Performance Enhancement of Unified Recommendation and Knowledge Graph Completion Learning by Relation Rotation

Lama Khalil
University of Windsor

Performance Enhancement of Unified Recommendation and Knowledge Graph Completion Learning by Relation Rotation

By

Lama Khalil

A Thesis
Submitted to the Faculty of Graduate Studies
through the School of Computer Science
in Partial Fulfillment of the Requirements for
the Degree of Master of Science
at the University of Windsor

Windsor, Ontario, Canada

2023

Performance Enhancement of Unified Recommendation and Knowledge Graph Completion Learning by Relation Rotation

by

Lama Khalil

APPROVED BY:

M. Hassanzadeh
Department of Electrical and Computer Engineering

K. Selvarajah
School of Computer Science

Z. Kobti, Advisor
School of Computer Science

April 4, 2023

DECLARATION OF ORIGINALITY

ABSTRACT

Advances in multi-task learning (MTL) models have improved the performance and explainability of recommender systems (RS) by jointly learning the recommendation and knowledge graph completion (KGC) tasks. Recent studies have established that considering the incomplete nature of knowledge graphs (KG) can further enhance the performance of RS. However, most existing MTL models depend on translation-based knowledge graph embedding (KGE) methods for KGC, which cannot capture various relation patterns, including composition relations that are prevalent in real-world KG.

To address this limitation, this thesis proposes a new MTL model, named rotational knowledge-enhanced translation-based user preference (RKTUP). RKTUP enhances the KGC task by incorporating rotational-based KGE techniques (RotatE or HRotatE) to model and infer diverse relation patterns. These relation patterns include symmetry/asymmetry, composition, and inversion. RKTUP is an advanced variant of the knowledge-enhanced translation-based user preference (KTUP) MTL model, which provides interpretations of its recommendations.

The experimental results demonstrate that RKTUP outperforms existing methods and achieves state-of-the-art performance on both recommendation and KGC tasks. Specifically, it shows a 13.7% and 11.6% improvement in F1 score for recommendations on DBbook2014 and MovieLens-1m, respectively, and a 12.8% and 13.6% increase in hit ratio for KGC on the same datasets, respectively.

The use of RotatE improves the two tasks' performance, while HRotatE enhances the two tasks' performance and the model's efficiency.

DEDICATION

This thesis is dedicated to my precious daughter Nora, whose arrival in my life during my master's degree pursuit reminded me of the importance of being a strong and determined role model. Her presence has been an inspiration and motivation for me to push through the challenges. To my loving husband, Mark, who is always my constant source of support and encouragement, and who believed in me even when I doubted myself. To my late father, Saeed, who instilled in me the value of education and the belief that nothing is impossible through hard work and determination. His guidance and love will forever live within me. Also, I am deeply indebted to my mother, Zehrie and to my siblings for their constant love and for providing a strong foundation throughout my life and academic journey.

AKNOWLEDGEMENTS

I would like to express my gratitude to everyone who assisted me in completing this thesis. Especially, I would like to convey my most sincere appreciation to my thesis advisor, Dr. Ziad Kobti, for his guidance, support and mentoring throughout my master's program. He has been more than just a mentor, but a true inspiration to me, as well. With his extensive knowledge and experience in the field, he always challenged me to think critically and pushed me to go beyond my limits. His guidance and constructive feedback were crucial in helping me to shape my thesis into something that has become a source of deep self-esteem by giving me a strong sense of accomplishment.

I would also like to thank my thesis committee, Dr. Kalyani Selvarajah, and Dr. Mohammad Hassanzadeh. I am greatly appreciative of their availability and valuable feedback. Additionally, I am particularly grateful to Dr. Selvarajah for sparking my interest in knowledge graphs.

Moreover, I would like to acknowledge Akshay Mukundbhai Shah for his insightful work in the field of knowledge graph completion.

