

ENHANCING GRAPH NEURAL NETWORK BASED SEQUENTIAL
RECOMMENDER MODEL WITH KNOWLEDGE GRAPH EMBEDDINGS

by

Kazım Emre Yüksel

B.S., Mathematics Engineering, Istanbul Technical University, 2021

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Computer Engineering
Boğaziçi University

2025

ACKNOWLEDGEMENTS

I would like to start by thanking my thesis advisor, Assist. Prof. Susan Michele Üsküdarlı, whose guidance and encouragement have made this journey both insightful and rewarding. Her thoughtful feedback and the support she offered to me at every stage have been truly inspiring. I am grateful for the patience and knowledge she shared throughout this process.

To my family and loved ones, thank you for always being there for me. Your support, kindness, and belief in me gave me the strength to keep moving forward. Knowing I had you by my side made even the most challenging moments more manageable, and this accomplishment would not have been possible without your encouragement.

I'm also deeply grateful to my colleagues and the Recommendation team at Getir. Working with you has been a fantastic experience. The discussions we had and the ideas you shared pushed me to think more creatively and improved this work in countless ways. Your collaboration and encouragement made this journey much more rewarding.

ABSTRACT

ENHANCING GRAPH NEURAL NETWORK BASED SEQUENTIAL RECOMMENDER MODEL WITH KNOWLEDGE GRAPH EMBEDDINGS

The growth of online interactions and user-generated data has presented new challenges and opportunities for personalized recommendation systems. Traditional recommendation approaches rely on static user-item attributes and fail to account for temporal dynamics, limiting their ability to adapt to evolving preferences. Sequential recommendation models address the challenge of next-item prediction by learning from action sequences; however, their performance is limited by the lack of large, diverse, and up-to-date data. This thesis explores the integration of open-source knowledge through knowledge graphs, in the form of knowledge graph embeddings (KGEs), into graph neural network-based sequential recommendation model to enhance both semantic and sequential patterns captured within user-item interactions. By leveraging embeddings from open-linked data sources such as Wikidata, we enrich item attributes, broadening the feature set and providing a more comprehensive understanding of user preferences. Using the Movielens dataset as a foundational benchmark, four translation-based KGE models are individually incorporated within a sequential model to evaluate their impact on recommendation accuracy. Offline evaluations with Hit@K and NDCG@K metrics reveal that this approach improves the relevance and personalization of recommendations by effectively capturing both semantic context and sequential behaviors. The findings underscore the value of incorporating open-source knowledge bases into recommendation systems, demonstrating that such enriched data sources yield more accurate and meaningful recommendations. This study demonstrates how external knowledge-enhanced user and item representations improve recommendation models.

ÖZET

BİLGİ ÇİZGE GÖMÜLERİYLE ÇİZGE SINIR AĞI TABANLI SIRALI ÖNERİ MODELİNİN GELİŞTİRİLMESİ

Çevrimiçi etkileşimlerin ve kullanıcı tarafından üretilen verilerin artışı, öneri sistemleri için yeni zorluklar ve fırsatlar sunmuştur. Geleneksel öneri yaklaşımları, statik kullanıcı-öğeler özelliklerine dayanır ve zamansal dinamikleri hesaba katamaz, bu da değişen tercihlere uyum sağlama yeteneklerini sınırlar. Sıralı öneri modelleri, eylem dizilerinden öğrenerek bir sonraki öğeyi tahmin etme sorununu ele alır; ancak, büyük, çeşitli ve güncel verilerin eksikliği performanslarını sınırlar. Bu tez, açık kaynaklı bilgiyi bilgi çizgeleri aracılığıyla, bilgi çizge gömülerini kullanarak çizge sınır ağı tabanlı sıralı öneri modellerine entegre etmeyi araştırmaktadır. Bu entegrasyon, kullanıcı-öğeler etkileşimlerinde hem anlamsal hem de sıralı desenlerin daha iyi yakalanmasını sağlamaktadır. Wikiveri gibi açık bağlantılı veri kaynaklarından elde edilen gömülerden yararlanarak, öğeler özellikleri zenginleştirilmekte, özellik seti genişletilmekte ve kullanıcı tercihlerini daha kapsamlı bir şekilde anlamak mümkün olmaktadır. Movielens veri seti temel referans olarak kullanılarak, dört çeviri tabanlı bilgi çizge gömü modeli sıralı öneri modeline ayrı ayrı entegre edilmiş ve öneri doğruluğuna etkileri değerlendirilmiştir. Hit@K ve NDCG@K metrikleriyle yapılan çevrimdışı değerlendirmeler, bu yaklaşımın hem anlamsal bağlamı hem de sıralı davranışları etkili bir şekilde yakalayarak önerilerin alaka düzeyini ve kişiselleştirmesini geliştirdiğini ortaya koymuştur. Bulgular, açık kaynaklı bilgi tabanlarının öneri sistemlerine dahil edilmesinin değerini vurgulamakta ve bu tür zenginleştirilmiş veri kaynaklarının daha doğru ve anlamlı öneriler sunduğunu göstermektedir. Bu çalışma, dışsal bilgiyle güçlendirilmiş kullanıcı ve öğeler temsillerinin öneri modellerini nasıl iyileştirdiğini göstermektedir.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS	iii
ABSTRACT	iv
ÖZET	v
LIST OF FIGURES	viii
LIST OF TABLES	xi
LIST OF ACRONYMS/ABBREVIATIONS	xiii
1. INTRODUCTION	1
2. RELATED WORK	9
2.1. Graph Neural Networks in Recommender Systems	9
2.2. Sequential Recommendations	11
2.3. Knowledge Graph Embeddings	13
2.4. Knowledge Graph Embedding Method in Recommender System	15
3. BACKGROUND	17
3.1. Knowledge Graph	17
3.1.1. Knowledge Graph Embedding Methods	18
3.1.1.1. TransE	18
3.1.1.2. TransH	19
3.1.1.3. TransR	19
3.1.1.4. TransD	20
3.2. Graph Neural Networks	21
3.2.1. Message Passing	22
3.2.2. Graph Convolution Networks	23
3.2.3. Graph Attention Networks	24
3.3. Semantic Technologies	26
3.3.1. Wikidata	26
3.3.2. Resource Description Framework (RDF)	26
3.3.3. SPARQL	28
4. METHODOLOGY	30

4.1. Problem Definition	30
4.2. Model Overview	31
4.3. Approach	33
4.3.1. Data Collection	33
4.3.2. Knowledge Graph Construction	34
4.3.3. Knowledge Graph Embedding	35
4.3.4. Sequential Recommendation Model	37
4.3.5. Our Model	39
5. IMPLEMENTATION	42
5.1. Data Processing	42
5.1.1. Obtaining Movie Informations	42
5.1.2. Fetching Wikidata Entities and Properties	44
5.2. Constructing Knowledge Graph	51
5.2.1. Analysis on Knowledge Graphs	52
5.3. Model Training	62
5.3.1. Knowledge Graph Embedding	62
5.3.2. Sequential Model Training	63
6. EXPERIMENTS and RESULTS	64
6.1. Datasets	64
6.2. Parameter Setting	66
6.3. Experiments Environment	66
6.4. Evaluation Metrics	66
6.4.1. Hit@K	66
6.4.2. Mean Reciprocal Rank (MRR)	67
6.5. Result and Analysis	68
6.5.1. Knowledge Graph Embedding Results	68
6.5.2. Sequential Recommendation Experiment Results	69
6.5.3. Qualitative Analysis	72
7. CONCLUSION & FUTURE WORK	75
REFERENCES	78
APPENDIX A: FIGURES USED IN THESIS	87

LIST OF FIGURES

Figure 3.1.	Illustrative example of the knowledge graph embedding process. This figure depicts a knowledge graph with entities and relationships, which are mapped into an embedding space and represented as vector form.	18
Figure 3.2.	An example of RDF Representation of the movie "Titanic".	27
Figure 3.3.	An example of SPARQL query.	28
Figure 4.1.	Overview of the architecture: ① illustrates the construction of the knowledge graph and the processing of knowledge graph embeddings, ② shows the node vector embeddings from a gated graph neural network, and ③ shows the concatenation of the embedding vectors and the prediction phase.	32
Figure 4.2.	An illustration comparing how TransE, TransH, TransR, and TransD compute distances between entity embeddings using shared or relation-specific spaces.	36
Figure 4.3.	A fragment of the knowledge graph corresponding to the triples in the enriched MovieLens dataset, where the nodes represent movie entities and the edges depict the relationships between them. Both nodes and edges are labeled with Wikidata entity and property identifiers.	37
Figure 4.4.	Overview of our proposed model.	40

Figure 5.1.	Algorithm for extracting title and release year from given movie titles in original dataset.	43
Figure 5.2.	SPARQL query for a movie titled 'Toy Story' that returns the name, description, and release year.	45
Figure 5.3.	Algorithm to retrieve properties for a given Wikidata entity QID.	47
Figure 5.4.	A fragment of the knowledge graph constructed for the MovieLens 100K dataset, where the degree of nodes is filtered between 100 and 1148.	55
Figure 5.5.	A fragment of the knowledge graph constructed for the MovieLens 1M dataset, where the degree of nodes is filtered between 160 and 3346.	56
Figure 5.6.	A fragment of the knowledge graph constructed for the MovieLens 20M dataset, where the degree of nodes is filtered between 200 and 17546.	57
Figure 5.7.	The visualization of the connection between movies and the Academy Award for Best Sound shows nodes related to movie titles that represent the most influential films deserving of awards for ml-100k dataset.	58
Figure 5.8.	The visualization of the connection between movies and the Academy Award for Best Sound shows nodes related to movie titles that represent the most influential films deserving of awards for ml-1m dataset.	59

Figure 6.1. Example of a recommendation test case: the first row shows the user’s last five movies, the second row displays recommendations from the baseline model, and the third row presents recommendations from one of the combined model versions. 72



LIST OF TABLES

Table 5.1.	An example for preprocessed movie data.	42
Table 5.2.	Preprocessed movie data with Wikidata enrichment. Columns include movie ID, title, genres, cleaned title, release year (given in Movielen), Wikidata entity name, and Wikidata QID.	46
Table 5.3.	Table showing the properties of movie entities fetched from Wikidata, represented as subject, predicate, and object with corresponding entity names.	48
Table 5.4.	Top Entities Across Datasets.	49
Table 5.5.	Top Predicates Across Datasets.	50
Table 5.6.	Comparison of number of nodes, number of edges, and average degree for each Knowledge Graph constructed for different size of MovieLens datasets.	52
Table 5.7.	The table provides a comparison of the total number of nodes, distinguishing between those representing movie entities and those retrieved from Wikidata as feature entities.	53
Table 5.8.	The table provides a comparison of the total unique number of nodes and edges including Wikidata entities after enrichment. . . .	54

Table 6.1.	The performance of knowledge graph embedding models on the ml-100K, ml-1M, and ml-20M datasets for the TransR, TransH, TransE, and TransD models was evaluated using Hit@5, Hit@10, and MRR metrics.	68
Table 6.2.	The performance of sequential recommendation models on the ml-100K, ml-1M, and ml-20M datasets for SRGNN and its combinations with TransR, TransH, TransE, and TransD was evaluated using Hit@10 and Hit@20 metrics.	70

LIST OF ACRONYMS/ABBREVIATIONS

2D	Two Dimensional
CNN	Convolutional Neural Network
GAT	Graph Attention Network
GCN	Graph Convolutional Network
GNN	Graph Neural Network
GRU	Gated Recurrent Unit
KG	Knowledge Graph
KGE	Knowledge Graph Embedding
LLM	Large Language Model
LOD	Linked Open Data
MRR	Mean Reciprocal Rank
QID	Wikidata Q Identifier
RDF	Resource Description Framework
RNN	Recurrent Neural Network
SPARQL	SPARQL Protocol and RDF Query Language
URI	Uniform Resource Identifier

1. INTRODUCTION

Recommendation systems are indispensable in today's digital landscape, crucial for enhancing user experience across a multitude of online platforms. These systems are key to boosting customer satisfaction and keeping users engaged, while also increasing profits for businesses. They provide personalized suggestions, from recommending products on shopping websites to improving content discovery on streaming platforms. These algorithms are expected to develop further as technology progresses, providing ever-more accurate and personalized recommendations to consumers' constantly-evolving requirements.

In the field of recommendation systems, foundational approaches such as collaborative filtering, content-based filtering, and hybrid methods form the main approaches. These methods leverage user behavior, item attributes, or a combination of both to deliver relevant and personalized recommendations, addressing diverse application needs across industries. Collaborative filtering utilizes user-item interactions to identify similarities between users or items, generating recommendations. Content-based filtering focuses on item attributes to identify items similar in content to user preferences, while hybrid systems combine both methods for more nuanced recommendations. While these methods are effective in providing static recommendations, they fall short in addressing temporal dynamics and the sequential nature of user interactions, which are critical for capturing evolving user preferences in dynamic contexts. For example, traditional methods might recommend items based solely on past behavior, failing to account for the order or timing of interactions, such as recommending a dessert immediately after a meal in food delivery systems. To address these limitations, sequential recommendation models have emerged as a solution, emphasizing the importance of understanding the sequence and context of user actions over time. These models are designed to capture evolving user preferences, enabling more accurate and contextually relevant recommendations in dynamic and fast-changing environments.

Sequential recommendation is a type of recommendation system approach that takes into account the sequentiality of user actions and the dynamics of user behavior. Unlike other recommendation methods that focus only on what users like now or how items are similar, sequential recommendation models look at the order of user actions (in other words sequence), like clicks, purchases, or ratings, to predict what user preferences [1]. Modeling preference and popularity dynamically rather than statically is important for developing recommendation system approaches through real-life problems because these are parameters that change over time. As the number of user-product interactions increase in sequences, it becomes crucial to capture higher-order sequential dependencies and learning long-term sequential dependencies. One of the most common method to model higher-order sequential dependencies is integrating user-level and item-level features in recommendation systems. While it provides more personalized recommendations based on individual preferences and diversity in recommendations, it also enhances the connections between item-item, user-item, and user-user relationships. The incorporation of features into recommender system algorithms is essential for accurately recording the interactions between users and items. These might include attributes specific to the person, such demographic data, past customer information, and item features like product prices and movie genres, among many more. Moreover, the incorporation of knowledge graphs enhances the feature representations by adding semantic links between items, improving recommendation quality, and deepening our understanding of user-item interactions.

Despite the advancements in sequential recommendation systems, significant challenges remain when these systems fail to deliver accurate, personalized, or contextually relevant suggestions. In scenarios where recommendation algorithms do not work effectively, users may feel frustrated due to irrelevant or repetitive suggestions, which can lower user engagement and satisfaction. Specifically, the inability to recommend complementary items in sequential contexts, such as suggesting a suitable accessory for a recently purchased product, weakens the core goal of these systems. This problem is even more noticeable in cases where personalization is key, as generic or poorly targeted recommendations may fail to reflect the user's changing preferences. While

existing research has made progress in context-aware recommender systems through improvements in architectural development and the incorporation of various features, the precise integration of dynamic, complementary, and personalized recommendations remains an ongoing area of research. Solving this problem is important because the success of sequential recommendation systems depends on their ability to adapt to different scenarios and user needs. Our research focuses on addressing this challenge by incorporating rich side features to improve the contextual and complementary nature of sequential recommendations, making them more relevant, accurate, and user-friendly.

One major obstacle to developing robust and enhanced recommendation systems is the lack of broad and diversified data sources. The effectiveness of a recommendation system rely on its ability to understand and predict user preferences, which can only be achieved with access to rich and varied data. Different types of data such as user interactions, preferences, and item related features are critical to creating personalized suggestions. Without a wide variety of data, these systems struggle to make relevant suggestions that appeal to a diverse group of users. Furthermore, access to current and accurate data is crucial, as good recommendations rely heavily on recent user interactions, preferences, and information. If a recommendation system is based on outdated or incomplete data, it may fail to capture shifts in user behavior, making its suggestions less relevant or timely. When an item catalog is limited or overly narrow, the system’s ability to offer diverse and meaningful recommendations is significantly reduced. This limitation can lead to a narrow selection of suggestions, which ultimately diminishes the effectiveness and appeal of the recommendation system. The second major challenge is the inability to effectively model and integrate side features, such as contextual information or item attributes, into the recommendation process. These side features often contain valuable insights that could improve recommendation quality, but many models fail to adequately capture or utilize them. Crafting useful representations of these features and integrating them into the recommendation model requires careful effort, as poorly integrated or ignored side information can result in suboptimal predictions. The lack of a robust framework for including such features reduces the model’s ability to make suitable predictions that reflect both user preferences and the broader

context of interactions. Especially in sequential recommendation approaches, this challenge becomes even more pronounced. Sequential recommendations rely on accurately capturing the evolving preferences of users based on their recent interactions. Without access to a diverse and up-to-date set of item features, the model may struggle to provide relevant suggestions at the item level. Additional information for items, such as their attributes or contextual details, can greatly improve the model’s ability. However, many models fail to effectively incorporate this item-level data, which reduces the accuracy and relevance of the recommendations.

One approach to overcoming these challenges and utilizing current, comprehensive data for recommendation systems is through Linked Open Data (LOD) [2]. Linked Open Data provides extensive, freely accessible, and up-to-date cross-domain information, which can be a game-changer for recommendation systems. Examples of such data sources include Freebase [3] and Wikidata [4], which offer vast amounts of structured knowledge that is publicly available and continuously updated. These resources are particularly valuable because they help create rich, interconnected knowledge graphs, which include complex semantic links between entities. These links enable the system to understand relationships between different items and users more deeply, enhancing its ability to make relevant recommendations.

Moreover, the use of open data in recommendation systems can provide better contextual understanding. For instance, LOD allows the system to incorporate information about the relationships between users, items, and categories, further improving the accuracy of recommendations. The integration of semantic resources, such as knowledge graphs, provides a more holistic view of items and their attributes, enabling systems to predict user preferences based on more than just past behavior but also on the deeper context and meaning of the items themselves. This leads to more personalized recommendations that better match users’ needs. Therefore, we propose that by incorporating such semantic resources into recommendation systems, we can significantly enhance the quality and accuracy of the generated embeddings of recommendation items. These embeddings can capture not only the basic characteristics of

the items but also the complex relationships and contexts in which they are used, leading to better, more personalized recommendations for users. In doing so, the system becomes not just a tool for suggesting items but a more intelligent system that learns and adapts to the evolving preferences of users in a dynamic, data-rich environment.

The main aim of the thesis is to integrate features by incorporating Knowledge Graph Embeddings into a Graph Neural Network-based Sequential Recommendation architecture to enrich the predictive capabilities of the model for next-item recommendations. This approach leverages the semantic relationships captured in the knowledge graph, obtained from Wikidata [4] as a knowledge base, by utilizing translation-based knowledge graph embedding methods. By concatenating these knowledge graph embedding vectors with the node vectors derived from the GNN [5], the model gains a comprehensive understanding of both the sequential dependencies and the semantic context of user-item interactions, thereby enhancing the accuracy and relevance of the recommendations provided.

In the proposed architecture, we aim to integrate open-source knowledge base data into the sequential recommendation model in the form of embedding vectors derived from a knowledge graph to enhance its predictive capabilities. The process begins by linking items in the dataset to entities in a knowledge graph. This requires identifying corresponding entities and extracting their relationships and attributes. Specifically, we use SPARQL queries to fetch Q-identifiers for each item based on its name and associated metadata. Once the entities are identified, we retrieve all first-order features for each item as Resource Description Framework (RDF) triples, capturing relationships such as directors, genres, or cast members. These triples are then used to construct a knowledge graph that provides a structured and semantically rich representation of the dataset. To incorporate the semantic relationships from the knowledge graph into our recommendation system, we apply various Knowledge Graph Embedding (KGE) methods to generate low-dimensional embedding vectors for each entity. These embeddings capture latent patterns and relational dependencies that are essential for enhancing recommendation performance. The embeddings are designed to

represent not just the entities themselves but also the semantic connections between them, enabling the model to leverage additional contextual information. For the recommendation model, we adopt the Session-based Recommendation with Graph Neural Networks (SRGNN) [5] framework, which is particularly effective in capturing sequential patterns in user behavior. The SRGNN model employs a gated Graph Neural Network (GNN) [6] to learn node representations from sequential data, capturing both local and global dependencies. To enhance the model’s capability further, we integrate the entity embeddings generated by the KGE methods with the node embeddings produced by the GNN. This concatenation of embeddings enables the model to benefit from both sequential patterns in user behavior and the semantic information derived from the knowledge graph. Finally, the effectiveness of this architecture is evaluated through both qualitative and quantitative analyses. We assess the impact of incorporating knowledge graph features on the performance of the sequential recommendation model, using the Movielens dataset as our primary testbed. By comparing the baseline SRGNN model with and without the integration of knowledge graph embeddings, we demonstrate the benefits of linking domain-specific datasets with open-source knowledge bases.

