendation domains, respectively, by incorporating multimodal information, and provides links to both the raw data and pre-trained features obtained using various state-of-the-art encoders. *ii)* When available, not only the visual and the textual multimodal features (as in most of the research in the field) are considered, but also acoustic and video features as well, all at the same time; *iii)* A benchmark analysis of different state-of-the-art multimodal recommendation models on this dataset is performed using a state-of-the-art framework, in order to foster the reproducibility of the results and assess the contribution of multimodal features in this setting as well.

The results show that multimodal information can further improve the quality of the recommendations in these domains compared to single collaborative filtering. The multimodal version of such datasets is released to foster this line of research,

[5]The links reported in the paper and on the platform `paperswithcode.com` redirect to the original GroupLens page in which the original MovieLens dataset was released, but multimodal features have never been publicly shared.

Modality	*Encoders*
Text	MiniLM [217], MPNet [162]
Image	ViT [54], ResNet152 [68], VGG [160]
Audio	VGGish [72], Whisper [142]
Video	R(2+1)D [195], I3D [28]

TABLE 4.9: Encoders we used per modality

including links to download the raw multimodal files and the encoded item features.

For more details about how these datasets have been extended, please refer to Section 2.3, where the datasets used thought this thesis have been introduced.

These results have been conducted during the PhD and published at ACM RecSys 2025 in the Reproducibility Track (Giuseppe Spillo et al. "See the Movie, Hear the Song, Read the Book: Extending MovieLens-1M, Last. fm-2K, and DBbook with Multimodal Data". In: *Proceedings of the Nineteenth ACM Conference on Recommender Systems*. 2025, pp. 847–856).

4.3.2 Experimental Session

To assess the effectiveness of the multimodal features collected and described in Section 2.3, raw data were encoded using state-of-the-art encoders, and a benchmark analysis was conducted using the well-known MMRec [257] multimodal recommendation framework. Experiments were conducted to answer the following Research Questions (RQs):

- **RQ4.3.1**: Can multimodal features improve recommendation performance compared to CF-based models?
- **RQ4.3.2**: How do different modalities and encoders affect recommendation performance?

Datasets Experiments were conducted in a *movie, book and music recommendation* scenario. In particular, the ML1M, DBbook and Last.fm datasets have been considered. More details about this datasets, including the item coverage after the extension with multimodal features, can be found in Section 2.3, where the problem has been formalized.

Multimodal Encoders All multimodal raw features were encoded with state-of-the-art encoders. The repository at [6] reports detailed instructions to create the environment used and the scripts run to encode the multimodal raw features. Table 4.9 reports the encoders used, and a brief discussion is provided here. The repository also contains all details about the encoders, including instructions on how to set the environment used to run them and the scripts employed.

For **text**, the sentence transformers MiniLM [217] and MPNet [162] are considered, provided by the `sentence-transformers` library.

For **images**, ResNet152 [68] and VGG19 [160] (both pre-trained on the ImageNet-1K dataset and implemented in the `torchvision` library), as well as Visual Transformers (ViT) [54] (pre-trained on the ImageNet-21K dataset and implemented in the `transformers` library). For ViT, image embeddings are considered as both the embedding of the $[CLS]$ token—ViT CLS—and the embedding obtained as the mean of all patches in which the image was split—ViT AVG.

For **audio** signals, VGGish [72] (pre-trained on the AudioSet dataset and implemented in Tensorflow[7]) and Whisper [142] (for which the `base` weights for pre-training are considered).

For **videos**, R(2+1)D [195] (pre-trained on the Kinetics-400 dataset and provided by Moabitcoin[8]) and I3D [29] (pre-trained on the Kinetics-400 dataset and implemented in `torchvision`).

Multimodal Recommendation Models As recommendation models, a subset of those implemented in MMRec and adopted in current research [190, 247, 258, 239] was considered as baseline to guarantee fair comparisons. In particular, VBPR [69], Lattice [247], Freedom [258], MMGCN [226] and SLMRec [190] have been considered as multimodal models, while BPR [146] has been considered as CF basline. More details about these models can be found in the original paper [176] and in Section 2.3.

In the following, the aspects of the experimental setting that are not reported in Section 2.3 are discussed.

[6] `https://github.com/giuspillo/extending_sota_data_mm_feat`

[7] `https://github.com/tensorflow/models/tree/master/research/audioset/vggish`

[8] `https://github.com/moabitcoin/ig65m-pytorch`

Core-k filtering Core-k filtering was applied to the datasets following prior work [4, 257] in order to retain only users and items with at least k interactions. K was set to 5.

Multimodal Feature Encoding Multimodal features of the collected data were encoded with the encoders discussed previously to conduct the benchmark analysis with MMRec. For LFM-2K, in which more than one image/song per artist is available, the centroid of the available image/audio features was computed to obtain a single representation for the artist.

Filtering items As required by MMRec, all items for which not all modalities were available were filtered out (the resulting statistics are reported in Table 4.10). It is acknowledged that other strategies exist [113] (such as replacing missing modalities with random or zero embeddings), but their exploration is left for future work, as this is not the focus of the study.

Uni-modal experiments In this step, each recommendation model was run using individual encoded multimodal features, one at a time. This allowed the identification, for each model and modality, of the encoder that performed the best.

Multi-modal experiments Experiments were conducted by combining multiple modalities, using the best encoder for each modality and model, as identified in the uni-modal phase. This strategy maximized each model's performance while maintaining a manageable number of total experiments.

Evaluation metrics Model performance have been evaluated in terms of recall, NDCG, precision and MAP, with $k = \{10, 20\}$.

More details about model training, hyperparameter tuning, and hardware details can be found in the related publication [176].

4.3.3 Results Discussion

RQ4.3.1: Comparison with CF model To answer RQ4.3.1, in Tables 4.11, 4.12, and 4.13 we report the performance of the BPR model (only using CF signals), and the best results obtained for each combination of modality and encoder on the ML1M, DBbook, and LFM-2K datasets, respectively.

	ML1M	*LFM2K*	*DBbook*
CF data (5-core filtering) + multimodal filtering			
Users	6040	1859	6179
Items	3051	1425	4197
Interactions	946780	56346	121924

TABLE 4.10: Datasets used for the experiment, in which we kept only the items for which *all* the modalities are available.

Model	Modality	Encoder	rec@10	ndcg@10	map@10	prec@10	rec@20	ndcg@20	map@20	prec@20
BPR	-	-	0.1468	0.2093	0.1053	0.1689	0.2335	0.2273	0.1001	0.141
VBPR	audio	Whisper	0.1235	0.1821	0.0883	0.1512	0.1994	0.1972	0.0825	0.128
	image	ViT (AVG)	0.1179	0.1717	0.0821	0.1417	0.1899	0.187	0.077	0.1208
	text	MiniLM	0.132	0.1975	0.0982	0.1624	0.2123	0.2127	0.0917	0.1358
	video	I3D	0.1164	0.169	0.08	0.1407	0.1914	0.1852	0.0754	0.1199
LATTICE	audio	Whisper	0.1573	0.2211	0.1128	0.1796	0.2483	0.24	0.1076	0.1492
	image	ViT (CLS)	0.1562	0.2196	0.1117	0.1775	0.2481	0.2393	0.1069	0.1484
	text	MPNet	0.158	0.2218	0.1133	0.1792	**0.2494**	0.2415	0.1084	**0.1501**
	video	I3D	0.1565	0.2195	0.1114	0.1773	0.2487	0.2397	0.107	0.1488
FREEDOM	audio	Whisper	0.1591	0.2256	0.1162	**0.1821**	0.246	0.242	0.1098	0.1495
	image	ViT (CLS)	0.1585	0.2249	0.1157	0.1812	0.2465	0.2419	0.1096	0.1494
	text	MPNet	0.1589	**0.2258**	**0.1164**	**0.1821**	0.2461	**0.2422**	**0.11**	0.1495
	video	I3D	0.1592	0.2256	0.1161	0.1819	0.2464	0.2421	0.1099	0.1495
MMGCN	audio	Whisper	0.1423	0.2085	0.1056	0.1698	0.2257	0.2241	0.0989	0.1412
	image	ResNet152	0.1439	0.209	0.1057	0.1711	0.2291	0.2258	0.0997	0.1426
	text	MiniLM	0.1409	0.2057	0.1038	0.1675	0.2276	0.2237	0.0985	0.1414
	video	R(2+1)D	0.1441	0.2093	0.1058	0.172	0.2299	0.2264	0.1	0.144
SLMRec	audio	Whisper	**0.1604**	0.2036	0.1005	0.1552	0.2447	0.2235	0.0986	0.1244
	image	ResNet152	0.1521	0.1878	0.0906	0.1434	0.2329	0.2084	0.0899	0.116
	text	MiniLM	0.1478	0.1819	0.0876	0.1377	0.2285	0.2032	0.0874	0.112
	video	R(2+1)D	0.1566	0.1998	0.0982	0.1531	0.2399	0.2199	0.0963	0.1238

TABLE 4.11: ML1M *uni-modal* results. The best results per model are colored, while the best overall result is in **bold.**

Model	Modality	Encoder	rec@10	ndcg@10	map@10	prec@10	rec@20	ndcg@20	map@20	prec@20
BPR	-	-	0.1528	0.1132	0.0692	0.0426	0.2259	0.1384	0.0763	0.0316
VBPR	image	ViT (CLS)	0.106	0.0787	0.0467	0.0297	0.1545	0.0955	0.0509	0.0218
	text	MPNet	0.1489	0.1097	0.0663	0.0412	0.2176	0.1334	0.0728	0.0302
LATTICE	image	ResNet152	0.1769	0.133	0.0825	0.0493	0.2538	0.1595	0.0901	0.0356
	text	MiniLM	0.1799	0.1348	0.0836	0.05	**0.2555**	0.161	0.0911	**0.0358**
FREEDOM	image	VGG	0.1708	0.1281	0.0792	0.0476	0.2403	0.1522	0.0861	0.0338
	text	MiniLM	0.1707	0.1279	0.079	0.0475	0.2402	0.152	0.0859	0.0337
MMGCN	image	ViT (CLS)	0.1089	0.0762	0.0433	0.0307	0.1679	0.0967	0.0486	0.0239
	text	MPNet	0.117	0.0813	0.0458	0.0329	0.1841	0.1043	0.0518	0.0259
SLMRec	image	VGG	0.1863	0.1436	0.0907	0.0518	**0.2555**	0.1674	0.0977	0.0356
	text	MPNet	**0.187**	**0.1471**	**0.0942**	**0.052**	0.2553	**0.1707**	**0.101**	0.0357

TABLE 4.12: DBbook *uni-modal* results. The best results per model are colored, while the best overall result is in **bold.**

Model	Modality	Encoder	rec@10	ndcg@10	map@10	prec@10	rec@20	ndcg@20	map@20	prec@20
BPR	-	-	0.2669	0.2227	0.1293	0.1081	0.3905	0.2704	0.1466	0.0784
VBPR	audio	Whisper	0.2272	0.1875	0.1046	0.0929	0.3365	0.23	0.1195	0.0684
	image	ViT (AVG)	0.2172	0.1765	0.096	0.0905	0.3279	0.22	0.1116	0.0678
	text	MiniLM	0.2689	0.2241	0.1315	0.1076	0.3841	0.269	0.1478	0.0768
LATTICE	audio	Whisper	0.2852	0.2377	0.1399	0.1145	**0.408**	0.2853	0.158	**0.0814**
	image	ResNet152	0.2812	0.234	0.1373	0.1131	0.406	0.2823	0.1555	0.0811
	text	MPNet	**0.2888**	**0.2409**	**0.1423**	**0.1157**	0.4072	**0.2868**	**0.1598**	0.0812
FREEDOM	audio	VGGish	0.2766	0.231	0.1358	0.1106	0.4012	0.2793	0.1535	0.0799
	image	VGG	0.2768	0.2313	0.1362	0.1108	0.4004	0.2793	0.1537	0.0797
	text	MPNet	0.2719	0.2267	0.1326	0.1093	0.3967	0.2751	0.1502	0.079
MMGCN	audio	Whisper	0.2286	0.191	0.1081	0.0958	0.3382	0.2341	0.1233	0.0699
	image	ResNet152	0.236	0.1959	0.1109	0.0975	0.3456	0.2386	0.1261	0.0707
	text	MPNet	0.2325	0.1954	0.1118	0.0962	0.3537	0.2422	0.1284	0.0716
SLMRec	audio	VGGish	0.2561	0.2106	0.1218	0.1018	0.368	0.2544	0.1377	0.0735
	image	VGG	0.2419	0.2001	0.1148	0.0975	0.3541	0.2441	0.1308	0.0717
	text	MPNet	0.2364	0.1942	0.1109	0.0949	0.3459	0.2371	0.1261	0.0696

TABLE 4.13: Last.fm-2K *uni-modal* results. The best results per model are colored, while the best overall result is in **bold.**

Model	Modalities	Encoders	rec@10	ndcg@10	map@10	prec@10	rec@20	ndcg@20	map@20	prec@20
FREEDOM	text	MPNet	0.1589	0.2258	0.1164	0.1821	0.2461	0.2422	0.11	0.1495
VBPR	text + image	MiniLM + ViT (AVG)	0.1236	0.1841	0.0905	0.1519	0.2002	0.199	0.0841	0.1283
	text + audio	MiniLM + Whisper	0.1265	0.1849	0.0901	0.1518	0.2052	0.2011	0.0845	0.1283
	text + video	MiniLM + I3D	0.1251	0.1825	0.0886	0.1524	0.2039	0.199	0.0833	0.129
	image + audio	ViT (AVG) + Whisper	0.12	0.1792	0.0871	0.1483	0.1933	0.1927	0.0803	0.1246
	image + video	ViT (AVG) + I3D	0.1202	0.1745	0.0838	0.1445	0.1941	0.1897	0.0785	0.1222
	audio + video	Whisper + I3D	0.117	0.1733	0.0828	0.1444	0.1941	0.1897	0.0777	0.1235
LATTICE	text + image	MPNet + ViT (CLS)	0.1607	0.2263	0.1162	0.1826	0.2516	0.2446	0.1106	0.1512
	text + audio	MPNet + Whisper	0.1594	0.2257	0.1161	0.1823	0.2511	0.2446	0.1108	0.1515
	text + video	MPNet + I3D	0.1587	0.2229	0.1139	0.1785	0.252	0.2436	0.1098	0.1499
	image + audio	ViT (CLS) + Whisper	0.1603	0.2262	0.1163	0.1832	0.2509	0.2444	0.1106	0.1514
	image + video	ViT (CLS) + I3D	0.1609	0.2275	0.1171	0.1834	0.2517	0.2457	0.1115	0.1519
	audio + video	Whisper + I3D	0.1604	0.2263	0.1164	0.1833	0.2525	0.245	0.1108	0.1517
FREEDOM	text + image	MPNet + ViT (CLS)	0.1625	0.2307	0.1191	0.1857	**0.2538**	**0.2487**	**0.1133**	**0.1535**
	text + audio	MPNet + Whisper	**0.1627**	**0.2307**	**0.1191**	**0.1863**	0.2529	0.248	0.1129	0.153
	text + video	MPNet + I3D	0.1625	0.2305	**0.1191**	0.1857	0.2533	0.248	0.113	0.1527
	image + audio	ViT (CLS) + Whisper	0.162	0.2301	0.1189	0.1854	0.2532	0.2479	0.113	0.1528
	image + video	ViT (CLS) + I3D	0.1614	0.2298	0.1189	0.1853	0.252	0.2474	0.1129	0.1531
	audio + video	Whisper + I3D	0.1624	0.2305	**0.1191**	0.1858	0.2536	0.2482	0.1131	0.1529
MMGCN	text + image	MiniLM + ResNet152	0.1413	0.2041	0.1024	0.1681	0.2288	0.2227	0.0973	0.1419
	text + audio	MiniLM + Whisper	0.1442	0.2085	0.1052	0.1703	0.2296	0.2257	0.0994	0.1426
	text + video	MiniLM + R(2+1)D	0.1444	0.2086	0.1052	0.1706	0.2305	0.2261	0.0996	0.1426
	image + audio	ResNet152 + Whisper	0.1415	0.2075	0.1051	0.1693	0.2276	0.2248	0.0994	0.1417
	image + video	ResNet152 + R(2+1)D	0.144	0.2084	0.1052	0.1701	0.2295	0.2258	0.0996	0.1428
	audio + video	Whisper + R(2+1)D	0.1436	0.209	0.106	0.1705	0.2296	0.2263	0.0999	0.143
SLMRec	text + image	MiniLM + ResNet152	0.1526	0.1935	0.0943	0.1476	0.2348	0.2137	0.0932	0.1189
	text + audio	MiniLM + Whisper	0.1583	0.202	0.0995	0.1541	0.2429	0.2225	0.0981	0.1244
	text + video	MiniLM + R(2+1)D	0.1576	0.2011	0.0989	0.1534	0.2432	0.2219	0.0974	0.1246
	image + audio	ResNet152 + Whisper	0.1597	0.2041	0.1009	0.1561	0.2474	0.2253	0.0994	0.1266
	image + video	ResNet152 + R(2+1)D	0.16	0.2032	0.1001	0.1559	0.2455	0.2238	0.0984	0.126
	audio + video	Whisper + R(2+1)D	0.1626	0.2097	0.1045	0.1605	0.2494	0.2301	0.1021	0.1301

TABLE 4.14: ML1M *multi-modal* results. The best results per model are colored, while the best overall result is in **bold.**

Model	Modalities	Encoders	rec@10	ndcg@10	map@10	prec@10	rec@20	ndcg@20	map@20	prec@20
SLMRec	text	MPNet	0.187	0.1471	0.0942	0.052	0.2553	0.1707	0.101	0.0357
VBPR	text + image	MPNet + ViT (CLS)	0.112	0.0797	0.0457	0.0313	0.1681	0.0992	0.0507	0.0238
LATTICE	text + image	MiniLM + ResNet152	**0.1892**	0.1415	0.088	0.0525	0.266	0.168	0.0956	**0.0372**
FREEDOM	text + image	MiniLM + VGG	0.1865	0.1379	0.085	0.0517	**0.2631**	0.1644	0.0928	0.0367
MMGCN	text + image	MPNet + ViT (CLS)	0.1149	0.0797	0.0448	0.0322	0.1826	0.103	0.0509	0.0257
SLMRec	text + image	MPNet + VGG	**0.1892**	**0.1475**	**0.0938**	**0.0529**	0.2617	**0.1725**	**0.101**	0.0366

TABLE 4.15: DBbook *multi-modal* results. The best overall result is in **bold**.

Model	Modalities	Encoders	rec@10	ndcg@10	map@10	prec@10	rec@20	ndcg@20	map@20	prec@20
LATTICE	text	MPNet	0.2888	0.2409	0.1423	0.1157	0.4072	0.2868	0.1598	0.0812
VBPR	text + image	MiniLM + ViT (AVG)	0.2462	0.2026	0.1147	0.0995	0.361	0.2474	0.1309	0.0727
	text + audio	MiniLM + Whisper	0.2464	0.2039	0.1161	0.0997	0.3564	0.2469	0.1314	0.0721
	image + audio	ViT (AVG) + Whisper	0.2145	0.1778	0.098	0.0884	0.3262	0.2215	0.1128	0.0667
LATTICE	text + image	MPNet + ResNet152	0.2875	0.2401	0.1417	0.115	0.4109	0.288	0.1599	0.082
	text + audio	MPNet + Whisper	0.2881	0.2414	0.1427	0.1151	0.4123	0.2897	0.1609	0.0822
	image + audio	ResNet152 + Whisper	0.29	0.2411	0.1423	0.1161	0.4146	0.2894	0.1604	0.0827
FREEDOM	text + image	MPNet + VGG	**0.2919**	**0.2419**	**0.1429**	**0.1165**	0.4161	0.2898	0.1611	0.0826
	text + audio	MPNet + VGGish	0.2908	0.2413	0.1426	0.1161	**0.4169**	0.29	0.1611	**0.0828**
	image + audio	VGG + VGGish	0.2899	0.2414	0.1428	0.1155	0.4166	0.2904	**0.1613**	**0.0828**
MMGCN	text + image	MPNet + ResNet152	0.2456	0.206	0.1189	0.0999	0.3611	0.2508	0.1347	0.0728
	text + audio	MPNet + Whisper	0.2389	0.2006	0.1144	0.0981	0.3569	0.2463	0.1306	0.0721
	image + audio	ResNet152 + Whisper	0.2364	0.196	0.1105	0.0972	0.3481	0.2397	0.1261	0.0711
SLMRec	text + image	MPNet + VGG	0.2362	0.1975	0.1137	0.0953	0.3558	0.2441	0.1305	0.0712
	text + audio	MPNet + VGGish	0.2468	0.2014	0.1149	0.0989	0.3632	0.2462	0.1312	0.0723
	image + audio	VGG + VGGish	0.2496	0.2086	0.1216	0.0999	0.3611	**0.252**	0.1376	0.072

TABLE 4.16: Last.fm-2K *multi-modal* results. The best results per model are colored, while the best overall result is in **bold.**

First, on the ML1M dataset (Table 4.11), multimodal features can effectively enhance recommendation performance relative to collaborative filtering (CF) alone, as represented by the BPR model. In particular, LATTICE and FREEDOM consistently outperform BPR across all metrics and modalities. SLMRec, while showing improvements in recall@10 and recall@20, is outperformed by BPR on the remaining metrics. The other multimodal models fail to surpass BPR across the board. FREEDOM with video features obtained the best overall recall@10 results, while ViT AVG got the highest recall@20. As for the NDCG, FREEDOM with text features outperformed the other models in terms of both NDCG@10 and NDCG@20.

On DBbook (Table 4.12), LATTICE, FREEDOM, and SLMRec consistently outperform BPR in terms of all the metrics, regardless of the modality or encoder used. Also in DBbook, VBPR and MMGCN were outperformed by BPR. Here, SLMRec achieves the best overall performance when text features are used (encoded by MPNet). This holds for all considered metrics at both @10 and @20, with the exception of precision@20, for which the best result is obtained by LATTICE with textual features.

Finally, results on LFM-2K (Table 4.13) show a pattern similar to ML1M, with LATTICE and FREEDOM outperforming BPR across all settings, while MMGCN is again outperformed by BPR. VBPR is outperformed by BPR only for precision@10, recall@20, and precision@20. LATTICE obtains the best overall performance when text features encoded by MPNet are used. This result holds for all the metrics @10 and @20, with the exception of recall@20, where the best performance is obtained by FREEDOM with audio features encoded by VGGish.

It is worth analyzing the behavior of the VBPR model, as it is an adaptation of BPR for handling multimodal features: it performs much worse than the original BPR model and the other multimodal models, and this holds for all the considered datasets. This may be due to VBPR being a very simple model that merely concatenates the item ID and multimodal embedding, without performing any other operation. This can lead to a suboptimal final item representation, which decreases the overall recommendation performance, and hints at the need for more sophisticated treatments of multimodal features to be effective. This is partially confirmed by LATTICE and FREEDOM, as these methods use multimodal features to build a similarity matrix between items before providing recommendations. By contrast, SLMRec might struggle to fully demonstrate its potential, as it is designed to align different modalities, but in this context only a single modality is used, resulting in suboptimal performance of this model.

Multimodal features can effectively improve CF performance, but such improvement is model-dependent, and simpler models (e.g., VBPR) might not be able to effectively capture the potential of multimodal information.

RQ4.3.2: Effect of Multimodal Encoders To answer RQ4.3.2, we analyze how multimodal features affect the performance when they are used in *uni-modal* scenarios (Tables 4.11, 4.12, 4.13) and *multi-modal* scenarios (Tables 4.14, 4.15, and 4.16).

On *uni-modal* ML1M (Table 4.11), textual features performed very well, obtaining the best overall results (FREEDOM with the MPNet encoder). It is also noteworthy that audio features performed well as well (SLMRec with Whisper, which obtained the overall highest recall@10). This provides an interesting hint: audio features, typically overlooked by current research, can effectively be used to improve recommendation performance. When fusing more modalities (Table 4.14), a general trend is observed: almost every model shows improvements in almost all metrics compared with the uni-modal scenario. Then, FREEDOM using textual and audio features obtained the overall best performance on all metrics @10, while FREEDOM with textual and image features (with ViT) obtained the best performance for all metrics @20. These results confirm that all included modalities (images, audio, video) are worth considering and can improve overall performance. In such cases, the encoders that lead to the best performance are MPNet for text, Whisper for audio, and R(2+1)D for video.

Next, on DBbook in the uni-modal scenario (Table 4.12), textual features encoded by MPNet enable SLMRec to achieve the best overall performance among all considered models. The other text encoder, MiniLM, also obtained good results, allowing LATTICE to be the second-best overall performing model. Another observation is related to FREEDOM with image features encoded by VGG, which obtained very good results as well. These results are confirmed when considering the multi-modal scenario, in which SLMRec empowered with image and textual features (encoded by the same encoders, MPNet and VGG, respectively) obtained the best overall results in terms of almost all the metrics. Moreover, all models outperformed their uni-modal variant. Such observation confirms that combining overlooked multimodal features (in particular, image features) improves performance on this dataset as well. The best-performing encoders in this setting were MPNet and VGG for text and images, respectively.

Finally, for LFM-2K in the uni-modal scenario (Table 4.13), textual features encoded with MPNet yield the best overall performance for all metrics with the LATTICE model. The exceptions are recall@20 and precision@20, where LATTICE again achieves the best performance when audio features encoded with Whisper are used. FREEDOM also performs very well when image features encoded by VGG are used. These results confirm that overlooked multimodal features, such as audio and images, can also improve music recommendation. In the multi-modal scenario (Table 4.16), FREEDOM outperforms LATTICE in the uni-modal scenario, confirming the intuition that combining several modalities improves the recommendation. Moreover, FREEDOM performs best for all @10 metrics when textual and image features are used. For @20 metrics, the best performance is obtained by FREEDOM when text and audio features are used (encoded by MPNet and VGGish, respectively) in terms of map@20, and when audio and image features are used (encoded by VGGish and VGG, respectively) in terms of map@20 and precision@20. Moreover, the same encoders, VGG and VGGish, lead to the best overall performance in terms of nDCG@20, obtained by SLMRec. These results further confirm that using and combining overlooked features (audio and images, in this scenario) can bring important improvements in terms of the accuracy of the recommendations.

For this dataset, MPNet, VGG and VGGish were found to be the best performing encoders for text, images and audio, respectively.

To answer RQ4.3.2, this study shows that all the multimodal features considered are useful to improve the recommendation accuracy, particularly when used together. The best overall text encoder is MPNet, while VGG and VGGish perform best for image and audio, respectively, and R(2+1)D yields the best video feature encoder.

4.3.4 Take-Home Messages

In the publication [176], three well-known recommendation datasets (ML1M, DBbook, LFM-2K) were extended, mostly used in CF settings or non-reproducible multimodal recommendation settings. For the first time, publicly available multimodal features for all these datasets are provided, encoded using state-of-the-art multimodal encoders. Moreover, the mapping to the multimodal raw data is released, so that others can download them and use other encoders or methodologies. A benchmark analysis was run on the extended datasets, showing that multimodal features can improve recommendation performance in these domains as well, even if overlooked by current multimodal recommendation research.

Main Findings

- **RQ4.3.1**: Multimodal features can improve the recommendation performance with respect to CF signals, but such features must be properly handled by the models.
- **RQ4.3.2**: The fusion of different sources systematically leads to improvements in the recommendation performance, thus confirming the usefulness of these approeaches.

These results have been conducted during the PhD and published at ACM RecSys 2025 in the Reproducibility Track (Giuseppe Spillo et al. "See the Movie, Hear the Song, Read the Book: Extending MovieLens-1M, Last. fm-2K, and DBbook with Multimodal Data". In: *Proceedings of the Nineteenth ACM Conference on Recommender Systems*. 2025, pp. 847–856).

4.4 *Gotta Embed Them All!*

4.4.1 Introduction

The featured publication [177] presents GETALL! (Gotta Embed Them All!), a novel and comprehensive multi-modal Knowledge-Aware Recommender System (KARS) designed to address the critical limitations of current uni-modal and partially multi-modal recommendation approaches. A major shortcoming in the current landscape of KARS is that most models still rely on a single modality or, in the best cases, combine a small number of them without deeply integrating the diverse semantic content they encode. This often results in partial, modality-biased representations that fail to fully reflect the nuanced characteristics of items and the complex preferences of users, especially in domains where information is inherently multi-modal, such as movies, books, and multimedia content (also referred to as audio/video content, or A/V for short).

To fill this gap, a deep learning-based methodology is proposed to encode and fuse a wide range of heterogeneous data sources into unified, multi-modal user and item representations. The core idea behind GETAll! is that each modality—be it structured data from knowledge graphs, textual descriptions from metadata or user reviews, or audio-visual content like posters, trailers, or covers—captures a distinct aspect of an item. Thus, a robust recommendation model should be capable of incorporating all these views simultaneously, allowing it to generalize better across users with diverse and modality-specific preferences. The approach begins by applying dedicated encoders to each modality in order to extract uni-modal embeddings. These include, for instance, graph-based encoders for structured knowledge, Transformer-based models for text, and convolutional or pretrained models for image and video features. A fusion mechanism relying on self-attention refines each modality-specific embedding internally, and cross-attention integrates signals across modalities, producing a rich and context-aware multi-modal representation of both users and items.

These fused embeddings are then used to compute user-item relevance scores, which feed into the final top-k recommendation list. By integrating all available content into a shared representational space and allowing attention mechanisms to weight the relative importance of each modality dynamically, the model learns to capture both broad contextual information and fine-grained user interests. This is particularly important in scenarios where certain users may be influenced more strongly by visual cues, while others may respond to narrative elements or relational signals captured in the KG.

The experimental results demonstrate that GETALL! consistently outperforms several state-of-the-art baselines across different evaluation metrics. The comparisons confirm that deeply integrating diverse data modalities—not just aggregating them post hoc—leads to significant improvements in recommendation performance. Additionally, a sensitivity analysis is conducted to evaluate the impact of different feature extraction strategies within each modality, providing further insights into which combinations yield the most effective representations for multi-modal recommendation tasks.

This work proposes a principled and scalable framework for multi-modal knowledge-aware recommendation, grounded in the fusion of structured and unstructured sources through attention-based learning. By emphasizing the complementary nature of heterogeneous modalities and enabling their joint modeling from the earliest stages of representation learning, GETALL! addresses the limitations of existing systems and paves the way for more accurate, nuanced, and user-centric recommendations in complex domains.

These results have been published in the Journal of Intelligent Information Systems (Giuseppe Spillo et al. "*Gotta Embed Them All!* - Knowledge-aware Recommendations Fusing Heterogeneous Multi-Modal Item Embeddings". In: *Journal of Intelligent Information Systems* (2025), pp. 1–27).

4.4.2 Methodology

This section introduces strategies employed for graph, text, and multimedia content (images, audio, video) encoding, grounded in the current literature in the related areas. Then, GETALL! is introduced, the architecture designed to learn users' and items' representations and to provide recommendations.

Heterogeneous Embedding Learning In the methodology, state-of-the-art encoders for the knowledge sources are adopted.

Following the methodology for knowledge source representations discussed in Section 2, user and item embeddings are learned for each available source. In particular, for the KG, the CompGCN [202] encoder is adopted, while for textual information sentence encoders [217] are used; for image, audio, and video, Visual

Transformers [54], VGGish [72], and R(2+1)D [195] are used, respectively. Further details on the operation of these encoders are provided in Chapter 2.

Let:

- u_g and i_g denote the user and item embeddings coming from KGs, respectively;
- u_t and i_t denote the user and item embeddings coming from textual descriptions, respectively;
- u_{img} and i_{img} denote the user and item embeddings coming from images, respectively;
- u_a and i_a denote the user and item embeddings coming from audio signals, respectively;
- u_v and i_v denote the user and item embeddings coming from videos, respectively.

Recommender System Architecture After the previous steps, the embeddings of items and users, encoding *modality-specific characteristics* and *modality-specific preferences*, respectively, are obtained[9]. Next, these embeddings are fed into GETALL!, whose goal is to fuse *modality-specific* embeddings to learn a unique *multi-modal* representation of users and items, which is used to make predictions.

The architecture exploits attention mechanisms based on two key ideas: *(a) self-attention* fuses user and item embeddings within each modality to capture modality-specific preferences; *(b) cross-attention* then integrates these modality-specific embeddings into a compact, unified representation by highlighting the most informative features [184]. This final embedding is used to predict user interest. The overall architecture is shown in Figure 4.6, and detailed step-by-step below.

First, we apply a linear transformation with ReLU to all the embeddings:

$$\begin{aligned} u_g &= ReLU(W_u * u_{g_{enc}} + b_u) \\ i_g &= ReLU(W_i * i_{g_{enc}} + b_i) \end{aligned} \tag{4.12}$$

[9]For readability, we omit arrows; all mentioned objects are vectors.