Lastly, I offer my heartfelt thanks to my classmates Regina Khalil, Kamonashish Saha, and Mrulay Mistry for their encouragement, and all the help they provided during the course of this program. I have been blessed to have such wonderful and supportive peers who have made this journey an enjoyable and meaningful experience. My humble gratitute is extended to all who helped to make my endeavour rewarding beyond anything that mere words can express.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

KG	Knowledge Graph
KGC	Knowledge Graph Completion
RS	Recommender System
MTL	Multi-task Learning
CF	Collaborative Filtering
KGE	Knowledge Graph Embedding
KTUP	Knowledge-enhanced Translation-based User Preference
RKTUP	Rotational Knowledge-enhanced Translation-based User Preference
AI	Artificial Intelligence
MR	Mean Rank
Precis.	Precision
Hit	Hit Ratio
nDCG	Normalized Discounted Cumulative Gain
CNN	Convolutional Neural Network
F1	F1 Score
FN	False Negative
FP	False Positive
TP	True Positive
TN	True Negative

CHAPTER 1

Introduction

Recommender systems (RS) are a type of information filtering system that uses historical data to recommend one or more unobserved items to a particular user based on their past behaviour and predicted preference [8]. RS are widely used in various applications such as e-commerce, online advertising, social media, and many more [39]. There are several types of RS algorithms, each with its unique approach to making recommendations [20].

The input to various recommendation algorithms is called the user-item interaction list [8]. It is typically represented as a matrix, where rows represent users, columns represent items, and each cell contains the interaction between the corresponding user and item [55]. These interactions can take many forms, depending on the specific application, but commonly include things like ratings and written reviews (known as explicit interactions) and actions such as purchases, views, and clicks (known as implicit interactions) [68].

The main idea behind the RS's algorithms is to use the observed interactions in the user-item interaction list to infer the preferences of users for items with which they have not yet interacted [65].

User interactions provide valuable information about the preferences and opinions of users, which can be used to make more accurate, personalized and relevant recommendations based on the items that users with similar preferences have liked in the past [39]. However, one of the biggest challenges the RSs face is data sparsity, which is the lack of reviews, feedback and information about the users and items [49]. This results in a large number of empty cells in the user-item interaction matrix, leading to a decrease in the performance of the RS.

Many research papers have focused on resolving some of the challenges RSs face, including data sparsity and decreased performance, by using knowledge graphs (KG). Nevertheless, most of these studies did not consider the incomplete nature of KGs [27]. A few papers have taken this into account by proposing the use of multi-task learning (MTL) models, which jointly learn both the knowledge graph completion (KGC) and the RS algorithms to enhance the performance of both tasks [11, 41]. The majority of these models use TransH [71] and TransR [45] for the KGC task. These translation-based knowledge graph embedding (KGE) approaches fail to capture different relation patterns, such as composition relations, which are prevalent in real-world KGs [32]. We suspect that the more relation patterns the KGE model can infer, the better the representation of the user's preference, ultimately enhancing the RS's performance.

This thesis aims to investigate the effect of capturing a wide range of relation patterns, including composition relations, on the performance of the RS in an MTL model that provides explanations for its recommendations. One possible solution is to improve the entity and relation representations in the KG by utilizing rotation-based KGE models.

1.0.1 Recommender System

This is a classification task where the system recommends the top-N items for a target user in the user–item interaction list $y = \{(u, i)\}$. The pair (u, i) means that user $u \in U$ interacted with item $i \in I$ by clicking, watching, purchasing, or reviewing [68], where U and I denote the sets of users and items. There are several types of Recommender Systems. The most common types include:

1. **Collaborative Filtering [8]:** This method is based on the assumption that users with similar preferences in the past will have similar preferences in the future. There are two types of collaborative filtering: user-based and item-based. The process of identifying similar users and recommending items that those users have liked is known as user-based collaborative filtering. Item-

based collaborative filtering, on the other hand, entails identifying items and recommending them to users who have previously liked similar items.

2. **Content-based Filtering [65]:** This strategy entails recommending items that are comparable to those that a user has previously liked. Content-based filtering recommends items based on their attributes, such as the movie's genre, director, and actors.

3. **Hybrid Recommender Systems [6]:** These systems use a combination of both collaborative filtering and content-based filtering to make recommendations. This approach combines the strengths of both methods in order to provide more accurate and diverse recommendations.

4. **Neural Network-based [77]:** In this approach, several types of neural networks are utilized in combination with collaborative filtering or content-based filtering. Neural network-based techniques have demonstrated impressive performance, particularly in feature extraction and the accuracy of the predictions. These techniques, however, have some drawbacks, such as the lack of interpretability. The neural network's black-box nature limits the use of this approach in situations where understanding the reasoning behind the recommender decision is critical.