For the evaluation phase, we utilize the MovieLens dataset, comprising movie ratings provided by a set of users, along with detailed movie properties such as release date and genre [7]. The dataset used in this study consists of three sizes: one with 100,000 ratings, another with 1 million ratings, and a third with 20 million ratings, providing flexibility for experimentation at various scales. Notably, despite the availability of features within the dataset, our study aims to leverage Wikidata as an open data source for information enrichment. Therefore, we refrain from utilizing the features contained in the dataset, aligning with our objective of exploring external data sources for enhancing recommendation algorithms. The ratings are grouped on a user-level and sorted by timestamp to form sequential user-item interactions. The objective is to predict the next movie that a user might prefer based on their past interactions.

Our main contribution can be summarized as follows:

- The proposal of integrating external knowledge bases through embedding vectors into a graph-based architecture to capture richer, context-aware relationships between items and users, aiming to improve the accuracy of sequential recommendations.
- The demonstration of how integrating Wikidata, as an Open Linked Data resource, with knowledge graph embedding techniques can effectively enrich recommendation models. By leveraging the rich semantic relationships in this open-source knowledge base, the model can incorporate valuable contextual information, improving its understanding of user-item interactions. This approach highlights the effectiveness of external knowledge in representing the complex relationships between items and users through knowledge graphs, and shows that available Open Linked Data is both rich enough and accessible for enhancing recommendation systems.
- The demonstration of a prototype implementation of the proposed approach, evaluated using different sizes of the MovieLens dataset, showed significant improvements in recommendation accuracy and highlighted the relationship between the quality of feature embeddings and their impact on performance. The evaluation, based on both quantitative metrics such as Hit@K and NDCG@K and qualitative assessments, demonstrated the effectiveness of leveraging open-source data containing semantic relationships to enhance recommendation algorithms.

The structure of the thesis is outlined as follows: Chapter 2 provides a comprehensive review of related works on Knowledge Graph Embeddings and Sequential Recommendation methods, particularly their applications in recommender systems, with a focus on GNN-based approaches. Chapter 3 offers the necessary background information to enhance the reader’s understanding of the study. Chapter 4 explained the details of the proposed method in detail. Chapter 5 delves into the implementation details, outlining the steps taken to realize the proposed method. The experiments conducted and their results are presented in Chapter 6. Chapter 7 discusses the experiment

results, explores potential avenues for future research, and provides a comprehensive summary of the findings and conclusions drawn from the study.



2. RELATED WORK

The focus of the thesis work is on enhancing sequential recommendation systems by integrating knowledge graph embeddings. The base methods employed include the Graph Neural Network model and Knowledge Graph Embedding methods. Thus, the related work section primarily centers on studies concerning graph neural network-based recommendation system applications, specialized methods utilized for constructing sequential recommendation systems, and knowledge graph embedding techniques.

2.1. Graph Neural Networks in Recommender Systems

Learning useful user and item representations from their interactions and any relevant side information is the primary challenge faced by recommender systems. Essentially, a significant portion of the data within recommender systems inherently exhibits a graph structure, reflecting complex relationships between users, items, and their interactions. Therefore, Graph neural networks (GNNs) hold particular significance due to their ability in modeling graph data effectively, enabling the capture of complex dependencies and relationships among items and users, thereby enhancing graph representation learning [8, 9].

Initial studies in this field highlighted the success of Graph Convolutional Networks (GCNs) in enhancing recommender systems. NGCF [10] and PinSage [11] both leverage GCN to learn embeddings for users and items by propagating interaction information across graph layers. While NGCF (Neural Graph Collaborative Filtering) explicitly captures higher-order user-item interactions by stacking multiple GCN layers, thereby learning richer, multi-hop connectivity for improved recommendation quality, PinSage was specifically designed for large-scale content recommendation on platforms like Pinterest, where it efficiently handles billions of nodes by employing importance-based neighborhood sampling and scalable graph aggregation. On the other hand, LightGCN [12] simplifies this approach by eliminating feature transformation and non-

linear activation functions, focusing solely on graph-based neighborhood aggregation. This simplified design enhances efficiency and scalability, especially for large datasets, while maintaining high recommendation accuracy. Consequently, LightGCN has a significant advantage over NGCF, achieving comparable or even superior recommendation quality with considerably lower computational overhead.

Beyond the popular Graph Convolutional Networks (GCNs) used in recommender systems, other graph neural network architectures have also been explored. For instance, Graph Attention Networks (GATs) [13] incorporate attention mechanisms that assign importance to different nodes in a graph based on their features and relationships, allowing the model to focus on the most relevant interactions. Several graph attention-based models have been proposed to address common challenges in recommendation, such as data sparsity and cold start. One such model is the Graph Attention Collaborative Filtering (GACF) [14], which integrates collaborative filtering with knowledge graph information to capture more complex user-item relationships. Unlike traditional collaborative filtering models that rely solely on user-item interaction graphs, GACF incorporates knowledge graph context to enrich item representations and uses a user-specific attention mechanism to enhance personalization. By jointly learning both collaborative signals and knowledge graph context, GACF effectively models multi-hop neighbors along with relation-specific and user-specific information, resulting in more robust recommendations. Another notable model, the Attention-based Graph Convolutional Network (AGCN) [15], embeds both rating data and auxiliary information for users and items into a unified space. AGCN leverages Chebyshev polynomial graph filters to reduce computational complexity and uses attention layers to weigh neighbor information selectively, improving prediction accuracy and accelerating convergence. Experimental results show that both models achieve superior performance over state-of-the-art methods, demonstrating the effectiveness of incorporating attention and knowledge graph information in recommendation systems.

In industrial-scale recommender systems, graph-based approaches have been instrumental in addressing challenges of scalability, sparsity, and real-time responsive-

ness. Pixie [16], deployed at Pinterest, exemplifies a highly scalable, real-time recommender system that relies on a custom Pixie Random Walk algorithm over Pinterest’s extensive object graph of billions of nodes and edges. By leveraging real-time random walk computations, Pixie achieves significant improvements in user engagement. Meanwhile, Spotify introduced 2T-HGNN [17], a Heterogeneous Graph Neural Network [18] model designed to overcome the unique recommendation challenges posed by introducing audiobooks to its music and podcast-dominated platform. The 2T-HGNN model combines a Two Tower (2T) [19] approach with HGNNs, incorporating music and podcast preferences to provide more relevant audiobook recommendations, even under extreme data sparsity. PinSage [11], also deployed at Pinterest, introduces a graph convolutional approach for embedding generation, designed specifically for the web-scale demands of recommendation. Combining random walks and graph convolutions, PinSage captures both structural and feature information of items and is trained on a graph with billions of nodes and edges.

2.2. Sequential Recommendations

Before we start looking deeply into the literature, it is important to clarify two main points. Firstly, sequential recommendation and session-based recommendation are closely related terms, with session-based recommendation considered a subtype of sequential recommendation [1]. Generally, sequential recommendation focuses on longer-term patterns in user behavior, while session-based recommendation focuses on shorter-term interactions within a single session. The second point is about two relative concepts in sequential recommendation: defining next-item prediction and next-basket prediction. Next-item recommendation focuses on predicting the next individual object (e.g., product, movie, song, or category) a user is likely to interact with based on their past behavior, while next-basket recommendation extends this concept to predict entire sets or sequences of one or more objects that a user may be interested in [20].

The first versions of studies on sequential recommendations is based on the study of Markov Chains. In FPMC [21], the proposed method combines the matrix factoriza-

tion to model long term preferences and Markov chains to model sequential patterns by utilizing the transition graphs over underlying Markov Chains. In the past few years, there has been a noticeable increase in the number of research studies exploring deep learning methods in sequential recommendation. In early stages, studies started with RNN-based method such as GRU4Rec [22] which uses gated recurrent units (GRUs) to capture patterns in user behavior by processing sequences of interactions. By incorporating additional item features, p-RNN [23] propose parallel RNN architecture to model sessions based on the clicks and item features such as image and text embeddings. By enhancing RNN-based models through the incorporation of an attention mechanism, NARM [24] also aims to capture the main intention of a user within the current sequence, achieved through its proposed encoder-decoder architecture. STAMP [25] is another attention-based architecture designed to capture the general interest of users based on both long-term sequences and recent interactions, emphasizing the importance of incorporating information from the most recent clicks. While the methods mentioned above uses vanilla attention mechanism, there are also studies using self attention mechanism to effectively balance the learning of short-term intent and long-term preferences. SASRec [26] adopts self-attention to capture intricate temporal patterns in user behavior sequences, whereas BERT4Rec [27] leverages transformers model to facilitate context-aware recommendations based on the entire input sequence.

Following the examination of Markov chains, RNNs, and attention-related studies, which respectively exhibit limitations such as the dependency assumption, the challenge of capturing complex transitions and maintaining long-term sequence representations, and the potential reduction in local dependencies between adjacent nodes, there has been an emergence of studies focusing on Graph Neural Networks.

The first study on representing the session as graph is the SRGNN model [5], which use attention network to represent session as the composition of the global preference and the current interest of that session. Item vectors are obtained through gated GNNs, and session embedding is acquired by concatenating local embeddings,

which represent the node vectors of the last item in the session, with global embeddings obtained through the adoption of a soft attention mechanism. This method is extended by incorporating a self-attention mechanism to better capture long-range dependencies, in conjunction with graph neural networks within GCSAN [28]. Leveraging both local and global contexts, GCE-GNN [29] and DAT-MDI [30] utilize item transitions across all sessions to enhance the transition patterns within the current sequence. KSR [31] combines a GRU network with a knowledge-enhanced key-value memory network (KV-MN) to capture user preferences by integrating sequential interactions and feature-level knowledge from TransE [32].

2.3. Knowledge Graph Embeddings

A knowledge graph is a model used to represent real-world entities and their relationships using a directed graph, where nodes represent entities and edges represent relationships. Each edge is expressed as a triple in the format (head entity, relation, tail entity), also known as a fact, signifying that two entities are linked by a particular relationship [33]. Despite their effectiveness in organizing structured data, the symbolic representation of these triples often complicates their processing and integration into more dynamic systems like machine learning applications. Additionally, knowledge graphs face challenges such as scalability, data sparsity, and incompleteness, particularly when dealing with large or rapidly changing datasets. That’s why knowledge graph embedding approaches have emerged, aiming to convert symbolic entities and relations into continuous vector spaces, which makes it easier to apply machine learning algorithms and improve the overall efficiency of downstream tasks like link prediction, entity classification, and recommendation systems.

The most common techniques in this field are under translation-based approaches which models the relationships between entities as translations in a continuous vector space. These approaches aim to represent entities and relations as low-dimensional vectors (embeddings) such that the relationship between two entities is captured as a vector translation. TransE [32] is the pioneering method in this field which models

relationships between entities as translations in the embedding space. It essentially minimizes the distance between the embedded head entity plus the relation and the embedded tail entity. Although it is computationally efficient, it is inadequate for modeling relationships beyond one-to-one relations due to embedding all relationships into a single vector space. To overcome this limitation, TransH [34] extends this idea by projecting the head and tail entities onto a relation-specific hyperplane and then applying the translation. This approach allows for better modeling of one-to-many and many-to-one relationships, though it comes with the trade-off of increased complexity. TransR [35] modifies the embedding process by separating the embedding spaces for entities and relations. It projects entities from the entity space into the relation space and builds translations between the projected entities. While TransD [36] takes this a step further by implementing dynamic mappings that enable each entity to be represented in relation-specific spaces.

On behalf of translation-based models, Tensor Decomposition-based Models offer an alternative approach to represent relationships in knowledge graphs by utilizing multi-dimensional arrays to capture interactions between entities and relations more comprehensively. RESCAL [37] leverages tensor factorization to represent relationships in knowledge graphs as a third-order adjacency tensor, facilitating the extraction of latent embeddings for entities and relations that capture complex interactions across multiple dyadic relations. DistMult [38] is a constrained version of RESCAL that forces relation matrices to take on diagonal forms, resulting in reduced model complexity. However, this constraint prevents DistMult from effectively modeling asymmetric relations due to its bilinear structure. Therefore, ComplEx [39] was proposed to address this issue by introducing complex-valued embeddings, allowing it to model both symmetric and asymmetric relations effectively.

With the development and growing popularity of neural networks, methods for knowledge graph embedding based on neural networks have also emerged in the literature, highlighting their importance in modeling non-linear relationships within the data. ConvE [40] models entity and relation embeddings as 2D matrices and ap-

plies convolutional filter operations on these matrices to capture complex interactions between entities and relations. R-GCN [41] adopts Graph Convolutional Network where nodes are updated based on neighborhood information to deal with highly multi-relational data. Similarly, RotatE [42] employs rotational embeddings in the complex plane to effectively represent various relational patterns, enhancing the model’s ability to capture intricate relationships within the knowledge graph.

2.4. Knowledge Graph Embedding Method in Recommender System

The primary objective of the Knowledge Graph Embedding (KGE) task is to represent entities and relations—the fundamental components of a knowledge graph—in continuous vector spaces. In this context, each entity and relation is depicted as a dense vector, preserving the inherent structural relationships within the knowledge graph [33]. KGE has numerous applications in the recommender system domain, particularly in addressing the sparsity problem by leveraging the rich contextual information and latent relationships captured by knowledge graphs [43].

One of the earliest studies in this area is the Collaborative Knowledge Base Embedding (CKE) model, which utilizes heterogeneous information from knowledge bases through TransR [35] to enhance the effectiveness of collaborative filtering methods [44]. Following this, the KGNN-LS model [45] employs graph neural networks to generate personalized user embedding vectors by transforming knowledge graphs into user-specific graphs, relying on the label smoothness assumption. KGAT [46] extends this concept by capturing high-order relations in a hybrid structure of knowledge graphs and user-item graphs, utilizing embedding propagation along with an attention mechanism. Additionally, MKGAT [47] incorporates multi-modal information, including text and visuals, in the knowledge graph construction and applies information propagation to enhance its effectiveness. DKN improves the user experience by combining knowledge graph embeddings through TransD [36] with textual sentence embeddings using a multi-channel convolutional neural network, effectively capturing user interests and enhancing click-through rate predictions. Similarly, KTUP [48] employs knowledge

graph facts as auxiliary data to improve user-item interaction modeling while completing missing facts in the knowledge graph based on an enhanced user-item model. MKR introduces a multi-task learning approach that utilizes knowledge graphs as side information to overcome the cold-start problem. This framework enhances recommendation tasks through a shared feature space, enabling better generalization and improved performance in scenarios with limited data. In contrast, DKFM uses TransE t [32] to leverage contextual information derived from location-based social networks, further enriching its recommendation capabilities.



3. BACKGROUND

3.1. Knowledge Graph

A graph G , written $(U, V) \in E$, is a mathematical structure consisting of a set of vertices V and a set of edges E connecting these vertices. In the field of graph theory, the edges $(U, V) \in E$ signify relationships between vertices U and V . Directed graphs, like (U, V) , indicate a one-way relationship from U to V , while undirected graphs, like (U, V) or (V, U) , depict mutual connections between U and V .

A knowledge graph is a directed labeled form of graph that undergoes a transformation in how relationships between entities are represented through the inclusion of semantic meaning in its edges [49]. In this knowledge graph, constructed by triples of subject-predicate-object within the Resource Description Framework (RDF), includes semantic context increases the depth and precision of knowledge representation. For example, (Jennifer Aniston, actedIn, Picture Perfect) is an example of a triple. This approach lays the foundation for a more complex understanding of interconnected information by capturing relationships and adding nuanced meaning to them.

Knowledge Graph Embedding (KGE) methods are playing a crucial role in many application areas, with recommender systems being a particularly notable field. In recommender systems, knowledge graphs help model the relationships between items, users, and other relevant entities by representing them as a network of interconnected nodes (entities) and edges (relations). This approach allows for capturing the underlying semantics of these relationships, enabling more personalized and context-aware recommendations. By embedding these entities and relations into low-dimensional spaces, KGE methods provide a way to efficiently encode complex interactions and support advanced reasoning in recommendation scenarios.

3.1.1. Knowledge Graph Embedding Methods

Knowledge Graph Embeddings (KGE) is a technique to transform entities and relationships within a knowledge graph into continuous vector spaces, preserving their semantic meanings and structural information. This process enables to make efficient computation and reasoning over the embedded representation.

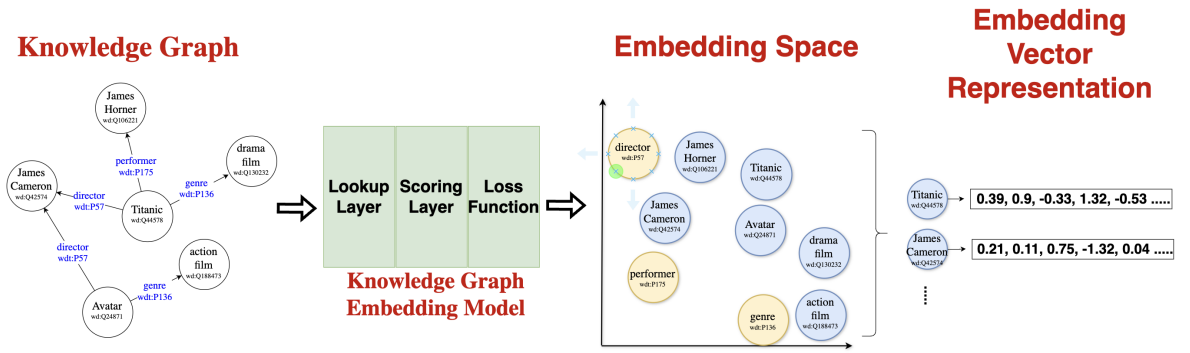


Figure 3.1. Illustrative example of the knowledge graph embedding process. This figure depicts a knowledge graph with entities and relationships, which are mapped into an embedding space and represented as vector form.

Various types of embedding techniques have been developed to optimize this knowledge representation process. These include translation-based models, which interpret relations as translations between entities, as well as semantic matching models, which focus on capturing similarities between entities based on their interactions [33]. Other approaches, such as graph neural network-based embeddings, leverage neural networks to learn richer representations by incorporating both node and edge information. Each of these embedding techniques offers unique strengths for capturing semantic relationships within a knowledge graph, providing a foundation for more accurate and meaningful recommendations.

3.1.1.1. TransE. TransE [32] is the pioneer study in the field of translation-based knowledge graph embedding. TransE introduces the idea of representing relation-

ships as translations in the embedding space, where each entity and relationship is represented as a vector, with the relationship vector acting as a translation from the head entity to the tail entity. The primary objective is to ensure that for a given triplet (h, r, t) (head entity, relation, tail entity), the following condition holds:

$$h + r \approx t, \quad (3.1)$$

where h , r , and t are the embeddings of the head entity, relation, and tail entity, respectively. Therefore, we can express the scoring function as follows:

$$\|\mathbf{h} + \mathbf{r} - \mathbf{t}\|_2^2. \quad (3.2)$$

However, it can only effectively model one-to-one relationships, while struggling to accurately represent one-to-many, many-to-one and many-to-many relationships.

3.1.1.2. TransH. To overcome the limitations encountered in TransE, TransH [34] extends the proposed idea by introducing a new hyperplane for each relation instead of a single translation vector, allowing for better representation of one-to-many and many-to-one relationships. For a given triplet (h, r, t) , the embeddings are adjusted such that they lie on the corresponding hyperplane defined by the relation vector $\mathbf{r}$. TransH represents a relation r as a vector $\mathbf{r}$ situated on a hyperplane, where $\mathbf{w}_r$ is the normal vector. The embeddings of the entities h and t are initially projected onto the $\mathbf{w}_r$ hyperplane, denoted as $\mathbf{h}_\perp$ and $\mathbf{t}_\perp$. The scoring function is:

$$\|\mathbf{h}_\perp + \mathbf{d}_r - \mathbf{t}_\perp\|_2^2, \quad (3.3)$$

where $\mathbf{h}_\perp = \mathbf{h} - \mathbf{w}_r^\top \mathbf{h} \mathbf{w}_r$ and $\mathbf{t}_\perp = \mathbf{t} - \mathbf{w}_r^\top \mathbf{t} \mathbf{w}_r$. If we restrict $\|\mathbf{w}_r\|^2 = 1$, it results in the scoring function:

$$f_r(\mathbf{h}, \mathbf{t}) = \|(\mathbf{h} - \mathbf{w}_r^\top \mathbf{h} \mathbf{w}_r) + \mathbf{d}_r - (\mathbf{t} - \mathbf{w}_r^\top \mathbf{t} \mathbf{w}_r)\|_2^2. \quad (3.4)$$

This approach enables entities to assume various roles across different relations by mapping entity embeddings onto relation hyperplanes.