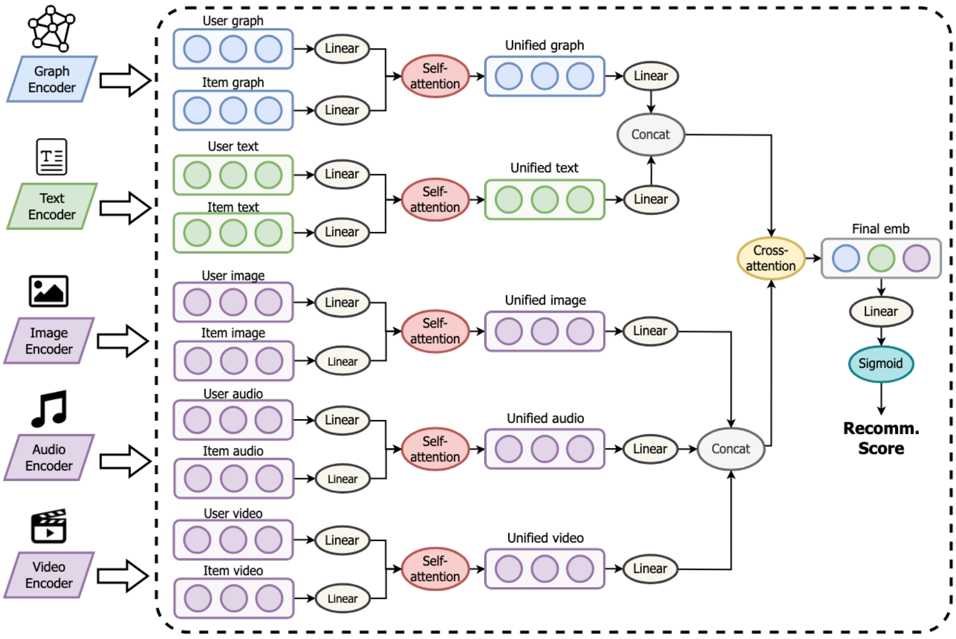
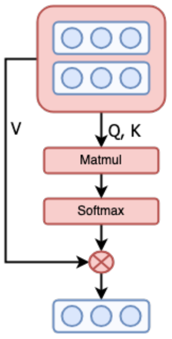
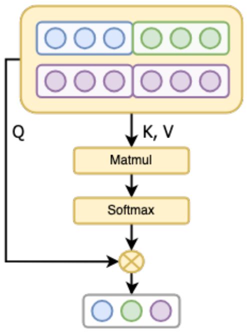


FIGURE 4.6: Architecture of GETALL!. For each source (graph, text, images, audio, video), users and item embeddings are fused with Self-attention, and the resulting embedding are fused through Cross-Attention, in which unified graph and text embeddings serve as query tensors, all the multimedia content derived embeddings serve as key and value tensors.



(A) Self-attention mechanism



(B) Cross-attention mechanism

FIGURE 4.7: Attention mechanisms adopted in GETALL! to fuse modality-specific embeddings.

where $u_{g_{enc}}$ and $i_{g_{enc}}$ are the original user and item embeddings learned by the Graph Encoder, respectively; W_u and W_i are learnable parameters, while b_u and b_i are biases[10].

Then, *self-attention* is exploited to fuse *users* and *items* embeddings for each knowledge source, to reveal the latent characteristics of the items that are relevant to the user. Formally, for each user-item pair, first we define the query, key, and value tensors:

$$\begin{aligned} Q_g &= Linear\left(\begin{bmatrix} u_g \\ \cdot \\ i_g \end{bmatrix}\right) = W_{Q_g}\begin{bmatrix} u_g \\ \cdot \\ i_g \end{bmatrix} + b_{Q_g} \\ K_g &= Linear\left(\begin{bmatrix} u_g \\ \cdot \\ i_g \end{bmatrix}\right) = W_{K_g}\begin{bmatrix} u_g \\ \cdot \\ i_g \end{bmatrix} + b_{K_g} \\ V_g &= Linear\left(\begin{bmatrix} u_g \\ \cdot \\ i_g \end{bmatrix}\right) = W_{V_g}\begin{bmatrix} u_g \\ \cdot \\ i_g \end{bmatrix} + b_{V_g} \end{aligned} \tag{4.13}$$

where W_{Q_g}, W_{K_g}, W_{V_g} are learnable parameters, and b_{Q_g}, b_{K_g}, b_{V_g} are biases.

Then, given d_g the dimension of users and item embeddings, the unified *graph* embedding h_g is obtained as

$$h_g = softmax\left(\frac{Q_g K_g^T}{\sqrt{d_g}}\right) V_g \tag{4.14}$$

Following Equations 4.13 and 4.14, we use *self-attention* to obtain a unique textual embedding h_t, and unique multimedia embeddings for each available multimedia source. Let such embeddings be h_{img}, h_a, h_v associated to images, audio signals and videos, respectively.

Next, the *cross-attention* is used. As explained earlier, the goal is to extract the most relevant features from each modality and fuse them into a unique, more compact and accurate representation [184]. To this end, we first define query, key, and value (as already discussed), and use them to obtain the final embedding

[10]Of course, this process is repeated for each embedding and for each source. For the sake of brevity, we report the formalization only for graph embeddings.

emb_{final} as follows:

$$\begin{aligned}
Q &= Linear\left(e_g \oplus e_t\right)\\
K &= Linear\left(e_{img} \oplus e_a \oplus e_v\right)\\
V &= Linear\left(e_{img} \oplus e_a \oplus e_v\right)\\
emb_{final} &= softmax\left(\frac{QK^T}{\sqrt{d}}\right)V
\end{aligned} \tag{4.15}$$

Note that these equations can be generalized to any number of the multimedia knowledge sources. For example, in some cases, only text and images could be available (e.g., in the book recommendation domains, where only book covers and textual description are available).

This embedding is finally passed through a series of linear layers, whose size is increasingly smaller, with the last one having a size of 1 and a sigmoid activation function, to obtain a recommendation score $score_{rec} \in [0, 1]$ that represents the user's interest in the item.

$$score_{rec} = sigmoid\left(ReLU\left(emb_{final} * W_{rec} + b_{rec}\right)\right) \tag{4.16}$$

In order to learn the recommendation model, GETALL! is trained by exploiting a subset of the ratings in the form $y_{u,i}$ available in Y (see Section 4.4.4) and by using *binary cross-entropy* as loss function. In particular, it learns the function $\widetilde{Y}$ (introduced in Section 3) that predicts to what extent user u will like item i. At prediction time, given a user u, GETALL! calculates the *scores* for the candidate items, which are then ranked based on the predicted relevance scores, and the *top-k* are returned as recommendations.

4.4.3 Discussion

To summarize the key points of this methodology:

- State-of-the-art encoders are used to extract information from several knowledge sources, including KGs, text, images, audio, and video. This enables the learning of *modality-specific* embeddings for users and items.
- User and item embeddings (related to each knowledge source) are fused using *self-attention* to highlight item features relevant to users, as described in [167].

- These embeddings are then combined through *cross-attention* to obtain a unified embedding capturing the most relevant modality-specific features for the recommendation task, as in [184].
- The resulting unified embedding is used to provide personalized recommendations.

The fusion of all these knowledge sources is overlooked in the current literature. Most KARS using KGs ignore multimodal features, while most multimodal RSs typically ignore structured knowledge deriving from KGs. In contrast, this approach fuses all these sources within a single architecture.This research direction could be pursued further, and the present contribution could foster this aspect in KARS.

In addition, the efficiency of the methodology is highlighted, as it relies on the fusion of pre-trained multimodal embeddings. This implies that multimodal embeddings need to be learned only once and can then be exploited in the recommendation process, rather than being trained from scratch each time the recommendation model is updated. This aspect is particularly crucial in real-world scenarios, where catalogs may contain millions (or even billions) of items, each associated with multiple modalities (e.g., text, images, audio, video). In such cases, training an end-to-end recommendation model on a daily basis would be highly resource-intensive and often impractical. In contrast, an architecture focusing solely on the fusion of pre-trained embeddings significantly improves overall efficiency.

Acknowledging that some limitations exist, attention mechanisms were adopted to fuse multimodal embeddings, rather than simpler strategies (such as concatenation). As shown by the ablation studies discussed in Section 4.4.6, this design choice proved effective. However, multimodal embedding fusion remains challenging, as features across modalities are not inherently aligned, and each embedding process typically ignores the others. A possible solution could involve adopting Large Multimodal Models (LMMs), defined by [231], such as CLIP or VLMo (introduced in [141] and [14], respectively), which enable embeddings to be aware of multiple modalities simultaneously. Nevertheless, this solution presents drawbacks as well: first, only a limited number of modalities can be considered (for instance, CLIP supports only text and image features, making alignment with KG data, video, and audio challenging); second, the large number of parameters in LMMs dramatically increases overall complexity, making it hard to efficiently apply this methodology in large-scale and real-world scenarios.

These considerations indicate that a trade-off between the complexity of the approach and the effectiveness of the recommendation must be sought. Future work will further explore and analyze these aspects.

4.4.4 Problem Formulation

This section extends the formalization introduced in Section 2.3, covering only the aspects that were not previously formalized for the sake of generalizability.

Learning User and Item Embeddings. Given all the available sources, a deep neural network is designed to learn user and item embeddings using *modality-specific* encoders.

Starting from graph $\mathcal{G}$, a graph encoder is used to learn user and item embeddings $\vec{u_g}$ and $\vec{i_g}$. Similarly, starting from the textual content $\mathcal{T}$, a sentence encoder is used to learn item embeddings $\vec{i_t}$. Next, user embeddings $\vec{u_t}$ are obtained as the *centroid* of the embeddings of the items liked by the user.

The same strategy is employed for each multimedia knowledge source. First, item embeddings $\vec{i_{img}}$, $\vec{i_a}$, $\vec{i_v}$ associated to each multimedia source are obtained. Then, user embeddings $\vec{u_{img}}$, $\vec{u_a}$, $\vec{u_v}$ are obtained as the *centroid* of the embeddings of the items liked by the user.

Next, embeddings are fused to obtain a *multi-modal* representation of users and items, labeled as $\vec{u}$ and $\vec{i}$. These vectors are finally used to predict the extent of interest of a user in an item. Let θ be the parameters of the network.

Problem Statement. Given $\mathcal{G}$, $\mathcal{T}$, $\mathcal{IMG}$, $\mathcal{A}$, $\mathcal{V}$ and θ, the deep neural network aims to predict the interest of a user u in an item i based on the available knowledge sources.

Formally: $\widetilde{Y}(u, i|\mathcal{G}, \mathcal{T}, \mathcal{IMG}, \mathcal{A}, \mathcal{V}, \theta) = y_{u,i}$. In experiments, the approach is evaluated using a *top-k* recommendation setting, with the available items ranked according to the scores predicted by $\widetilde{Y}$, and the top-k items are returned.

4.4.5 Experimental Evaluation

The following research questions were addressed:

- **RQ4.4.1 - Comparison to SOTA:** How does the proposed approach compare with competitive SOTA baselines?

- **RQ4.4.2 - Ablation Tests:** How do the graph, text, multimedia signals and architectural components contribute to overall performance?
- **RQ4.4.3 - Sensitivity Analysis:** How do different hyperparameters and encoding techniques impact model performance?

Datasets Experiments were conducted in a *movie, book and music recommendation* scenario. In particular, the ML1M, DBbook and Last.fm datasets have been considered. More details about this datasets can be found in Section 2.3, where the problem has been formalized.

Details related to the choice of the encoders implementations, training parameters, and hardware configuration can be found in the related publication [177].

Sensitivity Analysis To support the choice of hyperparameters and encoding techniques, a sensitivity analysis was conducted. For the graph encoder, various embedding sizes (128, 256, 384, 512) and a variant without descriptive properties (no $\mathcal{KG}$) were tested. For the text encoder, several Transformers (MPNet, DistilBERT, RoBERTa, ALBERT) from Hugging Face were evaluated. For the image encoder, VGG19 was tested alongside ViT, and ViT patch embeddings were averaged (excluding the $[CLS]$ token). For audio, VGGish and Whisper were compared, and for video, R(2+1)D and I3D were considered. All the results are discussed in Section 4.4.6.

Baselines The proposed model was compared with several baselines using implementations from RecBole [234] and [114]. CF, KARS based on KGs, and KARS based on multimedia soures have been considered. Hyperparameters for these models were optimized. Their description is discussed in Section 2.3, and is not reported here to avoid repetitions.

Evaluation Metrics For this work, precision, recall, F1 and NDCG were considered as accuracy metrics. Diversity was assessed through the Gini index, while novelty was assessed through the EPC metric. Fairness was assessed by APLT. Statistical significance was assessed using a t-test (P-value = 0.01).

4.4.6 Results Discussion

RQ4.4.1: Comparison to Baselines To answer RQ4.4.1, GETAll! performance was compared with the baseline models' performance. Tables 4.17 and 4.18 show

Model	Accuracy				Diversity	Novelty	Fairness
	Precision	Recall	F1	NDCG	Gini	EPC	APLT
Collaborative Filtering RSs							
ItemKNN	0.7238	0.4243	0.4531	0.9024	0.7938	0.5679	0.0294
BPR	0.7204	0.4215	0.4505	0.9014	0.7244	0.5872	0.0564
EASE	0.7336	0.4249	0.4557	0.9097	0.7714	0.5823	0.0331
SLIMElastic	0.7447	0.4053	0.4533	0.9137	0.7715	0.5692	0.0024
MultiVAE	0.7341	0.4295	0.4594	0.9051	0.8379	0.5451	0.0226
NeuMF	0.6954	0.4153	0.4414	0.8781	**0.6732**	0.5796	**0.0819**
Graph-based RSs							
LightGCN	0.7287	0.4238	0.4534	0.9063	0.7536	0.5836	0.0429
NGCF	0.7156	0.4209	0.4490	0.8950	0.7156	0.5854	0.0558
SGL	0.7173	0.4199	0.4492	0.8943	0.7089	0.5894	0.0546
(Uni-modal) Knowledge-aware RSs, exploiting either graph, text and multimedia signals							
CKE	0.7119	0.4188	0.4469	0.8954	0.7046	0.5866	0.0629
CFKG	0.7325	0.4268	0.4567	0.9069	0.7477	0.5890	0.0426
KGCN	0.7343	0.4280	0.4579	0.9044	0.8271	0.5519	0.0264
KGAT	0.7133	0.4195	0.4480	0.8952	0.7122	0.5844	0.0586
KGIN	0.7125	0.4200	0.4483	0.8916	0.7046	0.5864	0.0621
KTUP	0.7232	0.4229	0.4518	0.9019	0.7457	0.5801	0.0417
KGNNLS	0.7364	0.4281	0.4587	0.9063	0.8236	0.5585	0.0257
MKR	0.7060	0.4176	0.4449	0.8877	0.7067	0.5814	0.0641
VBPR	0.7222	0.4242	0.4482	0.9025	0.7493	0.5802	0.0396
Multi-modal KARS, exploiting graph, text and multimedia signals (fused)							
Lattice	0.7248	0.4242	0.4488	0.9067	0.7714	0.5732	0.0390
Freedom	0.7354	0.4308	0.4557	0.9138	0.8093	0.5656	0.0307
MMGCN	0.7205	0.4237	0.4475	0.9031	0.7387	0.5850	0.0553
GRCN	0.7134	0.4183	0.4467	0.8962	0.7145	0.5877	0.0569
BM3	0.7377	0.4302	0.4604	0.9092	0.8290	0.5564	0.0258
LightGCNM	0.7390	0.4289	0.4597	0.9116	0.8243	0.5614	0.0269
Polignano et al. [136]	0.8026	0.4588	0.4947	0.9422	0.7709	0.6605	0.0728
Spillo et al. [167]	0.8054	0.4592	0.4958	0.9453	0.7638	0.6635	0.0715
GETAll!	**0.8146***	**0.4630***	**0.5002***	**0.9490***	0.7519	**0.6730***	0.0815

TABLE 4.17: Comparison to baselines in the movie domain (ML1M dataset. The best overall value is highlighted in bold, while the best baseline value is underlined. (*) means the differences are statistically significant with $p < 0.01$.

Model	Accuracy				Diversity	Novelty	Fairness
	Precision	Recall	F1	NDCG	Gini	EPC	APLT
Collaborative Filtering RSs							
ItemKNN	0.6635	0.4817	0.5180	0.8655	0.6919	0.5899	0.1345
BPR	0.6714	0.5238	0.5437	0.8708	0.7185	0.5988	0.1043
EASE	0.6685	0.5211	0.5407	0.8708	0.7222	0.5960	0.1207
SLIMElastic	0.6578	0.5074	0.5293	0.8627	0.6535	0.5879	0.1824
MultiVAE	0.6705	0.5238	0.5427	0.8650	0.6993	0.5962	0.1210
NeuMF	0.6675	0.5225	0.5405	0.8660	0.6990	0.5953	0.1383
Graph-based RSs							
LightGCN	0.6748	0.5255	0.5460	0.8723	0.7139	0.6018	0.1069
NGCF	0.6732	0.5253	0.5450	0.8689	0.7073	0.6003	0.1113
SGL	0.6703	0.5211	0.5419	0.8705	0.6913	0.5993	0.1334
(Uni-modal) Knowledge-aware RSs, exploiting either graph, text and multimedia signals							
CKE	0.6714	0.5242	0.5437	0.8685	0.7108	0.5981	0.1107
CFKG	0.6737	0.5251	0.5456	0.8739	0.7285	0.6019	0.0904
KGCN	0.6712	0.5224	0.5428	0.8689	0.7188	0.5968	0.1106
KGAT	0.6717	0.5231	0.5433	0.8713	0.7104	0.5996	0.1137
KGIN	0.6703	0.5243	0.5432	0.8708	0.6935	0.5992	0.1239
KTUP	0.6734	0.5246	0.5450	0.8715	0.7435	0.5985	0.0873
KGNNLS	0.6717	0.5237	0.5438	0.8711	0.7128	0.5990	0.1121
MKR	0.6702	0.5229	0.5425	0.8679	0.7196	0.5960	0.1085
VBPR	0.6662	0.5226	0.5399	0.8691	0.7156	0.5944	0.1128
Multi-modal KARS, exploiting graph, text and multimedia signals (fused)							
Lattice	0.6713	0.5251	0.5436	0.8754	0.7177	0.6004	0.1095
Freedom	0.6716	0.5259	0.5440	0.8744	0.7235	0.5999	0.1104
MMGCN	0.6679	0.5220	0.5408	0.8689	0.7065	0.5959	0.1135
BM3	0.6885	0.5320	0.5420	0.8690	0.8290	0.5564	0.0258
GRCN	0.6689	0.5218	0.5414	0.8692	0.6973	0.5978	0.1378
BM3	0.6744	0.5263	0.5457	0.8720	0.7274	0.6011	0.0984
Polignano et al. [136]	0.6882	0.5471	0.5613	0.8794	0.6276	0.6185	0.2334
Spillo et al. [167]	0.6912	0.5498	0.5640	0.8811	**0.6258**	0.6222	**0.2371**
GETAll!	**0.7020***	**0.5580***	**0.5730***	**0.8914***	0.6812	**0.6329***	0.1972*

TABLE 4.18: Comparison to baselines in the movie domain (DBBOOK dataset). The best overall value is highlighted in bold, while the best baseline value is underlined. (*) means the differences are statistically significant with $p < 0.01$.

that GETALL! consistently outperforms all baselines across almost all accuracy and novelty metrics, with statistically significant improvements ($\alpha = 0.01$).

Models combining multiple knowledge sources tend to outperform others. The best-performing baselines, [136] and [167], exploit both graph and text embeddings fused through concatenation and attention, respectively. In general, on ML1M, uni-modal KARS outperform CF and graph-based models, while multi-modal KARS outperform uni-modal ones, highlighting the benefit of combining diverse sources. Freedom [258], BM3 [259] and LightGCNM [114] are among the best-performing A/V-based baseline models, showing very good accuracy, while NeuMF provides the most diverse movie recommendations, slightly affecting APLT, although its difference with GETALL! is not statistically significant.

On DBBOOK, GETALL! significantly outperforms all baselines in accuracy and novelty, and is the second-best-performing in terms of fairness as well; among A/V-based models, Freedom and BM3 remain competitive, further confirming that multimedia signals should not be overlooked when designing KARS due to their informative power. The performance of all baselines based on a single source ranges from 0.5419 (SGL) to 0.5456 (CFKG) in terms of F1-measure, with a tiny gap. This indicates that, regardless of the knowledge source used, *multi-modal* approaches based on the combination of heterogeneous data, including text, graphs, and multimedia signals, can improve the accuracy of KARS. **This further confirms that all the different *modalities* of an item, as well as all the different *modality-specific preferences*, are worth encoding in a KARS.** This is also confirmed by the most competitive baselines, [136] and [167], even if their performance is significantly lower with respect to GETALL!.

RQ4.4.2: Ablation Tests To answer RQ4.4.2, ablation tests on the knowledge sources as well as on the components of the architecture were conducted to assess their contribution to the overall performance. Results are reported in Table 4.19 and 4.20 for ML1M and DBBOOK, respectively.

For ML1M, ablation experiments on the single knowledge sources revealed that graph (G) and text (T) are the most informative sources in terms of accuracy, while the effectiveness of multimedia signals alone is relatively lower, particularly for video (V) and audio (A) sources. The Basic Model (G + T) performs better than the previous ones, in line with the outcomes of related literature [136, 167]. Therefore, the use of all available multimedia signals led to a significant drop in terms of accuracy, balanced by increases in *diversity* and APLT. In particular, **the significant**

Model	Accuracy				Diversity	Novelty	Fairness
	Precision	Recall	F1	NDCG	Gini	EPC	APLT
				Data sources			
Graph (G)	0.8063	0.4590	0.4957	0.9444	0.7435	0.6715	0.0921
Text (T)	0.8033	0.4583	0.4947	0.9425	0.7587	0.6590	0.0775
Video (V)	0.7843	0.4495	0.4844	0.9350	0.7950	0.6351	0.0569
Image (I)	0.8026	0.4575	0.4941	0.9428	0.7664	0.6597	0.0769
Audio (A)	0.7854	0.4496	0.4846	0.9348	0.7895	0.6375	0.0639
G + T (Basic)	0.8088*	0.4605**	0.4975*	0.9468	0.7712	0.6632*	0.0659*
V + I + A (A/V)	0.6235*	0.3846*	0.4044*	0.8286*	**0.6910**	0.5349*	**0.1583***
Basic + V	0.8070*	0.4601*	0.4967*	0.9429*	0.7686	0.6603*	0.0738*
Basic + I	**0.8159**	**0.4643**	**0.5015**	0.9483	0.7586	**0.6735**	0.0789**
Basic + A	0.8126	0.4624	0.4995	0.9472	0.7524	0.6713	0.0832
				Architecture			
w/o attention	0.8006*	0.4580*	0.4940*	0.9414*	0.7758	0.6542*	0.0669*
w/o dropout	0.7950*	0.4538*	0.4894*	0.9359*	0.7221	0.6694*	0.1237*
				Full model			
Basic + A/V	0.8146	0.4630	0.5002	**0.9490**	0.7519	0.6730	0.0815*

TABLE 4.19: Ablation tests performed on the movie domain (ML1M dataset). Best values are in **bold**, second best values are underlined. Differences statistically significant are highlighted with * ($p < 0.01$) and ** ($p < 0.05$).

Model	Accuracy				Diversity	Novelty	Fairness
	Precision	Recall	F1	NDCG	Gini	EPC	APLT
Data sources							
Graph (G)	0.6864*	0.5466*	0.5599*	0.876*	0.6269	0.6155*	0.2373*
Text (T)	0.6826*	0.5457*	0.5570*	0.8753*	**0.6177**	0.6136*	**0.2479***
Image (I)	0.6810*	0.5396*	0.5541*	0.8766*	0.6261	0.6126*	0.2418*
G + T (Basic)	0.6976**	0.5564	0.5693**	0.8886	0.6609	0.6295	0.2108*
G + A/V	0.6914*	0.5523**	0.5646*	0.8831*	0.6508	0.6244*	0.2172*
T + A/V	0.6864*	0.5439*	0.5587*	0.8862*	0.6672	0.6211*	0.2120*
Architecture							
w/o attention	0.6989	0.5576	0.5709	0.8900	0.6508	0.6315	0.2202*
w/o dropout	0.6853*	0.5428*	0.5580*	0.8774*	0.6390	0.6154*	0.2274*
Full model							
Basic + A/V	**0.7020**	**0.5580**	**0.5730**	**0.8914**	0.6812	**0.6329**	0.1972

TABLE 4.20: Ablation tests performed on the book domain (DBBOOK dataset). Best values are in **bold**, second best values are underlined. Differences statistically significant are highlighted with * ($p < 0.01$) and ** ($p < 0.05$).

increase in non-accuracy metrics is an interesting outcome, since it shows that multimedia sources may be exploited to tackle popularity bias in KARS.

Next, multimedia signals were integrated into the Basic Model. The results confirmed the initial intuition: **adding video, images, or audio consistently improves model accuracy across all metrics.** Interestingly, the best performance was achieved using only *images*: the $Basic + I$ configuration even outperformed the full model that fuses all sources, although the differences were not statistically significant. This outcome suggests a lower quality of video and audio encodings compared to other knowledge sources (see V and A in Table 4.19). In particular, video fusion (Basic+V) resulted in a performance drop for most metrics. This aspect will be explored further in future work, by experimenting with alternative encoding techniques and fusion strategies for multimedia signals. **In addition, attention mechanisms and dropout demonstrated their effectiveness in learning precise user and item representations, as removal led to a performance decrease.**

Results on DBBOOK (Table 4.20) are consistent with the previous findings. G and T remained the most informative sources, with their fusion boosting both accuracy

FIGURE 4.8: Dimension of Graph Embeddings

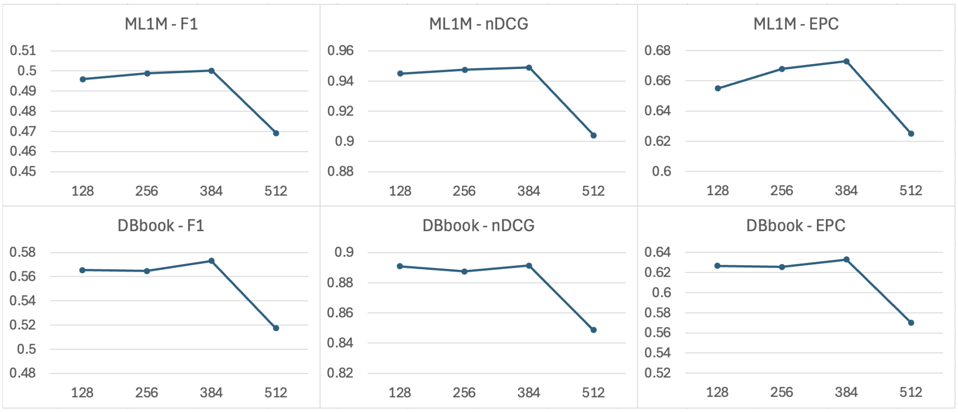
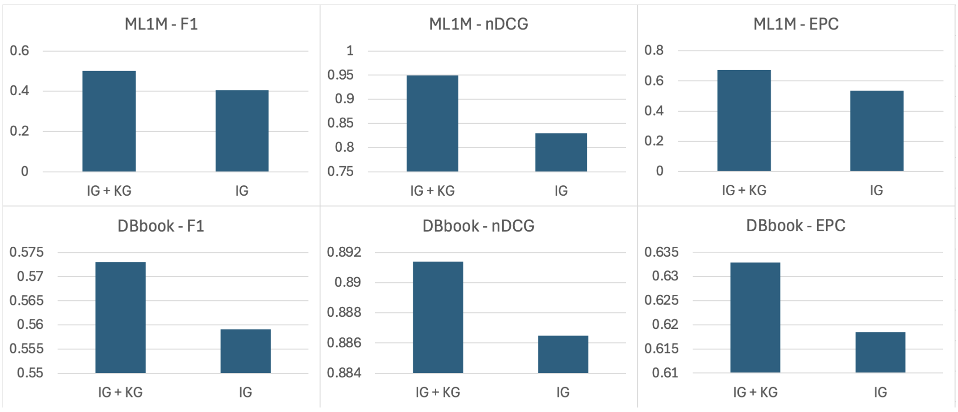


FIGURE 4.9: Information Encoded in the Graph



and novelty. **On DBBOOK, the best results were achieved by the full model combining all available sources.** However, since the only A/V input in this dataset is the book cover, the impact of audio and video remains an open question.

In conclusion, the ablation tests validate the design choices: *(a)* **attention mechanisms and dropout enhance the representation of modality-specific item and preference features**, improving recommendation accuracy; *(b)* **incorporating multimedia signals, especially images, further improves performance across both datasets.** These results support the intuition that fusing diverse knowledge sources—particular multimedia signals—is a promising direction for KARS research.

FIGURE 4.10: Sentence Embedding Techniques

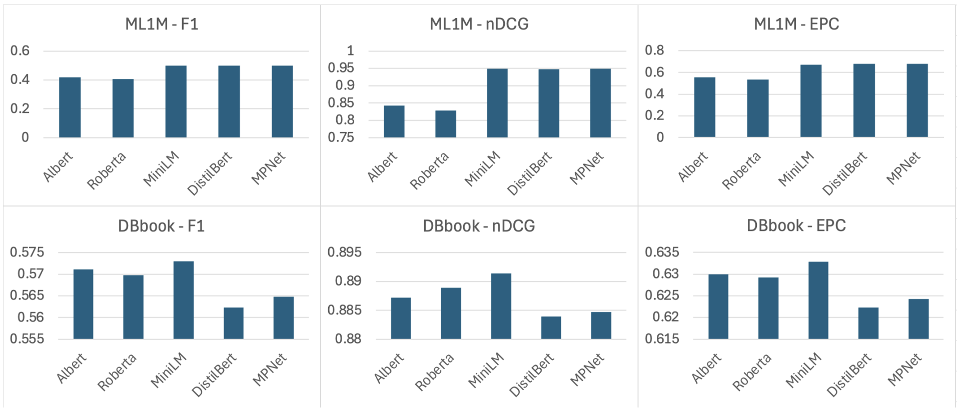


FIGURE 4.11: Image Embedding Techniques

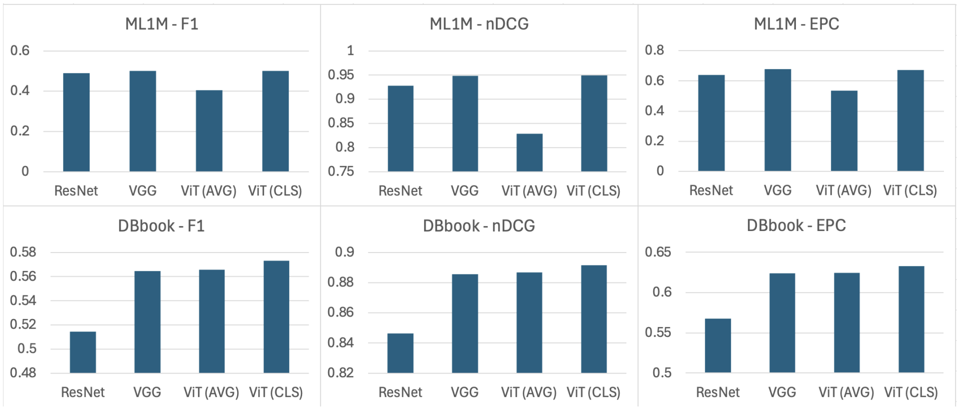


FIGURE 4.12: Audio Embedding Techniques (only for ML1M)

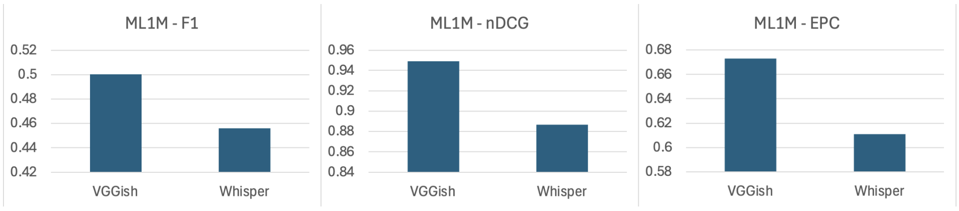
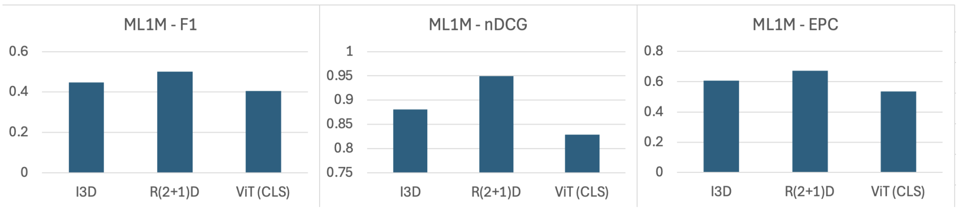


FIGURE 4.13: Video Embedding Techniques (only for ML1M)



RQ4.4.3: Sensitivity Analysis To address RQ4.4.3, a sensitivity analysis over hyper-parameter values and encoders in the full GETALL! model, focusing on F1, nDCG, and EPC (Figures 4.9–4.13). First, for graph embeddings, $d = 384$ provided the best performance across both datasets (Figure 4.8), confirming the design choices. Then, comparing CF-only ($\mathcal{IG}$) to full KG embeddings ($\mathcal{IG} + \mathcal{KG}$), the latter significantly outperformed the former (Figure 4.9), confirming that KGs improve recommendation performance. Regarding text encoders (Figure 4.10), MiniLM, MPNet, and DistillBERT performed similarly on ML1M, but MiniLM consistently performed well on both datasets, confirming the choice. For image encoding (Figure 4.11), ViT using the CLS token outperformed alternatives on DBbook, aligning with existing literature. ViT AVG and ResNet performed worse. Audio and video analysis (ML1M only) showed VGGish and R(2+1)D outperform Whisper, I3D, and ViT CLS.