5. **Knowledge Graph-based [27]:** In an attempt to solve the issues that arose with the previous approaches, this strategy combines KGs and other types of recommendation, such as collaborative filtering, content-based filtering etc. This approach utilizes the knowledge of items and their attributes, along with other information, such as user interactions, to make recommendations. The aim of this research is to enhance the performance of a multi-task learning RS, which falls under this category.

These are the main types of RSs, but many other variations and hybrids can be created to overcome the challenges these methods face. Classical RSs, such as collaborative filtering and content-based filtering, suffer from a variety of issues. The most common issues include:

1. **Cold-start Problem:** This occurs when an RS cannot make recommendations for a new user or item because there is insufficient historical data [42]. Knowledge graph-based RS can help to solve this problem by using item attributes and relationships to make recommendations for new items, even if there is no historical data available [27]. For example, if a new item is added to an entertainment service site such as Netflix, the system can use the item's attributes, such as movie genre, director, and actors, to recommend similar items that users might be interested in selecting. Additionally, Knowledge graph-based RS can use demographic information about a new user to make recommendations. For example, if a new user has not interacted with any items yet, the system can use the user's age, gender, and geographic location to recommend items that are likely to be relevant to them.

2. **Data Sparsity:** The sparsity problem occurs when the number of interactions or ratings between users and items is very low, making it difficult to find patterns or similarities [49]. The KG-based RS attempts to solve this issue by leveraging additional information about the users and items already contained in the system [27].

3. **The Long-tail Problem:** This occurs when a small number of items receive a large number of recommendations, while a large number of items receive very fewer recommendations. It is caused by the sparsity of the data and the diversity of user preferences. Since there are a limited number of interactions or ratings between users and items, it is difficult for the RS to find patterns or similarities among the vast number of available items [3]. KG-based RS can potentially address the long-tail problem [72]. For example, the RS can use item attributes in the KG, such as genre, director, and actors, to recommend similar items, even if those items have not been rated or interacted with by many users. This can help to uncover items that are not as well-known or popular, but still may

be relevant to a user's preferences.

4. **Performance:** Performance is a critical factor for RSs, and it often presents a challenge [4]. The goal of an RS is to make relevant recommendations to users [27]. Performance is a measure of how well the system is or is not achieving this goal. Performance enhancement of RSs is an active area of research [25, 4]. The field of RSs is constantly evolving, and researchers are continuously working to develop new algorithms and approaches that improve the performance of these systems. This is driven by the growing demand for personalized recommendations in various applications, such as e-commerce, entertainment, and information retrieval [38]. In practice, there are several factors that can impact the performance of an RS, such as the size of the dataset, the quality of the data, the type of the algorithms used, and the computational resources available [25]. Thus, the need for more efficient and effective RSs is becoming increasingly more important. In this research, we study the problem of performance enhancement of a KG-based multi-task learning RS.

It's worth mentioning that, even though KG-based systems have the potential to address these problems, it is not guaranteed to solve them completely, as it still depends on the quality and the completeness of the knowledge graph, the available data, and the effectiveness of the algorithms used.

1.0.2 Multi-task Learning Models

Multi-task learning models are a category of KG-based recommendation systems that use KGC methods to enhance the recommendations by predicting missing facts in the KG. These models align the items in the RS with corresponding entities in the KG and then alternate training between the recommendation and KGC tasks. The joint learning layer improves the embeddings in the RS with the help of KG embeddings. Most models only improve the item embeddings by adding them to the corresponding entity embeddings. However, the knowledge-enhanced translation-based user preference (KTUP) model enhances the representation of users' preferences and items by integrating the corresponding embeddings of entities and relations in the KG, resulting in more enhanced and explainable recommendations [27]. This thesis presents a new variant of the KTUP model called RKTUP, which improves KTUP's performance by incorporating a vast majority of relation types from the KG into the RS.

1.0.3 Knowledge Graphs

Knowledge graphs can be traced back to the early days of AI and the development of semantic networks [58]. However, Google popularized the term "knowledge graph" in 2012 [23] when it released the Knowledge Graph system, which leverages KGs to improve search results.

Knowledge graphs are widely used to power search and recommendation systems, knowledge base systems, and other applications that require an understanding of the links between pieces of information [27].