3.1.1.3. TransR. TransE and TransH put embeddings in the same semantic spaces, meaning they assume a consistent interpretation of the relationships across different entities. In contrast, TransR [35] assumes that an entity can possess multiple aspects,

with different relations emphasizing distinct aspects of the entities. Consequently, it maps entities and relations into separate semantic spaces.

For each triple (h, r, t) , the entity embeddings are denoted as $h, t \in \mathbb{R}^k$, and the relation embeddings are represented as $r \in \mathbb{R}^d$, where k and d correspond to the dimensionalities of the entity and relation embedding spaces, respectively. Importantly, the dimensionalities of the entity and relation embeddings are not required to be identical, allowing $k \neq d$.

To enable this separation, TransR introduces a unique projection matrix $\mathbf{M}_r \in \mathbb{R}^{k \times d}$ for each relation r . This matrix maps entities from the entity space to the relation space, enabling entities to be projected according to the specific relation. The projected vectors of entities are defined as follows:

$$h_r = hM_r, \quad (3.5)$$

$$t_r = tM_r. \quad (3.6)$$

The score function for a given triple (h, r, t) in TransR is defined as:

$$f_r(h, t) = \|h_r + r - t_r\|_2^2. \quad (3.7)$$

To ensure consistency and stability in the embeddings, TransR imposes constraints on the norms of both the embeddings and the mapping matrices. Specifically, for each h, r, t , and M_r , the following conditions are enforced:

$$\|h\|_2 \leq 1, \quad \|r\|_2 \leq 1, \quad \|t\|_2 \leq 1, \quad \|hM_r\|_2 \leq 1, \quad \|tM_r\|_2 \leq 1. \quad (3.8)$$

These constraints help ensure that the embeddings remain within a bounded range, facilitating stable training dynamics and effective knowledge representation in relation-specific spaces.

3.1.1.4. TransD. TransE, TransH, and TransR each use a single projection approach that assumes relations apply uniformly to entity representations across the embedding space. In contrast, TransD [36] introduces a pair of projection vectors for each entity and relation. One vector encodes the semantic meaning of the entity or relation, while

the other is utilized to dynamically construct the corresponding mapping matrix. This approach allows each entity-relation pair to be associated with a distinct mapping matrix tailored to their interaction.

For a given triplet (h, r, t) , let the vectors h , h_p , r , r_p , t , and t_p represent the entities and relation, where the subscript p denotes projection vectors, with $h, h_p, t, t_p \in \mathbb{R}^n$ and $r, r_p \in \mathbb{R}^m$. TransD introduces two mapping matrices, M_{rh} and $M_{rt} \in \mathbb{R}^{m \times n}$, to project entities from entity space to relation space, defined as:

$$M_{rh} = r_p h_p^\top + I^{m \times n}, \quad (3.9)$$

$$M_{rt} = r_p t_p^\top + I^{m \times n}. \quad (3.10)$$

This formulation allows the projection vectors to interact sufficiently. Initializing each mapping matrix with an identity matrix ensures stable transformations. Using these matrices, the projected vectors become:

$$h^\perp = M_{rh}h, \quad t^\perp = M_{rt}t. \quad (3.11)$$

The score function for each triplet is then given by:

$$f_r(h, t) = -\|h_\perp + r - t_\perp\|_2^2. \quad (3.12)$$

To maintain normalized embeddings, they enforce the constraints $\|h\|_2 \leq 1$, $\|t\|_2 \leq 1$, $\|r\|_2 \leq 1$, $\|h_\perp\|_2 \leq 1$, and $\|t_\perp\|_2 \leq 1$.

3.2. Graph Neural Networks

Graphs are used as a meaningful data structure to model complex problems and relationships exist in various domains, from computer science to operational research, from bioinformatics to social network analysis [50]. With the increasing developments in the field of deep learning, the learning ability of complex architectures on data has also increased. Graph Neural Networks (GNNs) emerged to enable representation learning of graph-structured data, compared to traditional artificial neural networks that have shown significant success on sequences and grid-structured data.

The first proposed Graph Neural Network models are essentially extensions of RNN models, designed to be applied on static graphs rather than dynamic ones due to a lack of inherent support in the update mechanisms for managing temporal dependencies and evolving relationships between nodes [51]. The update rule for node representation in a basic GNN is explained as follows:

$$h_v^{(l+1)} = f \left(\sum_{u \in \mathcal{N}(v)} g(h_u^{(l)}, h_v^{(l)}, e_{uv}) \right). \quad (3.13)$$

Here, $h_v^{(l+1)}$ represents the updated representation of node v at layer $(l + 1)$, $\mathcal{N}(v)$ denotes the set of neighboring nodes of v , e_{uv} is the edge feature between nodes u and v , g is a learnable aggregation function, and f is a non-linear activation function.

This formulation is referred to as a Spatial GNN, where information propagation and node updates primarily rely on the spatial relationships within the local neighborhood of each node. Apart from this particular GNN variant, a variety of other graph neural networks have been proposed, covering a broad spectrum of types, including graph convolution networks [52–54], graph attention networks [13, 55], and graph recurrent neural networks [56, 57].

3.2.1. Message Passing

One concept in GNNs is message passing which facilitates the exchange of information between nodes in a graph, allowing nodes to aggregate and update their representations based on the information from their neighbors formulated as follows:

$$h_u = \phi \left(x_u, \Delta_{v \in \mathcal{N}(u)} \psi(x_u, x_v) \right), \quad (3.14)$$

where h_u represents the node representation for node u , x_u denotes the input features of node u , ϕ is a function combining x_u with the aggregated results from neighboring nodes using a permutation-invariant aggregation function Δ , where $v \in \mathcal{N}(u)$ denotes iteration over neighboring nodes, and ψ is a function computing interactions between x_u and x_v for neighboring nodes for u and v . However, the message passing algorithm exhibits high computational costs when applied to large-scale graphs. Additionally, it

faces limitations in recognizing even basic substructures within graphs [58].

3.2.2. Graph Convolution Networks

Graph Convolutional Networks (GCNs) [53] extend the principles of Convolutional Neural Networks (CNNs) [59] to data organized in a graph structure. In the process of acquiring graph representations, the "graph convolution" operation is employed, involving the application of a consistent linear transformation to all neighboring nodes of a given node. This transformation is subsequently subjected to a nonlinear activation function, enhancing the network's capacity to capture intricate relationships within the graph [60].

Graph Convolutional Networks can be classified into two main categories: Spatial Graph Convolutional Networks and Spectral Graph Convolutional Networks. In a Spatial GCN, convolutional operations aggregate information from neighboring nodes with convolutional filters based on their geometric proximity in the graph to capture local patterns and relationships on the localized neighborhood of each node.

Spectral GCN lies in techniques such as graph Fourier transforms, which allow the representation of graph signals in the frequency space. In this context, convolutional operations are performed on the graph Laplacian matrix, which is defined in terms of its adjacency matrix A and degree matrix D as follows:

$$L = D - A, \tag{3.15}$$

where L , D and A represent laplacian matrix of the graph, degree matrix of the graph and adjacency matrix of the graph, respectively.

The degree matrix D is a matrix with diagonal elements D_{ii} representing the degrees of individual nodes, while the adjacency matrix A captures the connections between nodes within the graph. It enables the model to effectively capture and leverage the spectral characteristics of the graph for learning informative node representations.

In the initial version of the proposed architecture, challenges emerge in both forward and backward propagation due to the incorporation of computationally intensive tasks, including matrix multiplications and the eigendecomposition of the Laplacian matrix. Consequently, this results in prolonged training times for the model. In addition to that, the transductive nature of Graph Convolutional Networks (GCN) adds complexity to the generalization process, leading to variations in representation learning for different graph [61]. The following studies aim to address the downsides of the architecture, such as the expensive training process, by enabling mini-batch training [54] or mitigating issues like over-smoothing [53].

3.2.3. Graph Attention Networks

In standard Graph Neural Networks (GNNs) like Graph Convolutional Networks (GCNs), node feature updates involve averaging neighboring features without distinguishing the importance of individual neighbors. Instead of treating all neighbors equally, Graph Attention Networks (GATs) assign attention coefficients to each neighbor by using shared self-attention mechanism [62] to compute attention scores for node pairs without necessitating expensive matrix operations or relying on prior knowledge of the graph structure [13]. This approach allows GATs to capture more complex relationships between nodes by giving differential importance to various neighbors during feature updates. The final node representation, h'_i can be expressed as a weighted combination of its neighbors' features, determined by attention scores as follows:

$$h'_i = \sigma \left(\sum_{j \in \mathcal{N}_i} a_{ij} \cdot \mathbf{W}h_j \right), \quad (3.16)$$

where e_{ij} and a_{ij} represent the attention coefficient and attention scores assigned to the connection between a node i and one of its neighbors j , respectively.

$$a_{ij} = \frac{\exp(e_{ij})}{\sum_{k \in \mathcal{N}_i} \exp(e_{ik})}, \quad (3.17)$$

$$e_{ij} = \text{LeakyReLU}(\vec{a}^T [\mathbf{W}h_i \parallel \mathbf{W}h_j]). \quad (3.18)$$

e_{ij} and a_{ij} are formulated in Equations eqs. (3.17) and (3.18). $\mathcal{N}_i$ signifies the neighborhood of node i , $\vec{a}^T$ denotes a learnable weight vector, $\mathbf{W}$ is a learnable parameter

matrix, and $\parallel$ indicates concatenation, while LeakyReLU refers to the leaky rectified linear unit activation function. This formulation represents GAT with single-head attention. Additionally, the utilization of multi-head attention which helps to stabilise the learning process of self-attention, is possible [13].



3.3. Semantic Technologies

3.3.1. Wikidata

Wikidata, launched in October 2012 by the Wikimedia Foundation, is a free, openly accessible knowledge base designed to serve as a central repository for structured data [4] including Wikipedia. This collaborative platform allows both humans and machines to contribute to and access information on a wide array of topics, making the information human-readable. One of the great advantage of Wikidata is that it provides source of information that can be linked to other available data on the linked data standards.

It is a document-oriented database focused on items representing various topics, concepts, or objects. Each item is assigned a unique identifier known as a QID, which consists of the prefix "Q" followed by positive integers. This QID allows us to identify topics without needing translation into different languages. While an item can have multiple meanings, it must possess a unique identifier and distinct descriptions. The QID is linked to these descriptions to avoid ambiguity, ensuring that the combination of a QID and its description provides a unique meaning.

3.3.2. Resource Description Framework (RDF)

The Resource Description Framework (RDF) is a standardized framework for exchanging data on the web, enabling the sharing and reuse of structured information across various applications [63]. It enables the representation of information in a graph format in the form of triples which consist of a subject, predicate, and object. This structure allows for a flexible way to model relationships and attributes of resources. While subject represents the resource being described, the predicate indicates the property or relationship of the subject, and the object represents the value or another resource related to the subject.

For example, in the triple visualized in Figure 3.2 ("Titanic", "is a", "film"), "Titanic" is the subject, "is a" is the predicate, and "film" is the object. This feature supports various data formats of RDF, making it a part of the semantic web.

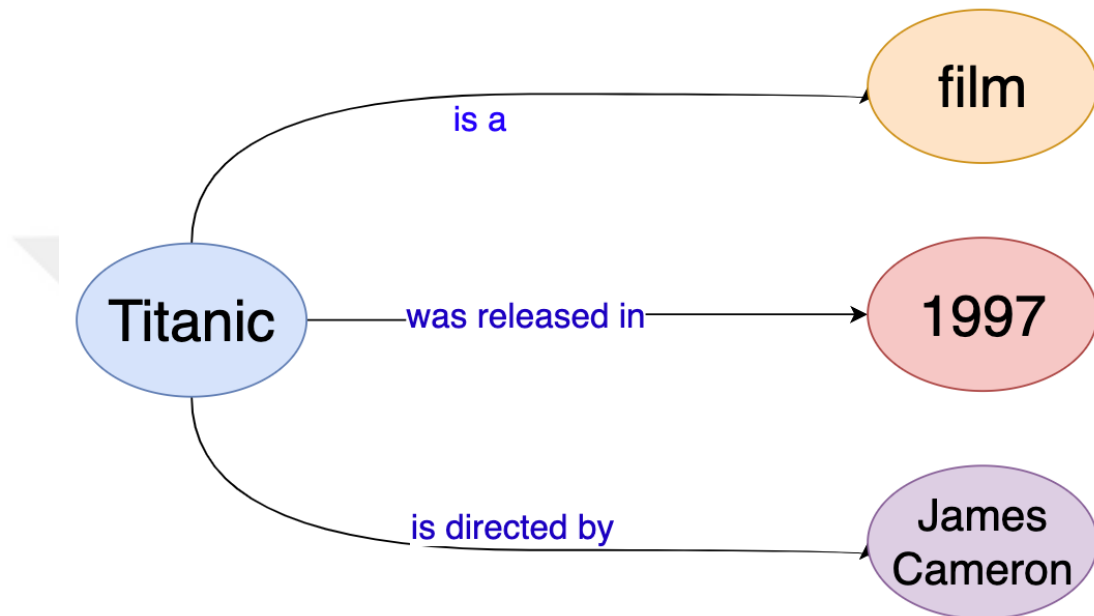


Figure 3.2. An example of RDF Representation of the movie "Titanic".

There are several methods to represent (or serialize) RDF such as Turtle, RDF/XML and JSON. By using URIs (Uniform Resource Identifiers) to uniquely identify resources, RDF ensures that data can be linked and accessed globally, enhancing interoperability among data sources.

The combination of Wikidata and RDF provides a powerful framework for representing and querying structured data, enabling more effective data integration and knowledge representation in applications such as semantic search, data analytics, and knowledge graphs.

3.3.3. SPARQL

SPARQL, which stands for SPARQL Protocol and RDF Query Language, is used to retrieve RDF triples from knowledge databases. It is used to query for key-value data or RDF graphs, enabling complex queries on structured data within these databases. This allows for the extraction and manipulation of data stored in Resource Description Framework (RDF) format, facilitating tasks such as data integration, linking, and enrichment from various sources.

To better understand SPARQL, let's look at a straightforward example of a SPARQL query given in Figure 3.3:

```
-- Movies
SELECT ?item ?itemLabel
WHERE
{
    ?item wdt:P31 wd:Q11424.
    SERVICE wikibase:label { bd:serviceParam wikibase:language "[AUTO_LANGUAGE],en". }
}
LIMIT 20
```

Figure 3.3. An example of SPARQL query.

This query retrieves information about movies from a Wikidata database. The "SELECT" statement specifies that we want to obtain two variables, "?item" and "?itemLabel". The "WHERE" clause contains the pattern to match data in the database. Here, "?item" represents an item in the database, and "wdt:P31" "wd:Q11424" restricts the query to items that are instances of a "movie" (where P31 is the property for "instance of" and Q11424 is the Wikidata entity for "movie"). The "SERVICE wikibase:label" part is used to get the labels (human-readable names) of the items in the specified language, which is set to automatically use the user's language or English. Finally, the "LIMIT 20" clause restricts the result set to 20 items, ensuring the query returns only the first 20 matches. Understanding this query helps illustrate the basic structure and components of SPARQL, including how to specify what data to retrieve,

how to filter the data, and how to limit the results. By using SPARQL, one can perform powerful and flexible queries on linked data, such as retrieving specific types of entities and their associated labels from a dataset like Wikidata.



4. METHODOLOGY

This section outlines the overall architecture of the study, detailing the methodological framework employed to integrate open source knowledge from knowledge base, construct a knowledge graph, and develop a sequential recommendation system utilized by graph neural networks (GNNs). The methodology is divided into several key phases: data integration, knowledge graph construction, knowledge graph embedding generation, and the integration of embeddings into GNN based sequential recommendation architecture.

4.1. Problem Definition

In the domain of recommender systems, sequential recommendation is one of the task that involves predicting the next item a user is likely to interact with, based on their historical interactions modeling as a sequence. In this study, we focus on predicting next movie that is likely to be watched by user leveraging their watch history and relations between items.

Formally, let $U = \{u_1, u_2, \dots, u_m\}$ and $M = \{m_1, m_2, \dots, m_n\}$ represent the set of users and the set of movies, respectively. Each user u_i has a watch history defined as $H_i = \{(m_{i1}, r_{i1}), (m_{i2}, r_{i2}), \dots, (m_{ik}, r_{ik})\}$, where $m_{ij} \in M$ is a movie and $r_{ij} \in \mathbb{R}$ is the rating given by user u_i to movie m_{ij} . The sequence H_i is ordered by the time of interaction. The goal is to develop a recommendation model that takes a user's watch history H_i and predicts the next movie $m_{i,k+1}$ that the user u_i is likely to watch.

Utilizing features related to both users and items enhances the predictive capability of models when predicting the next movie a user will watch. Examples include a user frequently watching films directed by their favorite director, films featuring the same actors, or films sharing common genres. This study aims to increase the success of prediction capability of sequential recommendation model, particularly by leveraging

additional item features available in open-source knowledge base.

4.2. Model Overview

Figure 4.1 provides an overview of our approach, which consists of three main tasks. The first task focuses on constructing a knowledge graph from Wikidata by extracting movie entities and their associated properties to generate embedding vectors that serve as additional features for the recommendation model. To begin, each movie in the MovieLens dataset is linked to its corresponding entity in Wikidata using SPARQL queries. This step is critical for ensuring accurate alignment between the MovieLens dataset and the knowledge base, enabling the retrieval of meaningful supplementary features. Once the correct entities are identified, the associated first-order features (directly connected properties) for each movie entity are extracted to enrich the context of the movie data. These features provide valuable semantic information about the movies, such as genres, directors, cast members, and release dates, enhancing the recommendation model’s ability to make informed suggestions. The extracted data is stored as RDF triples, a standard representation format for knowledge graphs, enabling efficient querying and facilitating the construction of the knowledge graph. Next, we construct a knowledge graph, where nodes represent Wikidata entities related to the movies (i.e., genre, actors, directors, and cast members) and edges represent relationships between these entities. This graph structure allows for a more detailed representation of the semantic relationships within the data. In this way, the relationships between different movies can be modeled at a more complex level. For example, relationships between films with the same director, common actors, or belonging to the same category can be illustrated. Since our study uses different sizes of the MovieLens dataset, we construct a separate knowledge graph for each dataset size. This approach is necessary because some movies present in one dataset size may not be included in others, leading to inconsistencies that could affect the integrity of the knowledge graph and the performance of the recommendation model. By creating separate knowledge graph for each dataset, we ensure that the additional features fetched from Wikidata accurately match the movies in the respective dataset.

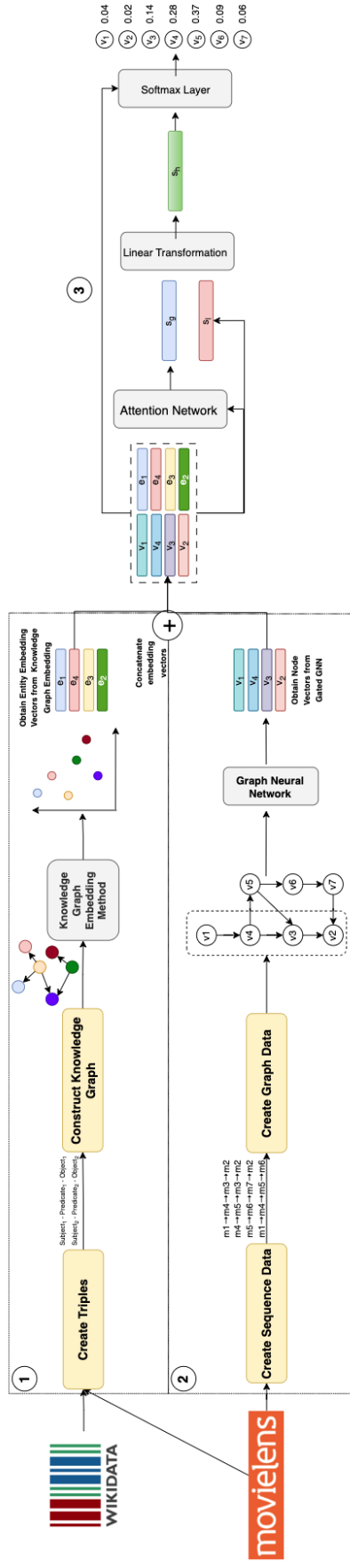


Figure 4.1. Overview of the architecture: ① illustrates the construction of the knowledge graph and the processing of knowledge graph embeddings, ② shows the node vector embeddings from a gated graph neural network, and ③ shows the concatenation of the embedding vectors and the prediction phase.