This sensitivity analysis confirms the design choices, showing that the architecture employs the most effective encoding strategies.

4.4.7 Take-Home Messages

The featured publication [176] presents GETALL!, a KARS that exploits heterogeneous knowledge sources, including graph, text, and multimedia sources to provide recommendations. A deep architecture is designed to fuse uni-modal user and item embeddings into a unified multi-modal representation using self-attention and cross-attention mechanisms. In the experiments, GETALL! was compared to several competitive baselines, and the results showed significant improvements in terms of accuracy and novelty. However, ablation studies indicated that further work is needed to identify more effective strategies to encode audio and video signals.

In future work, the methodology will be applied to new datasets, and other source fusion techniques will be investigated; moreover, qualitative analyses will be conducted to assess how different knowledge sources influence personalized suggestions.

Main Findings

- **RQ4.4.1**: Compared to baseline models, GETALL! outperforms most of the models, for most of the metrics, in both the datasets. This confirms that the fusion on KGs and multimedia signals should not be overlooked in the literature.
- **RQ4.4.2**: Ablation study reveal that multimedia sources can be effective in improving beyond-accuracy aspects of the recommendations, and that fusing *all* the available modalities leads to scenario-dependent improvements. In general, fusing heterogeneous embeddings improves the recommendation performance.
- **RQ4.4.3**: The sensitivity analysis confirms the design choices, as it shows that the architecture employs the most effective encoding strategies and training parameters.

These results have been carried out during the PhD, published in the Journal of Intelligent Information Systems (Giuseppe Spillo et al. "*Gotta Embed Them All!* - Knowledge-aware Recommendations Fusing Heterogeneous Multi-Modal Item Embeddings". In: *Journal of Intelligent Information Systems* (2025), pp. 1–27).

4.5 Conclusions

This section concludes the chapter and draws some conclusions to answer RQ2, introduced in Chapter 1.

First, this chapter investigated how to effectively fusing heterogeneous embeddings coming from two knowledge sources, KGs and textual descriptions. In contrast to state-of-the-art approaches that primarily rely on simple embedding concatenation, this thesis investigate the use of attention mechanisms (both self- and cross-attention) to enhance the resulting representations and improve overall performance. The findings contribute to the KARS literature by demonstrating that simplistic fusion strategies are not always optimal and can be substantially improved through more sophisticated techniques.

Then, the chapter investigated an alternative approach to fuse the same heterogeneous knowledge sources, KGs and text, based on pre-training strategies. To this aim, text embeddings have been used to initialize the embeddings of a GCN, which have then been fine-tuned with the message passing process. This way, the graph structure exploited the textual information to obtain more informative node representations within the KG. The experimental results showed a general improvement in performance, particularly in highly sparse scenarios, where textual information helped to compensate for the weak graph signals. This approach to fuse heterogeneous embeddings is scarcely investigated in KARS, and should be further explored in future works in the field.

In the subsequent section, multimedia embeddings have been considered, and a benchmark analysis run over the datasets extended with multimodal features has been presented. The dataset extension represents one of the contribution carried out during the PhD and discussed in this thesis, since multimedia data was missing for the considered datasets. Moreover, the experimental results consistently showed that multimedia knowledge sources outperform traditional CF methods, with performance improvements becoming even more pronounced when multiple modalities are fused together.

Finally, this chapter presented GETALL!, a recommendation architecture designed to effectively fuse *all* the available knowledge sources for the recommendation task by leveraging attention mechanisms. In particular, GETALL! integrates KGs, textual information, images, audio signals, and videos through the use of attention-based fusion. The experimental results confirmed that combining KGs with multimedia embeddings leads to a general improvement in accuracy, while multimedia knowledge sources enhance the diversity and fairness of the recommendations. This is a relevant result, as investigations addressing both accuracy and beyond-accuracy aspects are typically overlooked in the KARS literature, which tends to focus more on the former than the latter. Furthermore, this outcome is consistent with the findings observed in the user study presented in the previous chapter.

This discussion allows to answer the second RQ of this thesis, investigating how fusing heterogeneous embeddings affects the recommendation performance, and which are the most prominent strategies to perform such fusion.

Answer to RQ2: Fusion of Heterogeneous Embeddings

- Fusing heterogeneous embeddings from KGs and text using attention mechanisms (self- and cross-attention) yields more effective representations than simple concatenation, leading to improved recommendation performance.
- More sophisticated fusion strategies outperform simpler ones, highlighting the importance of modeling relationships between modalities rather than merely concatenating them.
- Pre-training text embeddings to initialize GCNs allows the graph structure to better exploit textual information, improving recommendation quality, especially in sparse data settings.
- The pre-training fusion strategy is a promising but underexplored approach in KARS literature, suggesting future potential for enhancing heterogeneous embedding integration.
- Integrating multimedia knowledge sources significantly boosts recommendation performance compared to traditional CF settings.
- Multimedia sources fusion further improves these gains, showing that combining multiple heterogeneous knowledge sources leads to more accurate models.
- The proposed GETALL! architecture, which fuses all available knowledge sources via attention mechanisms, achieves overall performance gains in both accuracy and beyond-accuracy metrics (diversity and fairness).
- Comprehensive fusion of heterogeneous knowledge sources, including KGs, text, and multimedia data, leads to significant improvements recommendation accuracy performance.
- Attention-based fusion architectures enable more effective integration of diverse modalities, allowing the system to leverage complementary information from different sources. This holds a higher number of sources as well.
- Considering beyond-accuracy aspects such as diversity and fairness together with accuracy provides a more holistic evaluation of recommendation quality, an area often overlooked in existing KARS literature.

Chapter 5

Sustainable and Multi-objective Perspective on RSs

This chapter addresses new challenges in the RS field that have been investigated during the PhD. In particular, this chapter provides a multi-objective perspective on the field, in which the focus is not solely on the accuracy of such systems, but also on *beyond-accuracy* metrics.

The first work described (Section 5.1) explores the application of RecJPQ [132] in large-scale and multimodal recommendation scenarios. The central idea is to quantize user and item ID embeddings to reduce the number of trainable parameters, thereby improving the memory efficiency of recommendation models while maintaining strong accuracy. The novelty of this work lies in the introduction of URecJPQ, the first approach in the multimodal RS literature that achieves a dramatic reduction in both trainable parameters and checkpoint sizes without compromising performance.

Section 5.2 introduces a novel dimension in the evaluation of RSs, focusing on the carbon footprint. This study examines the trade-off between state-of-the-art recommendation models (including knowledge-aware ones) and their emissions produced during the training phase. Unlike previous works, this contribution extends RS evaluation beyond traditional accuracy metrics by integrating environmental impact assessment, offering the first multi-objective framework for KARS evaluation that accounts for both performance and sustainability.

Finally, the chapter discusses two approaches aimed at enhancing environmental sustainability in RSs. The first study in this line (Section 5.3) investigates how applying data reduction strategies to training datasets influences the balance between performance and emissions in RSs and KARS. This work is novel in its systematic analysis of data reduction as a means to improve the performance–emission trade-off, an aspect previously unexplored in the field.

The second study (Section 5.4) continues this research direction by investigating whether emission data can be leveraged as a stopping criterion during training. Specifically, it explores the possibility of early-stopping the training process when further increases in emissions no longer justify improvements in validation performance. This represents the first attempt to integrate emission-awareness directly into the model optimization process, contributing a new perspective on sustainable machine learning practices for recommendation systems.

5.1 Improving Memory Efficeny of Multi-Modal Recommendation Models

5.1.1 Introduction

RSs typically learn k-dimensional embeddings to represent users and items, and these embeddings are commonly referred to as ID embeddings.

However, in large-scale scenarios, when the system has to manage a massive number of users and items, learning unique embeddings for each user and for each item requires a huge amount of memory for storing such embeddings and thus for providing recommendations.

This problem is further exacerbated in the case of multimodal recommendation [257]. Indeed, multimodal approaches generally improve recommendation performance [238], but require more trainable parameters to process and encode multimodal data, such as images or textual descriptions.

A possible solution to solve the aforementioned limitation is the Product Quantization (PQ) approach [59, 83], which is commonly used to compress collections of embeddings: its core idea consists in splitting each embedding into a set of smaller, non-overlapping *sub-embeddings*, and then independently quantizing each of them by assigning it to the index of its closest centroid from a learned codebook (*i.e., sub-IDs*), with the final compressed code being the concatenation of these indices; then, the full embedding can be reconstructed by concatenating its composing sub-embeddings, and used in any downstream task (including recommendation).

However, PQ requires enough memory to train the original full embeddings before compressing them and, since the quantization is not differentiable, it is impossible to jointly compress the embeddings and perform downstream tasks in an end-to-end manner.

This limitation is addressed by the Joint Product Quantization method [243] (JPQ), a PQ approach that assigns the sub-IDs before training the backbone model for the downstream task. The sub-ID assignment is the only non-differentiable operation in PQ, therefore, if the assignments occur before the model training, it can be trained in an end-to-end manner without learning the full embeddings; then, when the full embeddings are needed, they are reconstructed by concatenating the sub-embeddings.

In other words, JPQ can replace the embedding tensors of any backbone model without requiring other changes to the model itself.

Quantization approaches have also been exploited in recommendation scenarios. For example, Lian et al. proposed LightRec [98], a recommendation model that compresses only item ID embeddings through additive quantization [9] and uses the Deep Semantic Similarity Model [77] as the backbone model together with a knowledge distillation component. However, it learns full item ID embeddings, and it is not clear whether the quantization can be used outside of the Deep Semantic Similarity Model. Rajput et al. proposed TIGER [143], an extension of the Residual-Quantized Variational Autoencoder (RQ-VAE) [242] originally developed in the computer vision field. TIGER is a recommendation model inspired by generative retrieval methods [191] and uses textual information to generate item representations. Instead of relying solely on ID-based embeddings, TIGER uses textual side-information in the quantization process to better capture the hierarchy of the items, thereby providing next-item prediction in sequential recommendation. However, TIGER cannot be considered a multimodal model, since it is designed to model only textual information, while other modalities, such as images, are not supported. Petrov and Macdonald [132] proposed RecJPQ, an adaptation of the JPQ method to the sequential recommendation scenario. Differently from TIGER, RecJPQ does not use residual quantization and does not learn hierarchical or semantic embeddings. In RecJPQ, instead of learning full and unique item ID embeddings, the authors applied the JPQ principles to a number of backbone sequential models (BERT4Rec [187], GRU4Rec [73], SAS4Rec [73]) to significantly reduce the sizes of the trained models while maintaining comparable – or even improved – recommendation performance.

Training shared sub-embeddings dramatically reduces the number of trainable parameters and the gradients required for backpropagation, thereby lowering memory usage and enabling training on modern GPUs – even for large-scale catalogs.

However, the use of quantization to improve memory efficiency in traditional top-*k* recommendation scenarios remains largely overlooked, since prior works have focused primarily on sequential recommendation. In this work, the multimodal and top-*k* recommendation setting for large-scale catalogs is addressed to fill this gap.

This study makes the following contributions: (i) This work extends RecJPQ to the top-*k* recommendation scenario by quantizing not only item ID embeddings as in sequential recommendation, but also user ID embeddings – this is called **URecJPQ**; (ii) This work applies URecJPQ to multimodal recommendation in large-scale catalog scenarios, thus drastically reducing the number of trainable parameters; (iii) A quantitative analysis is provided of the trade-off between the decrease of the total number of trainable model parameters (and model checkpoint sizes) and the recommendation performance, in terms of both accuracy and beyond-accuracy metrics.

5.1.2 *URecJPQ* in Multi-modal Large-scale Scenarios

In this section, the original RecJPQ [132] approach is described and URecJPQ is introduced, an extension for top-*k* recommendation.

5.1.3 Basics of RecJPQ

In Figure 5.2, BPR [146] and its quantized version URecJPQ-BPR are compared on the Baby 23 dataset. In vanilla BPR (Fig. 5.2a), the total number of trainable parameters is computed as $(\#users + \#items) * k$, with k being the embedding size. In URecJPQ-BPR (Fig. 5.2b), the total number of trainable parameters becomes $256 * m * d$, with m being the number of sub-spaces and $d = \frac{k}{m}$ the sub-embedding size. URecJPQ dramatically decreases the number of trainable parameters: from

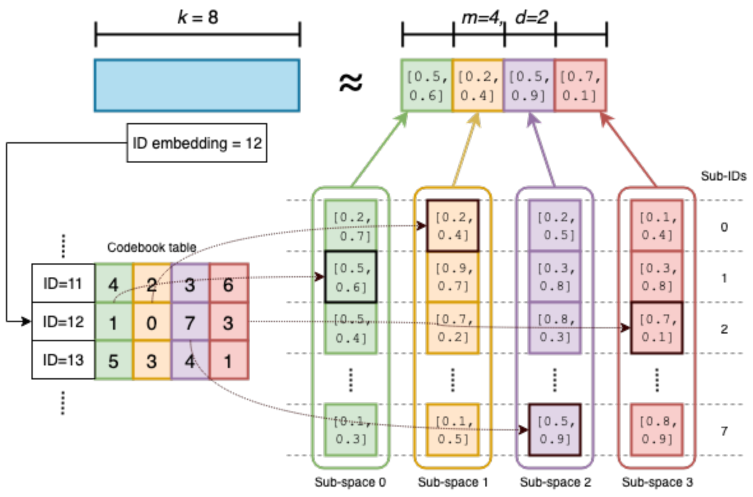


FIGURE 5.1: Embedding quantization in RecJPQ: an 8-dimensional embedding is approximated by the concatenation of 4 sub-embeddings, each 2-dimensional.

12M to 32K for BPR (99.7%), and from 21M to 170K for VBPR [69] (99.3%), which is the multimodal variant of BPR.

As mentioned above, RecJPQ allows an end-to-end training of sequential recommendation models with quantized item embeddings. In particular, given a collection of k-dimensional embeddings $v_1, v_2, ..., v_n$, RecJPQ learns a number of d-dimensional sub-embeddings such that $d \ll k$ and $d = \frac{k}{m}$, with m being the number of sub-spaces considered for the quantization. The number of the embeddings in each sub-space, denoted as the codebook length l, is typically a power of 256, so that the codes can be stored in an l-byte integer structure. Following [132, 243], we set $l = 1$, since in this way, each code can be represented by using a single byte. As a result, each sub-space has 256 sub-embeddings. Figure 5.1 illustrates this process applied to the BPR [146] model.

In addition, RecJPQ supports multiple strategies for code assignment. The simplest, **random**, assigns each item m random sub-IDs (from 0 to 255), one per sub-space. Its main limitation is that it may assign different sub-embeddings to similar items – or the same sub-embeddings to diverse items – risking a loss of semantic coherence.

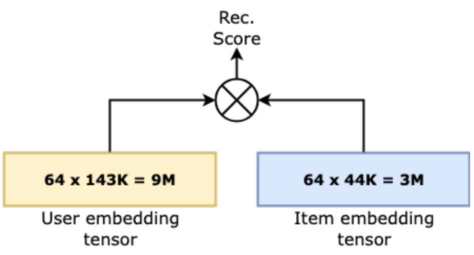
To preserve semantic consistency across items, Truncated SVD [66] offers an alternative code assignment strategy. Specifically, performing SVD with m latent factors on the interaction matrix Y provides low-dimensional user and item embeddings, such that similar users (items) are mapped to similar embeddings. Formally, let $Y \approx U \cdot \Sigma \cdot I^T$, where U and I are the user and item embeddings, and Σ is a diagonal matrix of singular values. The resulting item embeddings I are normalized, and a small random noise is added to avoid identical embeddings for items with identical interaction histories. Each of the m dimensions of the normalized embeddings is then discretized into l quantiles, each containing roughly the same number of items. These quantile bins define the m sub-IDs for each item – one per subspace.

URecJPQ In the original version of RecJPQ [132], the above-described process is applied only to quantize item embeddings. This stems from RecJPQ being applied in sequential recommendations, where only items are represented as embeddings, since users are represented through sequences of the interacted items. In contrast, the proposed **URecJPQ method also learns quantized user ID embeddings by applying the same procedure and code assignment strategies, along with the quantized item ID embeddings.** In the case of the random code assignment strategy, no changes are needed, since random sub-IDs are assigned to both users and items. In the case of Truncated SVD, user embeddings stored in U are exploited to obtain the user sub-IDs, and the item embeddings stored in I are used for the item embeddings, following the same procedure described for RecJPQ.

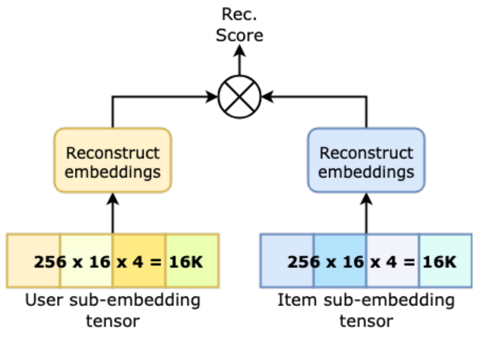
A distinctive trait of RecJPQ is its ability to support end-to-end training of the sequential recommendation models into which it is integrated. **URecJPQ inherits this property, making it applicable to *any* backbone model that explicitly uses user and item ID embeddings** to provide top-*k* recommendation (including most of multimodal models): this improves their efficiency in terms of memory occupancy and number of trainable parameters.

As an example, BPR and its multimodal variant, VBPR, require 12M and 21M trainable parameters on Amazon Baby23, respectively, when full embeddings are learned. In contrast, when URecJPQ is applied on the same dataset, these numbers decrease to 32K and 170K, respectively.

Summary of URecJPQ URecJPQ is an extension of the original RecJPQ applied in a top-*k* recommendation scenario. While RecJPQ only quantizes item embeddings, URecJPQ extends the approach by also quantizing user embeddings. This is necessary since top-*k* recommendation requires both user and item ID embeddings, while sequential recommendation only involves item embeddings. In this



(A) Traditional BPR model. The number of trainable parameters depends on the number of users, items and embedding size (64 in the example).



(B) URecJPQ applied to the BPR model. The number of trainable parameters depends on the number of sub-IDs per space, the number of spaces, and the original embedding size (64 in the example).

FIGURE 5.2: Comparison between the number of trainable parameters of the BPR model on the Baby23 dataset, for 143K users and 44K items. The vanilla version has 12M total trainable parameters, while URecJPQ has only 32K.

way, URecJPQ can be applied to *any* backbone recommendation model, thus replacing user and item tensors in the model design.

Given its flexibility, URecJPQ represents an important contribution to the field, as anyone can integrate URecJPQ in their designed model architectures, thus improving the memory efficiency and the total number of trainable parameters. Moreover, parameter sharing across users and items also acts as an effective regularization mechanism to reduce overfitting [132], and better clusters users and items. The investigation of such aspects is left for future work.

To illustrate its effectiveness, URecJPQ is applied in large-scale and multimodal recommendation scenarios, as discussed in the following sections.

With respect to other works in the area, such as LightRec, TIGER, and RecJPQ, URecJPQ enables end-to-end training of quantized user and item ID embeddings, rather than only item embeddings. Moreover, it is designed for the top-k recommendation setting, instead of sequential recommendation. Because of these aspects, it can be integrated into any backbone recommendation model, including multimodal, in contrast to TIGER, which only supports textual side information, and LightRec that does not support multi-modality at all. In Table 5.1, a table presents a comparison of the approach with other relevant works in the field, highlighting the novelty and the differences of URecJPQ.

Model	End-to-end	Multimodal	Quantization	Type
LightRec	✓	–	Item	Top-k
TIGER	✓	text only	Item	Sequential
RecJPQ	✓	–	Item	Sequential
URecJPQ	✓	✓	User + Item	Top-k

TABLE 5.1: Comparison of our URecJPQ with other quantization approaches, highlighting the novelty of the proposed model.

TABLE 5.2: Statistics of the datasets used in the experiments (on the left), compared to previous and smaller versions of the same datasets (on the right).

	Baby 23	Sports 23	Baby 14	Sports 14
# users	143,602	338,058	19,445	35,598
# items	44,340	165,334	7,037	18,287
# inters.	1,178,208	2,852,781	160,522	295,366
Sparsity	0.9998%	0.9999%	0.9998	0.9995

5.1.4 Experimental Design

Experiments are conducted to answer the following research questions:

- **RQ5.1.1:** How does URecJPQ affect the performance of state-of-the-art multimodal recommendation models?
- **RQ5.1.2:** How does the number of sub-spaces affect overall performance?
- **RQ5.1.3:** How do different code assignment strategies affect overall performance?

Datasets. This study evaluates its methodology on two large-catalog datasets provided with multimodal information. In particular, the *Baby23* and *Sports23* from the Amazon Review 2023[1] datasets [75]. To process these datasets, common approaches [238, 241] are followed: first, a *core-10* filtering is applied to keep only users and items with at least 10 interactions. Then, items for which multimodal information is not available are filtered out. As multimodal information, images and textual descriptions are used, since these are the available modalities for the datasets; however, the method can be applied to any number of modalities. The statistics of the post-processed datasets are reported in Table 5.2. The scripts to

[1]`https://amazon-reviews-2023.github.io`

download the original datasets and process them are provided in our repository[2], along with the post-processed version used in the experiments, and the source code to reproduce the results. A discussion of the scale of the datasets is provided by Hou et al. [75].

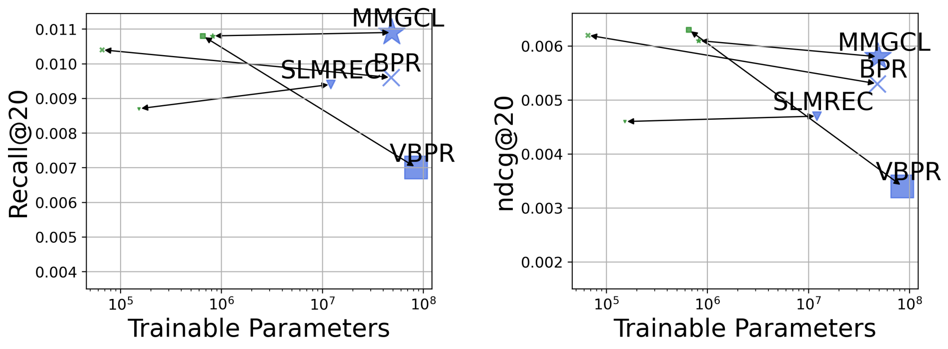
Recommendation models. The methodology is applied to four state-of-the-art recommendation models commonly used in collaborative filtering and multimodal recommendation, evaluated in a top-*k* recommendation setting. Note that each of these models requires learning user and item ID embeddings; thus, memory efficiency can be improved with URecJPQ, as previously discussed. In particular, the following models are used: BPR [146], VBPR [69], SLMRec [190], MMGCL [240]. More details about these models have been already discussed in Section 2.3.

Moreover, to guarantee and simplify replicability, the methodology is implemented within the MMRec [257] framework, which already provides implementations of all the aforementioned recommendation models.

Evaluation protocol. Following the literature in the field [248, 256, 238], the datasets are split into training, validation, and testing sets with an 80:10:10 ratio. Performance is evaluated using typical top-*k* recommendation metrics, namely Recall@20 and NDCG@20. Moreover, since the focus is to investigate how URecJPQ can be applied in large-scale scenarios, checkpoint sizes and the number of trainable parameters of each model are used as metrics. As the validation metric, Recall@20 from MMRec's default configuration with early stopping after 10 consecutive epochs without improvement is employed.

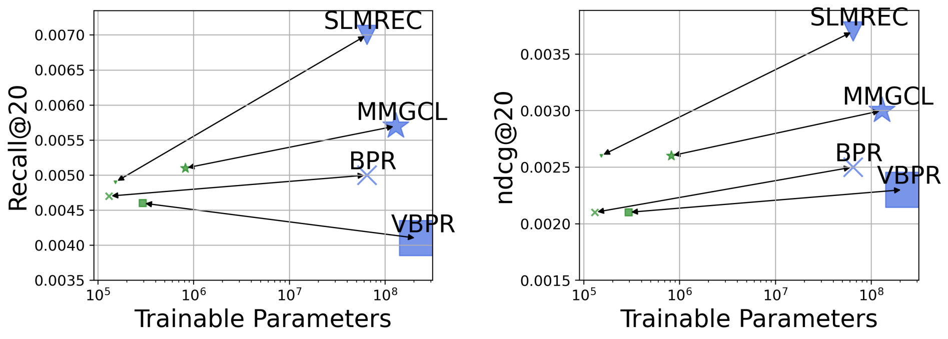
Hyperparameters. A grid search over the parameters of both the quantization performed by URecJPQ and the specific parameters of the backbone models is conducted. For the vanilla versions of the considered models, full embeddings of sizes $k = \{64, 128, 256\}$ are explored. Then, the quantization is applied with the codebook length fixed to $l = 1$, and different numbers of sub-spaces are explored in the range $m = \{8, 16, 32, 64, 128\}$, as well as two code assignment strategies, namely, *random* and *SVD*. Due to space constraints, the hyperparameter values specific to the backbone models here are not provided. Instead, the aforementioned repository provides full details. Notably, these parameters are optimized for both the vanilla and quantized versions of the models.

[2]The repository is currently anonymous due to the double-blind policy for the submitted paper: `https://anonymous.4open.science/r/large_mmrecjpq-839B/README.md`



(A) Recall on Baby23

(B) NDCG on Baby23



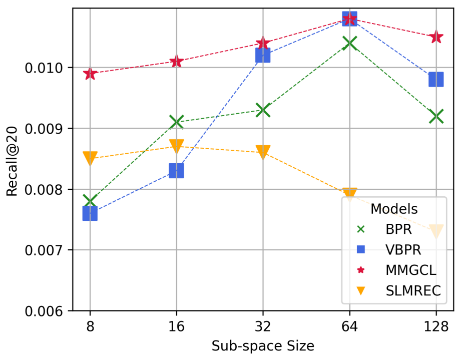
(C) Recall on Sports23

(D) NDCG on Sports23

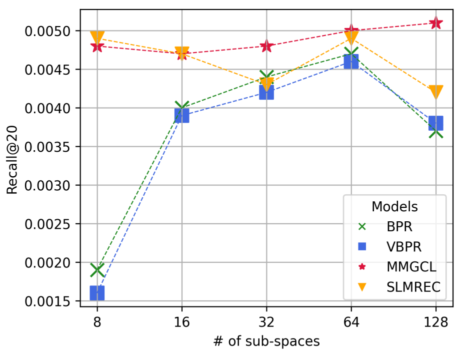
FIGURE 5.3: Model performance on the Baby23 and Sports23 datasets, in terms of Recall and NDCG. On the x-axis, the number of trainable parameters is reported, while on the y-axis the metric values are reported. The size of each dot indicates the size of the model checkpoint.

5.1.5 Results Discussion

To answer RQ5.1.1, Figure 5.3 reports the results obtained for the Baby23 and Sports23 datasets, for both the Recall and NDCG metrics. As shown by the plots, there is a marked decrease in the number of trainable parameters for both datasets: the number decreases by three orders of magnitude, from 10^8 when no quantization is applied to 10^5 when URecJPQ is applied. Also for the checkpoint sizes (depicted by the size of the dots), there is a dramatic decrease: for example, the checkpoint size of VBPR on Baby23 is 325 MB, which decreases to 14 MB with the application of URecJPQ. Similarly, the MMGCL's checkpoint size is almost 500 MB, while its quantized variant is 15 MB. In relation to recommendation performance, for the Baby23 dataset (Figures 5.3a and 5.3b), MMGCL and SLMRec



(A) Recall on Baby23



(B) Recall on Sports23

FIGURE 5.4: Impact of number of sub-spaces on performance.

perform slightly better than their quantized counterparts. In contrast, applying URecJPQ to BPR and VBPR leads to improvements in Recall and NDCG. Moreover, URecJPQ-MMGCL outperforms its vanilla version in terms of NDCG. Similar findings can be observed for the Sports23 dataset (Figures 5.3c and 5.3d). Interestingly, in this case, VBPR achieves a lower Recall than its quantized counterpart, while the other models all slightly outperform their quantized versions. A similar trend is observed for NDCG.

To conclude for RQ5.1.1, the quantized models exhibit only a marginal decrease in performance — averaging 8.5% in Recall and 16% in NDCG — while achieving substantial reductions in checkpoint sizes and in the number of trainable parameters, by up to 98% and 99%, respectively. These results show how URecJPQ can support the deployment of industrial RSs, where recommendations must be provided to a massive number of users.

To answer RQ5.1.2, Figure 5.4 reports how different numbers of sub-spaces m affect the performance of the recommendation. Due to space reasons, the focus is on Recall, but similar findings are observed for NDCG. For these plots, the best performance obtained by the quantized models for each number of sub-spaces is considered.

Overall, performance increases with the number of sub-spaces, peaking at 64, after which it starts to decrease. The only exceptions to this behavior are SLMRec on Baby23, whose performance drops earlier and shows a less monotonic trend on Sports23, and MMGCL on Baby23, whose peak performance is reached at 128 sub-spaces.

TABLE 5.3: Impact of code assignment strategy on performance (Recall@20).

Dataset	**Baby23**		**Sports23**	
Strategy	**random**	**svd**	**random**	**svd**
BPR	**0.0104**	0.0056	**0.0047**	0.0019
VBPR	**0.0108**	0.0061	**0.0046**	0.0017
SLMRec	0.0073	**0.0087**	0.0021	**0.0049**
MMGCL	**0.0108**	0.0094	0.0048	**0.0051**

To address RQ5.1.2, it is observed that using 64 sub-spaces for the quantization leads to the best performance in terms of Recall among the considered possibilities.

To address RQ5.1.3, Table 5.3 reports how random and SVD code-assignment strategies affect the performance in terms of Recall@20. The best performance achieved under each strategy is reported. The random strategy performs best in 3 of 4 Baby23 cases and 2 of 4 Sports23 cases. BPR and VBPR are more sensitive to code assignment than SLMRec and MMGCL, showing larger performance gaps. Moreover, on the Sports23 dataset, the SVD strategy seems to be suboptimal for BPR and VBPR. This is likely due to the higher sparsity of this dataset, in which a simple strategy such as SVD fails to capture meaningful connections between users and items, leading to the introduction of noise that decreases the overall performance. On the other hand, more complex models, such as SLMRec and MMGCL, benefit more from this strategy, as the performance gap between using random and SVD strategies is smaller.

The difference in the behavior of these models likely lies in their architecture and complexity. While BPR and VBPR learn only user and item ID embeddings (plus multimodal features, in the case of VBPR), replacing full embeddings with quantized ones drastically reduces the trainable parameters. In this setting, SVD-based sub-ID assignments may fail to provide enough information for capturing patterns in the data, leading to suboptimal performance. In contrast, random sub-ID assignments increase the chance of learning more diverse user preferences, leading to better results than SVD-based quantization. However, in the case of more complex models (SLMRec and MMGCL) that do not limit themselves to learning ID embeddings from training data but instead perform more sophisticated operations on interactions and multimodal information, the impact of the code assignment strategy diminishes. In such cases, the SVD strategy can even lead to better performance than the random strategy (Table 5.3), as happens for SLMRec on both the datasets, and for MMGCL for Sports23. Another aspect to consider is that Sports23 is considerably sparser than Baby23 (Table 5.2), making SVD sub-IDs less effective at fully capturing user–item connections. Consequently, BPR and VBPR perform up to three times better with random codes, whereas SLMRec and MMGCL benefit more from SVD. A deeper analysis of this effect is left for future work.

To conclude for RQ5.1.3, the code assignment strategy is a relevant factor for the overall effectiveness of the model: the random strategy works better for simpler models, while the SVD strategy is more suitable for complex models, particularly on sparser datasets.

5.1.6 Take-Home Messages

In this paper, URecJPQ is proposed, as an extension of RecJPQ to the top-k recommendation task. The experiments conducted in a large-scale and multimodal recommendation setting showed that URecJPQ is an effective solution for dramatically reducing the number of trainable parameters (with reductions of up to 98%), while incurring only marginal performance drops (8.5% on average), or even improvements when considering the BPR, VBPR and MMGCL models. The potential impact of this methodology in industrial large-scale scenarios is supported, where the quantization approach can offer significant practical benefits, enabling training recommendation models on even larger datasets.

In future work, the exploration of additional recommendation models (e.g., graph-based models such as LATTICE [247] and FREEDOM [258]) and other datasets to further generalize findings is planned. Furthermore, the investigation of strategies to also compress multimodal embeddings is planned. Finally, the quantization impact on beyond-accuracy aspects of recommendation models, including fairness, novelty and diversity of the recommendation lists, as well as the overall energy efficiency during training, is planned.