Since 2012, knowledge graphs have been one of the main research subjects, leading to numerous descriptions and definitions being published [23]. In general, though, a knowledge graph is a type of data model that represents information as a collection of items (nodes) and their interactions (edges). Each triple has a subject, a predicate (or " relation"), and an object. Typically, the subject and object are both entities, while the predicate describes their relationship.

Formally, A Knowledge graph is a directed triple $G = (E, R, T)$ in which E and R are two disjoint finite sets [73]. Each element of E is called an entity, and each element of R is called a relation. $T \subseteq E \times R \times E$ is called the triples set, where each triple $(e_s, r, e_t) \in T$ represents a fact in G such that the source entity e_s has relation r with tail entity e_t [12]. Fig 1.0.1 shows an example of a Knowledge Graph.

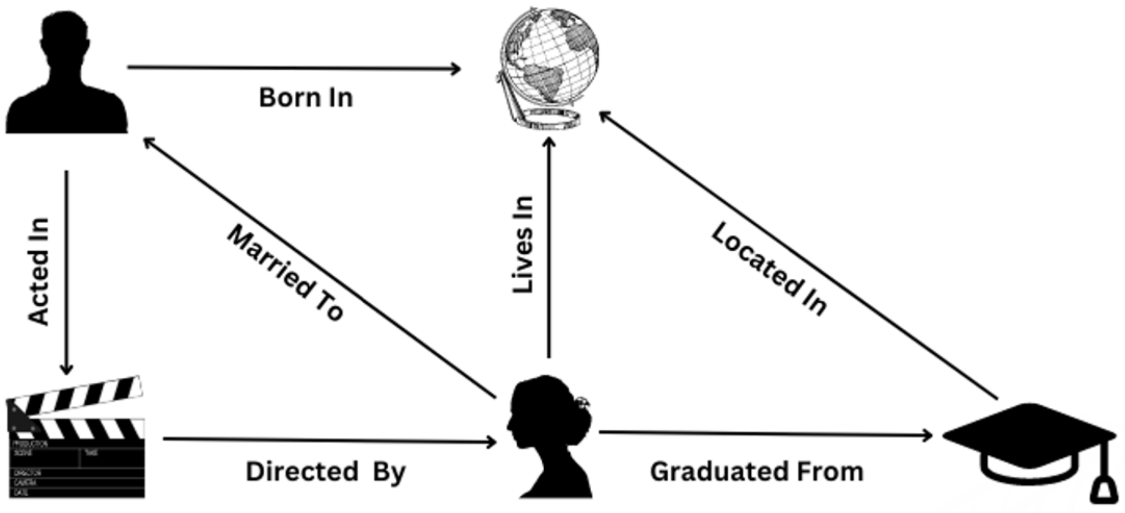

Fig. 1.0.1: An Example of a Knowledge Graph

1.0.4 Knowledge Graph Completion

Real-life KGs are inherently incomplete [43] because the information they represent is always a subset of the total knowledge that exists in the world. For this reason, we need Knowledge Graph Completion methods that can automatically infer missing information in a knowledge graph.

KGC methods use known facts in the KG to infer unknown facts using embedding-based [70], neural network-based [13] or multi-hop reasoning-based [13] approaches. KGC models either predict missing relations $(h, ?, t)$ or missing target entities $(h, r, ?)$ in the knowledge graph [61], where h, r, t, and ? denote the source entity, the relation, the head entity, and the missing fact, respectively. Fig 1.0.2 shows an example of Knowledge Graph Completion.