Subsequently, we generate knowledge graph embeddings, which are low-dimensional vector representations of the nodes in the graph. These embeddings enable the integration of additional contextual information by combining them with other embedding vectors produced by different components of the model architecture. This approach enriches the representation of user-item relationships by incorporating semantic and structural knowledge from the graph. To achieve this, we utilize translation-based knowledge graph embedding techniques, which effectively capture the relationships between entities in a continuous vector space. This ensures that both the semantic meaning and the structural information of the knowledge graph are preserved, enhancing the model’s ability to understand complex interactions and improving the overall recommendation quality.

The second task is to derive node embedding vectors using a gated Graph Neural Network (GNN) within the SRGNN model. We begin by converting each user’s watch history, which is originally unordered in the MovieLens dataset, into sequential data organized by timestamp. The MovieLens dataset contains unique movie and rating histories for each user, reflecting their individual preferences. To prepare this data for sequential modeling, we create a chronological sequence for each user, arranging their interactions with movies based on the recorded timestamps. This transformation enables the model to capture the temporal dynamics of user behavior, providing a foundation for predicting future preferences based on past interactions. This sequential data is then transformed into a graph structure to better capture relationships and patterns in user behavior. In this graph, each node represents a movie that a user has watched, and the edges between nodes represent the sequence in which the movies were watched. This representation allows the model to leverage graph-based techniques to analyze the connections between movies in a user’s watch history. The GNN processes this graph-based representation, outputting node embedding vectors that capture the nuanced relationships and sequential patterns within each user’s watch history.

The third task focuses on continuing the training of the SRGNN model. Instead of using the node embedding vectors as in the original model architecture, we com-

bine the node embeddings $\mathbf{v}$ with entity embeddings $\mathbf{e}$ obtained from the knowledge graph embedding (KGE) method into a single concatenated embedding vector. With these modifications, the adjacency matrix interacts with the concatenated embedding vectors, weight matrix, and bias vector. This integration allows the KGE vectors to serve as additional features in the sequential recommendation architecture, enabling the model to leverage both the structural information from the knowledge graph and the node-specific data, which enhances the system’s overall representation and predictive accuracy. The original training process then resumes with a soft attention layer that assigns varying weights to the different movies in the user’s watch history based on their order. This is followed by a linear transformation layer and a softmax layer to generate the final recommendation output.

4.3. Approach

4.3.1. Data Collection

In the study, two different data sources were utilized. First, the MovieLens dataset served as the primary dataset. This dataset includes information about users, the movies they watched, the ratings they gave to those movies, and various features of the movies. The task is to predict what users would watch next by sorting the movies they had previously rated according to time. Therefore, temporal sorting of user ratings was performed to create a chronological sequence of watched movies for each user. This will be used to create sequence data in order to be used sequential recommendation model as input.

Since another aim of the study is to incorporate additional information from open-source knowledge bases, the features of the movies in the dataset were not used. Instead, Wikidata was chosen as the knowledge base for this purpose. To achieve this, the methodology involved several key steps. The first step was to preprocess the MovieLens data to extract the movie titles and corresponding release dates. This involved separating the movie titles from their publication years, ensuring the titles

were in a clean format for further processing. After extracting the movie titles and release dates, the next step was to find the corresponding entities in Wikidata. This was achieved by matching the cleaned movie titles with Wikidata entities, using SPARQL queries to ensure accurate identification. The goal was to obtain unique identifiers (QIDs) for each movie. With the Wikidata QIDs, additional information about each movie was retrieved. This included various properties such as the director, cast, and genres, which were incorporated into the MovieLens dataset. This enrichment process provided a more comprehensive dataset, enabling deeper analyses and more accurate recommendations. There might be cases where an entity and its properties for the corresponding movie in the MovieLens dataset do not exist. In such scenarios, the movie is filtered out from the input data. This issue also poses potential risks for other domain cases, with details provided in Section 7.

4.3.2. Knowledge Graph Construction

The data sourced from Wikidata is inherently structured as triples, each consisting of a subject, predicate, and object. These triples are utilized to construct a knowledge graph, wherein nodes represent entities—both movies and related Wikidata objects—and edges represent the relationships, defined by predicates, between these entities. Each node encapsulates relevant attributes, such as titles, genres, and release years for movies, while edges denote the various Wikidata properties that connect these objects, such as relationships between directors, actors, and genres.

To enhance the analysis and facilitate tailored recommendations, we create separate knowledge graphs for different sizes of the MovieLens dataset. This stratification allows us to capture the unique characteristics and dynamics present in each dataset size, thereby improving the contextual relevance of the knowledge graph for the specific application of movie recommendation.

4.3.3. Knowledge Graph Embedding

The aim of the study is to incorporate movie embeddings enriched by additional entities from Wikidata. To accomplish this, four well-known translation-based knowledge graph embedding algorithms were used:

- TransE [32] was chosen for its simplicity and efficiency in representing entities and relations in the same continuous vector space, making it ideal for large-scale knowledge graphs.
- TransH [34] builds on TransE by allowing entities to have different representations depending on the relation they are involved in, offering more flexibility for modeling diverse relationships.
- TransR [35] separates entity and relation spaces entirely, projecting entities and relations into distinct spaces, making it well-suited for modeling more intricate, heterogeneous relations.
- TransD [36] enhances this idea by allowing for dynamic mappings, using relation-specific transformations to project entities into different spaces, which enables it to capture more complex interactions between entities and relations.

Each method has different scoring functions and representations, as explained in Section 3.1.1. Figure 4.2 illustrates how different knowledge graph embedding models compute distances between entity embeddings. In TransE, distances are measured within a single, shared embedding space, irrespective of the type of relation. In contrast, both TransH and TransR calculate the distances in relation-specific spaces, where each relation has its own dedicated embedding space for more accurate representation. TransD further refines this approach by dynamically adjusting the relation-specific spaces for each entity pair in the triple. The notation (h, r, t) represents a triple in the knowledge graph, where $\mathbf{h}$ is the head entity, $\mathbf{r}$ is the relation, and $\mathbf{t}$ is the tail entity. By embedding the knowledge graph using these models, we generate low-dimensional vector representations that capture the underlying semantic relationships among entities, which can then be integrated into our sequential recommendation architecture.

The reason for leveraging various Knowledge Graph Embedding (KGE) methodologies is to assess how the quality of these embeddings affects the accuracy and personalization of the recommendation outputs. Different configurations within a knowledge graph lead to varying vector representations for nodes and edges, which can influence the effectiveness of the recommendations, ultimately enhancing user satisfaction and engagement.

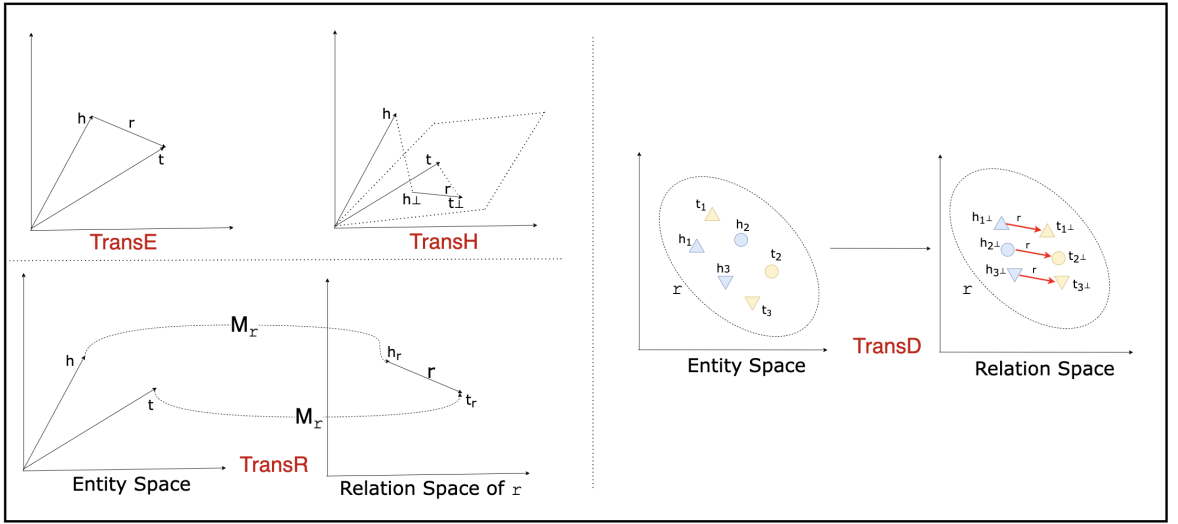


Figure 4.2. An illustration comparing how TransE, TransH, TransR, and TransD compute distances between entity embeddings using shared or relation-specific spaces.

The complexity and computational cost of translation-based models vary greatly, reflecting the balance between simplicity and the ability to represent relationships. TransE, being the most straightforward, operates in a single shared embedding space and has the lowest computational cost due to its minimal parameterization. TransH introduces relation-specific hyperplanes, adding moderate complexity and computational requirements as it projects entities onto these hyperplanes for improved handling of diverse relation types. TransR, by embedding entities and relations in distinct vector spaces, incurs a higher computational cost because of the additional matrix transformations required for each relation. Finally, TransD optimizes this process by using dynamic mapping matrices that depend on both entity and relation embeddings, re-

ducing parameter overhead while maintaining flexibility. The models can be ordered in terms of increasing complexity, starting with TransE, followed by TransD, TransH, TransR. These variations in computational requirements and complexity allow each model to cater to different scenarios, balancing efficiency and accuracy in capturing semantic relationships within knowledge graphs.

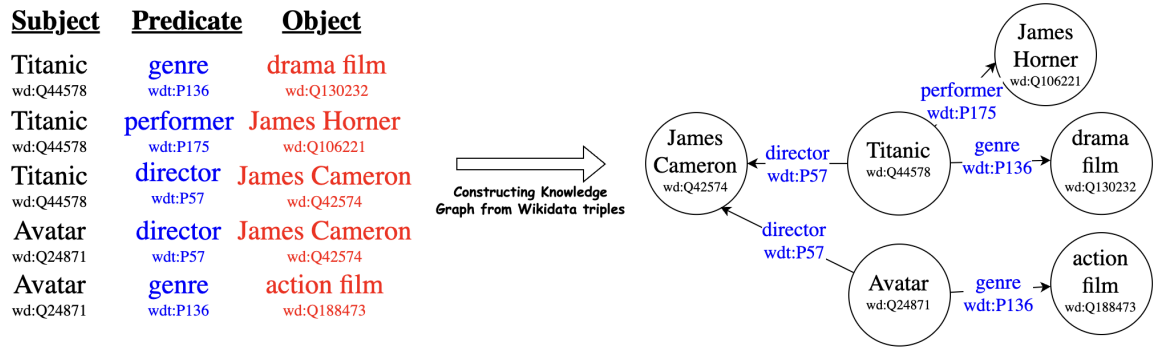


Figure 4.3. A fragment of the knowledge graph corresponding to the triples in the enriched MovieLens dataset, where the nodes represent movie entities and the edges depict the relationships between them. Both nodes and edges are labeled with Wikidata entity and property identifiers.

4.3.4. Sequential Recommendation Model

As a base model, we adopted Graph Neural Networks due to their ability to capture complex transitions between items, which are difficult to reveal with conventional sequential methods. Additionally, we specifically intended to use an architecture that employs a soft attention mechanism to give more weight to the last item in a user’s past interests. For this purpose, we decided to use the architecture in the paper titled ”Session-based Recommendation with Graph Neural Networks (SRGNN).” [5]

The SRGNN model leverages session-based data as a sequence and utilizes graph structures to represent the transitions between items within a session. In our case,

we create a single session for each user by arranging the rated movie data sorted by timestamp. Then, we create a directed graph $\mathcal{G}_t = (\nu_t, \epsilon_t)$ where each node corresponds to a movie $v_{t,i} \in \nu_t$ and each edge $(v_{t,i-1}, v_{t,i}) \in \epsilon_t$ corresponds to two consecutive movies rated by a user. In the original paper, the same item may appear more than once in a session sequence. Therefore, each edge is assigned a normalized weight, calculated as the occurrence of the edge divided by the outdegree of the edge's start node. However, in our movie dataset, each user can rate a movie only once. Consequently, we do not need to use this weighting method since each edge naturally represents a unique transition between two distinct rated movies.

The next step is to embed each movie node $v \in V$ into unified d dimensional embedding space $\mathbb{R}^d$ and obtain node vectors $\mathbf{v} \in \mathbb{R}^d$ by training graph neural network. Each session can be represented with node vectors of movies in that session. The update functions used during training can be expressed as follows:

$$\mathbf{a}_{s,i}^t = \mathbf{A}_{s,i} \left[\mathbf{v}_1^{t-1}, \dots, \mathbf{v}_n^{t-1} \right]^T \mathbf{H} + \mathbf{b}, \quad (4.1)$$

$$\mathbf{z}_{s,i}^t = \sigma \left(\mathbf{W}_z \mathbf{a}_{s,i}^t + \mathbf{U}_z \mathbf{v}_i^{t-1} \right), \quad (4.2)$$

$$\mathbf{r}_{s,i}^t = \sigma \left(\mathbf{W}_r \mathbf{a}_{s,i}^t + \mathbf{U}_r \mathbf{v}_i^{t-1} \right), \quad (4.3)$$

$$\tilde{\mathbf{v}}_i^t = \tanh \left(\mathbf{W}_o \mathbf{a}_{s,i}^t + \mathbf{U}_o \left(\mathbf{r}_{s,i}^t \odot \mathbf{v}_i^{t-1} \right) \right), \quad (4.4)$$

$$\mathbf{v}_i^t = (1 - \mathbf{z}_{s,i}^t) \odot \mathbf{v}_i^{t-1} + \mathbf{z}_{s,i}^t \odot \tilde{\mathbf{v}}_i^t. \quad (4.5)$$

where $H \in \mathbb{R}^{d \times 2d}$ controls the weight, $z_{s,i}$ and $r_{s,i}$ are the reset and update gates, respectively. $v_i \in \mathbb{R}^d$ represents the latent vector of node $v_{s,i}$, and $\left[\mathbf{v}_1^{t-1}, \dots, \mathbf{v}_n^{t-1} \right]$ is the list of node vectors that make up a session s . $\sigma(\cdot)$ is the sigmoid function, and $\cdot$ is the element-wise multiplication operator. The connection matrix $A_s \in \mathbb{R}^{n \times 2n}$ determines the information propagation across nodes. It represents the concatenation of two adjacency matrices for incoming and outgoing weighted edges, respectively. As stated in the original paper [5], the connection matrix can be modified or combined with additional features. As mentioned in the following sections, we will concatenate the embedding vectors obtained from the KGE method with the connection matrix.

The gated graph neural network processes nodes concurrently in each session graph G . It extracts latent vectors of neighborhoods as input and uses Equation (4.1) for information propagation between nodes as limited by matrix A . The update and reset gates decide which information to retain or discard given in Equation (4.2). The candidate state, constructed from the previous state, current state and reset gate, is combined with the previous hidden state under the control of the update gate to form the final state. We can get the final node vector by updating all nodes in session graphs till convergence.

For considering both long-term preference and current interest, the proposed model leverage soft-attention mechanism on both local embeddings and global embeddings. Local embedding represents the node vector of last movie in the session while global embedding is represented as aggregation of all node vectors.

$$\alpha_i = \mathbf{q}^T \sigma(\mathbf{W}_1 \mathbf{v}_n + \mathbf{W}_2 \mathbf{v}_i + \mathbf{c}), \quad (4.6)$$

$$\mathbf{s}_g = \sum_{i=1}^n \alpha_i \mathbf{v}_i, \quad (4.7)$$

$$\mathbf{s}_h = \mathbf{W}_3 [\mathbf{s}_1; \mathbf{s}_2]. \quad (4.8)$$

Different information priorities held in sequence is represented with soft-attention mechanism as described in Equation (4.7).

In the final step, the final embedding is calculated by applying a linear transformation to the concatenation of the local and global embeddings, and then multiplying by a weight matrix stated in Equation (4.8).

4.3.5. Our Model

The main idea of our method is to leverage open source knowledge-base information into the sequential recommendation method. As stated in Section 4.3.1, we use MovieLens as main dataset and Wikidata as knowledge base for the study. The aim is to increase the recommendation quality of sequential recommendation by incorporating the available outsource data.

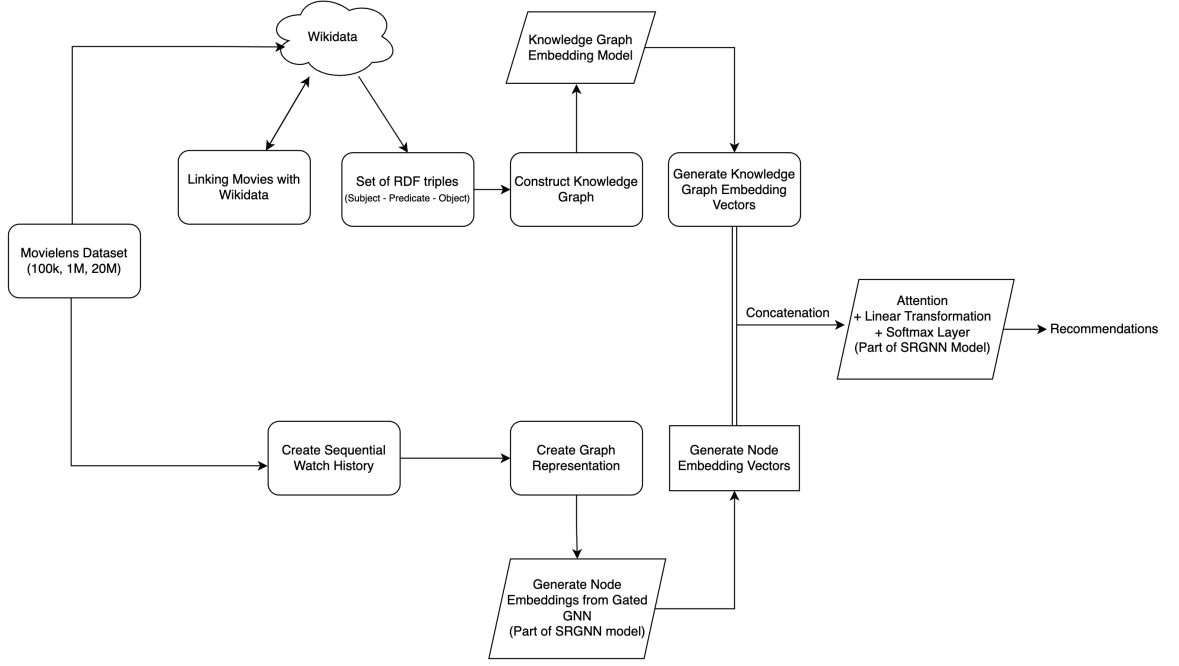


Figure 4.4. Overview of our proposed model.

The overall architecture started with creating knowledge graph by fetching the movie entities and its properties from Wikidata. The subject and object in triplets makes up node while predicates are edges in KG. In the first step, we obtain entity and relation embeddings by training knowledge graph embedding method. The node list consists of entities of movies in the MovieLens data and the entities in their first neighborhood in Wikidata while the edge list consists of properties. In the study, we only use the embedding vectors of the movie entities, we do not use the entity embeddings in their neighborhood of movie entity. However, entities not used in our study also creates a future work and is discussed in Section 7.