Main Findings

- **RQ5.1.1**: URecJPQ dramatically reduced the number of trainable parameters and checkpoint sizes by up to 99%, while showing a much smaller performance reduction, from 8.5% to 16%.
- **RQ5.1.2**: Among the possibilities considered in this work, a quantization exploiting 64 sub-spaces leads to the best performance, regardless the original embedding size.
- **RQ5.1.3**: The code assignment strategies is a crucial aspect of this work. Random strategies better work for simple model, while SVD is more suitable for complex models and sparser dataset.

These results have been carried out during a visiting period at the University of Glasgow, Scotland, and have been submitted to the ACM UMAP 2026 Conferece (Giuseppe Spillo et al. "URecJPQ: Memory-efficient Multimodal Recommendation Models through RecJPQ in Large-Scale Scenarios". In: *Submitted to ACM International Conference on User Modeling, Adaptation and Personalization (UMAP). 2026*. 2026).

5.2 Benchmarking State-of-the-art Recommendation Model Carbon Footprint

5.2.1 Introduction and Motivations

In recent years, AI algorithms have emerged as transformative tools, revolutionizing industries in several domains, ranging from healthcare to finance. Their ability to automate tasks, extract insights from vast datasets, and facilitate decision-making processes has led to significant advancements in technology and reshaped societal landscapes. However, this unprecedented progress has come at a significant cost, notably the escalating energy consumption associated with AI systems. As these algorithms become increasingly sophisticated and ubiquitous, their demand for computational resources has grown exponentially, leading to a surge in power consumption and environmental concerns [228]. The expanding carbon footprint of AI raises pressing questions about sustainability and the trade-offs between technological innovation and ecological impact. However, in spite of the interest in designing and implementing AI algorithms that are more energy-efficient, the environmental impact of complex algorithms, such as RSs, and the trade-offs between sustainability and their overall performance are generally overlooked.

The exponential rise in energy consumption by AI algorithms is particularly relevant. Indeed, as stated in [90], it impacts the concentration of greenhouse gases (GHGs) in the atmosphere, which in turn significantly contributes to climate change. As an example, Strubell et al. [185] found that training recent translation engines can emit almost 300 tonnes of CO_2. According to official estimates [3], this is equivalent to the yearly energy use of more than 40 houses. This underscores the urgent necessity to establish benchmarks for measuring and mitigating the environmental impact of these algorithms [155]. The problem of estimating the CO_2 footprint of AI algorithms has been addressed in a number of recent papers [88, 131].

The dichotomy between *Red* and *Green* AI was first introduced by Schwartz in 2020 [155]. Following the paper [179], the research area has rapidly spread, as shown by the recent review on Green AI [205], which compared several studies quantifying energy consumption of AI models. One of the first attempts in this direction is due to Strubell et al. [185], who estimated the emissions of training and fine-tuning a large Transformer model. The recent spread of large language models paved the way for a large number of studies trying to estimate both the effectiveness and the emissions of such algorithms. As an example, the carbon footprint of another popular 176-billion-parameter language model, *i.e.,* BLOOM, is estimated by Luccioni et al. in [111]. The emissions of BLOOM across its life-cycle are quantified as 24.7 tonnes of $CO2-eq$ in [111], and the carbon footprint of NLP algorithms is also studied in [13]. Other large benchmarks are presented in [110] and [30], presenting a survey of the carbon emissions of machine learning models. Finally, carbon emissions of state-of-the-art large language models are quantified in [105]. In this case, due to the huge number of parameters, these models have an enormous carbon footprint and emit a lot. Regarding neural networks, the perspective has been explored by Patterson et al. [131]. Finally, the carbon footprint of medical image segmentation pipelines is estimated in [157].

As regards tools for quantifying the carbon footprint of algorithms, mention is made of the Machine Learning Emissions Calculator [88] and the framework introduced in [90], which aims to estimate the carbon footprint of any computational task. Similarly, [22] presents ECO2AI, a package that tracks the energy consumption of AI models. As discussed in the next sections, CodeCarbon was used to estimate the carbon footprint of recommendation algorithms.

[3]`https://www.epa.gov/energy/greenhouse-gas-equivalencies-calculator`

The featured publications [179, 166] focus on RSs, whose carbon footprint is currently under-investigated. Generally speaking, RSs are a *data-intensive* technology, so they require long training times and a huge amount of energy. Indeed, as shown by the analyses carried out in [179, 166], training an RS for 1 year emits the same quantity of CO_2 as emitted by a car driven for about 450 miles. With respect to these works, the distinctive trait of the present approach is the analysis of the carbon footprint of recommender systems, a data-intensive family of algorithms whose carbon footprint is currently overlooked.

As regards the role of hyper-parameter tuning, many publications have shown that it significantly affects energy costs. Accordingly, some attempts first tried to reduce the number of iterations required to tune the parameters [182], or, alternatively, to develop and incorporate power consumption in the optimization process [33]. The impact of hyper-parameter optimization on energy consumption is also discussed in [20]. With respect to the above-mentioned work, the novelty of the present contributions lies in the focus on recommendation algorithms. To the best of knowledge, the impact of hyper-parameter tuning on the trade-off between accuracy and efficiency of recommendation algorithms has not been investigated.

Generally speaking, the works [179, 166] are framed in the general area of *sustainability of AI*, which refers to the capacity of AI systems and technologies to be developed, deployed, and utilized in a manner that promotes long-term environmental, social, and economic well-being, while minimizing adverse impacts on ecosystems, societies, and economies. This includes considerations such as energy efficiency, ethical use of data, fairness and transparency in decision-making processes, and the equitable distribution of benefits and risks associated with AI advancements [201]. In particular, attention is directed to the *environmental* pillar of sustainability [140] and to the recent area of *Green AI* [155]. Indeed, benchmarking several representative recommender systems in terms of performance and carbon footprint is undertaken. Moreover, the analysis also includes the study of the impact of hyper-parameters in the development of sustainable-aware recommender systems. By considering the general Sustainable Development Goals (SDGs)[4], it can be stated that the contribution impacts SDG 7 and SDG 13, related to clean energy and climate action, respectively.

In today's rapidly advancing technological landscape, prioritizing sustainability and reducing the carbon footprint in AI algorithms, such as *recommender systems*,

[4]`https://sdgs.un.org/2030agenda`

is fundamental for mitigating environmental impact and fostering a more eco-friendly technological landscape. Accordingly, the papers [179, 166] carry out extensive benchmarks of the trade-off between the *environmental impact* and the *performance* of the most popular recommendation algorithms, evaluated on three different state-of-the-art datasets. In particular, the investigation considered: *(a)* whether more recent recommendation algorithms justify their increase in carbon emissions with an equivalent increase in performance; *(b)* what is the impact of hyper-parameter tuning strategies on the trade-off between carbon footprint and performance. Overall, the work showed that early and less sophisticated algorithms still represent a good alternative for implementing *greener* recommendation algorithms, especially when parameter tuning is needed. In conclusion, this work discusses the challenges related to energy consumption in recommender systems, with the aim of initiating a new generation of algorithms that prioritize sustainability considerations. To guarantee reproducibility, all scripts and the source code are released on GitHub[5].

The publications [179, 166] shed light on the problem of carbon emissions and energy consumption of RSs. The main finding is that the choice of the optimal recommendation algorithm requires thorough analysis, since less sophisticated algorithms can still constitute a viable alternative to implementing *greener* recommendation algorithms, particularly when parameter tuning is required. The papers [179, 166] provide the following contributions: an analysis of the trade-off between energy consumption, carbon emissions and the predictive accuracy of state-of-the-art recommendation algorithms; an investigation of the impact of hyper-parameter tuning on the overall carbon emissions of the algorithms; a sketch of the take-home messages and the main limitations of this work. This work aims to foster research in the area and drive the development of a new generation of recommendation algorithms that also take sustainability into account.

5.2.2 Methodology

Selected Recommendation Algorithms In the featured publications [179, 166], 14 recommendation algorithms were benchmarked. Based on the analysis provided in the previous section, for each paradigm one or more representative models were selected (14 in total). In particular, general recommendation models, algorithms that exploit deep learning techniques, graph-based methods, and recent knowledge-aware algorithms were included. In particular, the following models

[5]`https://github.com/swapUniba/RecSysCarbonFootprint`

have been considered: BPR [146], ItemKNN [50], LINE [189], DMF [235], MultiDAE [99], NGCF [221], DGCF [218], LightGCN [70], CKE [246], CFKG [251], KGNN-LS [210], RippleNet [213]. Their description is discussed in Section 2.3, and is not reported here to avoid repetitions.

Tracking Carbon Emissions and Energy Consumption Nowadays, many organizations and governments use the carbon dioxide equivalent[6] (or CO_2 equivalent, abbreviated as CO_2-eq) as a standard metric to track the emissions of various greenhouse gases. Currently, the amount of CO_2-eq is widely used to simplify the quantification and comparison of the various emissions across different settings and domains, as shown in several pieces of literature, including construction industry [45], vehicle life-cycle [21], air conditioning [192], and AI [185, 111, 13], just to cite a few of them. Accordingly, CO_2-eq is used to track carbon emissions as well. A state-of-the-art method [130] to compute or estimate the CO_2-eq is based on the combination of the values of Carbon Intensity (CI) and Power Consumed (PC). Formally:

$$emissions = CI \cdot PC \qquad [kg/kWh \cdot kWh \equiv kg]$$

The Carbon Intensity (CI) of the consumed electricity is computed by combining the emissions of the different sources used to generate the electricity. For both fossil and renewable sources, each source is associated with a specific carbon intensity (i.e., a known amount of carbon dioxide emitted).

CI values generally depend on the energy mix used for the computation, with each source having its own CO_2 emissions; in general, the overall CI of an energy mix is computed as a weighted average of the emissions of each energy source in the mix. Formally, let S be an energy mix, composed of s energy sources denoted as e_s; each source e_s is present with a percentage of p_s in the mixture; then, the Carbon Intensity *CI* of the mix is computed as follows:

$$CI = \sum_{s \in S} e_s \cdot p_s$$

However, since different countries use different energy source mixtures, their carbon intensities differ; for example, in 2022 Italy used a mixture consisting of 8% coal, 51% gas, 2% oil, 11% hydroelectric, 7% wind, 10% solar, 6% bio energy, and

[6] `https://ec.europa.eu/eurostat/statistics-explained/index.php?title=Glossary:Carbon_dioxide_equivalent`

the remainder from other renewable sources (including geothermal, tidal, and wave generation); in the same year, the United Kingdom used a mixture consisting of 1.5% coal, 38% gas, 4% oil, 15% nuclear, 1.5% hydroelectric, 25% wind, 4% solar, and 11% bio-energy. Since these mixtures differ, the CI for Italy and the CI for the United Kingdom in 2022 are different: 261 grams of CO_2-eq per kilowatt-hour in the United Kingdom, and 373 grams of CO_2-eq per kilowatt-hour for Italy; the world average is about 475 grams of CO_2-eq per kilowatt-hour. In this case, the world average is used as the reference mixture. The mix is composed of 19% coal, 39% gas, 1% oil, 18% nuclear, 6% hydroelectric, 10% wind, 5% solar, 1% bio energy, and the remainder is represented by other renewable sources (including geothermal, tidal, and wave generation). This kind of analysis is non-trivial, since it motivates large companies to deploy their data centers in specific countries [199], given that the location of computation has a strong influence on the overall carbon footprint.

Next, with regard to the Power Consumed (PC), it reflects the total energy required by the hardware during the computation, and can be obtained by using low-level libraries that track the consumption of CPUs, GPUs, and other components (including RAM). In the experiments, PC was tracked every 15 seconds, and an aggregated result related to the energy consumed was obtained at the end of the computation.

Currently, several tools are available that can estimate or measure the CO_2-eq values of AI algorithms (and, more generally, any kind of computation) by estimating *CI* and *PC* values; for the experiments, CodeCarbon[7] was chosen, since other tools (such as ML CO_2 Impact[8]) estimate the energy consumption, whereas CodeCarbon measures it, resulting in a more precise and reliable tracking.

5.2.3 Experimental Evaluation

In the experimental session, a large benchmark of the previously presented recommendation model was run. Specifically, the experiments aimed to answer the following RQs:

- **RQ5.2.1 - Accuracy vs Sustainability:** Is there any trade-off between the performance and the carbon emissions of the main recommendation algorithms?

[7]`https://mlco2.github.io/codecarbon/`
[8]`https://mlco2.github.io/impact/`

- **RQ5.2.2 - Tuning of Hyper-Parameters:** What is the impact of hyper-parameter tuning on the trade-off between carbon footprint and performance of the algorithms?

To answer RQ5.2.1, an extensive benchmark was conducted to assess both the performance of the previously introduced recommendation algorithms and their CO_2-eq emissions. The high-level workflow is depicted in Figure 5.5. Specifically, for each recommendation model, the algorithm was supplied with the training data. Subsequently, the model was trained and the recommendations generated by the algorithm were collected, with CO_2 emissions tracked. To answer RQ5.2.2, the above-mentioned protocol was repeated for all combinations of parameters available for each algorithm, and the trade-off between the quality of the recommendations in their best-performing run and the overall cost was analyzed to identify the optimal combination of parameters.

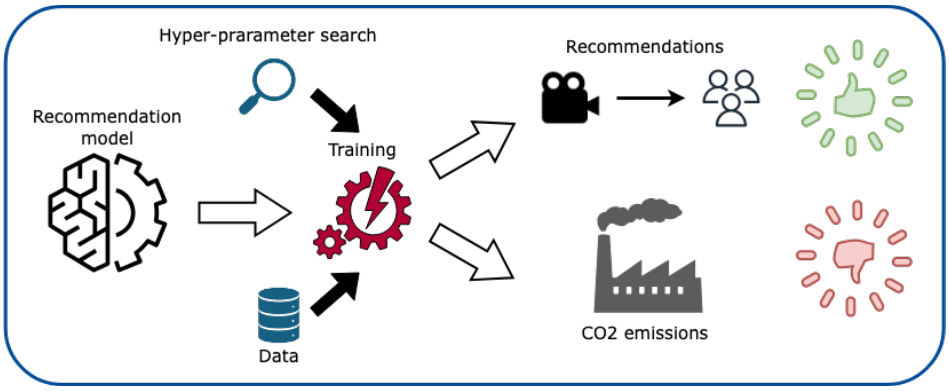


FIGURE 5.5: Experimental Protocol.

Datasets. To perform this analysis, three datasets in the areas of movies, books, and news recommendations were used, respectively. Benchmarks were run on Movielens-1M, Amazon-Books, and Mind, which have been discussed and presented in Section [118].

Recommendation Algorithms The RecBole [234] implementation of each recommendation algorithm was employed, and the selection was limited to those available in the library.

Recommendation Performance Evaluation metrics are calculated using RecBole, as well. The evaluation consider all the metrics implemented in RecBole, but for the sake of simplicity, results are presented in terms of Recall, NDCG, average popularity and Gini Index. This allows assessment of the accuracy, ranking, popularity bias and diversity of the recommendations.

Measuring Carbon Footprint. To track the emissions and the carbon footprint of the recommendation models, the Python library CodeCarbon was used; moreover, the world average value of carbon intensity[9] was used to provide general findings that reflect world average scenarios. Finally, emissions of the models are discussed in terms of grams (g) rather than kilograms (kg).

Hyper-parameter Tuning. To answer RQ5.2.2, a protocol for hyper-parameter tuning based on the iteration of several runs for each algorithm was designed. A set of suitable values for some of the parameters of each recommendation model was defined. These parameters include elements such as learning rate, number of epochs, or the size of the embeddings when available. The complete list of parameters considered for each model are reported in the related publication [166] for the sake of readability.

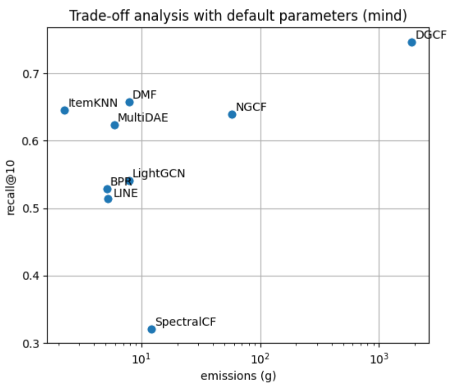
5.2.4 Results Discussion

RQ5.2.1: Accuracy vs Sustainability This study analyzes the trade-off between model performance and carbon emissions. The results, calculated for all datasets, are presented in Figures 5.6 (Mind), 5.7 (MovieLens), and 5.8 (Amazon).

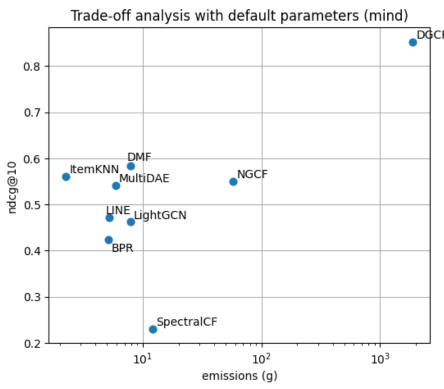
For the Mind datasets, the average carbon footprint of a run was equal to 221 grams of CO_2-equivalent. As expected, ItemKNN has the lowest carbon footprint (3.73 grams) while DGCF has the highest value (around 1.8 kilograms). As regards performance, DGCF achieved the best overall recall and NDCG. Consequently, the use of complex graph-based learning mechanisms such as those implemented in DGCF leads to an improvement in the accuracy of the recommendations, at the cost of an approximately 840x increase in emissions.

Another interesting outcome is that *high-emissions* models such as NGCF (emitting 57 grams of CO_2-eq) do not compensate their computational demands with an equivalent increase in performance. Indeed, their results are often worse with respect to those obtained by simpler and *low-emissions* algorithms. Such behavior is also confirmed when non-accuracy metrics are taken into account. Indeed,

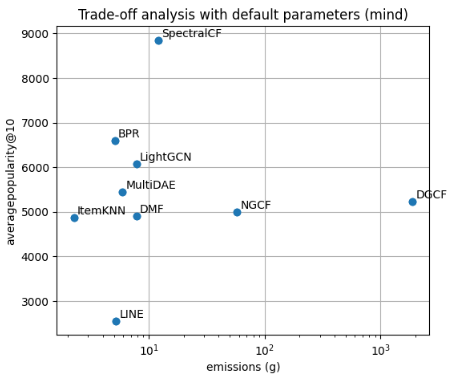
[9]https://www.iea.org/reports/global-energy-co2-status-report-2019/emissions



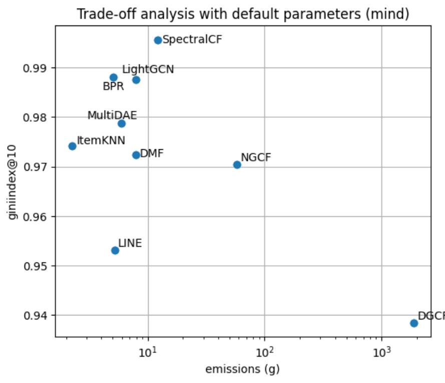
(A) Recall



(B) NDCG



(C) Avg. Pop.



(D) Gini Index

FIGURE 5.6: Comparison between carbon emissions (*X-axis*) and performance (*Y-axis*) on **Mind** data.

the increase in terms of CO_2-emissions justifies the relatively good performance of DGCF in terms of the Gini Index, whose scores are significantly lower with respect to the other algorithms considered. Conversely, the analysis of the average popularity shows that an algorithm with lower emissions, such as LINE, provides results that are better than those obtained by high-emissions algorithms. Clearly, all results were obtained using the default parameters of the algorithms. The impact and role of parameter tuning will be investigated in the next experiment.

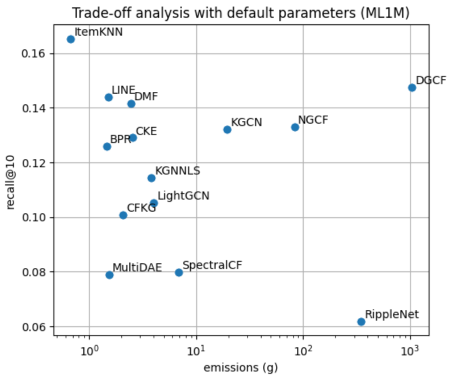
Next, the results for **MovieLens data** are analyzed. Simple models have very low emissions (0.67 grams of CO_2 equivalent for ItemKNN, 1.45 grams for BPR), while complex models have a carbon footprint which is comparable (1.03 kilograms for DGCF) to that obtained for Mind data. The first relevant outcome of this dataset is that models belonging to the *knowledge-aware* family are characterized by a large amount of carbon emissions (384 grams). Unfortunately, such a requirement is not supported by any relevant improvement in performance metrics.

As regards the results, the analysis of the NDCG provides findings in line with those obtained for Mind data, since DGCF justifies the increase in terms of carbon footprint with an equivalent increase in terms of accuracy, and similar results were obtained for NGCF and KGCN. Conversely, regarding Recall, ItemKNN has the best trade-off between recall and its carbon footprint. A final remark should be made regarding more sophisticated algorithms, such as RippleNet. Indeed, while the amount of emissions characterizing these models is significantly higher, most experiments show that their performance is lower than that of simpler methods. This should be taken into account when selecting the optimal recommendation algorithm. Finally, regarding non-accuracy metrics, results generally show that *high-emissions* approaches such as DGCF and RippleNet tend to reduce popularity bias and increase diversity. In this case, results clearly show that methods such as LINE generally have the best trade-off since they obtain a relatively low carbon footprint (just 1.5 grams of CO_2-eq), but they obtain similar performance in terms of average popularity and the Gini index.

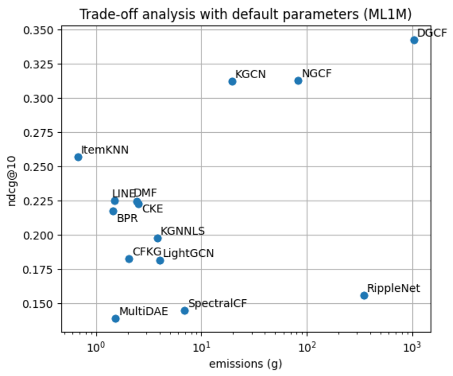
Finally, with regard to **Amazon Books**, the lowest emissions were observed for ItemKNN (3.4 grams), while the highest carbon footprint was observed for running DGCF (2.88 kg of CO_2 equivalent). Differing from previous analyses, ItemKNN provides the best trade-off between carbon emissions, accuracy, and non-accuracy metrics. In this case, algorithms with higher emissions such as DGCF and NGCF do not justify the increase in emissions with a commensurate gain in predictive accuracy. Moreover, knowledge-aware algorithms such as RippleNet typically require substantial computational resources, without achieving equivalent performances.

RQ5.2.1 can be answered by noting that the choice of the recommendation algorithm with the optimal trade-off between performance and carbon emissions is not trivial. ItemKNN emerged as the best option in terms of Recall on two of three datasets, with good performance on the third dataset as well. The only algorithm competing with ItemKNN is DGCF, which provides good recall and NDCG on all datasets. However, as previously stated, this always entails a substantial increase in emissions (up to 850x), with relative benefits in terms of accuracy.

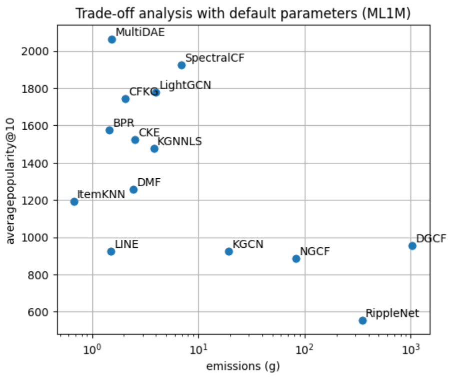
RQ5.2.2: Hyper-parameter tuning Next, to answer RQ5.2.2, the protocol presented previously was followed, and tuning of hyper-parameters was performed



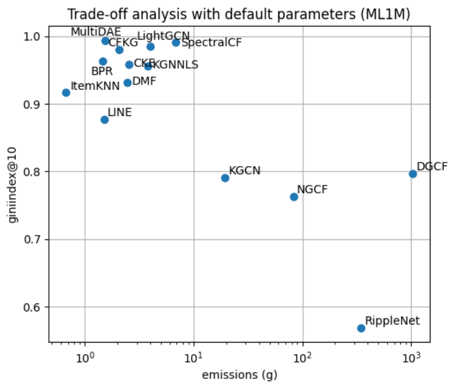
(A) Recall



(B) NDCG

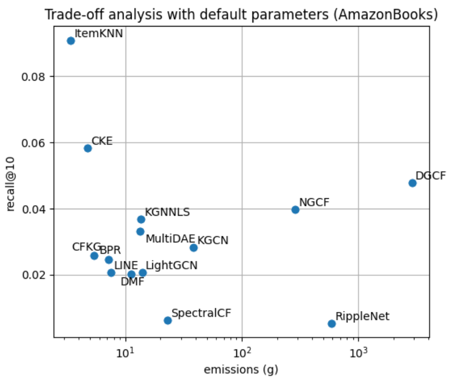


(C) Avg. Pop.

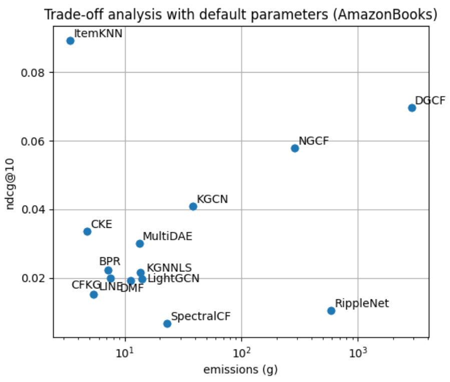


(D) Gini Index

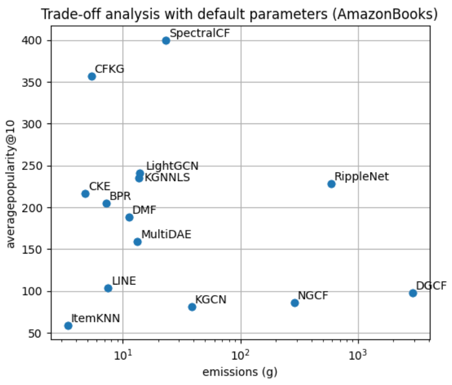
FIGURE 5.7: Comparison between carbon emissions (*X-axis*) and performance (*Y-axis*) on **MovieLens** data.



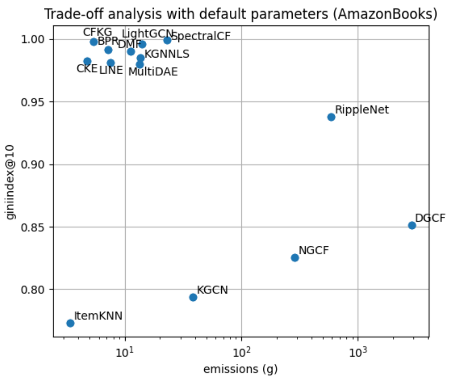
(A) Recall



(B) NDCG



(C) Avg. Pop.



(D) Gini Index

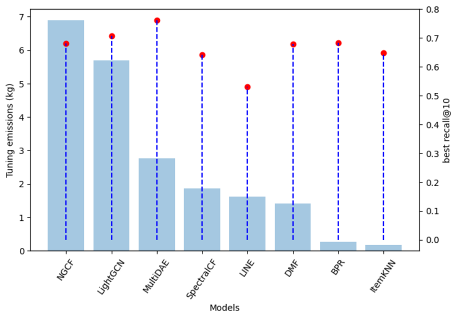
FIGURE 5.8: Comparison between carbon emissions (*X-axis*) and performance *(Y-axis*) on **Amazon-Books** data.

on all the recommendation models compared. The emissions of an algorithm are calculated as the sum of the emissions of each single run, while for each metric the best value obtained over all runs is considered. The aim of the experiment is to quantify how the search for the optimal hyper-parameters affects the trade-off between emissions and performance. The results are presented in Figure 5.9, Figure 5.10 and Figure 5.11.

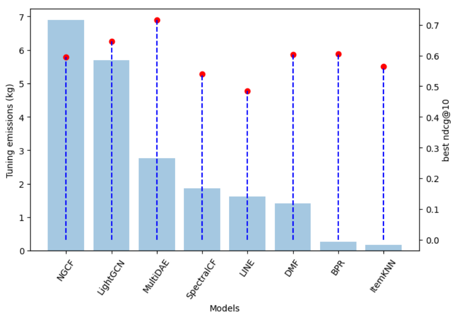
Overall, the results are in line with expectations, since all the datasets showed that the search for the optimal hyper-parameters significantly increases the overall emissions of the algorithms, but such an increase is not balanced by an equally significant increase in performance. For Recall, MultiDAE emerges as the best-performing algorithm, with a 14.6% increase in Recall (from 0.6241 to 0.7169). However, this comes at the cost of an exponential increase in emissions, exploding from 5.9 grams (with default parameters) to almost 7 kilos (by summing all the runs). Similar findings hold for LightGCN. Overall, less sophisticated algorithms such as BPR emerge as suitable alternatives, since BPR obtained a 28.59% increase in performance, at the cost of 260 grams of emissions, which is 35x smaller with respect to NGCF. Finally, the use of hyper-parameter tuning for very simple algorithms such as ItemKNN has to be carefully considered. Indeed, in this case the increase in performance is very small (from 0.6456 to 0.6481) and the emissions increase (from 2 grams to 170). Generally speaking, all these findings are valid also for NDCG, Gini and Average Popularity.

Next, by extending the analysis to other datasets, *i.e.*, ML1M, it is possible to study the impact of hyper-parameter tuning on knowledge-aware methods such as CKE. In this case, the introduction of knowledge features leads to outcomes similar to those observed for more sophisticated methods, with a 28.68% increase in recall, while emissions rise to 1.6 kilos (from 2.5 grams). On these data, MultiDAE emerged again as the algorithm with the best compromise between performance and emissions, together with CFKG.

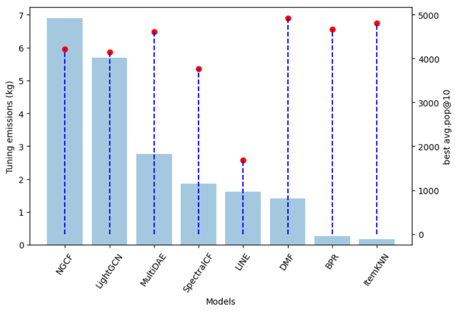
All issues characterizing the tuning of hyperparameters clearly emerge from analyzing Amazon results. In this case, algorithms such as NGCF, KGCN and LightGCN emit a huge amount of CO_2-eq (between 14 and 16 kilos), but their performance remains significantly below algorithms such as ItemKNN. Overall, methods such as CKE and CFKG emerged again as the strategies having the best compromise between increase in performance and increase in emissions.



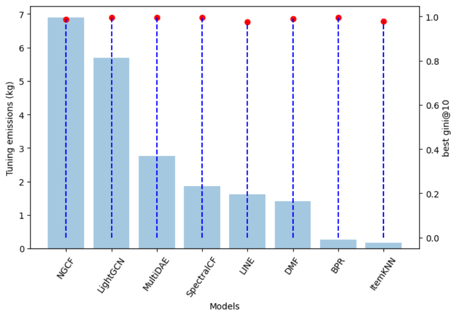
(A) Recall



(B) NDCG

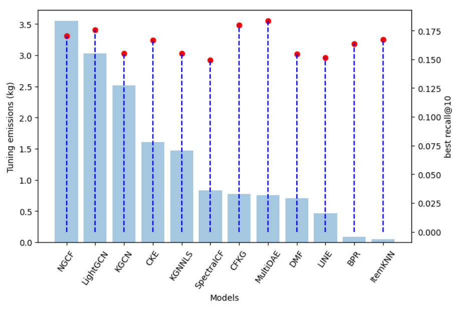


(C) Avg. Pop.

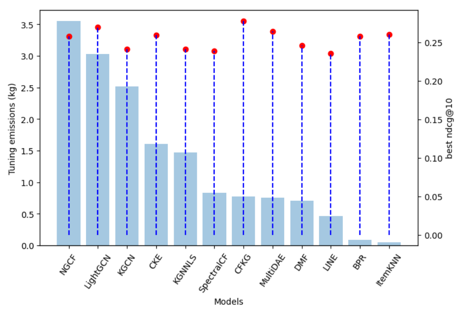


(D) Gini Index

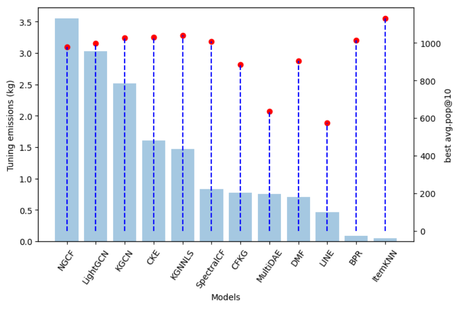
FIGURE 5.9: **Mind Dataset.** Cost of the hyperparameter tuning in terms of CO_2 emissions (kg) and performances obtained in terms of recall (dots). Vertical bars refer to the emissions, while red dots refer to the considered metric.