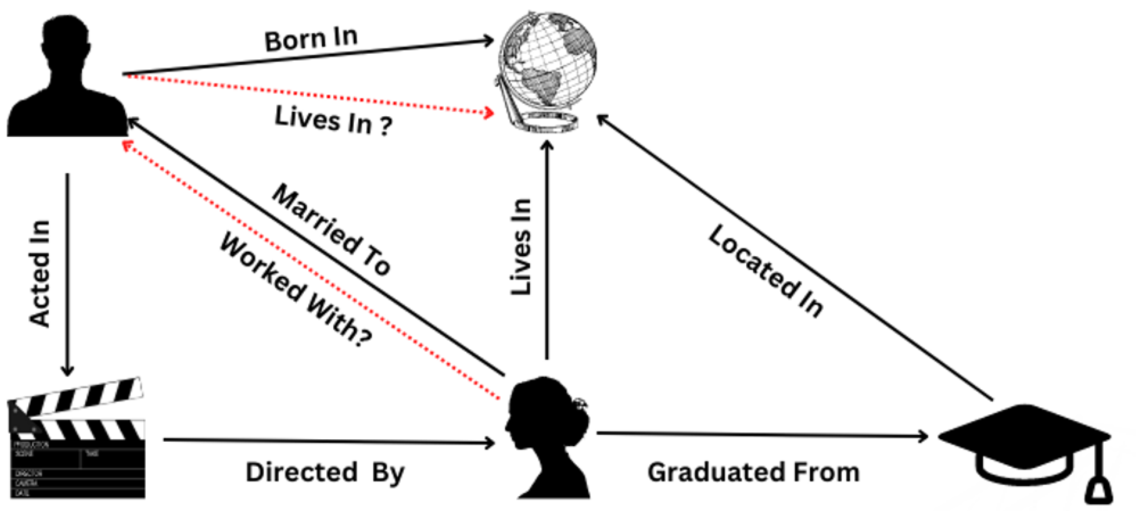

Fig. 1.0.2: An Example of Knowledge Graph Completion

1.0.5 Knowledge Graph Embedding

1.0.5.1 Definition and Background

Knowledge Graph Embedding (KGE) is the process of mapping entities and relationships in a KG to a low-dimensional vector space, where entities and relationships are represented as unique vectors [70]. The goal of knowledge graph embedding is to capture the underlying structure and semantic meaning of the knowledge graph in an efficient computational way. KGE can be used for downstream tasks such as link prediction, and knowledge graph completion [5].

The main KGE approaches are classified into three types: translation-based methods, semantic matching-based methods and neural network-based methods [58].

In the translation-based methods, entities are represented as points in a continuous vector space, and relationships between entities are modelled using a translation operation [70]. Some examples of translation-based methods are TransE [9], TransH [71], TransR [45], RotatE [64], and HRotatE [58].

The semantic matching-based methods use tensor factorization to break up high-dimensional relationship tensors into low-dimensional entity embeddings to find possible semantic connections between entities and relationships [13]. DistMult [74], and

ComplEx [66] are some examples in this category.

Lastly, the neural network-based methods use neural networks to learn the embeddings for the entities and relationships in a knowledge graph [19]. Examples of neural-network-based methods include ConvE [19], and E-MLP [61]. This thesis focuses on translation-based models only.

The ability to effectively capture the underlying relationship properties and patterns in a knowledge graph is crucial for the performance of translation-based models [32]. These models rely on the ability to translate between vectors to model the relationships in the graph [70].

Understanding the patterns of relationships such as inverse, hierarchical, composition and symmetry, as well as properties like the relation's cardinality, such as one-to-many and many-to-one allows these models to better reflect the structure of the knowledge graph. This leads to improved results in various downstream tasks [32].

Furthermore, handling these properties and patterns can help reduce the number of duplicated relationships in the KG and help learn more general representations of entities and relationships, making the models more robust and efficient [64]. Not all translation-based models can capture all relationship patterns and properties. However, RotatE and HRotatE can capture the majority of them [58, 64].

1.0.5.2 KGE Models at a Glance

Inspired by [16], this section provides an example to explain the steps taken by KGE models, in general.

Consider a knowledge graph with three entities " Toronto", " City", and " Canada" and two relationships " is-a" and " located-in". We want to train a knowledge graph embedding model to predict missing relationships in this knowledge graph. To do so, a knowledge graph embedding model would generally go through the following steps:

1. **Lookup layer:** It assigns an embedding to each node and relationship in the graph. It randomly initializes the embeddings for each entity and relationship.

For example, “ Toronto” is assigned an embedding [0.1, 0.2, 0.3], “ City” is assigned [0.4, 0.5, 0.6] and “ is-a” is assigned [0.7, 0.8, 0.9].