After carefully training the Knowledge Graph Embedding method, we obtain entity embedding vectors $\mathbf{e} \in \mathbb{R}^k$ and relation embedding vectors $\mathbf{r} \in \mathbb{R}^k$, where k is the dimension of the embedding space. $\mathbf{e}$ is a list of entity embeddings that contains the movie entities stored in the MovieLens data and entities in the one-hop neighborhood in Wikidata. In the SRGNN (Section 4.3.4) section, Equation (4.1) represents information propagation between different nodes, and $\mathbf{v}$ in that formula represents the list of node

vectors. The main integration of the proposed model is to concatenate the node vectors $\mathbf{v}$ with entity embeddings $\mathbf{e}$ from the KGE method. With this change, the adjacency matrix interacts with the new concatenated embedding vector along with the weight matrix and bias vector. The new formulation for obtaining activation is as follows:

$$\mathbf{a}_{s,i}^t = \mathbf{A}_{s,i} \left([\mathbf{v}_1^{t-1}, \dots, \mathbf{v}_n^{t-1}; \mathbf{e}_1, \dots, \mathbf{e}_m] \right)^T \mathbf{H} + \mathbf{b}. \quad (4.9)$$

Consequently, the model can leverage both structural information from the knowledge graph and node-specific information, enhancing the overall representation and predictive power of the system.

5. IMPLEMENTATION

5.1. Data Processing

5.1.1. Obtaining Movie Informations

In this study, the Movielens dataset serves as the primary data source, while Wikidata is utilized as the knowledge base to enhance the Movielens data by incorporating additional movie-related features. The original item dataset of Movielens contains movie titles along with their release dates, formatted as "Movie Title (Year)," and genre information for each corresponding movie. This dataset provides a foundation for linking the movies to their detailed properties in Wikidata, enabling a richer contextual understanding of each film through external knowledge.

Table 5.1. An example for preprocessed movie data.

movie_id	movie_title	genres	cleaned_title	year
0	Toy Story (1995)	Animation—Children’s—Comedy	Toy Story	1995
1	Jumanji (1995)	Adventure—Children’s—Fantasy	Jumanji	1995
2	Grumpier Old Men (1995)	Comedy—Romance	Grumpier Old Men	1995
3	Waiting to Exhale (1995)	Comedy—Drama	Waiting to Exhale	1995
4	Indian in the Cupboard, The (1995)	Adventure—Children—Fantasy	Indian in the Cupboard	1995

To link the movies in the Movielens dataset to their corresponding entities in Wikidata, the first step is to extract the movie titles and release years in the correct format. The release year is crucial for distinguishing between potential entities in Wikidata that share the same name but refer to different films released in different years. Since multiple entities may exist with identical titles but vary in other attributes such as release year, it is essential to use the release year as a key differentiator. By leveraging the release year information from the original dataset and matching it to the release year stored in the corresponding Wikidata entity, we can ensure accurate alignment and link the correct entity to each movie in our study.

```

1: Input: String movieDescription containing the movie title and release year
2: Output: Tuple (title, releaseYear)
3: Find the index of ", The" in movieDescription and store it in commaIndex
4: Find the index of "(" in movieDescription and store it in parenthesisIndex
5: if commaIndex  $\neq -1 \wedge$  (commaIndex < parenthesisIndex  $\wedge$  parenthesisIndex = -1) then
6:   splitIndex  $\leftarrow$  commaIndex
7: else
8:   splitIndex  $\leftarrow$  parenthesisIndex
9: end if
10: Extract the substring of movieDescription from the start to splitIndex, clean it by removing extra spaces, and store it in title
11: if ", A" is in title then
12:   Remove ", A" from title by splitting and keeping the first part, then clean the result by removing extra spaces
13: end if
14: Return: (title, releaseYear)

```

Figure 5.1. Algorithm for extracting title and release year from given movie titles in original dataset.

To achieve this, algorithm in Figure 5.1 is used to process each entry in the dataset. This function performs the following operations:

- (i) Identify the Separator: It locates the position of specific delimiters in the movie description to identify the release year in movie title. The function searches for the index of ", The" and "(" to determine where to split the title from the year. If ", The" appears before the year or if "(" is not present, the function uses ", The" as the separator.
- (ii) Extract Title: Using the identified index, the function extracts the substring that represents the movie title. It then removes any leading or trailing whitespace to clean up the title.
- (iii) Extract Year: The function employs a regular expression to locate the year enclosed in parentheses.
- (iv) Handle Special Cases: If the title contains ", A", the function removes this portion to ensure the title is in its correct form.

The function then returns a tuple consisting of the cleaned movie title and the release year. This process is crucial for aligning the extracted titles with their corresponding entries in Wikidata, thereby enriching the dataset with additional movie features.

5.1.2. Fetching Wikidata Entities and Properties

Once the titles are extracted and formatted correctly, the next step involves matching these titles with the corresponding entities in Wikidata. This matching process allows for the enrichment of the Movielens dataset by incorporating additional information from Wikidata, such as director, cast, and other relevant details about each movie. By linking these datasets, we can perform more comprehensive analyses, gain deeper insights, and enhance the quality of recommendations and other data-driven outcomes derived from the Movielens dataset.

After obtaining clean movie title and corresponding publish date, we firstly find the corresponding Wikidata entity by using SPARQL query given in Figure 5.2. The SPARQL query is designed to find the corresponding Wikidata entity for a given movie title. It begins by specifying the variables to be retrieved: the item (or entity), its label, description, and publication year. The query then searches for entities with a label that matches the provided movie title in English. It further ensures that the entity is categorized as a film by checking if it is an instance of or a subclass of the 'film' category in Wikidata. The properties P31 and P279 are entity IDs for 'instance of' and 'subclass of', respectively, while Q11424 represents the entity ID for movie.

```
SELECT DISTINCT ?item ?itemLabel ?itemDescription ?publicationYear
WHERE {
  ?item rdfs:label "Toy Story"@en.
  # Check if is an "instance of" or descendent of "film" (wd:Q11424)
  ?item wdt:P31/wdt:P279* wd:Q11424.
  OPTIONAL {
    ?item wdt:P577 ?publicationDate.
    BIND(YEAR(?publicationDate) AS ?publicationYear)
  }
  SERVICE wikibase:label {bd:serviceParam wikibase:language "en".}
}
```

Figure 5.2. SPARQL query for a movie titled 'Toy Story' that returns the name, description, and release year.

The query also attempts to retrieve the publication date of the movie, converting it to the publication year if available. This "publishDate" will be used to compare with the date stored in the Movielens dataset to ensure a more accurate match between the entities found in Wikidata and the original movies. This step is crucial because there might be multiple entities with the same or similar names, and matching both the name and publication date helps in identifying the correct entity. Duplicate movie titles across different release years can lead to mismatched entities during data integration. For example, there are two movies titled Titanic—one is a 1953 film by

Jean Negulesco, and the other is a 1997 film by James Cameron. To ensure accuracy, we verify the correct entity by comparing the release year in the MovieLens dataset with that in Wikidata. Additionally, the query includes a service that retrieves labels and descriptions in English to ensure consistency. This process allows us to accurately identify and retrieve detailed information about the movie from Wikidata, which can then be used to enrich the MovieLens dataset with attributes such as the director, cast, genre, and more.

Table 5.2. Preprocessed movie data with Wikidata enrichment. Columns include movie ID, title, genres, cleaned title, release year (given in MovieLens), Wikidata entity name, and Wikidata QID.

movie_id	movie_title	genres	cleaned_title	year	wikidata_entity_name	wikidata_qid
0	Toy Story (1995)	Animation—Children's—Comedy	Toy Story	1995	Toy Story	Q171048
1	Jumanji (1995)	Adventure—Children's—Fantasy	Jumanji	1995	Jumanji	Q222939
2	Grumpier Old Men (1995)	Comedy—Romance	Grumpier Old Men	1995	Grumpier Old Men	Q782465
3	Waiting to Exhale (1995)	Comedy—Drama	Waiting to Exhale	1995	Waiting to Exhale	Q736254
4	Waiting to Exhale pboard, The (1995)	Adventure—Children—Fantasy	Indian in the Cupboard	1995	The Indian in the Cupboard	Q1194561

Table 5.2 provides a dataset of movies with preprocessed titles and enriched metadata from Wikidata. It includes columns for a unique movie ID, the original title with release year, genres separated by vertical bars, a cleaned title, release year, the Wikidata entity name, and the corresponding Wikidata QID. This table serves as a reference, combining basic movie information with identifiers fetched from Wikidata, which enables the integration of additional properties to enrich the dataset.

After obtaining Wikidata identifiers for all movies, the next step is to retrieve all properties associated with each Wikidata entity using its QID. We only fetch the first-order features for each entity, which means retrieving properties directly linked to the movie entity without following any deeper connections to other entities. Rather than filtering the property list with a predefined set of properties, our goal is to leverage the full potential of the open-source knowledge base provided by Wikidata. To achieve this, we utilize the algorithm in Figure 5.3 to extract all relevant properties within a single hop of the given entity.

```

1: Description: Retrieve all properties of a Wikidata entity identified by its QID.
2: Input: QID of a Wikidata entity (qid)
3: Output: List of tuples containing QID, property ID, and property values
4: entity ← client.get(qid)                                ▷ Fetch the entity using its QID
5: propertiesList ← []                                     ▷ Initialize an empty list to store properties
6: for statementGroup ∈ entity.attributes do
7:   propertyID ← statementGroup    ▷ Extract the property ID from statement group
8:   for statement ∈ entity.attributes['claims']['statementGroup'] do
9:     try
10:      values ← statement['mainsnak']['datavalue']['value']    ▷ Extract property
      values
11:      propertiesList.append((qid, propertyID, values['id']))    ▷ Add a tuple of
      QID, property ID, and values to the list
12:    catch
13:      pass
14:    end try
15:  end for
16: end for
17: return propertiesList                                ▷ Return the list of properties

```

Figure 5.3. Algorithm to retrieve properties for a given Wikidata entity QID.

The function `get_properties` as given in Figure 5.3 operates as follows:

- (i) **Fetch Entity:** The function retrieves the entity from Wikidata using the provided QID.
- (ii) **Initialize List:** An empty list is initialized to store the properties of the entity.
- (iii) **Iterate Through Claims:** The function iterates over all claims associated with the entity. Each claim group is identified by a property ID.
- (iv) **Extract Values:** For each statement within a claim group, the function attempts to extract property values. If successful, it appends a tuple containing the QID, property ID, and property values to the list.

- (v) Error Handling: Any errors encountered during value extraction are ignored.
- (vi) Return Results: The function returns the list of properties.

Properties obtained from Wikidata are represented as triples (subject, predicate, object) as given in Table 5.3, making creating a knowledge graph easier. Each property for every movie is stored in a list to be used in constructing the knowledge graph. In this context, the subject and object represent entities, while the predicate denotes the relationship between them. Consequently, our entity space expands to include various entities beyond just movie titles.

Table 5.3. Table showing the properties of movie entities fetched from Wikidata, represented as subject, predicate, and object with corresponding entity names.

Subject	Predicate	Object
Toy Story (Q171048)	director (P57)	John Lasseter (Q269214)
Toy Story (Q171048)	producer (P162)	Bonnie Arnold (Q1132913)
Toy Story (Q171048)	characters (P674)	Woody (Q2290907)
Scream of Stone (Q2305376)	film editor (P1040)	Suzanne Baron (Q3505895)
Scream of Stone (Q2305376)	narrative location (P840)	Argentina (Q414)
Mixed Nuts (Q1824522)	cast member (P161)	Caroline Aaron (Q508404)
When We Were Kings (Q1779974)	director of photography (P344)	Maryse Alberti (Q45928)
8 Heads in a Duffel Bag (Q2478733)	distributed by (P750)	Netflix (Q907311)
Pocahontas (Q218894)	composer (P86)	Alan Menken (Q317272)

Table 5.3 shows a sample of triples retrieved from Wikidata. For each dataset size used in the study, features were fetched and stored separately. As the size of the Movielens dataset grows, the total number of triples also increases, since more movies lead to more data being gathered from Wikidata. As you can see in Table 5.4 and Table 5.5, the total number of each entity increases as the dataset size grows. However, the distribution of entities remains mostly the same in terms of percentage, which shows that the relative proportions of different entities are consistent despite the increase in total entities. This growth results in a larger and more detailed knowledge graph, capturing richer relationships and attributes associated with each movie.

Table 5.4. Top Entities Across Datasets.

Dataset	Identifier	Identifier Name	Count and Percentage	
			Count	Percentage (%)
ML-100k	Q11424	film	1448	5.62
	Q1860	English	1401	5.44
	Q22006653	color	1341	5.21
	Q723685	video on demand	1334	5.18
	Q30	United States of America	1262	4.90
	Q907311	Netflix	1110	4.31
	Q130232	drama film	770	2.99
	Q4165246	Bechdel test	767	2.98
	Q23817729	no age restriction	726	2.82
	Q20644796	FSK 12	424	1.65
ML-1M	Q11424	film	3335	7.38
	Q1860	English	3250	7.19
	Q723685	video on demand	3059	6.77
	Q22006653	color	3005	6.65
	Q30	United States of America	2917	6.45
	Q907311	Netflix	2547	5.63
	Q130232	drama film	1680	3.72
	Q4165246	Bechdel test	1675	3.71
	Q23817729	no age restriction	1581	3.50
	Q20644796	FSK 12	990	2.19
ML-20M	Q11424	film	17546	13.03
	Q1860	English	14568	10.81
	Q723685	video on demand	14299	10.61
	Q22006653	color	14025	10.41
	Q30	United States of America	12593	9.35
	Q907311	Netflix	10888	8.08
	Q130232	drama film	8667	6.43
	Q23817729	no age restriction	6001	4.45
	Q4165246	Bechdel test	5983	4.44
	Q20644796	FSK 12	3576	2.65

Table 5.5. Top Predicates Across Datasets.

Dataset	Identifier	Identifier Name	Count and Percentage	
			Count	Percentage (%)
ML-100k	P161	cast member	22503	30.68
	P136	genre	4718	6.43
	P750	distributed by	2715	3.70
	P58	screenwriter	2336	3.18
	P162	producer	2089	2.85
	P1411	nominated for	2080	2.84
	P5021	assessment	2031	2.77
	P495	country of origin	1996	2.72
	P364	original language of film or TV show	1746	2.38
	P57	director	1651	2.25
ML-1M	P161	cast member	49658	30.91
	P136	genre	10603	6.60
	P750	distributed by	6137	3.82
	P58	screenwriter	5318	3.31
	P162	producer	4611	2.87
	P495	country of origin	4568	2.84
	P1411	nominated for	4303	2.68
	P5021	assessment	4188	2.61
	P364	original language of film or TV show	4000	2.49
	P57	director	3891	2.42
ML-20M	P161	cast member	214762	31.89
	P136	genre	45704	6.79
	P750	distributed by	27253	4.05
	P495	country of origin	24474	3.63
	P58	screenwriter	23730	3.52
	P364	original language of film or TV show	20809	3.09
	P162	producer	20622	3.06
	P57	director	20568	3.05
	P31	instance of	19901	2.96
	P462	part of	17159	2.55

5.2. Constructing Knowledge Graph

Our knowledge graph is constructed from multiple triples fetched from Wikidata, as explained in previous sections. For our knowledge graph (KG) experiments, we use PyKEEN [64], a Python library specifically designed for knowledge graph embeddings. This library facilitates both data processing and training experiments through its comprehensive functions. To train Knowledge Graph Embedding (KGE) methods, we first need to create triples as input data to make suitable for the library. We leverage the `TriplesFactory.from_labeled_triples` function in PyKEEN, which requires a list of triples as input. This function efficiently handles the transformation of raw triplet data into a format suitable for KGE model training.

When we look at the initial analysis of the constructed knowledge graphs, across all datasets, the most frequent entity is Q11424 (film), consistently accounting for a significant percentage of the triples. Its representation increases with the dataset size, from 5.62% in ML-100k to 13.03% in ML-20M, indicating that the larger datasets capture more comprehensive film-related information. Similarly, Q1860 (English), Q723685 (video on demand), and Q30 (United States of America) appear frequently, reflecting the prevalence of English-language films and content distributed in the U.S. across these datasets. The dominance of Q907311 (Netflix) in all three datasets highlights its strong presence as a major distributor in the dataset.

Interestingly, while entities such as Q130232 (drama film) and Q4165246 (Bechdel test) remain consistent across datasets, their percentages vary, suggesting that genre-specific or socially relevant attributes (e.g., the Bechdel test) become more prominent with increasing data size. The low representation of age-restriction entities such as Q23817729 (no age restriction) and Q20644796 (FSK 12) indicates that content classification data about movies is relatively infrequent compared to other attributes.

The predicates (relations) exhibit similar patterns across datasets, with P161 (cast member) being the most frequently used relation by a wide margin in all datasets.

This is expected, as films often have extensive cast lists, contributing to the large proportion of cast-member relations. The next most common predicates, such as P136 (genre) and P750 (distributed by), show a strong correlation with the primary characteristics of films, including their genre and distribution channels.

The entity and predicate distributions highlight the robustness of the knowledge graph in capturing both general and specific film-related information. Larger datasets provide greater detail and a more comprehensive representation of entities and their relationships, especially around key characteristics such as cast members, language, and distribution. The consistency of top entities and predicates across datasets also suggests a well-structured knowledge graph, while the increasing prevalence of certain relations in larger datasets indicates enhanced semantic richness.

5.2.1. Analysis on Knowledge Graphs

Visualization and analysis of the knowledge graph are crucial steps in understanding the structure, relationships, and insights embedded within the data. Visualizing the graph helps us identify key entities and their interactions, revealing patterns that may not be immediately apparent from raw data. By filtering nodes based on the degree parameter (i.e., the number of connections a node has), we can focus on the most influential entities in the graph, such as high-degree nodes that represent central hubs in the network. These nodes often play a critical role in the graph, and their analysis can provide valuable insights into the most interconnected or important entities.

Table 5.6. Comparison of number of nodes, number of edges, and average degree for each Knowledge Graph constructed for different size of MovieLens datasets.

Knowledge Graph	Number of Nodes	Number of Edges	Average Degree
MovieLens 100k	25752	69140	2.685
MovieLens 1M	45200	151536	3.353
MovieLens 20M	134708	637407	4.732

Table 5.6 shows a clear progression in the size and complexity of the knowledge graphs as the dataset size increases. The number of nodes grows substantially from 25,752 in MovieLens 100k to 134,708 in MovieLens 20M, indicating that larger datasets capture a wider range of entities. Similarly, the number of edges increases significantly, with the MovieLens 20M graph containing over 637,000 edges, compared to just 69,140 in MovieLens 100k. The average degree of nodes also rises, from 2.685 in the smallest graph to 4.732 in the largest, suggesting that entities in the larger graphs are more interconnected, likely due to the richer information captured in the larger datasets.

Table 5.7. The table provides a comparison of the total number of nodes, distinguishing between those representing movie entities and those retrieved from Wikidata as feature entities.

Knowledge Graph	# of Nodes	# of Movie Names as Entity	# of Wikidata Entities
MovieLens 100k	25752	1625(%6.4)	24127(% 93.6)
MovieLens 1M	45200	3696(%8.1)	41504(%91.9)
MovieLens 20M	134708	17515(%13)	117193(%87)

Table 5.7 provides insights into the composition of the knowledge graphs constructed from the MovieLens datasets by differentiating between entities representing movie titles and those fetched from Wikidata. The proportion of movie names as entities increases with dataset size, rising from 6.4% in MovieLens 100k to 13% in MovieLens 20M. In contrast, the number of Wikidata entities remains significantly larger across all datasets, constituting over 90% of the total entities. For instance, while MovieLens 20M contains 17,515 movie entities, it features 117,193 entities sourced from Wikidata. The total count and proportion of Wikidata entities would be higher if we fetched entities from multiple neighborhood relationships. In other words, an additional higher-order neighborhood relationship would provide us with more entities related to the first-order neighborhood entities. In overall, numbers emphasize that most of the information utilized in constructing the knowledge graphs is derived from Wikidata, enriching the datasets with valuable context and relationships.

Table 5.8 provides insights into the scale of knowledge graph enrichment across the different MovieLens datasets, showing the total number of unique entities and relations after integrating Wikidata entities. As expected, the number of unique entities and relations increases with the size of the dataset, with the MovieLens 20M dataset containing the highest number of unique entities (134,708) and relations (303). The MovieLens 1M dataset follows with 45,200 unique entities and 230 relations, while the MovieLens 100K dataset has the smallest enrichment, with 25,752 unique entities and 191 relations. These variations highlight the increasing complexity and breadth of the knowledge graph as the dataset size grows, which in turn could influence the recommendation model’s ability to capture more diverse and complex relationships between entities.

Table 5.8. The table provides a comparison of the total unique number of nodes and edges including Wikidata entities after enrichment.