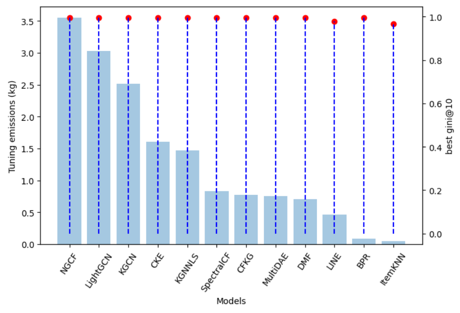
(A) Recall



(B) NDCG

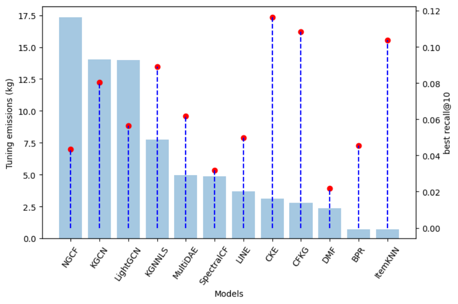


(C) Avg. Pop.

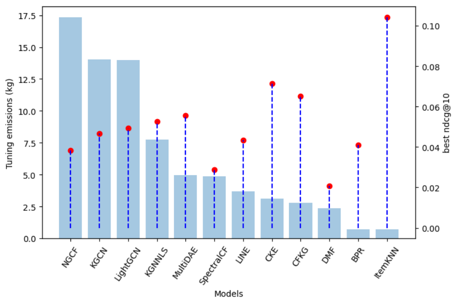


(D) Gini Index

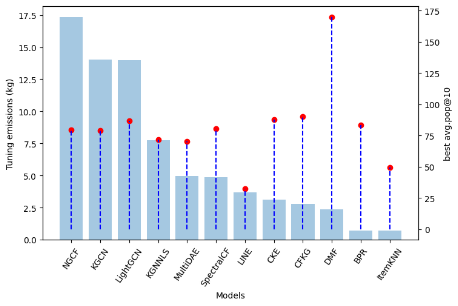
FIGURE 5.10: **ML1M Dataset.** Cost of the hyperparameter tuning in terms of CO_2 emissions (kg) and performances obtained in terms of recall (dots). Vertical bars refer to the emissions, while red dots refer to the considered metric.



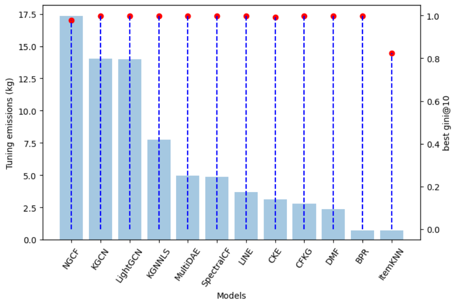
(A) Recall



(B) NDCG



(C) Avg. Pop.



(D) Gini Index

FIGURE 5.11: **Amazon-Books Dataset.** Cost of the hyperparameter tuning in terms of CO_2 emissions (kg) and performances obtained in terms of recall (dots). Vertical bars refer to the emissions, while red dots refer to the considered metric.

To answer RQ5.2.2, this study indicates that the common practice of tuning hyperparameters to optimize the performance of the algorithms should be considered with care. Indeed, in many settings, the increase in performance is not justified by the substantial emissions required to identify the best-performing configuration of the algorithm. Conversely, less sophisticated algorithms typically offer a better compromise, since they enable a better trade-off.

5.2.5 Discussion and Limitations

This study identifies several limitations and sketches the main findings of the benchmark.

- **Geographical Impact on Computation:** The analyses underscore the significance of computation location. While global average carbon intensity values are used to ensure fairness, the choice of computation site can substantially affect emissions. Selecting regions with lower carbon intensity can enable the use of more advanced algorithms with comparable emissions.

- **Prioritize Carbon Footprint Reduction:** Proactive measures are necessary to mitigate emissions after carbon footprint estimation. Strategies such as scheduling heavy workloads during periods of cleaner energy, training models in regions with renewable energy sources, and optimizing hyperparameters are essential for reducing emissions without compromising performance.
- **Scale Evaluation Imperatives:** Although the recorded maximum carbon equivalent is relatively low, larger-scale evaluations are necessary for comprehensive insights. At scale, differences in carbon footprint among algorithms may widen significantly, potentially exerting a substantial environmental impact when adopted across industries.
- **Demand for Multi-objective Approaches:** Achieving balance across sustainability pillars demands nuanced, multi-objective approaches. Balancing business objectives with sustainability goals requires careful algorithm and data selection to optimize fairness, emissions, and accuracy, suggesting avenues for developing novel multi-objective recommendation algorithms.

5.2.6 Take-Home Messages

The results of an extensive benchmark investigating the trade-off between performance and carbon emissions of state-of-the-art recommender systems are discussed in [179, 166]. Overall, basic recommendation algorithms, such as ItemKNN, generally achieve the best compromise between accuracy and carbon emissions, while more sophisticated algorithms do not often justify the increase in emissions with an equivalent improvement in performance. Moreover, it was shown that hyperparameter tuning, which is typically performed to find the combination of parameters that leads to the best predictive accuracy, can worsen the problem since more complex algorithms generally require many runs to find the optimal combination, and this further increases emissions.

As future work, the benchmark will be extended to consider different algorithms and different machines. Moreover, the direction of multi-objective recommendation will be pursued by designing new approaches that take into account sustainability and accuracy simultaneously.

Main Findings

- **RQ5.2.1**: The selection of a recommendation algorithm that balances performance and carbon efficiency is non-trivial. Overall, simpler approaches tend to be more energy-efficient while maintaining competitive performance.
- **RQ5.2.2**: Extensive hyperparameter tuning substantially increases emissions, often without proportionate performance improvements. Simpler models require fewer tuning iterations, thus providing a better performance–sustainability balance.

These results have been carried out during the PhD and have been published at ACM RecSys 2023 in the Short Papers Track (Giuseppe Spillo et al. "Towards sustainability-aware recommender systems: analyzing the trade-off between algorithms performance and carbon footprint". In: *Proceedings of the 17th ACM Conference on Recommender Systems*. 2023, pp. 856–862), presented during the poster session. An extended version of the paper (Giuseppe Spillo et al. "Balancing carbon footprint and algorithm performance in recommender systems: A comprehensive benchmark". In: *Sustainable Computing: Informatics and Systems* (2025), p. 101286) has been published in the Journal of Sustainable Computing: Informatics and Systems.

5.3 Data Reduction Approaches to Improve the Emissions-Performance Trade-off

5.3.1 Introduction

This section addresses the growing environmental concerns associated with increasing energy consumption and CO_2 emissions caused by modern AI systems, particularly in the context of RSs. As discussed in the previous section, AI has achieved significant success across fields, but this progress has led to rising carbon footprints due to prolonged training times and the computational intensity of large-scale models [116, 90, 155].

To contribute toward climate neutrality, the featured publications [174, 168] explore the use of data reduction strategies (i.e., training AI models on smaller datasets) as a model-agnostic and easily deployable approach to lower energy use and emissions [144]. While previous works have suggested that complex RS algorithms do not necessarily outperform simpler ones [179, 166], the environmental benefits of data reduction in this context remain underexplored. RSs are widely used in real-world applications and require frequent retraining, which intensifies their environmental impact [129, 103]. This study investigates whether reducing training data can decrease the carbon footprint of RSs without significantly compromising recommendation quality.

Following the principles of Green AI [155, 140], an extensive analysis across various datasets and RS algorithms is conducted to examine the trade-off between performance, energy consumption, and training data volume. The results indicate that data reduction can effectively enhance sustainability with only minor losses in predictive accuracy, while also reducing popularity bias. This investigation aims to promote environmentally sustainable RS practices and offer preliminary guidelines for selecting models that balance performance with lower environmental costs.

5.3.2 Methodology

In this section, all details concerning the methodology to assess the impact of data reduction strategies on the performance of RSs are provided. As shown in Figure 5.12, the pipeline can be roughly split into three steps. First, a recommendation model is selected. Next, the algorithm is trained with a subset of the data (*i.e.,* users' ratings and descriptive features of the items). At the end of the training, the amount of carbon emissions is quantified and the effectiveness of the recommendations generated by the algorithm is evaluated. In the following, the details of the different steps of the pipeline are provided. First, some basics of recommendation algorithms are presented. Next, the protocol to measure the carbon emissions of the algorithms is described, and finally the different data reduction strategies compared in the experiments are introduced.

For this work, the analysis has been conducted on different recommendation models belonging to both CF and KARS paradigm. In particular, the following models have been considered: BPR [146], DMF [235], NGCF [221], DGCF [218], MultiDAE [99], LightGCN [70], LINE [189], CKE [246], CFKG [251], KGCN [211], KGNN-LS [210]. Their description is discussed in Section 2.3, and is not reported here to avoid repetitions.

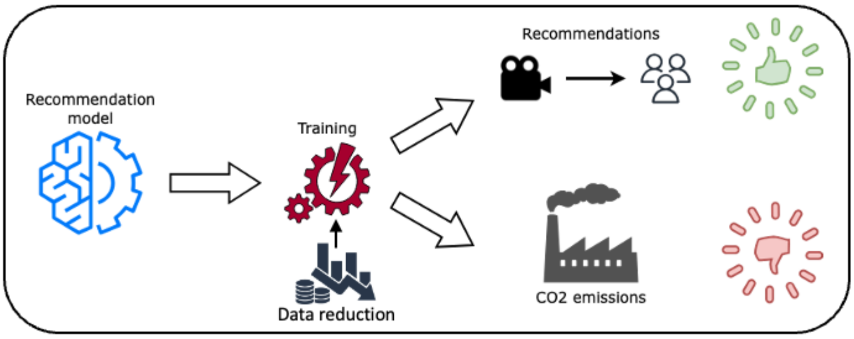


FIGURE 5.12: Overview of the main steps in our methodology pipeline.

Measuring Carbon Footprint The carbon footprint was quantified in terms of $CO_2 - eq$, as it is widely adopted by governmental bodies, international organizations, and the scientific research community. More details about how this metric was exploited can be found in Section 5.2.2, as the same methodology was adopted.

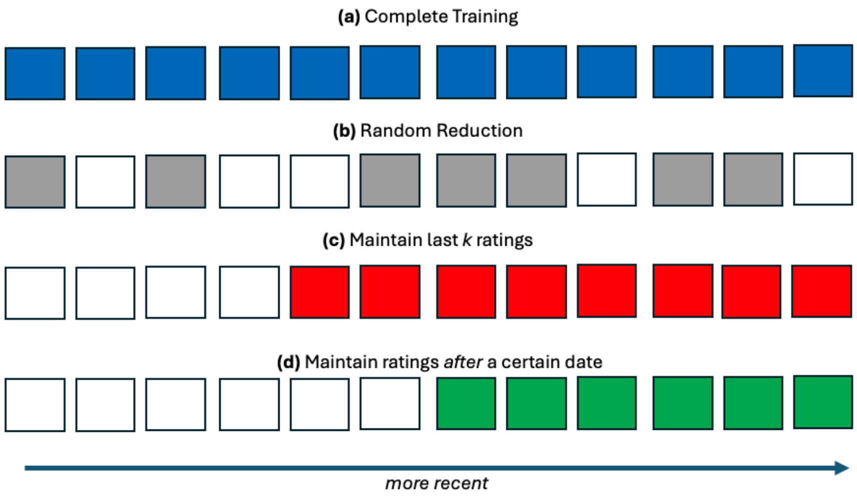

(a) Complete Training
(b) Random Reduction
(c) Maintain last *k* ratings
(d) Maintain ratings *after* a certain date
more recent

FIGURE 5.13: Overview of the Data Reduction Strategies.

Data Reduction Strategies To investigate the impact of data reduction on recommender system performance and carbon footprint, four distinct data reduction strategies were implemented, each designed to explore different aspects of data sparsity and temporal relevance ([91]). These strategies were chosen to provide a comprehensive evaluation of the effects of reduced training data on both model accuracy and environmental impact. A graphical overview of the proposed strategies can be found in Figure 5.13.

Random Sampling of User Ratings (User-Based Reduction) This strategy involved randomly removing 20% of each user's ratings from the training dataset. The rationale behind this approach is to simulate a scenario in which users have provided fewer interactions, thereby reducing the computational costs of training. By randomly selecting ratings for removal, the overall distribution of user preferences is intended to be maintained while reducing the volume of training data.

Random Sampling of Item Ratings (Item-Based Reduction) Complementary to user-based reduction, this strategy randomly removes 20% of the ratings associated with each item. This approach simulates a scenario where fewer users have interacted with specific items, increasing item sparsity and reducing computational costs. By randomly selecting ratings for removal, the overall distribution of item popularity is intended to be maintained while reducing the training dataset size. The intuition is that if a model can perform well with reduced item-specific data, it demonstrates resilience to item sparsity.

Maintaining Top-K Recent Ratings (User-Specific Temporal Reduction) This strategy introduced a temporal dimension to data reduction by maintaining only the k most recent ratings for each user. This approach reflects the assumption that more recent interactions indicate current user preferences. By focusing on the most recent interactions, the aim is to evaluate the model's ability to maintain good predictive accuracy while reducing environmental impact. The value of k was determined empirically, balancing the need for sufficient training data with the goal of significant data reduction. The rationale is to simulate a scenario in which the model is trained on a "sliding window" of recent user activities.

The user-specific temporal reduction strategy assumes that recent interactions are more indicative of current user preferences than older ones. This assumption is grounded in behavioral economics and user modeling theory, where preference drift, first introduced in [65], is a well-documented phenomenon.

Maintaining Ratings After a Cut-Off Date (Global Temporal Reduction) This strategy implemented a global temporal cut-off, maintaining all ratings recorded after a specific date. This approach simulates a scenario where only recent interactions are considered relevant for training. By establishing a fixed cut-off date, the aim is to assess the model's ability to capture the most recent trends and patterns in user-item interactions. This method reflects the assumption that older interactions may be less representative of current user preferences or item popularity. The cut-off date was determined based on the dataset's temporal distribution, ensuring a significant reduction in training data while maintaining a reasonable representation of recent interactions. This approach examines the impact of a strict 'freshness' requirement on recommender performance and the associated carbon footprint.

5.3.3 Experimental Evaluation

Based on the data reduction techniques introduced in the previous section, this study aimed to answer the following **research questions (RQs)**:

- **RQ5.3.1 - Impact on Emissions:** To what extent does data reduction influence the carbon footprint of diverse recommendation algorithms?
- **RQ5.3.2 - Impact on Recommendations:** What are the effects of data reduction on the performance metrics of different recommendation algorithms?
- **RQ5.3.3 - Trade-off analysis:** Is there any trade-off between carbon footprint and performance that emerges when data reduction is applied?

Experimental Design To address the research questions, the experimental protocol illustrated in Figure 5.12 was developed. Data reduction strategies were applied to each recommendation model to create subsets of the training data. Subsequently, the models were trained on these reduced datasets, the generated recommendations were collected, and the corresponding CO_2 emissions were recorded. The following sections provide a detailed description of each aspect of this protocol.

Datasets. Experiments were conducted in a *book and movie recommendation* scenario. In particular, Amazon-Books and ML1M have been considered. More details about this datasets can be found in Section 2.3, where the problem has been formalized.

Data Reduction. In the experiments, the effectiveness of the strategies introduced in Section 3 was compared. In particular, for the USER-BASED and ITEM-BASED reductions, $k\%$ of the training data available for each user and each item, respectively, were removed. In particular, $k = 20$. It is worth pointing out that data were removed in a stratified manner, *i.e.,* the ratio between positive and negative ratings per user and per item was maintained after the data reduction. Next, three different ranges were defined for the USER-SPECIFIC TEMPORAL Reduction. On ML1M, three configurations were evaluated by maintaining the last n ratings, with $n = 200, 250, 300$. Similarly, for Amazon, the last 65, 80, and 100 ratings were maintained. These values were empirically determined by analyzing the characteristics of the datasets and were chosen to remove approximately 20%, 25%, and 30% of the data, respectively. Finally, for the GLOBAL TEMPORAL reduction, three different ranges were set to evaluate data reduction at various strengths. In this case, July, August, and September 2000 were used as cut-off dates for ML1M and April 2012, November 2009, and September 2007 for Amazon. For each configuration, only the ratings collected after the date indicated in the configuration name were retained. Accordingly, April 2012 on Amazon is the configuration with the most substantial data reduction, while September 2007 retains most of the original ratings.

Recommendation Models. To ensure fair comparisons and facilitate replication, implementations of the selected recommendation models provided by the RecBole library were sued.

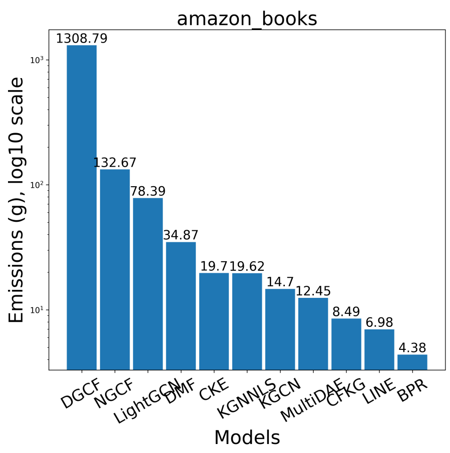
Evaluation Metrics. Evaluation metrics are calculated using RecBole, as well. The evaluation consider all the metrics implemented in RecBole, but for the sake of simplicity, results are presented in terms of Recall and average popularity. This allows assessment of the accuracy and popularity bias of the recommendations.

Measuring Carbon Footprint. The emissions and carbon footprint of the recommendation models were quantified using the CodeCarbon Python library. For comparability, the world average carbon intensity value was employed[10]. The emissions data are presented in grams (g).

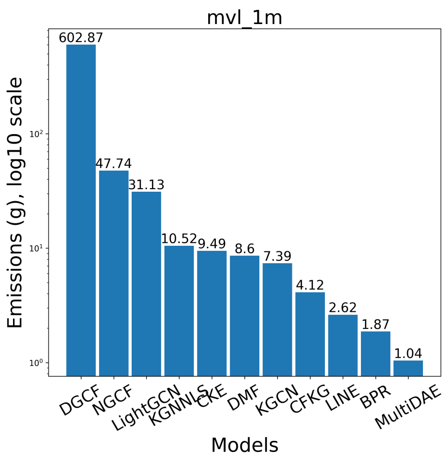
5.3.4 Discussion of the Results

The following presents the results of the experiments and discusses the findings of the research.

[10] `https://www.iea.org/reports/global-energy-co2-status-report-2019/emissions`



(A) AmazonBooks emissions



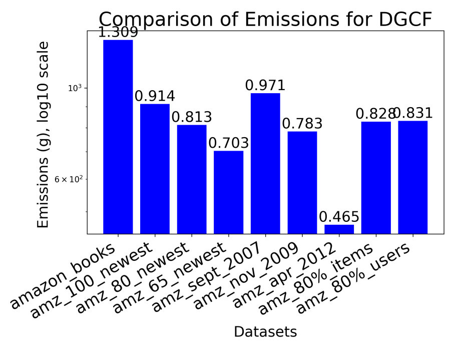
(B) ML1M emissions

FIGURE 5.14: Comparisons of the emissions on the *complete* AmazonBooks and ML1M datasets.

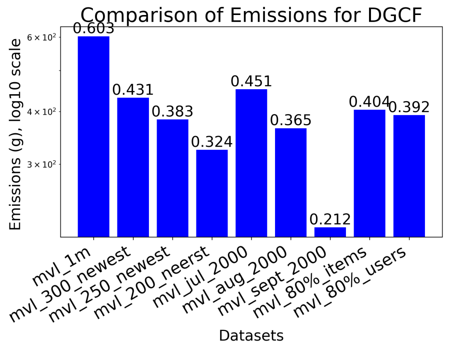
RQ5.3.1: Impact on Emissions To address **RQ5.3.1**, plots of the emissions of the different algorithms on the ML1M and AmazonBooks datasets are presented. As shown in Figure 5.14, some algorithms (such as DGCF, NGCF, LightGCN, and DMF) emit much more CO_2 than others. For example, the plots show that DGCF emits $10x$ more than NGCF on both datasets and more than $300x$ with respect to BPR. Due to space constraints and to focus on more relevant results, the analysis concentrates mainly on the algorithms with a higher carbon footprint. However, the complete results are available in the repository[11]. Regarding the behavior of individual algorithms, it is worth noting that some algorithms that use side information from knowledge graphs (i.e., KGNN-LS) emit less CO_2-eq than other models. While this result may be partially counterintuitive, it mainly depends on the intrinsic complexity of certain recommendation models. For example, performing complex operations on the graph, as occurs in DGCF and NGCF, impacts the overall emissions more than the use or absence of side information.

Next, the decrease in CO_2 emissions is quantified when the different data reduction techniques are applied. In Figure 5.15, 5.16, and 5.17, the results obtained for DGCF, NGCF, and LightGCN are presented. As shown in the figures, all data reduction strategies reduce the overall emissions of the models. Regarding DGCF, the decrease ranges between 23% and 64% when the *global* temporal reduction is

[11]https://github.com/swapUniba/datared-green-recsys



(A) AmazonBooks



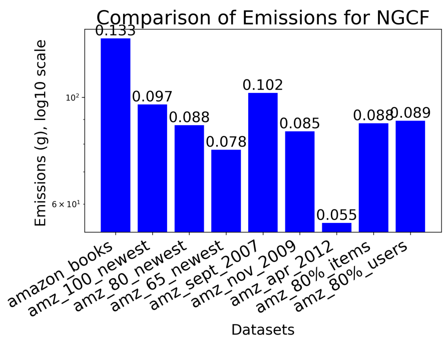
(B) ML1M

FIGURE 5.15: Impact of data reduction on CO_2 emissions of DGCF on AmazonBooks (left) and ML1M (right).

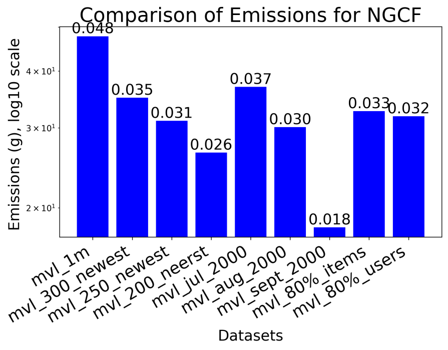
applied. Similarly, for NGCF, a decrease between 25% and 60% was observed. Finally, the same outcomes were obtained for LightGCN: using the global temporal data reduction as a strategy, the decrease ranges between 23% and 58%. All other algorithms evaluated followed a similar trend.

While the overall carbon footprint varies across algorithms, the decrease obtained by applying a specific data reduction technique is somewhat predictable. For example, when the *user-specific* temporal reduction with $k = 100$ is applied, the decrease in carbon emissions always ranges between 27%–30%. This is the first notable outcome of this experiment and may be useful for several practical applications. For example, a company may need to decrease emissions by a certain amount. Knowledge of the most suitable technique to reach this target can be important for applying these data reduction strategies in real-world scenarios.

To summarize, this study shows that all the techniques introduced reduce the emissions required to train a specific algorithm. Moreover, the experiment indicates that the decrease in emissions can be somewhat predictable, which may be useful for applying these techniques in real settings. Clearly, stronger data reduction leads to a greater decrease in the algorithm's carbon footprint. An interesting aspect emerging from the experiments is the predictability of the amount of CO_2 saved by applying a specific technique, which can be useful for exploiting these strategies in real applications. The following analysis examines whether and how the reduction of training data impacts the performance of the different algorithms.

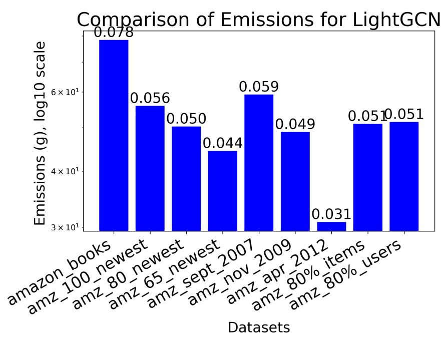


(A) AmazonBooks

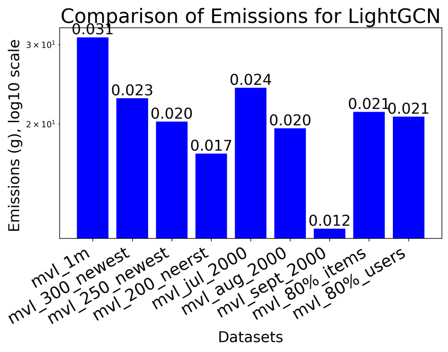


(B) ML1M

FIGURE 5.16: Impact of data reduction on CO_2 emissions of NGCF on AmazonBooks (left) and ML1M (right).



(A) AmazonBooks



(B) ML1M

FIGURE 5.17: Impact of data reduction on CO_2 emissions of LightGCN on AmazonBooks (left) and ML1M (right).

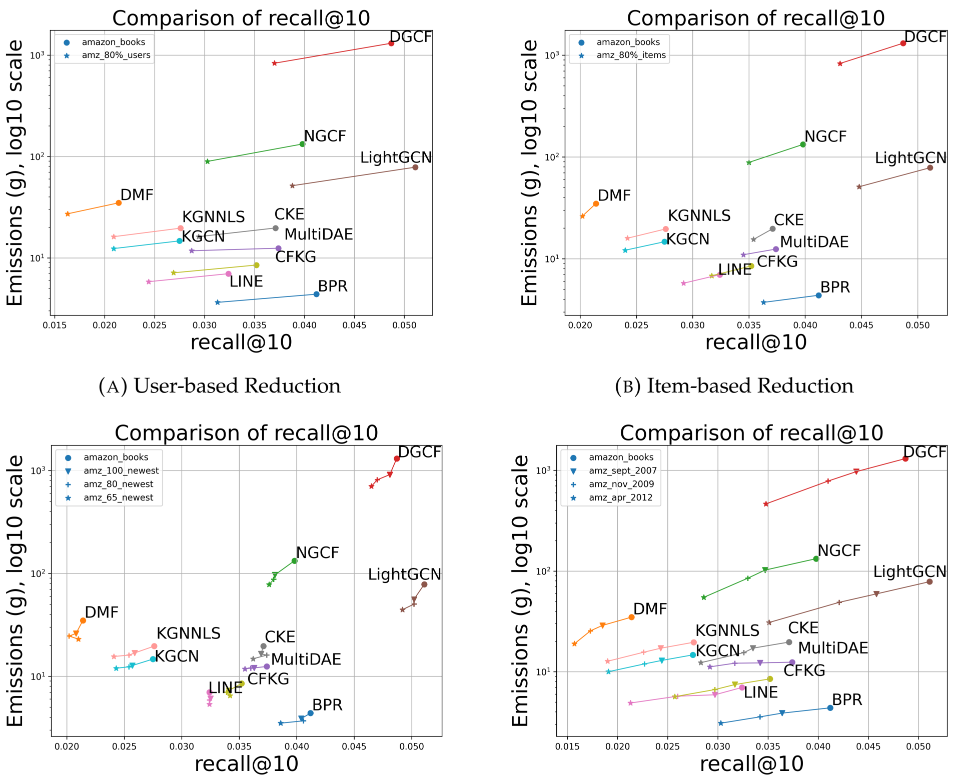
RQ5.3.2: Impact on Recommendations To address **RQ5.3.2**, the effects of the data reduction strategies on the performance of the recommendation algorithms are analyzed. Due to space limitations, results are presented for a single accuracy metric, *i.e.*, $Recall@10$, and a single non-accuracy metric, *i.e.*, $AveragePopularity$. The complete set of results is available in the repository. The results are presented from Figure 5.18 to Figure 5.21.

First, analysis of the AmazonBooks data shows that adopting *user-based* and *item-based* data reduction leads to decreased accuracy (Figures 5.18a and 5.18b). While this is an expected and unsurprising outcome, it is worth noting that the techniques affect performance differently. Considering the plots' lines, the ideal behavior would have been represented by an (almost) vertical line, *i.e.*, decreased emissions and identical or similar accuracy. Conversely, a horizontal line indicates that an equivalent decrease in emissions does not fully compensate for the reduction in accuracy. Accordingly, Figure 5.18b shows that *item-based* data reduction yields better performance than *user-based* reduction. This is not surprising since the removal of a substantial amount of user ratings increases data sparsity, making the recommendation task harder. The analysis on ML1M (see Figures 5.19a and 5.19b) yields similar results, since *item-based* reduction again appears more robust than *user-based* reduction. Moreover, the results indicate that, in this case, user-based reduction on ML1M has a negligible impact since the decrease in recall is not offset by a reduction in carbon emissions. This is probably due to the dataset's smaller size, making it harder to assess the positive impact of data reduction. With respect to the individual techniques, on Amazon Books, LightGCN and DGCF emerged as the most accurate when data reduction was applied, while MultiDAE and, again, LightGCN, obtained the best results on ML1M.

Next, the temporal reduction strategies were analyzed. The results (see Figures 5.18c and 5.18d for AmazonBooks, and 5.19c and 5.19d for ML1M) show a very interesting behavior for the *user-specific* temporal reduction. For nearly all algorithms, the lines follow the desired trend, approaching a vertical line. Maintaining the most recent ratings reduces carbon emissions due to shorter training times while maintaining good predictive accuracy. Conversely, the global temporal reduction strategy shows a trend similar to the user-based and item-based reductions, with a substantial decrease in performance that is not balanced by an equivalent reduction in emissions. Furthermore, a considerable decrease in performance is observed when global temporal reduction is applied. For example, LightGCN shows a decrease in recall from 0.053 to 0.035 on Amazon and from 0.015 to 0.009 on ML1M. Analysis of the individual techniques shows again that the DGCF and LightGCN emerge as the models with the best potential, exhibiting an almost horizontal line when data reduction is applied.

Overall, RQ5.3.2 can be addressed by observing that the USER-SPECIFIC temporal data reduction emerges as the most promising strategy, capable of significantly reducing the carbon emissions of the algorithms while maintaining satisfactory accuracy. The results are particularly relevant for the AmazonBooks dataset, where the techniques emit a significantly higher amount of CO_2-eq.

Next, the impact of data reduction on AveragePopularity was analyzed. In this



(A) User-based Reduction

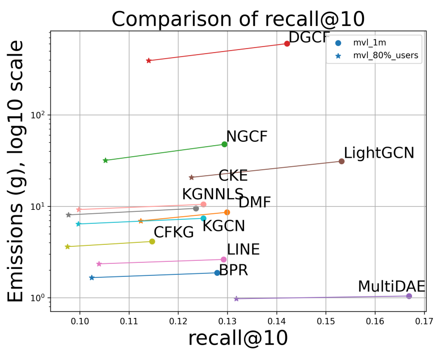
(B) Item-based Reduction

(C) User-Specific Temporal Reduction

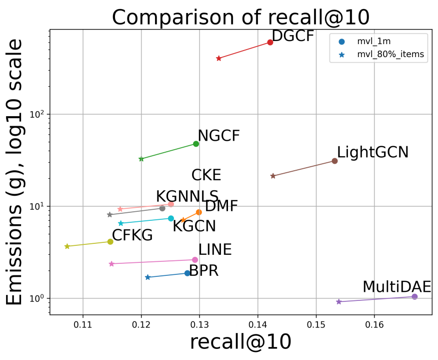
(D) Global Temporal Reduction

FIGURE 5.18: Effect of the Data Reduction Approaches on the Recall on AmazonBooks.