Knowledge Graph	# of Unique Entities	# of Unique Relations
MovieLens 100k	25752	191
MovieLens 1M	45200	230
MovieLens 20M	134708	303

In Figure 5.4, 5.5, and 5.6, small fragments of KGs for each size of the MovieLens datasets are visualized. To enhance readability and interpretation, the graphs are filtered using the degree of nodes within a specific interval. The degree can be defined as the number of connections (edges) a node has to other nodes in the graph, indicating its level of connectivity. Setting the degree to a high number results in filtering out nodes with fewer connections to other nodes, which means that we obtain a graph containing more influential nodes and edges. There are common entities coming from Wikidata such as United States of America, English, and Bechdel test. It highlights that the country of origin for many movies is America and the original language is English. Also, many movies have Bechdel Test information, which is used as a demonstration of the active presence of women in film and other fiction in general, and to draw attention to gender inequality in fiction.

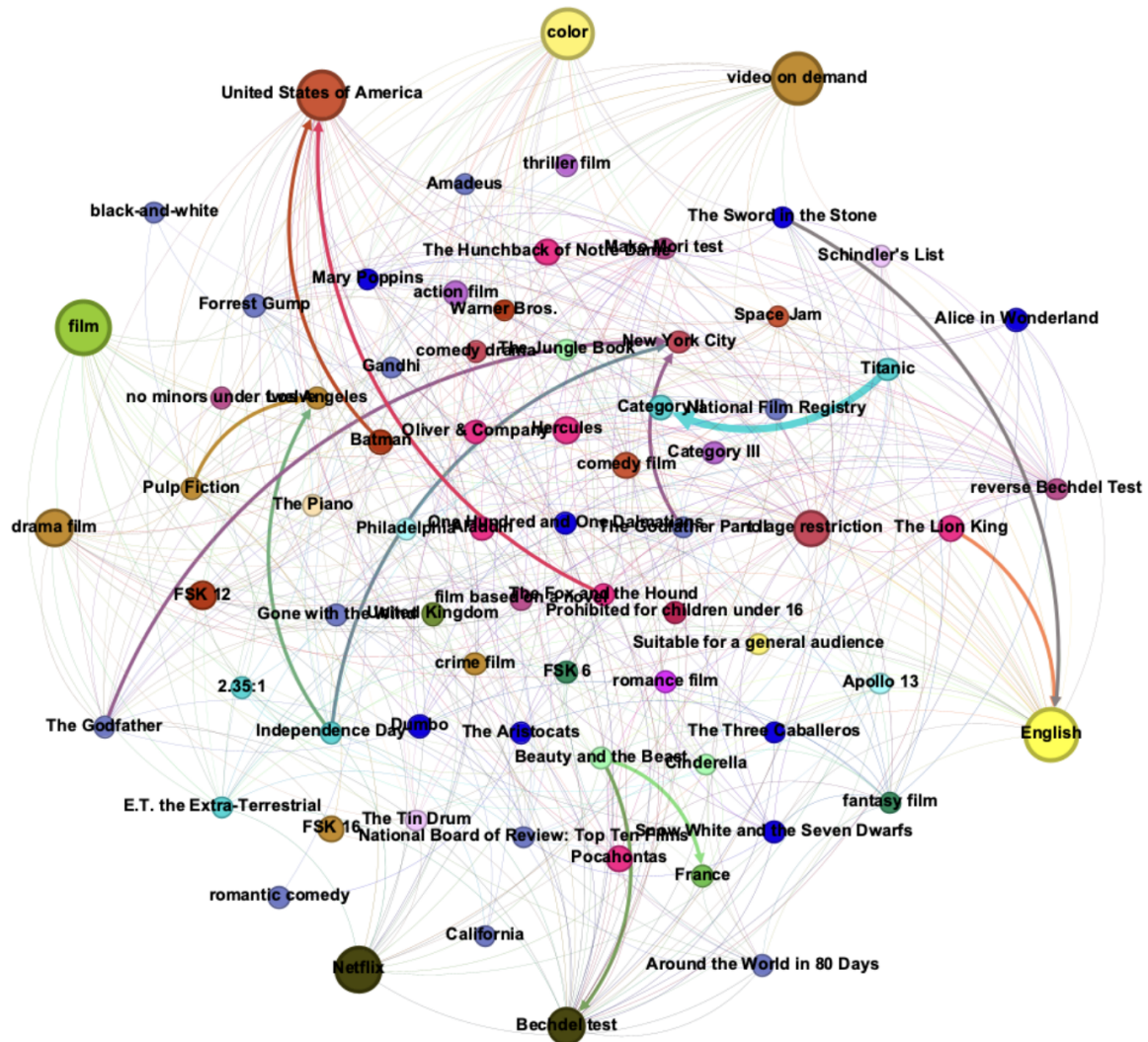


Figure 5.4. A fragment of the knowledge graph constructed for the MovieLens 100K dataset, where the degree of nodes is filtered between 100 and 1148.

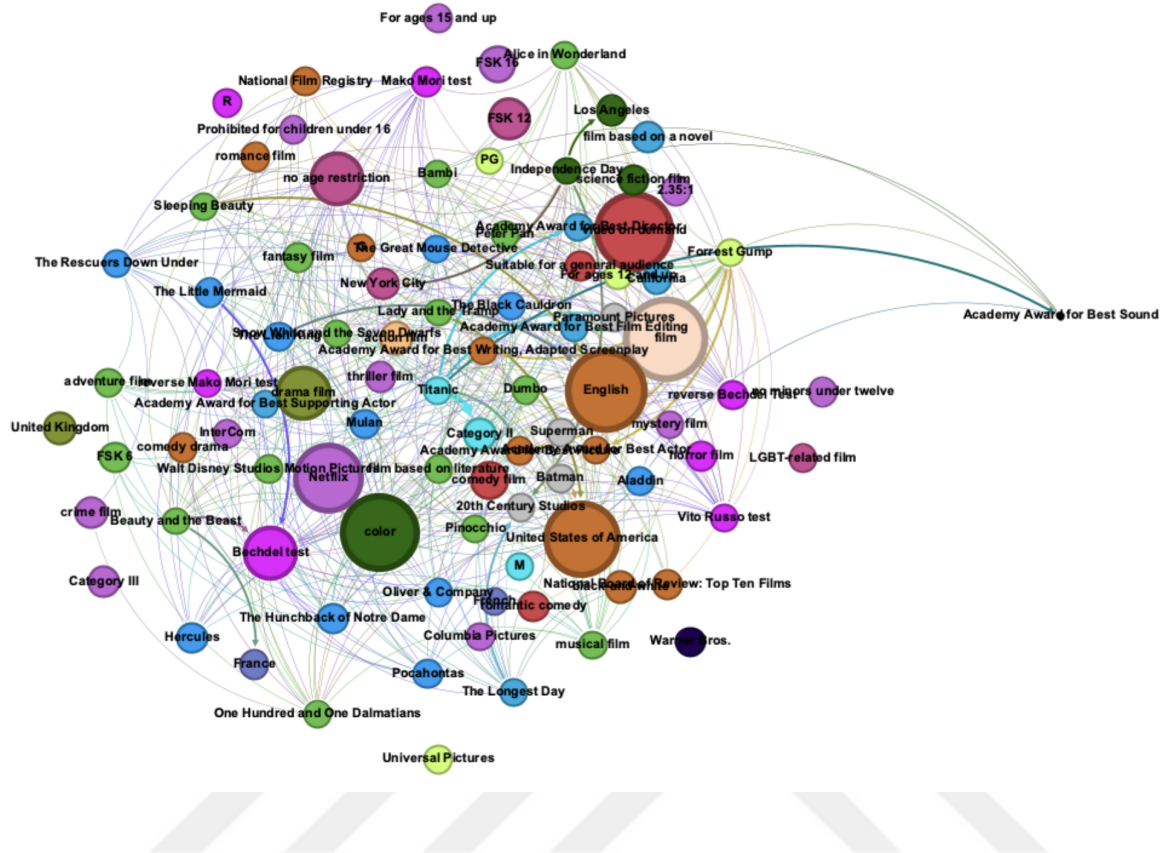


Figure 5.5. A fragment of the knowledge graph constructed for the MovieLens 1M dataset, where the degree of nodes is filtered between 160 and 3346.

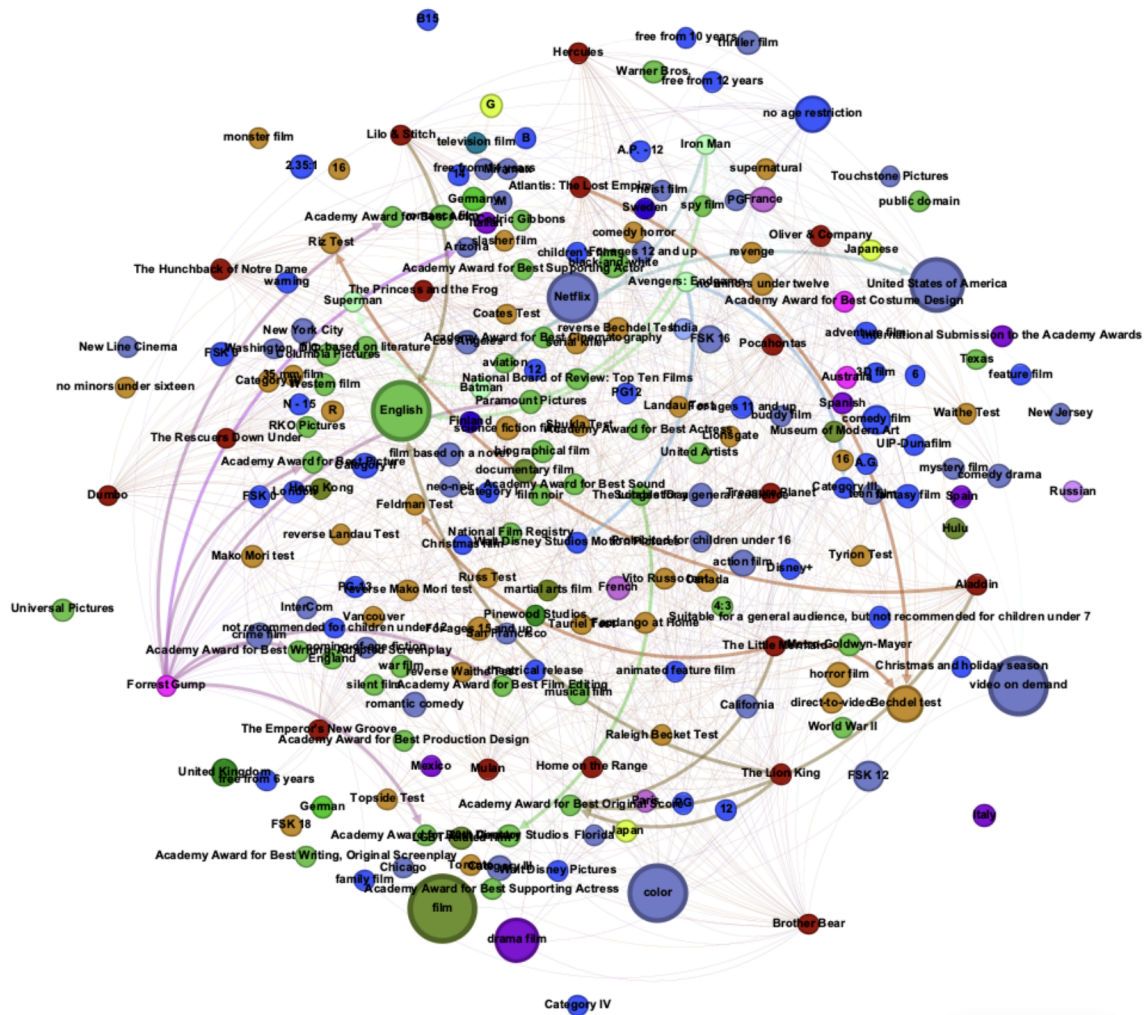


Figure 5.6. A fragment of the knowledge graph constructed for the MovieLens 20M dataset, where the degree of nodes is filtered between 200 and 17546.

When a more detailed analysis is made, Figure 5.4 illustrates how films are interconnected based on their genres, countries of origin, and other attributes, with genres such as drama, comedy, thriller, and action represented by lines connecting various films. Countries like the United States and France are linked to these films, and additional attributes, including black-and-white and video on demand, contribute to these connections. The central node Film serves as a hub that connects various genres, countries, and attributes, while another central node, United States of America, links many American films to different genres and characteristics.

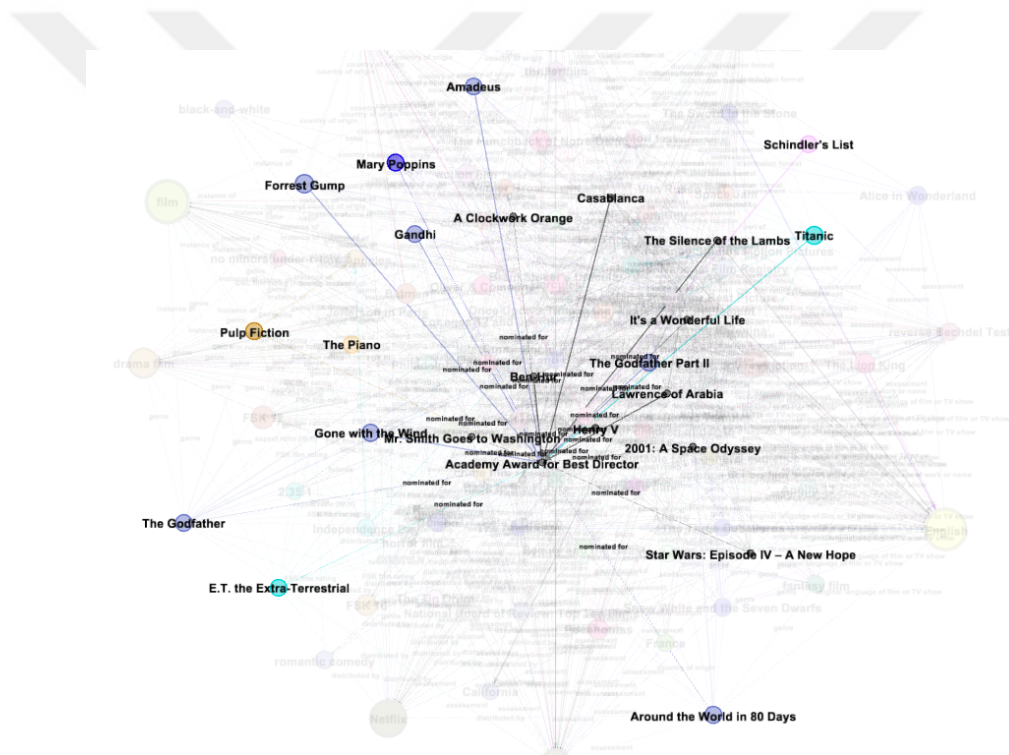


Figure 5.7. The visualization of the connection between movies and the Academy Award for Best Sound shows nodes related to movie titles that represent the most influential films deserving of awards for ml-100k dataset.

Genre clusters, such as those for drama films and comedy films, are formed by groups of films that share similar genres. Specific relationships are also evident, as seen with *The Godfather*, which is connected to both drama and crime genres, reflecting its themes, while *Pulp Fiction* is linked to drama, crime, and comedy, highlighting

its genre mix. Additionally, Forrest Gump connects to drama, comedy, and romance, showcasing its diverse emotional tone.

This interpretation is not limited to the filtered version of the knowledge graph (KG). When the degree filter for nodes is increased from 100 to 180 in Figure 5.7, new relationships emerge in the KG between movies and the Academy Award for Best Director. This type of connection is useful in recommendation scenarios, as it can suggest movies that users might enjoy watching for award-winning films. If we fetch additional connections from Wikidata, we are likely to discover new movie entities that deserve the same award, which could potentially be used as recommendations for new films.

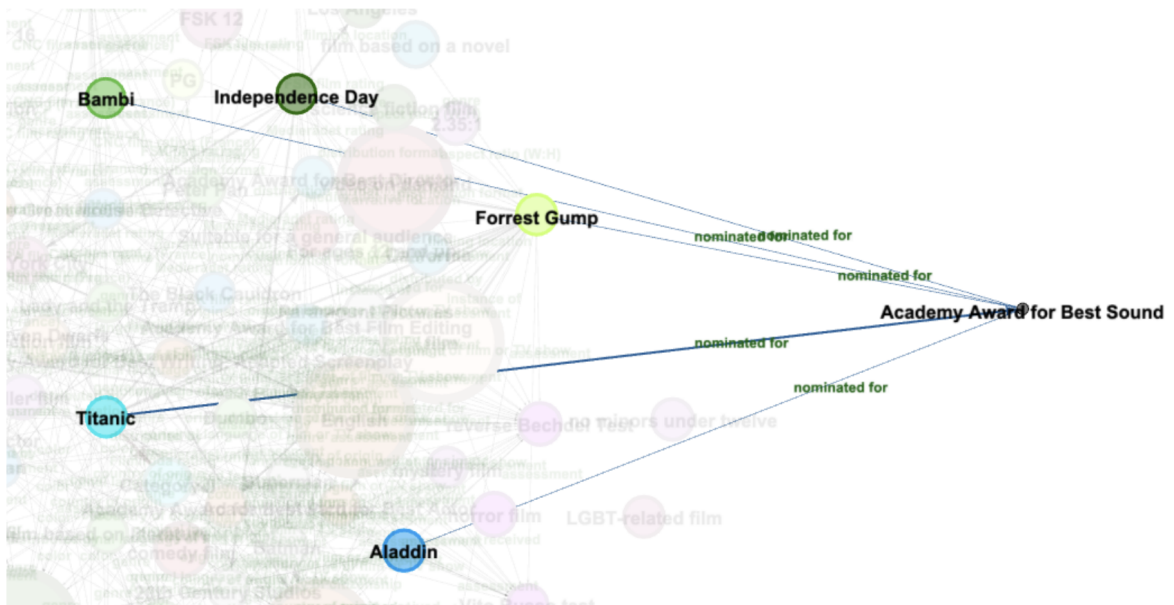


Figure 5.8. The visualization of the connection between movies and the Academy Award for Best Sound shows nodes related to movie titles that represent the most influential films deserving of awards for ml-1m dataset.

Figure 5.5, which is constructed for MovieLens 1M, is organized into different categories, such as adventure, drama, comedy, musical, thriller, and war, suggesting that the relationships between films are often based on their genres or themes. Sev-

eral nodes are connected to Academy Award nodes, which are visualized in Figure 5.8, indicating that these films have won or been nominated for Academy Award, highlighting the prestige and recognition associated with these films. The graph includes nodes representing production companies like Walt Disney Studios, Columbia Pictures, and Paramount Pictures, indicating that the relationships between films may also be influenced by their production companies. There are interconnected nodes representing films from the same series or franchise, such as *The Rescuers Down Under* and *The Little Mermaid*, suggesting that films within a series or franchise often share similar themes, characters, or storylines. Some nodes are labeled with countries or regions, such as United Kingdom and United States of America, indicating that the relationships between films may also be influenced by their cultural or regional origins. Additionally, nodes are labeled with ratings like FSK 16 and PG, suggesting that the relationships between films may be influenced by their target audience or content.

The graph in the Figure 5.6, designed for MovieLens 20m, highlights several film categories, including action, adventure, animation, comedy, drama, family, fantasy, horror, mystery, romance, sci-fi, thriller, and war. Awards such as the Academy Awards, BAFTA, Golden Globes, and National Board of Review awards are represented, indicating the critical acclaim received by certain films. Major production companies like Warner Bros., Disney, Pixar, and Paramount Pictures are clearly shown, indicating their strong impact on the film industry. The connections between films and their production companies show how these companies have been involved in producing a wide range of films across different genres. The lines connecting films indicate relationships between them, possibly based on shared actors, directors, production companies, or themes, while clusters of films suggest connections within specific genres or time periods.

Even though the filtered versions of the knowledge graphs (KGs) reflect strong connections between some entities, the unfiltered KGs contain many more connections, which are planned to be used as side features for the sequential recommendation model in the form of embedding vectors. Representing this kind of structural information in

a graph format allows us to model complex relationships from different perspectives. Our hypothesis is that the connections between movies, based on their characteristics, have the potential to positively impact the recommendation task.



5.3. Model Training

In this section, we focus on two key components: Knowledge Graph Embedding and Sequential Model Training. The Knowledge Graph Embedding subsection discusses the process of transforming entities and relationships from the knowledge graph constructed with Wikidata entities into continuous vector representations. The Sequential Model Training subsection explains the process of training models that leverage the time-ordered interactions between users and items. These models are designed to predict the next likely action or item of interest for users based on their historical interaction data.