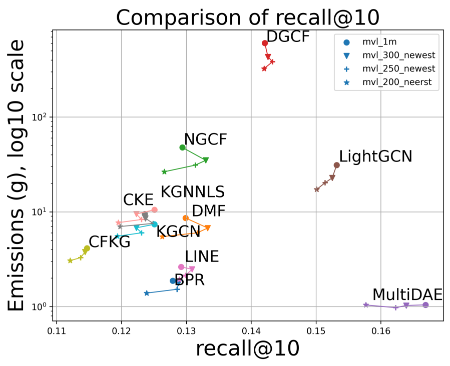
case, *lower is better*, so the ideal behavior is represented by a line that goes toward the lower-left corner. The different data reduction strategies showed heterogeneous behaviors. As regards the *item-based* data reduction (see Figure 5.20b), almost all the methods decrease the emissions but also increase the popularity bias, *i.e.*, higher average popularity. This is not surprising since the removal of random items is likely to drive the algorithms toward more popular recommendations. The same behavior is also confirmed on ML1M data (see Figure 5.21b). Conversely, the use of the *user-based* data reduction provides interesting insights. First, the lines showing the performance gap are longer and go in the desired direction. This indicates that the use of this specific data reduction technique can reduce popularity bias and emissions simultaneously. In this case, NGCF and LightGCN emerged as the techniques characterized by the best impact. Results on ML1M (see Figure 5.21a) confirm the same trend, with an interesting decrease in popularity and an equivalent decrease in emissions. In both cases, while the



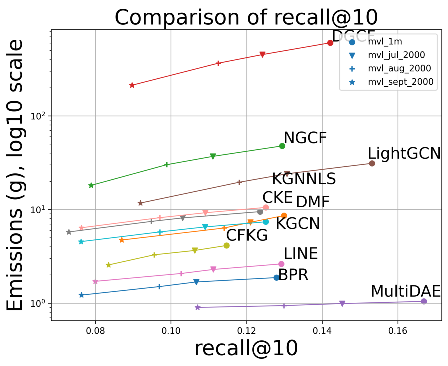
(A) User-based Reduction



(B) Item-based Reduction



(C) User-Specific Temporal Reduction

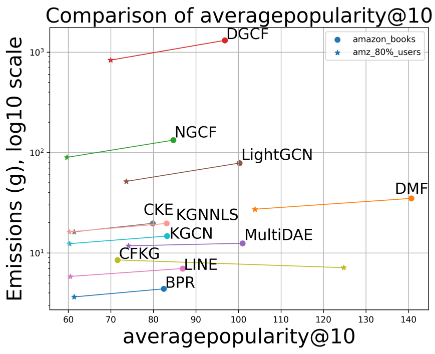


(D) Global Temporal Reduction

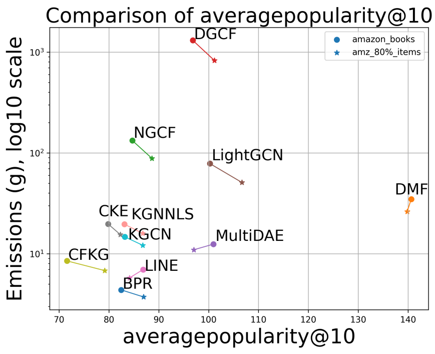
FIGURE 5.19: Effect of the Data Reduction Approaches on the Recall on ML1M.

lines are almost horizontal (i.e., minimum decrease in terms of emissions), the unexpected reduction of the popularity bias still makes these techniques interesting to analyze.

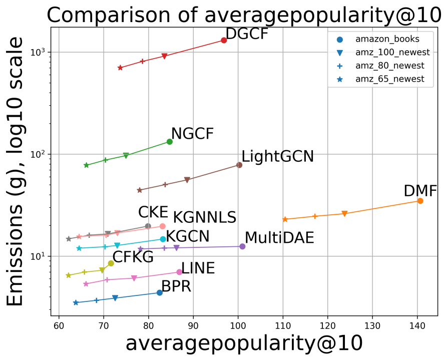
Finally, regarding the impact on AveragePopularity of temporal data reduction (see Figures 5.20c and 5.20d for AmazonBooks, and 5.21c and 5.21d for ML1M), the results show that both these techniques successfully optimize both objectives, since the reduction in terms of carbon footprint is also followed by a reduction of the popularity bias, *i.e.*, the lowest averagePopularity. This is generally confirmed by all the techniques proposed on the datasets and all the algorithms, with just a few exceptions. As regards the single techniques, LightGCN and, in particular, NGCF, emerge as the techniques with the lower popularity bias while significantly reducing their carbon footprint when the global temporal reduction is applied. The same outcome is also confirmed on ML1M data. Regarding this dataset, it



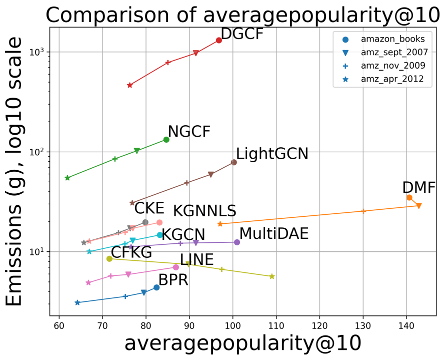
(A) User-based Reduction



(B) Item-based Reduction



(C) User-Specific Temporal Reduction



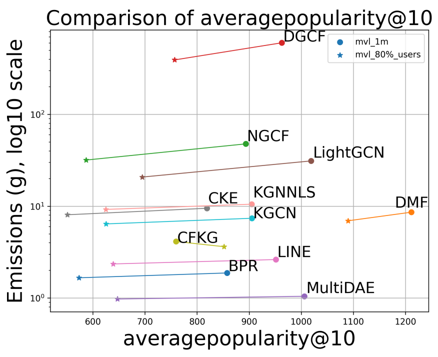
(D) Global Temporal Reduction

FIGURE 5.20: Effect of the Data Reduction Approaches on the Average Popularity on AmazonBooks.

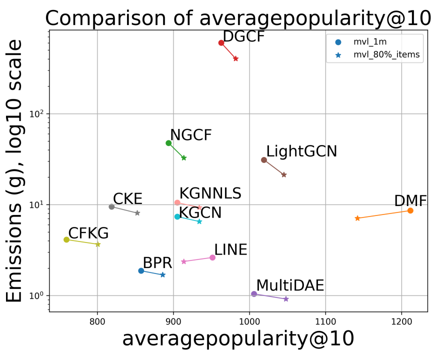
is worth noting that all techniques strongly reduce their *averagePopularity* scores. This is confirmed by the length of the lines, showing that the popularity bias can be reduced by two-thirds (from 900/100 to less than 400 for many techniques).

To sum up, the analysis emphasizes the unexpected effect of data reduction on popularity bias. Indeed, the results show that the use of data reduction reduces both the carbon footprint and the popularity bias of recommendation algorithms. As regards the techniques, LightGCN and NGCF emerged as those with the best trade-off, while temporal reduction emerged as the most effective strategy.

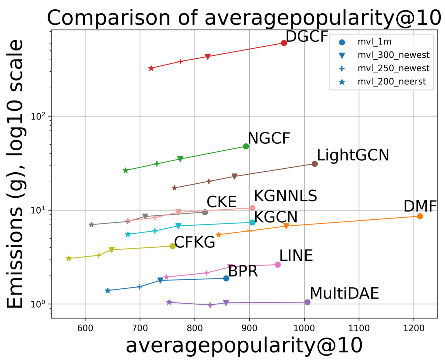
RQ5.3.3: Trade-off Analysis Finally, to answer **RQ5.3.3**, the percentage decrease in both carbon footprint and recall was calculated for the different data reduction strategies. Due to space limits, the analysis is limited to the *temporal data reduction*



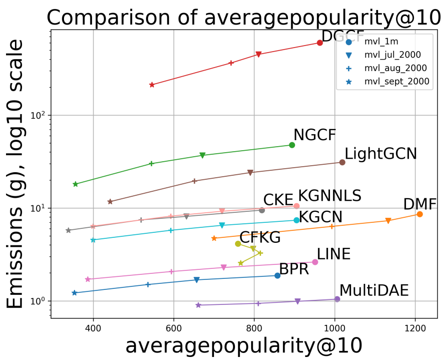
(A) User-based Reduction



(B) Item-based Reduction



(C) User-Specific Temporal Reduction



(D) Global Temporal Reduction

FIGURE 5.21: Effect of the Data Reduction Approaches on the Average Popularity on ML1M.

strategies, since these emerged as the most promising in the previous discussion. The results are presented in Tables 5.4 and 5.5. In each table, the first two columns refer to the *user-based* temporal data reduction, while the last two refer to the *global temporal* data reduction.

As shown in the tables, the user-based temporal data reduction generally provides the best trade-off between the metrics. Indeed, the reduction in recall is much smaller than that obtained when the global temporal data reduction is applied. This generally holds for all the techniques and both datasets. Of course, stronger data reduction (i.e., keeping the 100 or 300 newest ratings vs. keeping just the last 65 or 200) directly impacts the trade-off since it leads to a higher decrease in emissions but also in performance. As previously stated, the choice of the most suitable configuration is strictly scenario-dependent since different applications may have different tolerances for the overall amount of emissions as well as for

Algorithm	*User-Specific Temporal Reduction*				*Global Temporal Reduction*			
	100 newest per user		*65 newest per user*		*September 2007*		*April 2012*	
	% Emissions	% Recall	% Emissions	% Recall	% Emissions	% Recall	% Emissions	% recall
BPR	-11.59%	-1.94%	-20.16%	-6.31%	-11.42%	-11.65%	-29.57%	-26.46%
DMF	-25.30%	-2.80%	-34.12%	-1.87%	-17.29%	-13.55%	-45.73%	-26.64%
NGCF	-27.15%	-4.27%	-41.35%	-5.53%	-23.00%	-12.81%	-58.68%	-28.14%
DGCF	-30.16%	-1.23%	-46.29%	-4.52%	-25.81%	-10.06%	-64.50%	-28.54%
MultiDAE	-3.10%	-2.94%	-5.52%	-5.08%	-1.67%	-8.56%	-10.20%	-21.93%
LightGCN	-28.71%	-1.76%	-43.38%	-3.72%	-24.38%	-10.37%	-60.69%	-31.31%
LINE	-12.87%	0.31%	-23.41%	0.00%	-15.39%	-8.33%	-29.86%	-34.26%
CKE	-16.27%	-0.54%	-24.98%	-2.43%	-12.95%	-9.70%	-37.56%	-23.72%
CFKG	-14.50%	-3.13%	-23.62%	-2.84%	-12.18%	-9.94%	-33.45%	-26.99%
KGCN	-13.48%	-6.55%	-18.58%	-11.64%	-11.92%	-11.27%	-31.89%	-30.55%
KGNN-LS	-14.15%	-6.16%	-20.70%	-12.68%	-12.77%	-11.96%	-35.27%	-31.16%

TABLE 5.4: Percentage variation of Emissions and Recall@10 for the AmazonBooks datasets.

Algorithm	*User-Specific Temporal Reduction*				*Global Temporal Reduction*			
	300 newest per user		*200 newest per user*		*July 2000*		*September 2000*	
	% Emissions	% Recall	% Emissions	% Recall	% Emissions	% Recall	% Emissions	% recall
BPR	-5.04%	0.70%	-25.83%	-3.13%	-10.16%	-16.58%	-34.99%	-40.34%
DMF	-21.62%	2.62%	-36.44%	-2.77%	-15.19%	-6.85%	-45.12%	-33.03%
NGCF	-26.81%	2.78%	-44.58%	-2.16%	-22.66%	-14.14%	-62.08%	-39.03%
DGCF	-28.45%	0.35%	-46.18%	-0.07%	-25.11%	-12.60%	-64.76%	-36.88%
MultiDAE	-1.58%	-1.80%	-0.38%	-5.51%	-5.25%	-12.94%	-13.83%	-35.89%
LightGCN	-26.73%	-0.46%	-44.76%	-2.02%	-22.71%	-14.69%	-62.38%	-40.01%
LINE	-5.63%	1.32%	-26.38%	-0.23%	-12.80%	-13.93%	-34.90%	-38.08%
CKE	-10.03%	0.08%	-26.30%	-3.07%	-14.38%	-16.59%	-39.32%	-40.94%
CFKG	-8.63%	-0.26%	-26.05%	-2.27%	-11.50%	-7.32%	-37.99%	-27.20%
KGCN	-8.27%	-2.24%	-25.28%	-4.56%	-11.89%	-12.87%	-38.58%	-39.09%
KGNN-LS	-10.03%	-2.24%	-26.99%	-4.48%	-12.26%	-12.79%	-39.34%	-39.01%

TABLE 5.5: Percentage variation of Emissions and Recall@10 for the ML1M datasets.

the decrease in performance that may be acceptable. This aspect will be discussed next.

Regarding the individual techniques, it is worth noting that techniques such as LINE and CKE, when user-specific temporal reduction is applied, have a negligible decrease in performance (i.e., lower than 1%). The decrease in terms of emissions is lower with respect to other carbon-intensive algorithms such as LightGCN or NGCF. Considering performance in terms of absolute value (see the discussion of RQ5.3.2), LightGCN and NGCF emerge again as those with the best trade-off, since data reduction can provide an important reduction in emissions (between approximately 30% and 40%, depending on the configurations) with only a small reduction in performance. Overall, user-specific temporal reduction appears more promising than global temporal reduction, since the latter results in a substantial decrease in recall (up to 40%, when the stronger configuration is applied).

To conclude, RQ5.3.3 can be addressed by stating that all presented techniques exhibit different trade-offs. Generally, stronger data reduction leads to a greater decrease in performance, as expected. While some techniques tend to provide a better trade-off, a decrease in performance should be expected. Overall, user-specific data reduction emerges as the most promising strategy, particularly for carbon-intensive algorithms. The choice of the most suitable data reduction technique is strictly scenario-dependent and domain-dependent. If low emissions are required, this benchmark may identify the most suitable algorithm to provide the most effective trade-off.

5.3.5 Take-Home Messages

These published works [178, 168] present a comparative analysis of the trade-off between the performance of state-of-the-art recommendation algorithms and their environmental sustainability with different data reduction strategies. The outcomes of the experiments show that selecting the optimal recommendation algorithm requires a thorough analysis balancing the main pillars of sustainability, not only the environmental one.

This framing of data reduction as a tool for environmental and social sustainability in RS design constitutes a conceptual contribution beyond algorithmic novelty. Moreover, the reproducible experimental framework (based on RecBole, CodeCarbon, stratified sampling, and diverse datasets) provides generalizable findings. Indeed, the experiments show predictable reduction patterns (e.g., 30% emissions reduction with <5% accuracy loss) that provide practical decision-making support for real-world deployment. In this light, this work is foundational in initiating a new line of inquiry for sustainability-aware recommender systems. Indeed, the experiments open new questions regarding optimal training windows, data utility vs. carbon cost, and user dynamics modeling, laying the foundation for future theoretical and algorithmic exploration.

Future work includes replicating the experiments on different machines and employing other frameworks to assess whether different implementations lead to different carbon footprints.

Main Findings

- **RQ5.3.1**: All proposed data reduction techniques decrease training-related carbon emissions, with predictable patterns on such emission decrease.
- **RQ5.3.2**: Data reduction mitigates popularity bias while simultaneously lowering emissions. In particular, user-specific temporal emerged as the most effective strategy.
- **RQ5.3.3**: Each data reduction technique exhibits a distinct performance–emission trade-off. User-specific data reduction offers the most promising balance, especially for carbon-intensive algorithms. LightGCN achieves the best performance–sustainability trade-off.

These results have been carried out during the PhD and published at ACM RecSys 2023 in the Short Papers Track (Giuseppe Spillo et al. "Towards Green Recommender Systems: Investi- gating the Impact of Data Reduction on Carbon Footprint and Model Performances". In: *Proceedings of the 18th ACM Conference on Recommender Systems*. 2024), presented during the poster session, and subsequently extended and published in the Journal of Intelligent Information Systems (Giuseppe Spillo et al. "Comparing data reduction strategies for energy-efficient green recommender systems". In: *Journal of Intelligent Information Systems* (2025), pp. 1–27).

5.4 Green Early Stopping Criterion for Sustainable Training

5.4.1 Introduction

Relevant works in the RS literature have introduced tools and metrics to quantify the carbon footprint of RSs [224, 12], and scholars have increasingly called for more sustainable practices in their design and training [15, 82]. In line with this perspective, the previous sections empirically showed that simpler models such as CF can achieve competitive performance with considerably lower emissions compared to more complex systems, including KARS [204]. These findings reinforce the notion that pursuing increasingly sophisticated models does not always yield a favorable trade-off between accuracy and environmental impact.

However, the challenge of effectively reducing the carbon emissions of RSs without significantly compromising their predictive quality remains largely unaddressed. This study aims to contribute to this growing research area by proposing a novel Green Early Stop (GES) strategy that incorporates environmental awareness directly into the training process of recommender models.

For this work, the Accuracy Emission Ratio (AER) is introduced, a metric that quantifies the relationship between performance gains (based on validation scores) and the carbon emissions produced during training. By monitoring AER at each training epoch, the GES criterion can identify points at which continued training yields diminishing returns relative to environmental cost. If this ratio remains unfavorable across several epochs, the training is automatically stopped. Experimental evaluations demonstrate that this method can substantially reduce emissions with only minor sacrifices in predictive accuracy, thereby validating the hypothesis and offering a promising path toward greener recommender systems.

This study aims to establish a foundation for more environmentally responsible practices in recommender systems development, and future directions are laid out to make the approach increasingly robust and fully automated.

5.4.2 Methodology

Green Early Stop Early stopping strategies commonly used to train RS rely on the analysis of validation scores epoch by epoch. By contrast, the novel GES criterion also considers the amount of $CO_2 - eq$ emitted to decide whether training should be stopped ahead of time.

To implement this strategy, two novel constructs are relied upon: *(a)* the Accuracy Emission Ratio, calculated as the ratio between the gain in performance and the increase in carbon emissions in two consecutive epochs of training; *(b)* the *tolerance* of the GES criterion when monitoring the AER epoch by epoch.

In particular, AER is inspired by the finite difference method [96], which is designed to approximate the derivatives (the *rates of change*) of a function by using the difference between function values at discrete points. In this context, the function is the model's accuracy as given by the validation score. At a given epoch, the slope of this curve represents how favorable the gain in accuracy is with respect to the increase in emissions. Based on the slope, a decision is made whether to stop the training. Formally, AER can be defined as follows:

$$AER(r) = \frac{v\_score_{i+1} - v\_score_i}{e_{i+1} - e_i} \tag{5.1}$$

where r is a recommendation algorithm, v_score_i and v_score_{i+1} are the validation metric values observed at training epochs i and $i+1$, respectively, and e_i and e_{i+1} are the CO_2 emissions produced after i and $i+1$ training epochs. Based on these constructs, the GES criterion can be sketched as follows:

1. For each epoch, the AER score is calculated. Any accuracy metric, i.e., NDCG, can be used as a validation metric.
2. Based on the size of the datasets and on the characteristics of the recommendation algorithm, i.e., the average emission, a threshold score for AER is set. Let n be the threshold.
3. If the AER score is lower than n, it means that the increase is unfavorable. If this holds for a consistent number of epochs (*i.e.,* the tolerance), training is stopped.

Generally speaking, the training stops when, for too many epochs, the improvement in predictive accuracy is not justified by the increase in carbon emissions. Formally, let k be the tolerance and r be the recommendation algorithm; training is stopped if $AER(r) < n$ for more than k consecutive epochs. Higher tolerance results in a larger number of epochs not surpassing the threshold that is needed to stop training. Conversely, low tolerance values may stop the training even after a few epochs, where the ratio between accuracy and emissions is unfavorable.

Figure 5.22 sketches an example that illustrates the underlying intuition. At the beginning of the training, the validation score increases rapidly, as shown by the

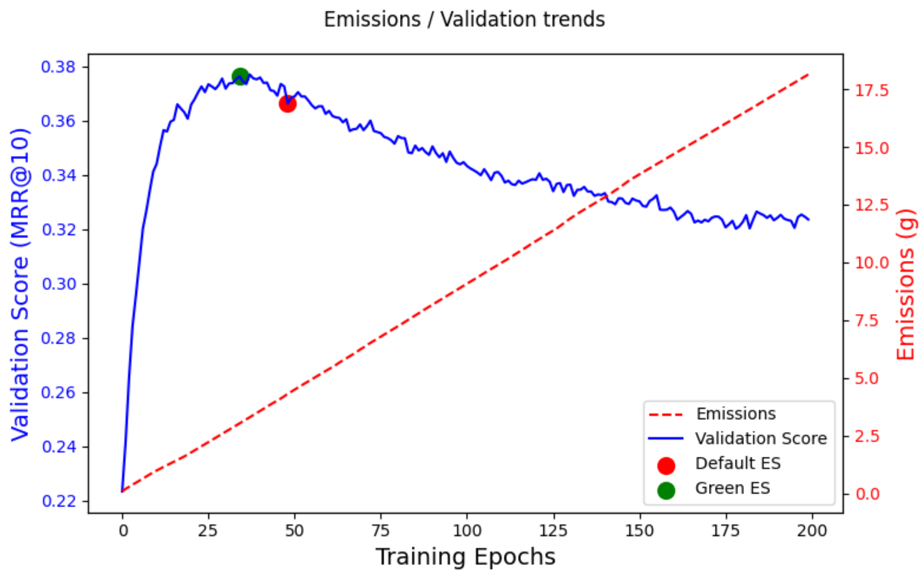


FIGURE 5.22: An example of the accuracy/emission curve graph

blue line in the picture. However, after some epochs, the improvement of the validation score decreases; in this case, the training process could be stopped after a few epochs (red dot), but the GES is able to stop it even before (green dot), by considering the emission produced during the training process (red dotted line). This results in a generally lower carbon footprint.

Discussion In this section, the intuition behind the design of the GES criterion is discussed, and the reasons why it can be regarded as an effective solution for a better trade-off between performance and the carbon footprint of RS models.

Traditional early-stopping approaches stop the training of a model when no improvements are observed over a number of subsequent epochs. By contrast, the GES takes into account the AER (defined in Equation 5.1), in which the carbon footprint is considered in the denominator and increases linearly with each epoch (as depicted in Figure 5.22).

For a given model, the emission trend increases linearly over time; however, each model exhibits a different slope: more complex models emit more CO_2 per epoch, resulting in a steeper line (i.e., a higher slope coefficient), while simpler models have a lower carbon footprint per epoch, corresponding to a less steep line (i.e., a lower slope coefficient).

In this context, the strength of the GES lies in rewarding models that, for the same improvement in the validation metric, have a lower carbon footprint. In other words, models with higher carbon footprints should guarantee, for each epoch, a higher gain in accuracy and thereby justify the continuation of the training process.

5.4.3 Experimental Session

Experiments were designed to answer the following **Research Questions (RQs)**:

- **RQ5.4.1 - Emissions Impact:** How does the early stopping criterion affect the emissions of RS models?
- **RQ5.4.2 - Impact on Recommendation Metrics:** How does the early stopping criterion affect the performance of RS models?
- **RQ5.4.3 - Trade-off analysis:** Can the early stopping criterion be considered a possible solution to improve the trade-off between emissions and performance of RS models?

Experimental Design

Dataset To answer the RQs, experiments were performed on the ML1M dataset, whose description and details are reported in Section 2.3.

Recommendation Models For the experimental session, the following recommendation models have been considered: BPR [146], DMF [235], LINE [189], MultiDAE [99], NGCF [221], DGCF [218], LightGCN [70], CKE [246], CFKG [251], KGCN [211], KGNN-LS [210]. Their discussion is reported in Section 2.3, and is not reported here to avoid repetitions.

GES Parameters. As previously stated, the threshold score used to evaluate whether AER is favorable strongly depends on both the dataset (i.e., number of users and the number of items) and the average emissions of the algorithms. In the paper [180], two different threshold values are evaluated: 30 and 40. From this point onward, the terms *low* and *high* threshold values are used. For the tolerance, three values are evaluated, i.e., 5, 6, and 7. These values are referred to as *low*, *moderate*, and *high* tolerance. All values are set heuristically after several runs of the pipeline and in accordance with the orders of magnitude of the data. This constitutes the first exploratory step in this direction within RS. For future work, a fully automatic strategy to define the optimal values for threshold and tolerance is needed to improve the soundness of the approach.

Implementation Details. To ensure fair comparisons and reproducible results, RecBole was used as the recommendation library. Accordingly, the GES criterion was implemented in this framework to enable reproducibility.

Evaluation Metrics. MRR was considered as an accuracy metric to compute the AER, since it is the default validation metric used by RecBole. The RS performance was evaluated in terms of Recall, NDCG, the Gini index, and Average Popularity. Due to space constraints, results related to Recall and the Gini index are presented.

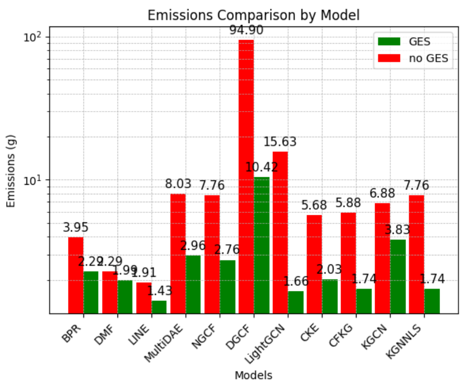
Results of the Experiments Figure 5.23a shows the effect of applying the GES criterion on the $CO_2 - eq$ emissions of recommendation models. Due to space constraints, results obtained under the *high* threshold and *low* tolerance setting are reported. This implies that training continues only if performance increases rapidly. Conversely, if the slope of the curve decreases, even for a few epochs, training stops. In other terms, this represents the most *aggressive* early stopping configuration evaluated.

5.4.4 Results Discussion

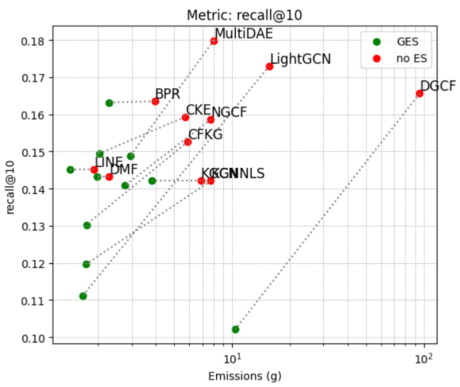
As shown in Figure 5.23a, the GES criterion strongly affects the carbon footprint of RSs. This holds for complex models such as LightGCN and DGCF, with $CO_2 - eq$ emissions approximately 9-times lower, as well as for simpler methods such as BPR, whose emissions are halved when the GES criterion is applied. Although this outcome is expected, the emissions saved by incorporating carbon footprint data directly into the training of recommendation models should be quantified.

A more comprehensive overview of this finding is provided in Tables 5.7 and 5.6, which show the percentage decrease when the GES criterion is applied to varying thresholds and tolerance values. As expected, the higher the tolerance, the lower the amount of emissions that are saved. Conversely, higher thresholds and lower tolerance lead to a reduction that may almost reach 90% of the emissions, as shown by LightGCN and DGCF.

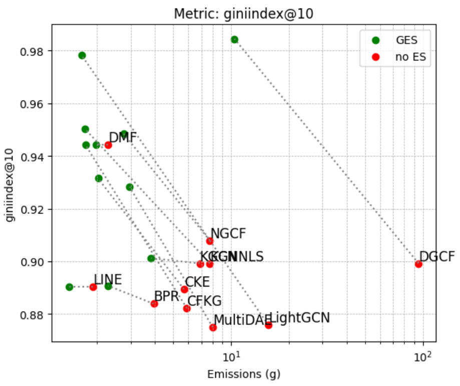
Based on these results, the GES criterion succeeds in decreasing the $CO_2 - eq$ emitted during the training of RS models. For complex models, the amount of emissions saved can be considerable.



(A) Impact of our GES criterion on emissions



(B) Impact of our GES criterion on Recall



(C) Impact of our GES criterion on Gini index

FIGURE 5.23: Results obtained by applying our GES criterion, with *high* treshold and *low* tolerance. *a)* compares the emissions produced by each model whether the GES criterion is applied (green) or not (red); *b)* and *c)* focus on the trade-off between $C0_2$ emissions (*X-axis*) and recommendation performance *(Y-axis)*.

Model	*low tolerance*			*moderate tolerance*			*high tolerance*		
	Emiss.	Rec	Gini	Emiss.	Rec	Gini	Emiss.	Rec	Gini
BPR	-42.01	-0.18	-0.77	-37.12	0	0	-17.99	0	0
DMF	-13.27	0	0	-12.87	0	0	-13.08	0	0
LINE	-25.03	0	0	-27.02	0	0	-27.42	0	0
MultiDAE	-63.68	-17.23	6.09	-62.04	-14.90	5.19	-60.33	-13.90	4.97
NGCF	-64.46	-11.22	4.46	-63.64	-9.33	4.43	-59.43	-6.75	3.50
DGCF	-89.02	-38.29	9.49	-87.48	-34.78	9.21	-85.89	-32.91	8.89
LightGCN	-89.35	-35.78	11.69	-88.85	-34.91	11.51	-88.14	-32.89	11.29
CKE	-64.34	-6.21	4.73	-63.39	-4.77	4.18	-59.18	-3.14	3.44
CFKG	-70.39	-14.61	7.04	-69.33	-14.74	6.56	-67.39	-11.93	6.18
KGCN	-44.29	0	0.22	-42.25	0	-0.09	-41.27	0	0
KGNN-LS	-77.61	-15.76	5.67	-75.90	-13.30	5.09	-73.95	-11.68	4.76

TABLE 5.6: Sensitivity analysis under **high threshold** settings for different levels of user tolerance. Values represent percentage change in emissions, recall, and Gini index.

Model	*low tolerance*			*moderate tolerance*			*high tolerance*		
	Emiss.	Rec	Gini	Emiss.	Rec	Gini	Emiss.	Rec	Gini
BPR	-36.13	0	0	-20.93	0	0	-19.97	0	-
DMF	-9.03	0	0	-12.93	0	0	-13.53	0	-
LINE	-25.17	0	0	-27.14	0	0	-24.98	0	-
MultiDAE	-50.49	-10.01	4.01	-53.42	-9.73	3.47	-50.25	-8.00	3.04
NGCF	-65.30	-11.22	4.46	-62.81	-9.33	4.43	-59.72	-6.75	3.50
DGCF	-89.03	-38.35	9.49	-87.29	-34.78	9.21	-85.90	-32.85	8.89
LightGCN	-87.84	-32.89	11.29	-86.46	-30.40	10.85	-86.66	-30.40	10.85
CKE	-63.99	-6.21	4.73	-63.44	-4.77	4.18	-59.23	-3.14	3.44
CFKG	-67.91	-14.74	6.56	-67.35	-11.93	6.18	-17.81	0	-
KGCN	-44.79	0	0.22	-42.85	0	-0.09	-41.21	0	-
KGNN-LS	-43.76	0.14	0.23	-38.19	0	-0.10	-36.07	0	-

TABLE 5.7: Sensitivity analysis under **low threshold** settings for different levels of user tolerance. Values represent percentage change in emissions, recall, and Gini index.

To address RQ5.4.2, Figures 5.23b and 5.23c show that a decrease in performance generally occurs when GES is applied. Indeed, most lines exhibit a decreasing trend toward the left (for recall, where *higher is better*) or toward the right (for the Gini index, where *lower is better*). This is particularly observed for models emitting more CO_2-eq, such as DGCF, LightGCN, and MultiDAE. However, other models, including BPR, DMF, and KGCN, do not exhibit a decrease in performance, and some techniques, *i.e.,* CKE, are minimally affected by GES. This result is also confirmed in Tables 5.7 and 5.6. As discussed for the previous RQ, higher tolerance generally yields performance closer to the original values. However, the reduction in emissions is smaller as well. **To summarize, GES generally decreases the performance of some RS, but this is not valid for all algorithms. In some settings, only a tiny decrease in performance is observed.**

Finally, RQ5.4.3 is addressed and the suitability of GES as a solution for trading off performance and carbon footprint in RS models is assessed. As previously stated, some models, such as DGCF and LightGCN, strongly reduce their emissions at the cost of an equivalent reduction in performance. Other models, such as MultiDAE or CFKG, show a smaller decrease in performance. Generally, one of the best trade-offs emerges for CKE. Indeed, this model reduces emissions by approximately 60%, with a tiny performance decrease. Overall, it is the second best-performing approach in terms of recall after the application of GES. Finally, the performance of simpler models such as LINE, DMF, and BPR is not affected by GES. In particular, BPR achieves the best accuracy when early stopping is applied.

To address RQ5.4.3, it can be concluded that different algorithms show heterogeneous behavior. Generally, the impact of GES appears to be scenario-dependent. However, simpler algorithms generally provide a good trade-off between the metrics.

5.4.5 Take-Home Messages

The featured publication [180] introduces a novel strategy to reduce the emissions of a recommendation algorithm by design, *i.e.,* by early stopping the training when the ratio between the increase in performance and the amount of emissions is not favorable. This is achieved by introducing a new metric inspired by the finite-difference method that compares the slopes of the curves modeling performance and emissions epoch by epoch. Overall, the analysis showed that the method effectively reduces the emissions of recommendation algorithms, even at a significant rate for carbon-intensive models such as LightGCN and DGCF.

However, especially for more complex techniques, the decrease in emissions often comes at the cost of a decrease in performance. This outcome is not surprising. Indeed, as shown in [31], it has been acknowledged since the early 1990s that early-stopping methods follow the *no-free-lunch* theory. Accordingly, the reduction in terms of emissions needs to be balanced by a decrease in performance.

Generally speaking, the identification of the ideal trade-off between these dichotomous aspects is not trivial. In some settings, it may be fundamental to reduce emissions, regardless of the loss in performance. Conversely, in particular domains, the accuracy of the algorithms may be indispensable, so the *tolerance* needs to be increased. This is a domain- and scenario-specific design choice.

Automatic tuning of such a trade-off is left for future work, so that the criterion can be extended and made fully automatic. Indeed, the main issue encountered was the different magnitudes of the values of performance and emissions, which made it difficult to define a fixed threshold for comparing the curves. Possible solutions include: *(a)* the adoption of a logarithmic formula that flattens the curves and brings their magnitudes closer; *(b)* the use of a *min-max* normalization strategy, where normalization is obtained by considering as *maximum* the scores of performance and the emissions benchmarked in previous related works, such as [179, 204].

Main Findings

- **RQ5.4.1**: The GES criterion dramatically decreases the emission of RSs, especially for more complex models.
- **RQ5.4.2**: The GES criterion also affects recommendation performance, which show an expected decrease, but this does not hold for all the models. Some of them show only a marginal performance decrease.
- **RQ5.4.3**: Simpler models provide a better trade-off between performance and carbon footprint when GES is applied, while for the other models, the effect is scenario-dependent.

These results have been carried out during the PhD and have been published at ACM UMAP 2025, in the Main Track (Giuseppe Spillo et al. "Training Green and Sustainable Recommendation Models: Introducing Carbon Footprint Data into Early Stopping Criteria". In: *Proceedings of the 33rd ACM Conference on User Modeling, Adaptation and Personalization*. 2025, pp. 341–346), presented at the poster session.

5.5 Conclusions

This section concludes the chapter and draws some conclusions to answer RQ3, introduced in Chapter 1. This chapter focused on beyond-accuracy aspects of KARS which are overlooked in the literature.

First, it investigated quantization approaches in large-scale multimodal scenarios for top-*k* recommendation. To this aim, URecJPQ has been designed, and its integration in state-of-the-art recommendation models clearly showed how it is possible to dramatically reduced the number of trainable parameters with only a marginal decrease in performance. This kind of analysis is poorly investigated in KARS, as the literature focused more on integrating the quantization on sequential recommendation models that ignore side information.

Then, the chapter introduced a novel evaluation metric for RS and KARS, the carbon footprint. This has been performed by running a benchmark analysis investigating the trade-off between recommendation performance and emissions produced by RS during training, both when default hyperparameter values are set, and when they are fine-tuned. The results discussion showed how simpler recommendation models show a more favorable trade-off with respect to more complex models, as they typically show marginal performance improvements at the cost of a huge emission increase. This is a novel contribution for the RS and KARS literature, as no previous work investigated these aspects. As confirmation of this, this work initiated a new research line within the RecSys community, thereby reinforcing its relevance and impact in the literature.

The subsequent sections extended this research line by investigating how to improve this trade-off. It is worth pointing out that these strategies were never been investigated in the literature before, thus these works represent a relevant advance in the RS and KARS literature.

To do so, first, novel data reduction strategies have been discussed. The core idea of data reduction is to decrease the quantity of training data following different heuristics, so to reduce training times, thus improving the model's energy efficiency. The experimental session clearly showed that data reduction is an effective approach to improve the mentioned trade-off. Moreover, the recommendations provided following this approach are more diverse and less affected by the popularity bias, which an unexpected but positive side-effect related.

Then, the problem of improving the trade-off has been faced under a different perspective: instead of reducing the training data, a novel emissions-aware early stop criterion has been designed. Its key idea consists in stopping the training process when the increase in the emissions does not justify the small improvement in the validation metric. Similarly to the previous case, this approach reduced the emissions of RS and KARS models, with a performance decrease scenario-dependent, and a general improvement as for the popularity bias.

The results of this chapter highlight that beyond-accuracy aspects of RS and KARS should be explicitly considered during evaluation. In particular, strategies such as model quantization, data reduction, and emissions-aware early stopping show that dramatic improvements in efficiency, sustainability, and recommendation diversity can be achieved with only marginal decreases in traditional accuracy metrics. For example, simpler models or quantized architectures achieve nearly the same top-*k* performance as complex models while dramatically reducing computational cost and carbon footprint. Similarly, reducing training data or applying an emissions-aware early stop criterion can lower energy consumption and mitigate popularity bias without significantly harming recommendation quality.

These findings suggest that accuracy alone is insufficient as a metric for evaluating RSs, as large gains in beyond-accuracy objectives often come at a marginal cost in performance. Consequently, a multi-objective evaluation perspective is warranted, where RS design and assessment explicitly balance accuracy and beyond-accuracy metrics. Adopting such a framework encourages the development of models that are not only accurate but also environmentally sustainable, efficient, and fair, opening a new line of research in the RS and KARS literature.

This discussion helps at answering RQ3 of this thesis, introduced in Chapter 1.

Answer to RQ3: Beyond-Accuracy Implications

- Accuracy alone is insufficient to assess the overall performance and impact of RSs and KARS. Metrics such as resource usage, carbon footprint and diversity, should also be considered.
- Quantization dramatically reduces trainable parameters with only marginal performance decreases.
- Similarly, reducing the training dataset decreases training time and emissions, with positive side-effect on fairness metrics.
- In addition, stopping the training when emissions outweigh small gains in validation performance minimizes carbon footprint. Also in this case, improvements in fairness metrics can be observed.
- A Multi-Objective perspective for evaluating RS is needed: RS evaluation should explicitly balance accuracy and beyond-accuracy objectives, and adopting this perspective supports the design of models that are both effective and sustainable.

Chapter 6

Conclusions

This chapter summarizes the various contributions of this thesis and how the research conducted during the PhD advances the current state-of-the-art in related areas. In addition, it highlights the limitations and sketches future works for improving the discussed contributions.

6.1 Contributions of this Thesis

Throughout this thesis, several aspects of RSs and KARS have been discussed, aiming to contribute to the research in the area. To better assess each contribution, the Research Questions introduced in Chapter 1 are discussed here, and answers are provided.

Research Question 1: Contribution of Individual Knowledge Sources: Throughout the research discussed in Chapters 3 and 4, it has been observed how the different knowledge sources affect the overall recommendation performance when considered separately.

First, KGs are a useful knowledge source for providing recommendations; in addition, injecting FOL rules (see Section 3.1) during the KG embedding process can further improve such performance. However, the choice of First-Order Logic rules to be injected is a key aspect of the process. Experiments showed that rules with a confidence score higher than a certain threshold tend to provide the best results.

Logic rules are not the only means of exploiting KGs to derive new knowledge; KGs encoding item features can be exploited by LLMs to reason over user preferences and derive new triples encoding user preferences into *groups* of items (see Section 3.2). Improvements in recommendation accuracy relative to KARS based only on popular KGs were observed, particularly in sparser scenarios, which nevertheless still provide very good performance. LLMs can be exploited to generate textual user profiles starting from item descriptions (Section 3.3), showing encouraging results in comparison with simpler approaches, such as the centroid, especially in sparser scenarios.

After focusing on KGs and textual knowledge sources, attention is turned to multimedia sources, such as images, audio and video data. A key challenge in this research line (commonly referred to as multimodal recommendation) is the availability of datasets, since collecting multimedia data is challenging. In this context, another contribution of this thesis is the release of three benchmark datasets (ML1M, Last.fm-2K, DBbook) extended with multimodal features, discussed in Section 3.4. In Section 4.3 it is shown that these information sources can improve non-accuracy metrics, such as the diversity of the recommendations or their novelty, while not always providing better recommendations with respect to CF data. This result is also confirmed by the conducted user study (Section 3.4), in which it was observed that, while graph-based recommendations provide more accurate results, multimedia signals improve diversity and user engagement when interacting with the system.

From the pure accuracy perspective, graph-based embeddings are the most effective knowledge source for providing recommendations, whereas multimedia embeddings provide complementary knowledge that contributes to increasing diversity and engagement.

Research Question 2: Fusion of Heterogeneous Embeddings Findings from the previous discussion show that multimedia embeddings can be used to enhance beyond-accuracy capabilities of graph-based embeddings. To this aim, Section 4.1 shows that combining graph and text embeddings leads to better recommendation performance than in scenarios where the embeddings are treated separately. The section discusses a work [167], published at ACM UMAP 23, which has been positively received by the scientific community, as this paper won the James Chen Best Student Paper Award. The results discussed in Section 4.1 are also confirmed by the research discussed in Section 4.2, in which multimedia embeddings are integrated and the fusion architecture is extended to incorporate multimedia features, including audio, video, and images. Moreover, it is observed that an effective technique to fuse heterogeneous embeddings relies on attention mechanisms, which provide better performance than simple concatenation.

Another effective strategy to improve the recommendation performance consists in pre-training the graph embeddings with multimedia embeddings (e.g., textual); this approach fuses the semantics provided by the multimedia embeddings with the graph structure provided by the interaction matrix, leading to slightly better results, particularly in sparser scenarios.

To address RQ2, it is stated that fusing heterogeneous embeddings effectively improves the recommendation performance; in particular, fused graph and text embeddings provide better performance than solely graph or solely text, and the improvement further increases when multimedia embeddings are considered. On the other hand, multimedia embeddings still provide the most diverse recommendations, so future work should investigate how to fuse heterogeneous embeddings to improve both accuracy and beyond-accuracy metrics.

Research Question 3: Beyond-accuracy Implications and Multi-Objective Perspective Although recommendation accuracy is a crucial aspect of evaluating RSs, it is not exhaustive for defining the quality of the recommendations. To this end, beyond-accuracy metrics are relevant to measure other aspects of the recommendation, such as efficiency or fairness, as discussed in Chapter 1.

Under this perspective, in cooperation with the University of Glasgow, experiments were conducted to reduce the total number of trainable parameters and improve memory efficiency in the context of large-scale catalog multimodal recommendation. To this aim, URecJPQ was proposed, a novel quantization approach which showed that quantizing user and item ID embeddings can lead to a negligible decrease in accuracy with a substantial decrease in the number of trainable parameters. This result suggests that not only accuracy metrics should be considered, and that a favorable trade-off between accuracy and efficiency can be found.

Another relevant aspect concerns environmental sustainability. In cooperation with the University of Bologna, a benchmark experiment was conducted to assess the trade-off between accuracy and the carbon footprint of popular CF RSs and KARS. Results, deeply discussed in Section 5.2, showed that in many scenarios more complex models produce much higher emissions than simpler models, with only tiny improvements in accuracy. This work has been positively perceived by the RecSys community; indeed, it influenced the community in several ways after its presentation at the ACM RecSys 23 Conference: first, a reproducibility paper [204] published at the ACM RecSys 24 Conference confirmed the results; subsequently, a new workshop, RecSoGood [18] focusing on sustainability in RSs, was organized and co-located with ACM RecSys, the main RS conference; a call-to-action [16] was advanced, with the idea of designing Green RSs and using the carbon footprint as an additional evaluation metric.

Further investigations were conducted on data reduction strategies to improve the trade-off between accuracy and carbon footprint, as discussed in Section 5.3, and by designing a custom Green Early Stop criterion that stops the training when the improvements in the accuracy do not justify the increase in the carbon footprint, discussed in Section 5.4. In both cases, it was observed that more complex models (including several KARS) benefit from these strategies, at the expense of accuracy, while achieving a relevant decrease in the carbon footprint.

To address RQ3, a multi-objective perspective of RSs and KARS is required: as demonstrated in the literature, including the results discussed in this thesis, tiny accuracy improvements often come at the cost of large increases in other aspects, such as the carbon footprint, and such aspects should be considered when evaluating recommendation models. The approaches proposed to tackle this challenge include quantization techniques, data reduction strategies, and a Green Early Stop criterion.

6.2 Future Works

The results discussed throughout this thesis highlight several open challenges and research opportunities that emerge from both the strengths and the limitations of the proposed approaches. While the presented contributions advance the state of the art in knowledge-aware and multimodal recommendation, they also expose fundamental open problems related to representation alignment, multi-objective optimization, and sustainable system design. In the following, the most promising future research directions are discussed.

Learning Aligned Multimodal Representations for Recommendation. A recurring theme in this thesis is the effectiveness of fusing heterogeneous embeddings, such as graph-based and multimodal representations, to improve recommendation performance. However, the empirical results also suggest that such embeddings are often misaligned, as they are pre-trained independently and optimized for tasks that are only partially related to recommendation. This misalignment may introduce noise and limit the effectiveness of fusion strategies, even when attention mechanisms are adopted. An important open research problem, therefore, concerns how to learn multimodal representations that are explicitly aligned with both the recommendation objective and the underlying semantics of the items.

A promising direction consists in integrating Large Multimodal Models, such as CLIP [141] or BEiT-3 [216], within recommendation architectures, enabling end-to-end learning of aligned multimodal features [238]. Such an approach could mitigate the semantic gap between modalities and allow the model to adapt multimodal representations directly to user preferences. At the same time, this direction raises non-trivial challenges, including a significant increase in model complexity, higher computational costs, and the difficulty of jointly aligning multiple modalities (e.g., text, images, audio, and video). Future research should therefore investigate lightweight or modular solutions that balance representation alignment with scalability and efficiency.

Fusion Strategies Beyond Accuracy-Oriented Objectives. Another key finding of this thesis is that different knowledge sources contribute differently to recommendation quality: structured sources such as knowledge graphs primarily improve accuracy, whereas unstructured multimodal sources tend to enhance beyond-accuracy aspects, such as novelty and diversity. Interestingly, while fusing these sources leads to further gains in accuracy, it does not consistently translate into improvements in beyond-accuracy metrics. This observation suggests a structural limitation of current fusion strategies, which are typically optimized for a single accuracy-centric objective.

Future research should therefore move beyond traditional fusion architectures and investigate training paradigms that explicitly account for multiple, and potentially conflicting, objectives. This includes the design of multi-objective loss functions, adaptive fusion mechanisms that dynamically prioritize different knowledge sources, and learning strategies that incorporate beyond-accuracy metrics directly into the optimization process. Such approaches could help bridge the gap between accuracy-driven learning and user- centric evaluation criteria, leading to recommendation systems that are both effective and experience-aware.

Toward a Unified Multi-Objective Perspective on Recommendation Systems. The investigations presented in this thesis clearly demonstrate that recommendation systems should be evaluated and designed from a multi-objective perspective. Small improvements in accuracy are often accompanied by disproportionately large increases in model complexity, computational cost, and carbon footprint. This raises fundamental questions about the sustainability and practicality of increasingly complex recommendation models.

Future work should extend the multi-objective framework explored in this thesis by incorporating additional dimensions, such as fairness, robustness, and transparency, alongside accuracy, efficiency, and environmental impact. In particular, the interaction between fairness-aware recommendation and sustainability constraints remains largely unexplored, and designing models that balance these objectives represents an important open challenge. Moreover, the concept of Green RS could be further developed by investigating ranking strategies that consider sustainability-aware scores in addition to relevance, enabling greener suggestions outputs rather than only greener training and inference processes. For instance, a Green RS could prioritize recommending a smartphone with a more sustainable lifecycle instead of simply selecting the most recent or most powerful device.

Advanced Training Control and Sustainable Optimization Strategies. This thesis introduced a simple yet effective Green Early Stop criterion to limit unnecessary training epochs when accuracy improvements no longer justify the associated emissions. While promising, this approach represents only a first step toward more principled and adaptive training control mechanisms. Future research should investigate more sophisticated strategies, such as early stopping criteria based on the Exponential Moving Average [93] of the ratio between validation performance and carbon emissions, or learning- based controllers that dynamically adjust training behavior according to sustainability constraints.

More broadly, these directions point toward a rethinking of how recommendation models are trained, evaluated, and deployed, emphasizing responsible and resource-aware machine learning practices. Addressing these challenges will be essential for ensuring that future recommendation systems are not only accurate and engaging, but also scalable, fair, and environmentally sustainable.

Bibliography

[1] Niran A Abdulhussein and Ahmed J Obaid. "User recommendation system based on MIND dataset". In: *arXiv preprint arXiv:2209.06131* (2022).

[2] Fatemeh Alyari and Nima Jafari Navimipour. "Recommender systems: a systematic review of the state of the art literature and suggestions for future research". In: *Kybernetes* (2018).

[3] Nicholas Ampazis and Theodoros Emmanouilidis. "Exploring semantic features for producing top-n recommendation lists from binary user feedback". In: *Semantic Web Evaluation Challenge*. Springer, 2014, pp. 157–162.

[4] Vito Walter Anelli et al. "Elliot: A comprehensive and rigorous framework for reproducible recommender systems evaluation". In: *Proceedings of the 44th international ACM SIGIR conference on research and development in information retrieval*. 2021, pp. 2405–2414.

[5] Vito Walter Anelli et al. "How to make latent factors interpretable by feeding factorization machines with knowledge graphs". In: *International Semantic Web Conference*. Springer. 2019, pp. 38–56.

[6] Vito Walter Anelli et al. "Knowledge-aware and conversational recommender systems". In: *Proceedings of the 12th ACM Conference on Recommender Systems*. RecSys '18. Vancouver, British Columbia, Canada: Association for Computing Machinery, 2018, 521–522. ISBN: 9781450359016. URL: `https://doi.org/10.1145/3240323.3240338`.

[7] Taushif Anwar, V Uma, and Gautam Srivastava. "Rec-cfsvd++: Implementing recommendation system using collaborative filtering and singular value decomposition (svd)++". In: *International Journal of Information Technology & Decision Making* 20.04 (2021), pp. 1075–1093.

[8] Sören Auer et al. "DBpedia: A nucleus for a web of open data". In: *The semantic web*. Springer, 2007, pp. 722–735.

[9] Artem Babenko and Victor Lempitsky. "Additive quantization for extreme vector compression". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2014, pp. 931–938.

[10] Prafulla Bafna, Dhanya Pramod, and Anagha Vaidya. "Document clustering: TF-IDF approach". In: *2016 International conference on electrical, electronics, and optimization techniques (ICEEOT)*. IEEE. 2016, pp. 61–66.

[11] Ashmi Banerjee. "Fairness and sustainability in multistakeholder tourism recommender systems". In: *Proceedings of the 31st ACM Conference on User Modeling, Adaptation and Personalization*. 2023, pp. 274–279.

[12] Ashmi Banerjee et al. "Modeling sustainable city trips: integrating CO 2 e emissions, popularity, and seasonality into tourism recommender systems". In: *Information Technology & Tourism* (2025), pp. 1–38.

[13] Nesrine Bannour et al. "Evaluating the carbon footprint of NLP methods: a survey and analysis of existing tools". In: *Proceedings of the second workshop on simple and efficient natural language processing*. 2021, pp. 11–21.

[14] Hangbo Bao et al. "VLMo: Unified Vision-language Pre-training With Mixture-of-modality-experts". In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 32897–32912.
DOI: `https://doi.org/10.48550/arXiv.2111.02358`.

[15] Joeran Beel et al. "Green Recommender Systems–A Call for Attention". In: ().

[16] Joeran Beel et al. "Green Recommender Systems: A Call for Attention". In: *ACM SIGIR Forum*. Vol. 58. 2. ACM New York, NY, USA. 2025, pp. 1–5.

[17] S Bhaskaran et al. "Recommendation system using inference-based graph learning–modeling and analysis". In: *2022 2nd International Conference on Innovative Sustainable Computational Technologies (CISCT)*. IEEE. 2022, pp. 1–5.

[18] Ludovico Boratto et al. "First International Workshop on Recommender Systems for Sustainability and Social Good (RecSoGood 2024)". In: *Proceedings of the 18th ACM Conference on Recommender Systems*. 2024, pp. 1239–1241.

[19] Antoine Bordes et al. "Translating embeddings for modeling multi-relational data". In: *Advances in neural information processing systems* 26 (2013), pp. 2787–2795.

[20] Alexander EI Brownlee et al. "Exploring the accuracy–energy trade-off in machine learning". In: *2021 IEEE/ACM International Workshop on Genetic Improvement (GI)*. IEEE. 2021, pp. 11–18.

[21] Johannes Buberger et al. “Total CO2-equivalent life-cycle emissions from commercially available passenger cars”. In: *Renewable and Sustainable Energy Reviews* 159 (2022), p. 112158.

[22] Semen Andreevich Budennyy et al. “Eco2ai: carbon emissions tracking of machine learning models as the first step towards sustainable ai”. In: *Doklady Mathematics*. Vol. 106. Suppl 1. Springer. 2022, S118–S128.

[23] Robin Burke et al. “Preface to the special issue on fair, accountable, and transparent recommender systems”. In: *User Modeling and User-Adapted Interaction* 31.3 (2021), pp. 371–375.

[24] Hongyun Cai, Vincent W Zheng, and Kevin Chen-Chuan Chang. “A comprehensive survey of graph embedding: Problems, techniques, and applications”. In: *IEEE Transactions on Knowledge and Data Engineering* 30.9 (2018), pp. 1616–1637.

[25] Pedro G Campos, Fernando Díez, and Manuel Sánchez-Montañés. “Towards a more realistic evaluation: testing the ability to predict future tastes of matrix factorization-based recommenders”. In: *Proceedings of the fifth ACM conference on Recommender systems*. 2011, pp. 309–312.

[26] Iván Cantador, Peter Brusilovsky, and Tsvi Kuflik. “Second workshop on information heterogeneity and fusion in recommender systems (HetRec2011)”. In: *Proceedings of the fifth ACM conference on Recommender systems*. 2011, pp. 387–388.

[27] Yixin Cao et al. “Unifying knowledge graph learning and recommendation: Towards a better understanding of user preferences”. In: *The world wide web conference*. 2019, pp. 151–161.

[28] Diego Carraro and Derek Bridge. “Enhancing recommendation diversity by re-ranking with large language models”. In: *ACM Transactions on Recommender Systems* (2024).

[29] Joao Carreira and Andrew Zisserman. “Quo vadis, action recognition? a new model and the kinetics dataset”. In: *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 6299–6308.

[30] Joel Castaño et al. “Exploring the Carbon Footprint of Hugging Face’s ML Models: A Repository Mining Study”. In: *2023 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*. IEEE. 2023, pp. 1–12.

[31] Zehra Cataltepe, Yaser S Abu-Mostafa, and Malik Magdon-Ismail. “No free lunch for early stopping”. In: *Neural computation* 11.4 (1999), pp. 995–1009.

[32] Dong-Kyu Chae et al. “Cfgan: A generic collaborative filtering framework based on generative adversarial networks”. In: *Proceedings of the 27th ACM*

international conference on information and knowledge management. 2018, pp. 137–146.

[33] Lucas Høyberg Puvis de Chavannes et al. "Hyperparameter power impact in transformer language model training". In: *Proceedings of the second workshop on simple and efficient natural language processing*. 2021, pp. 96–118.

[34] Li Chen et al. "Human decision making and recommender systems". In: *ACM Transactions on Interactive Intelligent Systems (TiiS)* 3.3 (2013), pp. 1–7.

[35] Shu Chen et al. "A survey on cross-domain sequential recommendation". In: *arXiv preprint arXiv:2401.04971* (2024).

[36] Xiaojun Chen, Shengbin Jia, and Yang Xiang. "A review: Knowledge reasoning over knowledge graph". In: *Expert systems with applications* 141 (2020), p. 112948.

[37] Zefeng Chen et al. "Data Scarcity in Recommendation Systems: A Survey". In: *ACM Trans. Recomm. Syst.* 3.3 (Mar. 2025). URL: `https://doi.org/10.1145/3639063`.

[38] Zhe Chen et al. "Knowledge graph completion: A review". In: *Ieee Access* 8 (2020), pp. 192435–192456.

[39] Janneth Chicaiza and Priscila Valdiviezo-Diaz. "A comprehensive survey of knowledge graph-based recommender systems: Technologies, development, and contributions". In: *Information* 12.6 (2021), p. 232.

[40] Davide Chicco. "Siamese neural networks: An overview". In: *Artificial Neural Networks* (2021), pp. 73–94.

[41] Francesco Colace et al. "A content-based recommendation approach based on singular value decomposition". In: *Connection Science* 34.1 (2022), pp. 2158–2176.

[42] Allegra De Filippo, Ludovico Boratto, and Giuseppe Spillo. "Human-Centered and Sustainable Recommender Systems". In: *Adjunct Proceedings of the 33rd ACM Conference on User Modeling, Adaptation and Personalization*. 2025, pp. 10–12.

[43] Allegra De Filippo et al. "Recommender systems and sustainability: a dual perspective". In: *Computer Science Review* 60 (2026), p. 100912.

[44] Marco De Gemmis et al. "Semantics-aware content-based recommender systems". In: *Recommender systems handbook*. Springer, 2015, pp. 119–159.

[45] Catherine De Wolf, Francesco Pomponi, and Alice Moncaster. "Measuring embodied carbon dioxide equivalent of buildings: A review and critique of current industry practice". In: *Energy and Buildings* 140 (2017), pp. 68–80.

[46] Yashar Deldjoo et al. “A Review of Modern Fashion Recommender Systems”. In: *ACM Comput. Surv.* 56.4 (Oct. 2023). ISSN: 0360-0300. URL: `https://doi.org/10.1145/3624733`.

[47] Yashar Deldjoo et al. “Content-based video recommendation system based on stylistic visual features”. In: *Journal on Data Semantics* 5.2 (2016), pp. 99–113.

[48] Yashar Deldjoo et al. “Multimedia recommender systems: Algorithms and challenges”. In: *Recommender systems handbook*. Springer, 2021, pp. 973–1014.

[49] Yashar Deldjoo et al. “Recommender Systems Leveraging Multimedia Content”. In: *ACM Comput. Surv.* 53.5 (Sept. 2020). ISSN: 0360-0300. URL: `https://doi.org/10.1145/3407190`.

[50] Mukund Deshpande and George Karypis. “Item-Based Top-N Recommendation Algorithms”. In: *ACM Trans. Inf. Syst.* 22.1 (2004), 143–177. ISSN: 1046-8188. URL: `https://doi.org/10.1145/963770.963776`.

[51] Tim Dettmers et al. “Convolutional 2d knowledge graph embeddings”. In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 32. 1. 2018.

[52] Jacob Devlin et al. “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”. In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*. Ed. by Jill Burstein, Christy Doran, and Thamar Solorio. Association for Computational Linguistics, 2019, pp. 4171–4186.

[53] Karlijn Dinnissen. “Fairness and Transparency in Music Recommender Systems: Improvements for Artists”. In: *Proceedings of the 18th ACM Conference on Recommender Systems*. 2024, pp. 1368–1375.

[54] A. Dosovitskiy et al. “An image is worth 16x16 words: Transformers for image recognition at scale”. In: *arXiv preprint arXiv:2010.11929* (2020).

[55] Luis Galárraga et al. “Fast rule mining in ontological knowledge bases with AMIEPlus”. In: *The VLDB Journal* 24.6 (2015), pp. 707–730.

[56] Luis Antonio Galárraga et al. “AMIE: association rule mining under incomplete evidence in ontological knowledge bases”. In: *Proceedings of the 22nd international conference on World Wide Web*. 2013, pp. 413–422.

[57] Yunfan Gao et al. “Retrieval-augmented generation for large language models: A survey”. In: *arXiv preprint arXiv:2312.10997* (2023).

[58] Palash Goyal and Emilio Ferrara. "Graph embedding techniques, applications, and performance: A survey". In: *Knowledge-Based Systems* 151 (2018), pp. 78–94.

[59] Robert Gray. "Vector quantization". In: *IEEE Assp Magazine* 1.2 (1984), pp. 4–29.

[60] Yue Gu et al. "Hybrid attention based multimodal network for spoken language classification". In: *Proceedings of the Conference. association for Computational Linguistics. meeting*. Vol. 2018. NIH Public Access. 2018, p. 2379.

[61] Arūnas Gudinavičius and Andrius Šuminas. "Choosing a book by its cover: analysis of a reader's choice". In: *Journal of Documentation* 74.2 (2018), pp. 430–446.

[62] Qingyu Guo et al. "A survey on knowledge graph-based recommender systems". In: *IEEE Transactions on Knowledge and Data Engineering* (2020).

[63] Shu Guo et al. "Jointly embedding knowledge graphs and logical rules". In: *Proceedings of the 2016 conference on empirical methods in natural language processing*. 2016, pp. 192–202.

[64] Petr Hájek. *Metamathematics of fuzzy logic*. Vol. 4. Springer Science & Business Media, 2013.

[65] Xiao Han, Chen Zhu, Xiao Hu, et al. "Adapting Job Recommendations to User Preference Drift with Behavioral-Semantic Fusion Learning". In: *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 2024, pp. 1004–1015.
DOI: `https://doi.org/10.1145/3637528.3671759`.

[66] Per Christian Hansen. "The truncated SVD as a method for regularization". In: *BIT Numerical Mathematics* 27 (1987), pp. 534–553.

[67] Hebatallah A Mohamed Hassan et al. "BERT, ELMo, USE and InferSent Sentence Encoders: The Panacea for Research-Paper Recommendation?" In: *RecSys (Late-Breaking Results)*. 2019, pp. 6–10.

[68] Kaiming He et al. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[69] Ruining He and Julian McAuley. "VBPR: Visual Bayesian Personalized Ranking from implicit feedback". In: *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*. AAAI'16. Phoenix, Arizona: AAAI Press, 2016, 144–150.

[70] Xiangnan He et al. "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. 2020, pp. 639–648.

[71] Xiangnan He et al. "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, 173–182.
ISBN: 9781450349130. URL: `https://doi.org/10.1145/3038912.3052569`.

[72] Shawn Hershey et al. "CNN architectures for large-scale audio classification". In: *2017 ieee international conference on acoustics, speech and signal processing (icassp)*. IEEE. 2017, pp. 131–135.

[73] Balázs Hidasi et al. "Session-based recommendations with recurrent neural networks". In: *arXiv preprint arXiv:1511.06939* (2015).

[74] Aidan Hogan et al. "Knowledge graphs". In: *ACM Computing Surveys (Csur)* 54.4 (2021), pp. 1–37.

[75] Yupeng Hou et al. "Bridging Language and Items for Retrieval and Recommendation". In: *arXiv preprint arXiv:2403.03952* (2024).

[76] Lei Huang et al. "A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions". In: *ACM Transactions on Information Systems* (2025).

[77] Po-Sen Huang et al. "Learning deep structured semantic models for web search using clickthrough data". In: *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*. 2013, pp. 2333–2338.

[78] Wenqing Huang et al. "Dual-LightGCN: Dual light graph convolutional network for discriminative recommendation". In: *Computer Communications* 204 (2023), pp. 89–100.

[79] Zan Huang, Hsinchun Chen, and Daniel Zeng. "Applying associative retrieval techniques to alleviate the sparsity problem in collaborative filtering". In: *ACM Transactions on Information Systems (TOIS)* 22.1 (2004), pp. 116–142.

[80] Antonio Raffaele Iacovazzi et al. "E-Mealio: An LLM-Powered Conversational Agent for Sustainable and Healthy Food Recommendation". In: *Second International Workshop on Recommender Systems for Sustainability and Social Good. In press.* 2025.

[81] Dietmar Jannach et al. *Recommender systems: an introduction*. Cambridge University Press, 2010.

[82] Dietmar Jannach et al. “Recommender Systems for Good (RS4Good): Survey of Use Cases and a Call to Action for Research that Matters”. In: *arXiv preprint arXiv:2411.16645* (2024).

[83] Herve Jegou, Matthijs Douze, and Cordelia Schmid. “Product quantization for nearest neighbor search”. In: *IEEE transactions on pattern analysis and machine intelligence* 33.1 (2010), pp. 117–128.

[84] Hamed Karimi and Ali Kamandi. “A learning-based ontology alignment approach using inductive logic programming”. In: *Expert Systems With Applications* 125 (2019), pp. 412–424.

[85] Shima Khoshraftar and Aijun An. “A survey on graph representation learning methods”. In: *ACM Transactions on Intelligent Systems and Technology* 15.1 (2024), pp. 1–55.

[86] Yehuda Koren, Robert Bell, and Chris Volinsky. “Matrix factorization techniques for recommender systems”. In: *Computer* 42.8 (2009), pp. 30–37.

[87] Sotiris Kotsiantis and Dimitris Kanellopoulos. “Association rules mining: A recent overview”. In: *GESTS International Transactions on Computer Science and Engineering* 32.1 (2006), pp. 71–82.

[88] Alexandre Lacoste et al. “Quantifying the carbon emissions of machine learning”. In: *arXiv preprint arXiv:1910.09700* (2019).

[89] Jonathan Lajus, Luis Galárraga, and Fabian Suchanek. “Fast and exact rule mining with AMIE 3”. In: *European Semantic Web Conference*. Springer. 2020, pp. 36–52.

[90] Loïc Lannelongue, Jason Grealey, and Michael Inouye. “Green algorithms: quantifying the carbon footprint of computation”. In: *Advanced science* 8.12 (2021), p. 2100707.

[91] Martha Larson, Alessandro Zito, Babak Loni, et al. “Towards Minimal Necessary Data: The Case for Analyzing Training Data Requirements of Recommender Algorithms”. In: *FATREC Workshop on Responsible Recommendation Proceedings*. 2017.
DOI: `https://doi.org/10.18122/b2vx12`.

[92] Dhyaa-Al Rahman Lateef and Ahmed J Obaid. “News Recommendation System Based Deep Learning on MIND Dataset”. In: *International Conference on Biologically Inspired Techniques in Many-Criteria Decision-Making Technologies*. Springer. 2024, pp. 264–275.

[93] AJ Lawrance and PAW Lewis. “An exponential moving-average sequence and point process (EMA1)”. In: *Journal of Applied Probability* 14.1 (1977), pp. 98–113.

[94] Kuang-Huei Lee et al. "Stacked cross attention for image-text matching". In: *Proceedings of the European conference on computer vision (ECCV)*. 2018, pp. 201–216.

[95] Anchen Li and Bo Yang. "GSIRec: Learning with graph side information for recommendation". In: *World Wide Web* 24.5 (2021), pp. 1411–1437.

[96] Changpin Li and Fanhai Zeng. "Finite difference methods for fractional differential equations". In: *International Journal of Bifurcation and Chaos* 22.04 (2012), p. 1230014.