5.3.1. Knowledge Graph Embedding

After constructing the knowledge graph using Wikidata entities, the next step is to generate embedding vectors that represent both entities and relationships within the graph. In this study, we only use the entity embeddings, which are movie entities in our case, to be fed into sequential recommendation architecture as additional feature vectors. Since the quality of embeddings depends on the effectiveness of the knowledge graph embedding method, we leverage four popular knowledge graph embedding models —TransH [34], TransE [32], TransD [36], and TransR [35]— each applying different geometric approaches to capture the graph’s structural and semantic information. The PyKEEN Python library [64] is for training and evaluating knowledge graph embeddings.

The PyKEEN Python library [64] is utilized for training and evaluating these knowledge graph embeddings. By using this library, we ensure efficient data processing and model training. Our hypothesis is that higher-quality embedding vectors will lead to an improvement in the performance of the sequential recommendation model, as the richer semantic representation of movie entities can better inform the recommendation process. We evaluate the success of incorporating these open-source linked data embeddings in our experiments section.

5.3.2. Sequential Model Training

For the sequential recommendation model, we use SRGNN [5] as a baseline model for our study. The model consists of a graph neural network (GNN) architecture and leverages gated recurrent units (GRUs) to propagate information between nodes, allowing the model to learn sequential dependencies effectively. We conduct our experiment in two phases: one using knowledge graph embedding vectors as features, obtained from different models, and the other without these embeddings.

The SRGNN model is designed to handle sequential recommendation tasks by modeling user behavior within individual sessions, treating these sessions as graph structures. Firstly, we convert the tabular data into sequences by grouping each user sorted by timestamp. Then, we convert this sequence data into directed graph structure where nodes represent movie that customer has watched and edges represents transitions between movies based on the order in which they appear in the user history.

Next, SRGNN applies a Graph Neural Network (GNN) layer to propagate information between the nodes. The GNN layer aggregates information from neighboring nodes through message passing, which allows the model to learn the relationships between items within the session. After message passing, Gated Recurrent Units (GRUs) are used to further refine the node representations by capturing the temporal dependencies in the session sequence. For the baseline model, we only use the node representations during training. In the feature-based variant, we combine the KGE embedding vectors with the node embedding vectors before passing them through the following steps. Once the GNN and GRUs have processed the graph, the model computes a representation by applying a global attention mechanism that weighs the importance of each item, resulting in an embedding that summarizes the interactions and focuses on key items likely to influence the next one. The model is then trained to predict the next item by minimizing cross-entropy loss between the predicted and actual next items, adjusting the item embeddings and GNN parameters accordingly.

6. EXPERIMENTS and RESULTS

6.1. Datasets

Since the aim of the study is to enrich the dataset by using open source knowledge bases and to increase the success of the sequential recommendation system architecture with this method, it is important to choose a dataset that corresponded to the entities in the dataset, especially in the knowledge bases. Therefore, we prefer to use MovieLens dataset [7] which was published by the GroupLens research laboratory in 1998. Datasets are available in various sizes; however, only the 1M and 20M-sized datasets were used in this study. These datasets share some similarities and differences. All datasets include the user ID, movie ID, rating (where users express their preferences with a discrete value between 0 and 5), and a timestamp.

The MovieLens 100K dataset consists of 100,000 ratings provided by 943 users on 1,682 movies. Each user in the dataset has rated at least 20 movies, allowing for an adequate understanding of individual user preferences. Unlike the larger datasets, MovieLens 100K does not include demographic data such as age, gender, or occupation. However, it still retains essential information about movies, including the title, release year, and associated genres, which are one of 19 unique genres. This smaller dataset has been widely used in recommender system research due to its balance of size and comprehensiveness.

MovieLens 1M dataset contains 1,000,209 anonymous ratings given by 6040 MovieLens users to 3706 movie. Each user in the dataset has at least 20 ratings which is important to understand the user preferences about movies. Besides other dataset, 1M version contains the user demographic data such as gender, age group and occupation. The movie file includes the title of the movie as it appears on IMDb, along with its release date and genre, which can be one of 18 unique potential genres.

Movielens 20M dataset contains 20,000,263 anonymous ratings given by 138,493 Movielens users to 27278 movies between January 09, 1995 and March 31, 2015. The same is true for the 20M dataset; each user had at least 20 ratings and was randomly selected for inclusion. However, this dataset does not include demographic data of the users. Unlike the other dataset, users can provide free-text tags for movies, consisting of single words or short phrases. The dataset contains a separate line for each tag given by the user for a movie. Users can express anything with these tags, such as "quirky" and "sad" for the mood of the movie, "Jason Bateman" for the actor, and "rise to power" to express the theme of the film. In addition to that, the dataset also contains tag genome [65] which is tag relevance scores for movies. After eliminating misspellings and infrequently used tags, a machine learning technique is applied to calculate the relevance score between the movie and tags, determining whether the tags are appropriate for the movie by leveraging the ratings, tags and textual reviews. Lastly, it contains the genres for the movies.

One important aspect of our study is our preference for using the Movielens dataset for experiments because it allows us to find entities for all movies on Wikidata. This capability enables easy linkage of movies to additional details available on Wikidata. By leveraging open-source data in this way, we can enrich the dataset with a rich of extra information not initially included, such as actors, directors, main subjects, and more. This potential opens avenues to uncover previously unknown relationships between films. Furthermore, storing this information in triplets format facilitates the construction of a knowledge graph. Although this study does not currently utilize more than one linked data, another advantage is the straightforward integration of other open-source linked datasets in the future. For these reasons, the features provided by the Movielens dataset are not used, despite its inclusion of various contextual features about users and movies.

6.2. Parameter Setting

For all knowledge graph embedding (KGE) methods, we employed consistent parameter settings: an embedding size of 64, a learning rate of 0.005, and a batch size of 256. For the sequential recommendation model, SRGNN, the parameters are configured as outlined in the original paper [5]: the dimensionality of latent vectors d is set to 100 and initialized with a Gaussian distribution (mean = 0, standard deviation = 0.1); the mini-batch Adam optimizer is used with an initial learning rate of 0.001, which decays by a factor of 0.1 every 3 epochs; the batch size is 100; and an L2 regularization penalty of 10^{-5} is applied.

6.3. Experiments Environment

The experiments were performed on a system equipped with an Intel(R) Xeon(R) Gold 6342 CPU @ 2.80GHz, 40 GB of RAM, and running Ubuntu 20.04.6 LTS with Linux kernel version 5.4.0-198-generic. GPU acceleration was achieved using an NVIDIA A40 GPU.

6.4. Evaluation Metrics

The success of the trained models is evaluated using Hit@K for both knowledge graph embedding and sequential recommendation experiments, as it provides a reliable measure of the accuracy of predicted rankings. In addition, for knowledge graph embedding experiments, we incorporate Mean Reciprocal Rank (MRR) to offer a more detailed evaluation of the ranking quality.

6.4.1. Hit@K

Hit@K is a widely used metric to evaluate the performance of recommendation systems and ranking models. It measures the proportion of times the correct item or entity appears within the top K recommendations. A higher Hit@K score indicates

that the model is more effective at ranking relevant items within the top positions. For each query, if the true relevant item is present among the top K results, it is considered a "hit". The Hit@K metric is calculated as follows:

$$\text{Hit@K} = \frac{1}{Q} \sum_{i=1}^Q \text{hit}_i, \quad (6.1)$$

where Q is the total number of queries and hit_i is a binary indicator that equals 1 if the relevant item is in the top K recommendations for the i -th query, and 0 otherwise. This metric provides a straightforward evaluation of the model's ability to rank relevant items higher in the list of recommendations.

6.4.2. Mean Reciprocal Rank (MRR)

Mean Reciprocal Rank (MRR) is a metric commonly used in information retrieval, recommendation systems, and ranking problems to assess the performance of models in producing relevant results. It is used to evaluate the effectiveness of ranking algorithms by measuring the average rank at which the first relevant item appears. For each query, MRR is computed as the reciprocal of the rank position of the first relevant item. The overall MRR score is the average of these reciprocal ranks across all queries:

$$\text{MRR} = \frac{1}{Q} \sum_{i=1}^Q \frac{1}{\text{rank}_i}, \quad (6.2)$$

where rank_i is the rank position of the first relevant item for the i -th query. MRR provides insight into how well the model ranks relevant items at the top of the list and is particularly useful in scenarios where it is important to identify the first relevant result quickly. In practice, MRR values range from 0 to 1. A value closer to 1 indicates that relevant items appear closer to the top of the ranking for most queries, while a value closer to 0 indicates that relevant items are ranked much lower or not at all.

6.5. Result and Analysis

6.5.1. Knowledge Graph Embedding Results

The performance of four knowledge graph embedding models (TransR, TransH, TransE, and TransD) was evaluated across three MovieLens datasets (ml-100k, ml-1m, and ml-20m) using the Hit@5, Hit@10, and MRR metrics. Results indicate that the TransE and TransD models consistently outperform TransR and TransH on most metrics, particularly on larger datasets.

Table 6.1. The performance of knowledge graph embedding models on the ml-100K, ml-1M, and ml-20M datasets for the TransR, TransH, TransE, and TransD models was evaluated using Hit@5, Hit@10, and MRR metrics.

Model	ml-100k			ml-1m			ml-20m		
	hit@5	hit@10	MRR	hit@5	hit@10	MRR	hit@5	hit@10	MRR
TransR	0.127	0.145	0.080	0.086	0.108	0.057	0.065	0.079	0.040
TransH	0.165	0.202	0.129	0.133	0.161	0.090	0.113	0.137	0.080
TransE	0.484	0.539	0.400	0.404	0.474	0.258	0.187	0.227	0.117
TransD	0.341	0.389	0.273	0.495	0.545	0.414	0.319	0.360	0.255

The hypothesis is that using high-quality embeddings as features in sequential recommendation methods will improve their performance. Translation-based models provide meaningful embeddings that contains patterns in user-item interactions. As a result, when combined with sequential recommendation techniques, these embeddings can lead to better recommendations because they include additional meaningful information.

For the ml-100k dataset, TransE achieved the highest scores with a notable Hit@5 of 0.484, Hit@10 of 0.539, and MRR of 0.400, making it the best performer in capturing relevant recommendations. Compared to TransD, which scored 0.341 on Hit@5, TransE demonstrated an increase of approximately 14.3 percentage points (48.4% vs. 34.1%).

Similarly, TransE’s Hit@10 score is 15 percentage points higher than TransD (53.9% vs. 38.9%). These improvements underscore TransE’s superiority in smaller datasets. TransR and TransH lagged behind on all metrics.

In the ml-1m dataset, TransD surpassed all other models, achieving the best Hit@5 of 0.495, Hit@10 of 0.545, and MRR of 0.414. TransD’s Hit@5 outperformed TransE by 9.1 percentage points (49.5% vs. 40.4%), while the Hit@10 showed a 7.1 percentage point increase (54.5% vs. 47.4%). These results suggest that TransD more effectively embeds knowledge graph structures in larger datasets, maintaining a substantial lead over other models.

In the ml-20m dataset, TransD again led with Hit@5 of 0.319, Hit@10 of 0.360, and MRR of 0.255, showing its scalability and robustness in handling complex datasets. TransE, although it excelled in smaller datasets, showed a significant decline in performance with Hit@5 dropping by 13.2 percentage points from ml-1m (from 40.4% to 18.7%), demonstrating that TransD’s strength grows with dataset size.

Overall, TransE dominates in the smaller ml-100k dataset, while TransD excels as dataset size increases, highlighting its capacity to handle larger and more complex datasets effectively.

6.5.2. Sequential Recommendation Experiment Results

The table 6.2 reports the results of the sequential recommendation models (SRGNN and its combinations with TransR, TransH, TransE, and TransD) evaluated using the Hit@10 and Hit@20 metrics across the ml-100k, ml-1m, and ml-20m datasets. For the ml-100k dataset, both SRGNN + TransE and SRGNN + TransD achieved the highest Hit@10 score of 0.137, slightly outperforming the base SRGNN model, which scored 0.132. In terms of Hit@20, SRGNN + TransE reached the highest score of 0.240, marginally surpassing SRGNN + TransD (0.236) and the base SRGNN (0.211). These results indicate that adding knowledge graph embeddings from TransE and TransD

models slightly improves the recommendation performance for smaller datasets.

In the ml-1m dataset, the SRGNN + TransD model led the performance, achieving the highest Hit@10 score of 0.149, followed closely by SRGNN + TransE (0.148). Both models performed significantly better than the base SRGNN, which scored 0.115, reflecting an improvement of around 3.3 percentage points. Similarly, SRGNN + TransD achieved the best Hit@20 score of 0.237, with SRGNN + TransE close behind at 0.235, both improving considerably upon the base SRGNN’s score of 0.191. These results underscore the effectiveness of combining SRGNN with TransE and TransD embeddings, especially for medium-sized datasets.

Table 6.2. The performance of sequential recommendation models on the ml-100K, ml-1M, and ml-20M datasets for SRGNN and its combinations with TransR, TransH, TransE, and TransD was evaluated using Hit@10 and Hit@20 metrics.

Model	ml-100k		ml-1m		ml-20m	
	hit@10	hit@20	hit@10	hit@20	hit@10	hit@20
SRGNN	0.132	0.211	0.115	0.191	0.079	0.129
SRGNN + TransR	0.125	0.200	0.142	0.227	0.075	0.126
SRGNN + TransH	0.133	0.215	0.137	0.219	0.084	0.132
SRGNN + TransE	0.137	0.240	0.148	0.235	0.085	0.139
SRGNN + TransD	0.137	0.236	0.149	0.237	0.086	0.142

For the ml-20m dataset, SRGNN + TransD slightly outperformed SRGNN + TransE, achieving Hit@10 and Hit@20 scores of 0.086 and 0.142, respectively, compared to 0.085 and 0.139 for SRGNN + TransE. Both models surpassed the base SRGNN, which had Hit@10 and Hit@20 scores of 0.079 and 0.129, respectively, demonstrating a small but consistent improvement in larger datasets.

Even though TransE is a less complex model with a lower computational cost compared to TransD, the difference in performance between these two models is minimal, which is somewhat contrary to expectations. This could be due to the fact that while TransE operates in a single shared embedding space, which limits its ability to

capture more complex relational structures, the dynamic mapping matrices of TransD offer flexibility but with minimal overhead in practice. As a result, both models can capture enough of the underlying relationships to provide comparable performance on medium to large datasets like ml-1m and ml-20m.

When comparing the results from the KGE experiments (Table 6.1) and sequential experiments (Table 6.2), it is clear that TransD emerges as the most effective model in both knowledge graph embedding and sequential recommendation tasks, particularly on the larger ml-1m and ml-20m datasets. Its consistent performance highlights its ability to handle more complex datasets effectively. TransE, while demonstrating strong performance in the smaller ml-100k dataset in both experiments, shows a noticeable decline as the dataset size increases, where TransD becomes the more scalable model.

The combination of SRGNN with TransR and TransH yielded only marginal improvements compared to the base model, while the combinations with TransE and TransD consistently led to better results, reflecting the higher quality of embeddings provided by these models. Overall, TransD proved to be the best model in larger datasets, aligning with its performance in knowledge graph embedding tasks, while TransE excels in smaller datasets, maintaining consistency across both embedding and recommendation tasks.

The improvements observed in recommendation performance can be attributed to the use of knowledge graph embedding vectors as features in the sequential recommendation models. Knowledge graphs constructed from Wikidata triples about movies encapsulate rich and meaningful information about entities and their relationships. These embeddings allow for a more comprehensive representation of user-item interactions by capturing complex patterns and connections that might not be explicitly present in the raw data. By representing this information in low-dimensional vectors, knowledge graph embeddings provide a more structured and semantically enriched feature space, enhancing the model’s ability to make more accurate recommendations. This demonstrates that incorporating knowledge graph embeddings into sequential rec-

ommendation systems effectively improves their predictive power, especially in more complex datasets where relational information plays a crucial role. Furthermore, we demonstrate the potential of using open-source knowledge bases to enhance recommendation systems by providing insights for future research and practical applications.

6.5.3. Qualitative Analysis

In the qualitative analysis section, we present a test case example shown in Figure 6.1. The first row displays the last five movies in the customer’s watch history. The movie on the right-hand side represents the ground-truth prediction, which was watched by the same user after the last five movies.

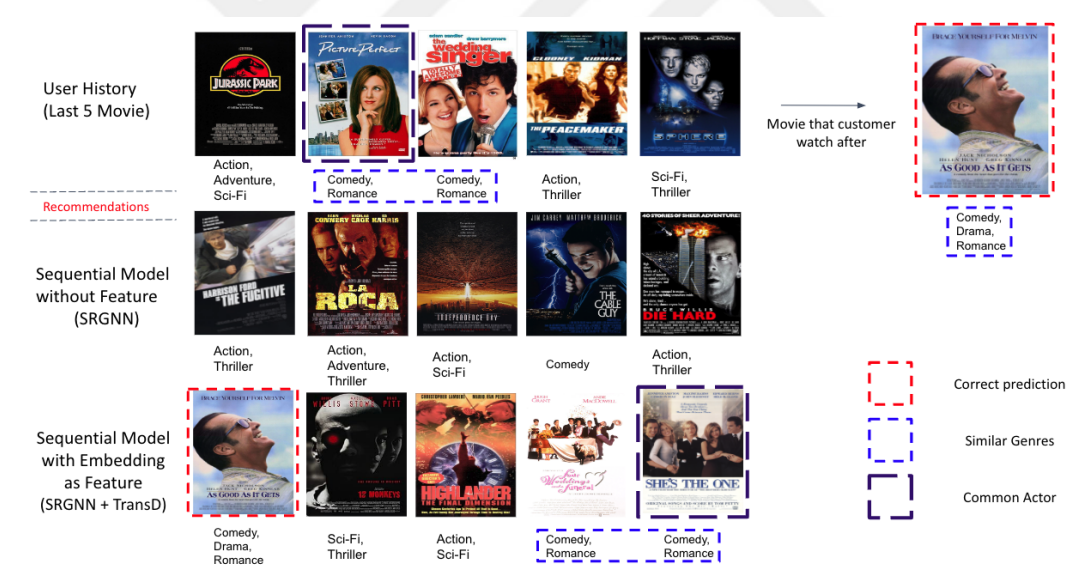


Figure 6.1. Example of a recommendation test case: the first row shows the user’s last five movies, the second row displays recommendations from the baseline model, and the third row presents recommendations from one of the combined model versions.

The second row, labeled "Sequential Model without Feature (SRGNN)", shows the recommendation list from the baseline model, which does not use any features in its training. The third row, titled "Sequential Model with Embedding as Feature (SRGNN + TransD)", represents one of the combined versions of our sequential recommendation

model, which incorporates knowledge graph embedding using the TransD method. We analyzed our results in several ways.

Three important points should be highlighted:

- Ground-truth prediction: The proposed model’s recommendation list includes the ground-truth movie in the top position, while the baseline recommendation does not feature it among the first five movies.
- Recommending more suitable genres: In examining the genre distribution of the customer’s watch history, we note that two of the last five movies belong to the Comedy and Romance genres, which is also the genre of the ground-truth prediction. In the proposed model’s recommendations, two out of the five recommended movies share the Comedy and Romance genres, whereas the baseline recommendation list contains only one movie in the Comedy genre, which does not match the Comedy and Romance genres.
- Movies with the same actor: One of the movies in the customer’s watch history is ”Picture Perfect”, which features Jennifer Aniston. In the proposed model’s recommendations, there is a movie titled ”She’s the One”, which also stars Jennifer Aniston.

Incorporating Linked Open Data through Knowledge Graphs into sequential recommendation models significantly enhances the quality and relevance of recommendations. By integrating embedding vectors generated from Knowledge Graph Embedding models, the sequential recommendation system can capture rich semantic relationships and contextual information about users and items. The qualitative analysis of the recommendation results demonstrates that the proposed model significantly outperforms the baseline. Notably, it successfully prioritizes the ground-truth movie in its recommendations, aligning closely with user preferences and enhancing the viewing experience. Furthermore, the model effectively captures genre distribution in the user’s watch history, offering two recommended movies from the Comedy and Romance genres, which aligns with the user’s tastes, while the baseline model only suggests one movie

with a limited genre classification. Additionally, the proposed model’s ability to recommend movies featuring the same actor, such as Jennifer Aniston, indicates a sophisticated understanding of user interests. Overall, these strengths—prioritizing relevant content, recommending suitable genres, and leveraging familiar actors—underscore the model’s effectiveness in delivering personalized recommendations that foster deeper user engagement. We can infer that the relations in the Knowledge Graph constructed for movie datasets enhance the connections between movies regarding actors, genres, and other information, thereby positively impacting the quality of recommendations.