[97] Yang Li and Tao Yang. "Word embedding for understanding natural language: a survey". In: *Guide to big data applications*. Springer, 2018, pp. 83–104.

[98] Defu Lian et al. "Lightrec: A memory and search-efficient recommender system". In: *Proceedings of the web conference 2020*. 2020, pp. 695–705.

[99] Dawen Liang et al. "Variational autoencoders for collaborative filtering". In: *Proceedings of the 2018 world wide web conference*. 2018, pp. 689–698.

[100] Hsiu-Ping Lin et al. "Amazon books rating prediction & recommendation model". In: *arXiv preprint arXiv:2310.03200* (2023).

[101] Yankai Lin et al. "Learning Entity and Relation Embeddings for Knowledge Graph Completion". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 29.1 (2015). URL: `https://ojs.aaai.org/index.php/AAAI/article/view/9491`.

[102] Peng Liu, Lemei Zhang, and Jon Atle Gulla. "Dynamic attention-based explainable recommendation with textual and visual fusion". In: *Information Processing & Management* 57.6 (2020), p. 102099.

[103] Pingshan Liu and Guoxin Lu. "Addressing data imbalance for federated recommender systems: a rebalancing framework with gradient alignment regularization". In: *Journal of Intelligent Information Systems* 63.2 (2025), pp. 657–679.
DOI: `10.1007/S10844-024-00910-8`. URL: `https://doi.org/10.1007/s10844-024-00910-8`.

[104] Qidong Liu et al. "Multimodal Recommender Systems: A Survey". In: *ACM Comput. Surv.* 57.2 (Oct. 2024).
ISSN: 0360-0300. URL: `https://doi.org/10.1145/3695461`.

[105] Vivian Liu and Yiqiao Yin. "Green AI: Exploring Carbon Footprints, Mitigation Strategies, and Trade Offs in Large Language Model Training". In: *arXiv preprint arXiv:2404.01157* (2024).

[106] Yinhan Liu et al. "RoBERTa: A Robustly Optimized BERT Pretraining Approach". In: vol. abs/1907.11692. 2019. arXiv: `1907.11692`. URL: `http://arxiv.org/abs/1907.11692`.

[107] Pasquale Lops, Marco De Gemmis, and Giovanni Semeraro. "Content-based recommender systems: State of the art and trends". In: *Recommender systems handbook* (2011), pp. 73–105.

[108] Pasquale Lops et al. "A semantic content-based recommender system integrating folksonomies for personalized access". In: *Web Personalization in Intelligent Environments* (2009), pp. 27–47.

[109] Pasquale Lops et al. "ClayRS: An end-to-end framework for reproducible knowledge-aware recommender systems". In: *Information Systems* 119 (2023), p. 102273.

[110] Alexandra Sasha Luccioni and Alex Hernandez-Garcia. "Counting carbon: A survey of factors influencing the emissions of machine learning". In: *arXiv preprint arXiv:2302.08476* (2023).

[111] Alexandra Sasha Luccioni, Sylvain Viguier, and Anne-Laure Ligozat. "Estimating the Carbon Footprint of BLOOM, a 176B Parameter Language Model". In: *Journal of Machine Learning Research* 24.253 (2023), pp. 1–15. URL: `http://jmlr.org/papers/v24/23-0069.html`.

[112] Qiyao Ma, Xubin Ren, and Chao Huang. "Xrec: Large language models for explainable recommendation". In: *arXiv preprint arXiv:2406.02377* (2024).

[113] Daniele Malitesta et al. "Do We Really Need to Drop Items with Missing Modalities in Multimodal Recommendation?" In: *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*. 2024, pp. 3943–3948.

[114] Daniele Malitesta et al. "Formalizing Multimedia Recommendation through Multimodal Deep Learning". In: *CoRR* abs/2309.05273 (2023).

[115] Kelong Mao et al. "UltraGCN: ultra simplification of graph convolutional networks for recommendation". In: *Proceedings of the 30th ACM international conference on information & knowledge management*. 2021, pp. 1253–1262.

[116] Eric Masanet et al. "Recalibrating global data center energy-use estimates". In: *Science* 367.6481 (2020), pp. 984–986.

[117] Tomas Mikolov et al. "Distributed representations of words and phrases and their compositionality". In: *Advances in neural information processing systems*. 2013, pp. 3111–3119.

[118] Tomas Mikolov et al. "Efficient estimation of word representations in vector space". In: *arXiv preprint arXiv:1301.3781* (2013).

[119] Ranjan Mishra et al. "Reproducibility Study of" XRec: Large Language Models for Explainable Recommendation"". In: *arXiv preprint arXiv:2510.06275* (2025).

[120] Cataldo Musto, Giuseppe Spillo, Giovanni Semeraro, et al. "Harnessing distributional semantics to build context-aware justifications for recommender systems". In: *User Modeling and User-Adapted Interaction* (2023).

[121] Cataldo Musto et al. "Context-aware graph-based recommendations exploiting Personalized PageRank". In: *Knowledge-Based Systems* 216 (2021), p. 106806.

[122] Cataldo Musto et al. "Deep content-based recommender systems exploiting recurrent neural networks and linked open data". In: *Adjunct Publication of the 26th conference on user modeling, adaptation and personalization*. 2018, pp. 239–244.

[123] Cataldo Musto et al. "Learning word embeddings from wikipedia for content-based recommender systems". In: *European conference on information retrieval*. Springer. 2016, pp. 729–734.

[124] Cataldo Musto et al. "Semantics and content-based recommendations". In: *Recommender systems handbook*. Springer, 2022, pp. 251–298.

[125] Cataldo Musto et al. "Semantics-aware Recommender Systems exploiting Linked Open Data and graph-based features". In: *Knowledge-Based Systems* 136 (2017), pp. 1–14.

[126] Jianmo Ni, Jiacheng Li, and Julian McAuley. "Justifying recommendations using distantly-labeled reviews and fine-grained aspects". In: *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*. 2019, pp. 188–197.

[127] Xia Ning and George Karypis. "Slim: Sparse linear methods for top-n recommender systems". In: *2011 IEEE 11th International Conference on Data Mining*. IEEE. 2011, pp. 497–506.

[128] Enrico Palumbo et al. "Translational models for item recommendation". In: *European Semantic Web Conference*. Springer. 2018, pp. 478–490.

[129] Deepak Kumar Panda and Sanjog Ray. "Approaches and algorithms to mitigate cold start problems in recommender systems: a systematic literature review". In: *Journal of Intelligent Information Systems* 59.2 (2022), pp. 341–366.
DOI: `10.1007/S10844-022-00698-5`. URL: `https://doi.org/10.1007/s10844-022-00698-5`.

[130] Divya Pandey, Madhoolika Agrawal, and Jai Shanker Pandey. “Carbon footprint: current methods of estimation”. In: *Environmental monitoring and assessment* 178 (2011), pp. 135–160.

[131] David Patterson et al. “The carbon footprint of machine learning training will plateau, then shrink”. In: *Computer* 55.7 (2022), pp. 18–28.

[132] Aleksandr V Petrov and Craig Macdonald. “RecJPQ: training large-catalogue sequential recommenders”. In: *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*. 2024, pp. 538–547.

[133] Alessandro Petruzzelli et al. “Improving transformer-based sequential conversational recommendations through knowledge graph embeddings”. In: *Proceedings of the 32nd ACM Conference on User Modeling, Adaptation and Personalization*. 2024, pp. 172–182.

[134] Guangyuan Piao and John G Breslin. “A study of the similarities of entity embeddings learned from different aspects of a knowledge base for item recommendations”. In: *The Semantic Web: ESWC 2018 Satellite Events: ESWC 2018 Satellite Events, Heraklion, Crete, Greece, June 3-7, 2018, Revised Selected Papers 15*. Springer. 2018, pp. 345–359.

[135] Guangyuan Piao and John G Breslin. “Transfer learning for item recommendations and knowledge graph completion in item related domains via a co-factorization model”. In: *The Semantic Web: 15th International Conference, ESWC 2018, Heraklion, Crete, Greece, June 3–7, 2018, Proceedings 15*. Springer. 2018, pp. 496–511.

[136] Marco Polignano et al. “Together is Better: Hybrid Recommendations Combining Graph Embeddings and Contextualized Word Representations”. In: *Fifteenth ACM Conference on Recommender Systems*. 2021, pp. 187–198.

[137] Pearl Pu, Li Chen, and Rong Hu. “A user-centric evaluation framework for recommender systems”. In: *Proceedings of the fifth ACM conference on Recommender systems*. 2011, pp. 157–164.

[138] Michael Pulis and Josef Bajada. “Siamese Neural Networks for Content-based Cold-Start Music Recommendation.” In: *Proceedings of the 15th ACM conference on recommender systems*. 2021, pp. 719–723.

[139] Sanjay Purushotham, Yan Liu, and C-C Jay Kuo. “Collaborative topic regression with social matrix factorization for recommendation systems”. In: *Proceedings of the 29th International Coference on International Conference on Machine Learning*. 2012, pp. 691–698.

[140] Ben Purvis, Yong Mao, and Darren Robinson. “Three pillars of sustainability: in search of conceptual origins”. In: *Sustainability science* 14 (2019), pp. 681–695.

[141] Alec Radford et al. "Learning transferable visual models from natural language supervision". In: *International conference on machine learning*. PmLR. 2021, pp. 8748–8763.

[142] Alec Radford et al. "Robust speech recognition via large-scale weak supervision". In: *International Conference on Machine Learning*. PMLR. 2023, pp. 28492–28518.

[143] Shashank Rajput et al. "Recommender systems with generative retrieval". In: *Advances in Neural Information Processing Systems* 36 (2023), pp. 10299–10315.

[144] Muhammad Habib ur Rehman et al. "Big data reduction framework for value creation in sustainable enterprises". In: *International journal of information management* 36.6 (2016), pp. 917–928.

[145] Nils Reimers and Iryna Gurevych. "Sentence-bert: Sentence embeddings using siamese bert-networks". In: *arXiv preprint arXiv:1908.10084* (2019).

[146] Steffen Rendle et al. "BPR: Bayesian Personalized Ranking from Implicit Feedback". In: *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*. UAI '09. Montreal, Quebec, Canada: AUAI Press, 2009, 452–461.
ISBN: 9780974903958.

[147] Paul Resnick and Hal R Varian. "Recommender systems". In: *Communications of the ACM* 40.3 (1997), pp. 56–58.

[148] Francesco Ricci, Lior Rokach, and Bracha Shapira. "Recommender Systems: Techniques, Applications, and Challenges". In: *Recommender Systems Handbook* (2022), pp. 1–35.

[149] Tim Rocktäschel, Sameer Singh, and Sebastian Riedel. "Injecting logical background knowledge into embeddings for relation extraction". In: *Proceedings of the 2015 conference of the north American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 2015, pp. 1119–1129.

[150] Tim Rocktäschel et al. "Low-dimensional embeddings of logic". In: *Proceedings of the ACL 2014 workshop on semantic parsing*. 2014, pp. 45–49.

[151] Gerard Salton, Anita Wong, and Chung-Shu Yang. "A vector space model for automatic indexing". In: *Communications of the ACM* 18.11 (1975), pp. 613–620.

[152] Kumari Samridhi, Bam Bahadur Sinha, and Trinanjan Das. "Optimizing User Recommendations with Variational Autoencoders: Insights from MovieLens-1M and BookCrossing Datasets". In: *2024 IEEE International Conference on Computer Vision and Machine Intelligence (CVMI)*. IEEE. 2024, pp. 1–6.

[153] Md Kamruzzaman Sarker et al. “Neuro-symbolic artificial intelligence”. In: *AI Communications* Preprint (2021), pp. 1–13.

[154] Badrul Sarwar et al. “Incremental singular value decomposition algorithms for highly scalable recommender systems”. In: *Fifth international conference on computer and information science*. Vol. 1. 012002. 2002, pp. 27–8.

[155] Roy Schwartz et al. “Green ai”. In: *Communications of the ACM* 63.12 (2020), pp. 54–63.

[156] Suvash Sedhain et al. “Autorec: Autoencoders meet collaborative filtering”. In: *Proceedings of the 24th international conference on World Wide Web*. 2015, pp. 111–112.

[157] Raghavendra Selvan et al. “Carbon footprint of selecting and training deep learning models for medical image analysis”. In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer. 2022, pp. 506–516.

[158] Baoxu Shi and Tim Weninger. “Open-world knowledge graph completion”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 32. 1. 2018.

[159] Philip Simon, Peter Bamidele Shola, and Ovye John Abari. “Application of content-based approach in research paper recommendation system for a digital library”. In: *International Journal of Advanced Computer Science and Applications* 5.10 (2014).

[160] Karen Simonyan and Andrew Zisserman. “Very deep convolutional networks for large-scale image recognition”. In: *arXiv preprint arXiv:1409.1556* (2014).

[161] Raymond M Smullyan. *First-order logic*. Courier Corporation, 1995.

[162] Kaitao Song et al. “MPNet: Masked and Permuted Pre-training for Language Understanding”. In: *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*. Ed. by Hugo Larochelle et al. 2020.

[163] Weiping Song et al. “Explainable Knowledge Graph-based Recommendation via Deep Reinforcement Learning”. In: *arXiv preprint arXiv:1906.09506* (2019).

[164] Giuseppe Spillo. “Combining Heterogeneous Embeddings for Knowledge-Aware Recommendation Models”. In: *Proceedings of the 31st ACM Conference on User Modeling, Adaptation and Personalization*. 2023, pp. 269–273.

[165] Giuseppe Spillo. “Knowledge-Aware Recommender Systems based on Multi-Modal Information Sources”. In: *Proceedings of the 17th ACM Conference on Recommender Systems*. 2023, pp. 1312–1317.

[166] Giuseppe Spillo et al. “Balancing carbon footprint and algorithm performance in recommender systems: A comprehensive benchmark”. In: *Sustainable Computing: Informatics and Systems* (2025), p. 101286.

[167] Giuseppe Spillo et al. “Combining graph neural networks and sentence encoders for knowledge-aware recommendations”. In: *Proceedings of the 31st ACM Conference on User Modeling, Adaptation and Personalization*. 2023, pp. 1–12.

[168] Giuseppe Spillo et al. “Comparing data reduction strategies for energy-efficient green recommender systems”. In: *Journal of Intelligent Information Systems* (2025), pp. 1–27.

[169] Giuseppe Spillo et al. “Evaluating Content-based Pre-Training Strategies for a Knowledge-aware Recommender System based on Graph Neural Networks”. In: *Proceedings of the 32nd ACM Conference on User Modeling, Adaptation and Personalization*. 2024, pp. 165–171.

[170] Giuseppe Spillo et al. “Exploiting Neuro-Symbolic Graph Embeddings based on First-Order Logical Rules for Knowledge-aware Recommendations.” In: *DP@ AI* IA*. 2022, pp. 1–11.

[171] Giuseppe Spillo et al. “GAL-KARS: Exploiting LLMs for Graph Augmentation in Knowledge-Aware Recommender Systems”. In: *Proceedings of the 33rd ACM Conference on User Modeling, Adaptation and Personalization*. 2025, pp. 73–82.

[172] Giuseppe Spillo et al. “Graph Augmentation with LLMs for Knowledge-Aware Recommender Systems”. In: *Proceedings of the 5th Italian Information Retrieval Workshop*. Vol. 4026. CEUR Workshop Proceedings. CEUR-WS.org, 2025.

[173] Giuseppe Spillo et al. “Knowledge-aware Recommendations Based on Neuro-Symbolic Graph Embeddings and First-Order Logical Rules”. In: *Proceedings of the 16th ACM Conference on Recommender Systems*. 2022, pp. 616–621.

[174] Giuseppe Spillo et al. “Recommender systems based on neuro-symbolic knowledge graph embeddings encoding first-order logic rules”. In: *User Modeling and User-Adapted Interaction* 34.5 (2024), pp. 2039–2083.

[175] Giuseppe Spillo et al. “Recsys CarbonAtor: Predicting carbon footprint of recommendation system models”. In: *International Workshop on Recommender Systems for Sustainability and Social Good*. Springer. 2024, pp. 98–110.

[176] Giuseppe Spillo et al. “See the Movie, Hear the Song, Read the Book: Extending MovieLens-1M, Last. fm-2K, and DBbook with Multimodal Data”. In: *Proceedings of the Nineteenth ACM Conference on Recommender Systems*. 2025, pp. 847–856.

[177] Giuseppe Spillo et al. “*Gotta Embed Them All!* - Knowledge-aware Recommendations Fusing Heterogeneous Multi-Modal Item Embeddings”. In: *Journal of Intelligent Information Systems* (2025), pp. 1–27.

[178] Giuseppe Spillo et al. “Towards Green Recommender Systems: Investigating the Impact of Data Reduction on Carbon Footprint and Model Performances”. In: *Proceedings of the 18th ACM Conference on Recommender Systems*. 2024.

[179] Giuseppe Spillo et al. “Towards sustainability-aware recommender systems: analyzing the trade-off between algorithms performance and carbon footprint”. In: *Proceedings of the 17th ACM Conference on Recommender Systems*. 2023, pp. 856–862.

[180] Giuseppe Spillo et al. “Training Green and Sustainable Recommendation Models: Introducing Carbon Footprint Data into Early Stopping Criteria”. In: *Proceedings of the 33rd ACM Conference on User Modeling, Adaptation and Personalization*. 2025, pp. 341–346.

[181] Giuseppe Spillo et al. “URecJPQ: Memory-efficient Multimodal Recommendation Models through RecJPQ in Large-Scale Scenarios”. In: *Submitted to ACM International Conference on User Modeling, Adaptation and Personalization (UMAP). 2026*. 2026.

[182] Dimitrios Stamoulis et al. “Hyperpower: Power-and memory-constrained hyper-parameter optimization for neural networks”. In: *2018 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE. 2018, pp. 19–24.

[183] Harald Steck. “Embarrassingly Shallow Autoencoders for Sparse Data”. In: *The World Wide Web Conference*. San Francisco, CA, USA: ACM, 2019, 3251–3257.
ISBN: 9781450366748.

[184] Matteo Stefanini et al. “A novel attention-based aggregation function to combine vision and language”. In: *2020 25th International Conference on Pattern Recognition (ICPR)*. IEEE. 2021, pp. 1212–1219.

[185] Emma Strubell, Ananya Ganesh, and Andrew McCallum. “Energy and policy considerations for deep learning in NLP”. In: *arXiv preprint arXiv:1906.02243* (2019).

[186] Alessandro Suglia et al. “A deep architecture for content-based recommendations exploiting recurrent neural networks”. In: *Proceedings of the 25th conference on user modeling, adaptation and personalization*. 2017, pp. 202–211.

[187] Fei Sun et al. “BERT4Rec: Sequential recommendation with bidirectional encoder representations from transformer”. In: *Proceedings of the 28th ACM international conference on information and knowledge management*. 2019, pp. 1441–1450.

[188] Zhu Sun et al. “Recurrent knowledge graph embedding for effective recommendation”. In: *Proceedings of the 12th ACM Conference on Recommender Systems*. 2018, pp. 297–305.

[189] Jian Tang et al. “Line: Large-scale information network embedding”. In: *Proceedings of the 24th international conference on world wide web*. 2015, pp. 1067–1077.

[190] Zhulin Tao et al. “Self-supervised learning for multimedia recommendation”. In: *IEEE Transactions on Multimedia* 25 (2022), pp. 5107–5116.

[191] Yi Tay et al. “Transformer memory as a differentiable search index”. In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 21831–21843.

[192] Peng Tianduo, Ou Xunmin, et al. “Scenario analysis on CO2-equivalent emissions from alternative mobile air conditioning refrigerants in China”. In: *Energy Procedia* 142 (2017), pp. 2617–2623.

[193] Hongzhi Tong, Di-Rong Chen, and Lizhong Peng. “Analysis of support vector machines regression”. In: *Foundations of Computational Mathematics* 9.2 (2009), pp. 243–257.

[194] Hugo Touvron et al. “Llama: Open and efficient foundation language models”. In: *arXiv preprint arXiv:2302.13971* (2023).

[195] Du Tran et al. “A closer look at spatiotemporal convolutions for action recognition”. In: *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*. 2018, pp. 6450–6459.

[196] Du Tran et al. “Learning spatiotemporal features with 3d convolutional networks”. In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 4489–4497.

[197] Théo Trouillon et al. “Complex embeddings for simple link prediction”. In: *International conference on machine learning*. PMLR. 2016, pp. 2071–2080.

[198] Quoc-Tuan Truong, Aghiles Salah, and Hady Lauw. “Multi-Modal Recommender Systems: Hands-On Exploration”. In: *Proceedings of the 15th ACM Conference on Recommender Systems*. RecSys ’21. Amsterdam, Netherlands: Association for Computing Machinery, 2021, 834–837.

ISBN: 9781450384582. URL: https://doi.org/10.1145/3460231.3473324.

[199] Mueen Uddin et al. “Evaluating power efficient algorithms for efficiency and carbon emissions in cloud data centers: A review”. In: *Renewable and Sustainable Energy Reviews* 51 (2015), pp. 1553–1563.

[200] Om Uparkar et al. “Vision Transformer Outperforms Deep Convolutional Neural Network-based Model in Classifying X-ray Images”. In: *Procedia Computer Science* 218 (2023). International Conference on Machine Learning and Data Engineering, pp. 2338–2349.
ISSN: 1877-0509. URL: https://www.sciencedirect.com/science/article/pii/S1877050923002090.

[201] Aimee Van Wynsberghe. “Sustainable AI: AI for sustainability and the sustainability of AI”. In: *AI and Ethics* 1.3 (2021), pp. 213–218.

[202] Shikhar Vashishth et al. “Composition-based multi-relational graph convolutional networks”. In: *arXiv preprint arXiv:1911.03082* (2019).

[203] Ashish Vaswani et al. “Attention is All you Need”. In: *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*. Ed. by Isabelle Guyon et al. 2017, pp. 5998–6008.

[204] Tobias Vente et al. “From Clicks to Carbon: The Environmental Toll of Recommender Systems”. In: *Proceedings of the 18th ACM Conference on Recommender Systems*. RecSys ’24. Bari, Italy: Association for Computing Machinery, 2024, 580–590.
ISBN: 9798400705052. URL: https://doi.org/10.1145/3640457.3688074.

[205] Roberto Verdecchia, June Sallou, and Luís Cruz. “A systematic review of Green AI”. In: *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* (2023), e1507.

[206] Alessandro Vicenti et al. “Estimating Product Carbon Footprint via Large Language Models for Sustainable Recommender Systems”. In: *Second International Workshop on Recommender Systems for Sustainability and Social Good. In press.* 2025.

[207] Aristidis G Vrahatis, Konstantinos Lazaros, and Sotiris Kotsiantis. “Graph attention networks: a comprehensive review of methods and applications”. In: *Future Internet* 16.9 (2024), p. 318.

[208] Denny Vrandečić and Markus Krötzsch. “Wikidata: a free collaborative knowledge base”. In: *Communications of the ACM* 57.10 (2014). Publisher: ACM New York, NY, USA, pp. 78–85.

[209] Sanne Vrijenhoek. “Do you MIND? Reflections on the MIND dataset for research on diversity in news recommendations”. In: *International Workshop on Algorithmic Bias in Search and Recommendation*. Springer. 2023, pp. 147–154.

[210] Hongwei Wang, Fuzheng Zhang, et al. “Knowledge-aware Graph Neural Networks with Label Smoothness Regularization for Recommender Systems”. In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. KDD ’19. Anchorage, AK, USA: ACM, 2019, 968–977.
ISBN: 9781450362016.

[211] Hongwei Wang et al. “Knowledge Graph Convolutional Networks for Recommender Systems”. In: *The World Wide Web Conference*. WWW ’19. San Francisco, CA, USA: Association for Computing Machinery, 2019, 3307–3313.
ISBN: 9781450366748. URL: `https://doi.org/10.1145/3308558.3313417`.

[212] Hongwei Wang et al. “Multi-Task Feature Learning for Knowledge Graph Enhanced Recommendation”. In: *The World Wide Web Conference*. WWW ’19. San Francisco, CA, USA: ACM, 2019, 2000–2010.
ISBN: 9781450366748.

[213] Hongwei Wang et al. “RippleNet: Propagating User Preferences on the Knowledge Graph for Recommender Systems”. In: *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. CIKM ’18. Torino, Italy: Association for Computing Machinery, 2018, 417–426.
ISBN: 9781450360142. URL: `https://doi.org/10.1145/3269206.3271739`.

[214] Jinglong Wang et al. “Metric learning with adversarial hard negative samples for tag recommendation”. In: *The Journal of Supercomputing* 80.14 (2024), pp. 21475–21507.

[215] Meihong Wang, Linling Qiu, and Xiaoli Wang. “A Survey on Knowledge Graph Embeddings for Link Prediction”. In: *Symmetry* 13.3 (2021).
ISSN: 2073-8994. URL: `https://www.mdpi.com/2073-8994/13/3/485`.

[216] Wenhui Wang et al. “Image as a foreign language: Beit pretraining for vision and vision-language tasks”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 19175–19186.

[217] Wenhui Wang et al. “Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers”. In: *Advances in neural information processing systems* 33 (2020), pp. 5776–5788.

[218] Xiang Wang et al. “Disentangled Graph Collaborative Filtering”. In: *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR ’20. Virtual Event, China: Association for Computing Machinery, 2020, 1001–1010. ISBN: 9781450380164. URL: `https://doi.org/10.1145/3397271.3401137`.

[219] Xiang Wang et al. “KGAT: Knowledge graph attention network for recommendation”. In: *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*. 2019, pp. 950–958.

[220] Xiang Wang et al. “Learning intents behind interactions with knowledge graph for recommendation”. In: *Proceedings of the web conference 2021*. 2021, pp. 878–887.

[221] Xiang Wang et al. “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, 165–174. ISBN: 9781450361729. URL: `https://doi.org/10.1145/3331184.3331267`.

[222] Yufeng Wang et al. “ADCB: adaptive dynamic clustering of bandits for online recommendation system”. In: *Neural Processing Letters* 55.2 (2023), pp. 1155–1172.

[223] Zhen Wang et al. “Knowledge Graph Embedding by Translating on Hyperplanes”. In: *Proceedings of the AAAI Conference on Artificial Intelligence* 28.1 (2014).

[224] Lukas Wegmeth et al. “Emers: Energy meter for recommender systems”. In: *arXiv preprint arXiv:2409.15060* (2024).

[225] Yinwei Wei et al. “Graph-refined convolutional network for multimedia recommendation with implicit feedback”. In: *Proceedings of the 28th ACM international conference on multimedia*. 2020, pp. 3541–3549.

[226] Yinwei Wei et al. “MMGCN: Multi-modal graph convolution network for personalized recommendation of micro-video”. In: *Proceedings of the 27th ACM international conference on multimedia*. 2019, pp. 1437–1445.

[227] Arif Wijonarko, Dade Nurjanah, and Dana Sulistyo Kusumo. “Hybrid recommender system using random walk with restart for social tagging system”. In: *2017 International Conference on Data and Software Engineering (ICoDSE)*. IEEE. 2017, pp. 1–6.

[228] Carole-Jean Wu et al. “Sustainable ai: Environmental implications, challenges and opportunities”. In: *Proceedings of Machine Learning and Systems* 4 (2022), pp. 795–813.

[229] Fangzhao Wu et al. “Mind: A large-scale dataset for news recommendation”. In: *Proceedings of the 58th annual meeting of the association for computational linguistics*. 2020, pp. 3597–3606.

[230] Jiancan Wu et al. “Self-supervised Graph Learning for Recommendation”. In: *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR ’21. Virtual Event, Canada: Association for Computing Machinery, 2021, 726–735.

[231] Jiayang Wu et al. “Multimodal Large Language Models: A Survey”. In: *2023 IEEE International Conference on Big Data (BigData)*. IEEE. 2023, pp. 2247–2256.
DOI: `https://doi.org/10.1109/BigData59044.2023.10386743`.

[232] Shiwen Wu et al. “Graph neural networks in recommender systems: a survey”. In: *ACM Computing Surveys* 55.5 (2022), pp. 1–37.

[233] Zonghan Wu et al. “A comprehensive survey on graph neural networks”. In: *IEEE transactions on neural networks and learning systems* 32.1 (2020), pp. 4–24.

[234] Lanling Xu et al. “Towards a More User-Friendly and Easy-to-Use Benchmark Library for Recommender Systems”. In: 2023, 2837–2847.

[235] Hong-Jian Xue et al. “Deep Matrix Factorization Models for Recommender Systems”. In: *Proceedings of the 26th International Joint Conference on Artificial Intelligence*. IJCAI’17. Melbourne, Australia: AAAI Press, 2017, 3203–3209.

[236] Ikuya Yamada et al. “Wikipedia2Vec: An efficient toolkit for learning and visualizing the embeddings of words and entities from Wikipedia”. In: *arXiv preprint arXiv:1812.06280* (2018).

[237] Bishan Yang et al. “Embedding entities and relations for learning and inference in knowledge bases”. In: *arXiv preprint arXiv:1412.6575* (2014).

[238] Zixuan Yi et al. “Enhancing recommender systems: Deep modality alignment with large multi-modal encoders”. In: *ACM Transactions on Recommender Systems* 3.4 (2025), pp. 1–25.

[239] Zixuan Yi et al. “Enhancing recommender systems: Deep modality alignment with large multi-modal encoders”. In: *ACM Transactions on Recommender Systems* 3.4 (2025), pp. 1–25.

[240] Zixuan Yi et al. "Multi-modal graph contrastive learning for micro-video recommendation". In: *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2022, pp. 1807–1811.

[241] Zixuan Yi et al. "Multi-modal graph contrastive learning for micro-video recommendation". In: *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2022, pp. 1807–1811.

[242] Neil Zeghidour et al. "Soundstream: An end-to-end neural audio codec". In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 30 (2021), pp. 495–507.

[243] Jingtao Zhan et al. "Jointly optimizing query encoder and product quantization to improve retrieval performance". In: *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 2021, pp. 2487–2496.

[244] Chunhong Zhang et al. "Knowledge graph embedding for hyper-relational data". In: *Tsinghua Science and Technology* 22.2 (2017), pp. 185–197.

[245] Chunxu Zhang et al. "Dual personalization on federated recommendation". In: *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence*. 2023, pp. 4558–4566.

[246] Fuzheng Zhang et al. "Collaborative Knowledge Base Embedding for Recommender Systems". In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '16. San Francisco, California, USA: Association for Computing Machinery, 2016, 353–362.

[247] Jinghao Zhang et al. "Mining latent structures for multimedia recommendation". In: *Proceedings of the 29th ACM international conference on multimedia*. 2021, pp. 3872–3880.

[248] Jinghao Zhang et al. "Mining latent structures for multimedia recommendation". In: *Proceedings of the 29th ACM international conference on multimedia*. 2021, pp. 3872–3880.

[249] Si Zhang et al. "Graph convolutional networks: a comprehensive review". In: *Computational Social Networks* 6.1 (2019), pp. 1–23.

[250] Yongfeng Zhang, Xu Chen, et al. "Explainable recommendation: A survey and new perspectives". In: *Foundations and Trends® in Information Retrieval* 14.1 (2020), pp. 1–101.

[251] Yongfeng Zhang et al. "Learning over knowledge-base embeddings for recommendation". In: *arXiv preprint arXiv:1803.06540* (2018).

[252] Yue Zhang et al. "Siren's song in the AI ocean: a survey on hallucination in large language models". In: *arXiv preprint arXiv:2309.01219* (2023).

[253] Zhenyu Zhang et al. "KRAN: Knowledge refining attention network for recommendation". In: *ACM Transactions on Knowledge Discovery from Data (TKDD)* 16.2 (2021), pp. 1–20.

[254] Wayne Xin Zhao et al. "RecBole 2.0: Towards a More Up-to-Date Recommendation Library". In: *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*. 2022, pp. 4722–4726.

[255] Lei Zheng et al. "Spectral collaborative filtering". In: *Proceedings of the 12th ACM conference on recommender systems*. 2018, pp. 311–319.

[256] Hongyu Zhou et al. "A comprehensive survey on multimodal recommender systems: Taxonomy, evaluation, and future directions". In: *arXiv preprint arXiv:2302.04473* (2023).

[257] Xin Zhou. "Mmrec: Simplifying multimodal recommendation". In: *Proceedings of the 5th ACM International Conference on Multimedia in Asia Workshops*. 2023, pp. 1–2.

[258] Xin Zhou and Zhiqi Shen. "A tale of two graphs: Freezing and denoising graph structures for multimodal recommendation". In: *Proceedings of the 31st ACM International Conference on Multimedia*. 2023, pp. 935–943.

[259] Xin Zhou et al. "Bootstrap latent representations for multi-modal recommendation". In: *Proceedings of the ACM web conference 2023*. 2023, pp. 845–854.

[260] Xin Zhou et al. "LessLeak-Bench: A First Investigation of Data Leakage in LLMs Across 83 Software Engineering Benchmarks". In: *arXiv preprint arXiv:2502.06215* (2025).

[261] Yuqi Zhu et al. "LLMs for knowledge graph construction and reasoning: Recent capabilities and future opportunities". In: *World Wide Web* 27.5 (2024), p. 58.