7. CONCLUSION & FUTURE WORK

In this study, we explored the integration of knowledge graph embeddings (KGE) into sequential recommendation models to enhance predictive performance. By leveraging session-based graph neural networks (SRGNN) and augmenting the node representations with KGE vectors, we aimed to capture both sequential patterns and the rich semantic relationships present in knowledge graphs. Our experiments evaluated several KGE models, including TransE, TransH, TransD, and TransR, and demonstrated their impact on improving the recommendation accuracy when compared to baseline models that based on only item interactions. Also, we demonstrated the utilization effect of Linked Open Data (LOD) by enriching the recommendation model with external information such as movie attributes, directors, and genres from knowledge bases like Wikidata. This external data enabled the model to understand deeper semantic relationships between items, contributing to more informed and accurate predictions, especially in cases where direct interaction data was limited or unavailable. The integration of LOD not only improved model accuracy but also provided a scalable way to introduce diverse and rich data sources into the recommendation pipeline.

We can conclude and summarize the outcomes of the study under two headings:

- **Enhanced Feature Set and Recommendations:** The primary advantage of integrating knowledge graph embeddings (KGEs) into the SRGNN model is the ability to capture both semantic relationships between items and sequential patterns of user behavior. Traditional sequential recommendation models, like SRGNN, rely heavily on temporal order and interaction frequency, which can limit their understanding of deeper item connections. By incorporating KGEs, we enhanced the model’s feature set, allowing it to not only learn from the order of interactions but also from the rich semantic links between items, such as shared genres, directors, or other attributes obtained from the knowledge graph.

In our study, this dual approach of combining sequential and semantic knowl-

edge significantly improved the model’s performance. The experiments demonstrated that the SRGNN model enriched with KG embeddings from various models—TransE, TransH, TransD, and TransR—was able to better distinguish between items that share underlying semantic properties. This enhanced feature set improved the overall learning process, enabling the model to make more informed predictions by leveraging both the graph structure of the session and the meaningful connections between items. We also observed this outcome on quantitative analysis in recommendations.

- **Leveraging Linked Open Data:** Another key contribution of this thesis is the demonstration of the value that Linked Open Data (LOD) can bring to recommendation models. We utilized open-source knowledge bases, such as Wikidata, to generate embeddings that enriched the feature space of the SRGNN model. By linking movies in our dataset to Wikidata entities and extracting valuable metadata (e.g., genres, directors, cast), we introduced an external layer of information that was not available in the raw user interaction data.

Our study showed that embeddings generated from open linked data not only added diversity to the model’s understanding but also provided it with structured and reliable information. This additional knowledge allowed the model to discover latent relationships between items, such as thematic or collaborative connections (e.g., movies directed by the same person or with similar storylines). The result is more personalized and contextually aware recommendation system that could suggest movies based on not only past watching history, also the characteristic of an items such as actor and genres.

The following areas present promising directions for future research in enhancing recommendation systems using knowledge graph embeddings:

- **Investigate Higher-Order Relationships:** Future work can focus on studying how deeper relationships within Wikidata impact recommendation quality. By examining higher-order relationships, we can gain insights into the potential for improved recommendations. Understanding these deeper connections may reveal

new patterns in user behavior and item similarities, leading to more contextually relevant recommendations. However, it might lead to an increase in the size of the data, which could bring challenges such as complexity in model training and data quality management which should be handled carefully.

- **Integrate Domain-Specific Data:** Another area for future research is to add information from domain-specific databases or ontologies to Knowledge Graph. This addition could help fill in gaps where there are missing entities and relationships that are important for certain recommendation tasks. For instance, while finding corresponding movie entities in Wikidata is relatively straightforward, obtaining entities for other domains, such as grocery products, is significantly more challenging due to issues like inconsistent naming conventions and region-specific variations. These challenges complicate the integration of additional entities in scenarios like product recommendations. By enriching the knowledge graph with data from these specialized sources, we can enhance the model's ability to grasp important details relevant to the domain.
- **Evaluate Embedding in Different Architectures:** We need to further evaluate knowledge graph embeddings in different model structures to find the best setup for sequential recommendations. By looking at how various embedding methods work with different recommendation systems, we can discover the configurations that give the best predictions and user satisfaction. This study can help guide the design of future models, which may lead to important improvements in recommendation systems. Additionally, we can explore other recommendation model approaches, such as transformers or large language models (LLMs), by incorporating the same feature embeddings, as these models may offer improved performance through their ability to capture complex patterns in data.

REFERENCES

1. Quadrana, M., P. Cremonesi and D. Jannach, “Sequence-Aware Recommender Systems”, *ACM Computing Surveys (CSUR)*, Vol. 51, No. 4, pp. 1–36, 2018.
2. Musto, C., P. Basile, P. Lops, M. de Gemmis and G. Semeraro, “Introducing Linked Open Data in Graph-based Recommender Systems”, *Information Processing & Management*, Vol. 53, No. 2, pp. 405–435, 2017.
3. Bollacker, K., C. Evans, P. Paritosh, T. Sturge and J. Taylor, “Freebase: A Collaboratively Created Graph Database for Structuring Human Knowledge”, *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pp. 1247–1250, Vancouver, British Columbia, Canada, 2008.
4. Vrandečić, D. and M. Krötzsch, “Wikidata: A Free Collaborative Knowledgebase”, *Communications of the ACM*, Vol. 57, No. 10, pp. 78–85, 2014.
5. Wu, S., Y. Tang, Y. Zhu, L. Wang, X. Xie and T. Tan, “Session-Based Recommendation with Graph Neural Networks”, *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33, pp. 346–353, USA, 2019.
6. Li, Y., D. Tarlow, M. P. Zinkevich and K. Swersky, “Gated Graph Sequence Neural Networks”, *International Conference on Learning Representations*, pp. 123–134, San Juan, Puerto Rico, 2016.
7. Harper, F. M. and J. A. Konstan, “The Movielens Datasets: History and Context”, *ACM Transactions on Interactive Intelligent Systems*, Vol. 5, No. 4, pp. 1–19, 2015.
8. Wu, S., F. Sun, W. Zhang, X. Xie and B. Cui, “Graph Neural Networks in Recommender Systems: A Survey”, *ACM Computing Surveys*, Vol. 55, No. 5, pp. 1–37, 2022.

9. Hamilton, W. L., *Graph Representation Learning*, Morgan & Claypool Publishers, San Rafael, California, 2020.
10. Ying, R., R. He, K. Chen, P. Eksombatchai, W. L. Hamilton and J. Leskovec, “Graph Convolutional Neural Networks for Web-scale Recommender Systems”, *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 974–983, London, UK, 2018.
11. Wang, X., X. He, M. Wang, F. Feng and T.-S. Chua, “Neural Graph Collaborative Filtering”, *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 165–174, Paris, France, 2019.
12. He, X., K. Deng, X. Wang, Y. Li, Y. Zhang and M. Wang, “LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation”, *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 639–648, Xian, China, 2020.
13. Veličković, P., G. Cucurull, A. Casanova, A. Romero, P. Lio and Y. Bengio, “Graph Attention Networks”, *arXiv preprint arXiv:1710.10903*, 2017.
14. Elahi, E. and Z. Halim, “Graph Attention-Based Collaborative Filtering for User-Specific Recommender System Using Knowledge Graph and Deep Neural Networks”, *Knowledge and Information Systems*, Vol. 64, No. 9, pp. 2457–2480, 2022.
15. Feng, C., Z. Liu, S. Lin and T. Q. Quek, “Attention-Based Graph Convolutional Network for Recommendation System”, *ICASSP IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 7560–7564, Brighton, UK, 2019.
16. Eksombatchai, C., P. Jindal, J. Z. Liu, Y. Liu, R. Sharma, C. Sugnet, M. Ulrich and J. Leskovec, “Pixie: A System for Recommending 3+ Billion Items to 200+

- Million Users in Real-time”, *Proceedings of the World Wide Web Conference*, pp. 1775–1784, Lyon, France, 2018.
17. De Nadai, M., F. Fabbri, P. Gigioli, A. Wang, A. Li, F. Silvestri, L. Kim, S. Lin, V. Radosavljevic, S. Ghael *et al.*, “Personalized Audiobook Recommendations at Spotify Through Graph Neural Networks”, *Companion Proceedings of the ACM on Web Conference*, pp. 403–412, Singapore, 2024.
 18. Chen, C., W. Ma, M. Zhang, Z. Wang, X. He, C. Wang, Y. Liu and S. Ma, “Graph Heterogeneous Multi-relational Recommendation”, *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35, pp. 3958–3966, Virtual Conference, 2021.
 19. Yi, X., J. Yang, L. Hong, D. Z. Cheng, L. Heldt, A. Kumthekar, Z. Zhao, L. Wei and E. Chi, “Sampling-Bias-Corrected Neural Modeling for Large Corpus Item Recommendations”, *Proceedings of the 13th ACM Conference on Recommender Systems*, pp. 269–277, Copenhagen, Denmark, 2019.
 20. Fang, H., D. Zhang, Y. Shu and G. Guo, “Deep Learning for Sequential Recommendation: Algorithms, Influential Factors, and Evaluations”, *ACM Transactions on Information Systems (TOIS)*, Vol. 39, No. 1, pp. 1–42, 2020.
 21. Rendle, S., C. Freudenthaler and L. Schmidt-Thieme, “Factorizing Personalized Markov Chains for Next-Basket Recommendation”, *Proceedings of the 19th International Conference on World Wide Web*, pp. 811–820, North Carolina, USA, 2010.
 22. Jannach, D. and M. Ludewig, “When Recurrent Neural Networks Meet the Neighborhood for Session-Based Recommendation”, *Proceedings of the Eleventh ACM Conference on Recommender Systems*, pp. 306–310, Como, Italy, 2017.
 23. Hidasi, B., M. Quadrana, A. Karatzoglou and D. Tikk, “Parallel Recurrent Neural

- Network Architectures for Feature-Rich Session-Based Recommendations”, *Proceedings of the 10th ACM Conference on Recommender Systems*, pp. 241–248, Boston, MA, 2016.
24. Li, J., P. Ren, Z. Chen, Z. Ren, T. Lian and J. Ma, “Neural Attentive Session-Based Recommendation”, *Proceedings of the ACM on Conference on Information and Knowledge Management*, pp. 1419–1428, Singapore, 2017.
 25. Liu, Q., Y. Zeng, R. Mokhosi and H. Zhang, “STAMP: Short-Term Attention/memory Priority Model for Session-based Recommendation”, *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 1831–1839, London, UK, 2018.
 26. Kang, W.-C. and J. McAuley, “Self-Attentive Sequential Recommendation”, *IEEE International Conference on Data Mining (ICDM)*, pp. 197–206, Singapore, 2018.
 27. Sun, F., J. Liu, J. Wu, C. Pei, X. Lin, W. Ou and P. Jiang, “BERT4Rec: Sequential Recommendation with Bidirectional Encoder Representations from Transformer”, *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pp. 1441–1450, Beijing, China, 2019.
 28. Fu, Z., C. Wang and J. Xu, “Graph Contextualized Self-attention Network for Software Service Sequential Recommendation”, *Future Generation Computer Systems*, Vol. 149, pp. 509–517, 2023.
 29. Wang, Z., W. Wei, G. Cong, X.-L. Li, X.-L. Mao and M. Qiu, “Global Context Enhanced Graph Neural Networks for Session-Based Recommendation”, *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 169–178, Xian, China, 2020.
 30. Chen, C., J. Guo and B. Song, “Dual Attention Transfer in Session-Based Recommendation with Multi-Dimensional Integration”, *Proceedings of the 44th Inter-*

- national ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 869–878, Montréal, Quebec, 2021.
31. Wang, B. and W. Cai, “Knowledge-Enhanced Graph Neural Networks for Sequential Recommendation”, *Information*, Vol. 11, No. 8, p. 388, 2020.
 32. Bordes, A., N. Usunier, A. Garcia-Duran, J. Weston and O. Yakhnenko, “Translating Embeddings for Modeling Multi-Relational Data”, *Advances in Neural Information Processing Systems*, Vol. 26, pp. 2787–2795, 2013.
 33. Wang, Q., Z. Mao, B. Wang and L. Guo, “Knowledge Graph Embedding: A Survey of Approaches and Applications”, *IEEE Transactions on Knowledge and Data Engineering*, Vol. 29, No. 12, pp. 2724–2743, 2017.
 34. Wang, Z., J. Zhang, J. Feng and Z. Chen, “Knowledge Graph Embedding by Translating on Hyperplanes”, *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 28, 2014.
 35. Lin, Y., Z. Liu, M. Sun, Y. Liu and X. Zhu, “Learning Entity and Relation Embeddings for Knowledge Graph Completion”, *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 29, pp. 2181 – 2187, Québec, Canada, 2015.
 36. Ji, G., S. He, L. Xu, K. Liu and J. Zhao, “Knowledge Graph Embedding via Dynamic Mapping Matrix”, *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pp. 687–696, Beijing, China, 2015.
 37. Nickel, M., V. Tresp and H.-P. Kriegel, “A Three-Way Model for Collective Learning on Multi-Relational Data”, *Proceedings of the 28th International Conference on International Conference on Machine Learning*, p. 809–816, Madison, WI, USA, 2011.
 38. Yang, B., W. tau Yih, X. He, J. Gao and L. Deng, “Embedding Entities and Re-

- lations for Learning and Inference in Knowledge Bases”, *International Conference on Learning Representations*, Banff, Canada, 2014.
39. Trouillon, T., J. Welbl, S. Riedel, É. Gaussier and G. Bouchard, “Complex Embeddings for Simple Link Prediction”, *International Conference on Machine Learning*, pp. 2071–2080, New York City, NY, USA, 2016.
 40. Dettmers, T., P. Minervini, P. Stenetorp and S. Riedel, “Convolutional 2D Knowledge Graph Embeddings”, *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 32, pp. 1811–1818, New Orleans, Louisiana, USA, 2018.
 41. Schlichtkrull, M., T. N. Kipf, P. Bloem, R. Van Den Berg, I. Titov and M. Welling, “Modeling Relational Data with Graph Convolutional Networks”, *The Semantic Web: 15th International Conference*, pp. 593–607, Heraklion, Crete, Greece, 2018.
 42. Sun, Z., Z.-H. Deng, J.-Y. Nie and J. Tang, “RotatE: Knowledge Graph Embedding by Relational Rotation in Complex Space”, *arXiv preprint arXiv:1902.10197*, pp. 1–18, 2019.
 43. Zhang, J.-C., A. M. Zain, K.-Q. Zhou, X. Chen and R.-M. Zhang, “A Review of Recommender Systems Based on Knowledge Graph Embedding”, *Expert Systems with Applications*, p. 123876, 2024.
 44. Zhang, F., N. J. Yuan, D. Lian, X. Xie and W.-Y. Ma, “Collaborative Knowledge Base Embedding for Recommender Systems”, *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 353–362, San Francisco, CA, 2016.
 45. Wang, H., F. Zhang, M. Zhang, J. Leskovec, M. Zhao, W. Li and Z. Wang, “Knowledge-Aware Graph Neural Networks with Label Smoothness Regularization for Recommender Systems”, *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 968–977, Anchorage, AK,

2019.

46. Wang, X., X. He, Y. Cao, M. Liu and T.-S. Chua, “KGAT: Knowledge Graph Attention Network for Recommendation”, *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 950–958, Anchorage, AK, USA, 2019.
47. Sun, R., X. Cao, Y. Zhao, J. Wan, K. Zhou, F. Zhang, Z. Wang and K. Zheng, “Multi-Modal Knowledge Graphs for Recommender Systems”, *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pp. 1405–1414, Beijing, China, 2020.
48. Cao, Y., X. Wang, X. He, Z. Hu and T.-S. Chua, “Unifying Knowledge Graph Learning and Recommendation: Towards a Better Understanding of User Preferences”, *The World Wide Web Conference*, pp. 151–161, San Francisco, California, USA, 2019.
49. Ji, S., S. Pan, E. Cambria, P. Marttinen and S. Y. Philip, “A Survey on Knowledge Graphs: Representation, Acquisition, and Applications”, *IEEE Transactions on Neural Networks and Learning Systems*, Vol. 33, No. 2, pp. 494–514, 2021.
50. Bondy, J. A. and U. S. R. Murty, *Graph Theory*, Springer Publishing Company, Incorporated, New York, 2008.
51. Scarselli, F., M. Gori, A. C. Tsoi, M. Hagenbuchner and G. Monfardini, “The Graph Neural Network Model”, *IEEE Transactions on Neural Networks*, Vol. 20, No. 1, pp. 61–80, 2008.
52. Bruna, J., W. Zaremba, A. Szlam and Y. LeCun, “Spectral Networks and Locally Connected Networks on Graphs”, *arXiv preprint arXiv:1312.6203*, 2013.
53. Kipf, T. N. and M. Welling, “Semi-Supervised Classification with Graph Convolutional Networks”, *arXiv preprint arXiv:1609.02907*, 2016.

54. Chen, J., T. Ma and C. Xiao, “FastGCN: Fast Learning with Graph Convolutional Networks via Importance Sampling”, *arXiv preprint arXiv:1801.10247*, 2018.
55. Lee, J. B., R. Rossi and X. Kong, “Graph Classification Using Structural Attention”, *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 1666–1674, London, UK, 2018.
56. Ruiz, L., F. Gama and A. Ribeiro, “Gated Graph Recurrent Neural Networks”, *IEEE Transactions on Signal Processing*, Vol. 68, pp. 6303–6318, 2020.
57. Hajiramezanali, E., A. Hasanzadeh, K. Narayanan, N. Duffield, M. Zhou and X. Qian, “Variational Graph Recurrent Neural Networks”, *Advances in Neural Information Processing Systems*, Vol. 32, pp. 10701–10711, 2019.
58. Xu, K., W. Hu, J. Leskovec and S. Jegelka, “How Powerful Are Graph Neural Networks?”, *arXiv preprint arXiv:1810.00826*, 2018.
59. LeCun, Y., Y. Bengio *et al.*, “Convolutional Networks for Images, Speech, and Time Series”, *The Handbook of Brain Theory and Neural Networks*, Vol. 3361, No. 10, p. 1995, 1995.
60. Chen, M., Z. Wei, Z. Huang, B. Ding and Y. Li, “Simple and Deep Graph Convolutional Networks”, *International Conference on Machine Learning*, pp. 1725–1735, Vienna, Austria, 2020.
61. Zhang, S., H. Tong, J. Xu and R. Maciejewski, “Graph Convolutional Networks: A Comprehensive Review”, *Computational Social Networks*, Vol. 6, No. 1, pp. 1–23, 2019.
62. Vaswani, A., N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser and I. Polosukhin, “Attention Is All You Need”, *Proceedings of the 31st International Conference on Neural Information Processing Systems*, p. 6000–6010, Curran Associates Inc., Red Hook, NY, USA, 2017.

63. Allemang, D. and J. Hendler, *Semantic Web for the Working Ontologist: Effective Modeling in RDFS and OWL*, Elsevier, Burlington, Massachusetts, 2011.
64. Ali, M., M. Berrendorf, C. T. Hoyt, L. Vermue, S. Sharifzadeh, V. Tresp and J. Lehmann, “PyKEEN 1.0: A Python Library for Training and Evaluating Knowledge Graph Embeddings”, *Journal of Machine Learning Research*, Vol. 22, No. 82, pp. 1–6, 2021.
65. Vig, J., S. Sen and J. Riedl, “The Tag Genome: Encoding Community Knowledge to Support Novel Interaction”, *ACM Transactions on Interactive Intelligent Systems (TiiS)*, Vol. 2, No. 3, pp. 1–44, 2012.
66. Wang, S., L. Hu, Y. Wang, L. Cao, Q. Z. Sheng and M. Orgun, “Sequential Recommender Systems: Challenges, Progress and Prospects”, *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, pp. 6332–6338, International Joint Conferences on Artificial Intelligence Organization, Macao, China, 2019.
67. Wang, H., F. Zhang, X. Xie and M. Guo, “DKN: Deep Knowledge-Aware Network for News Recommendation”, *Proceedings of the 2018 World Wide Web Conference*, pp. 1835–1844, Lyon, France, 2018.

APPENDIX A: FIGURES USED IN THESIS

The figures that emerged as part of this thesis study and whose copyrights were transferred to the publishing house have been used in the thesis book in accordance with the publishing house's own web page policy regarding the 'reuse of articles and graphics produced by the author'.