2. **Scoring function:** Different models have different scoring functions. This example uses a distance scoring function. The scoring function assigns a score to each triple (head entity, relationship, tail entity) in the knowledge graph. For example, the score of the triple (Toronto, is-a, City) would be computed as $||[0.1, 0.2, 0.3] + [0.7, 0.8, 0.9] - [0.4, 0.5, 0.6]|| = ||[0.8, 1.0, 1.2] - [0.4, 0.5, 0.6]|| = ||[0.4, 0.5, 0.6]|| = 0.782$

3. **Loss Function (training):** Different models have different loss functions. The loss function used in this example is the margin-based loss function. For each positive triple (head entity, relationship, tail entity), the model pays a penalty if the score of a positive triple is lower than the score of a negative triple by a margin γ. The negative triple is generated by corrupting the head or tail of the positive triple. For example, if the positive triple is (“ Toronto”, “ is-a”, “ City”), the negative triple could be (“ Toronto”, “ is-a”, “ Canada”). The loss function is used to measure the error of the model’s predictions and to guide the training process.

4. **Negative samples generations:** Generating negative samples is done by corrupting the head or tail of the positive triples. For each positive triple, KGE models generate one negative triple by corrupting the head and one negative triple by corrupting the tail. There are several methods to generate negative samples.

   - Uniform sampling: To generate negative triples, the uniform sampling approach generates all possible synthetic negatives randomly and samples n negatives for each positive triple.
   - Complete set: In the complete set approach, all possible synthetic negatives are used for each positive triple without any sampling.
   - Self-adversarial sampling: This approach generates negatives based on the

current embedding mode which allows the model to continuously improve its performance during training.

5. **Optimizer:** The optimizer is responsible for updating the embeddings in the lookup layer during the training process. It minimizes the loss function by adjusting the embeddings in the lookup layer. Common optimizers used in knowledge graph embeddings include Adam and AdaGrad.

6. **Downstream task:** In this example, the downstream task is link prediction, where the goal is to predict missing relationships between entities in the knowledge graph. The trained model can be used to predict the missing relationship " located-in" between " Toronto" and " Canada".

Fig 1.0.3 shows the input, the components and the output of a KGE model.

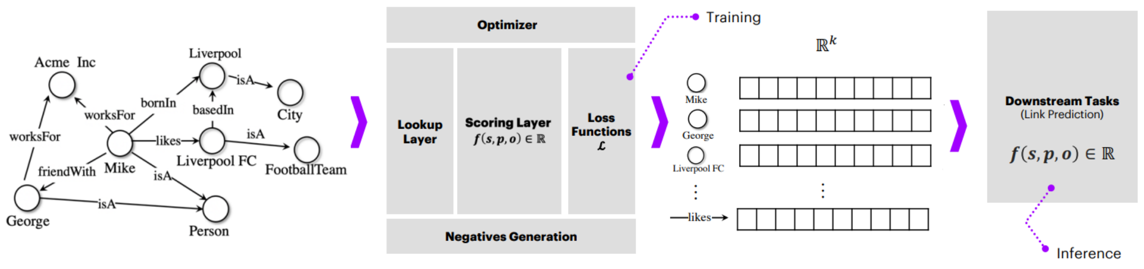

Fig. 1.0.3: KGE Models at a Glance [16]

1.0.6 Relationship Patterns and Multi-folds

Each relation in the KG has its corresponding relation pattern and multi-fold relation type [32]. For example, the relation *marriedTo* is symmetric and one-to-one. Thus, it is crucial for the KGE to recognize the relations' pattern and multi-fold [32]. Many translation-based KGE models exist in the literature, such as TransE and its extension. However, many of them cannot represent all types of the relation's pattern and multi-folds [32]. In this section, we provide some definitions of common relationship patterns and multi-folds.

1.0.6.1 Relationship Pattern

This section presents definitions for several common relationship patterns, drawn from [64]. Each pattern is accompanied by an example from the movie RS.

Definition 1 *A relation r is symmetric (antisymmetric) if*

$$\forall x, y : r(x, y) \Rightarrow r(y, x) \; (r(x, y) \Rightarrow \neg r(y, x)) \tag{1.0.1}$$

The similarity between two movies is a symmetric relation. If movie A is similar to movie B, then it follows that B is also similar to A. For example, if a user has watched and liked the movie " The Dark Knight," the recommendation system could suggest other similar movies such as " Batman Begins". The system could also recommend " The Dark Knight" to users who have liked " Batman Begins".

Definition 2 *Relation r_1 is inverse to relation r_2 if*

$$\forall x, y : r_2(x, y) \Rightarrow r_1(y, x) \tag{1.0.2}$$

The "users who liked this movie also liked" is an inverse relation. The inverse of this relation would be "movies that are liked by users who liked this movie". For example, if a user likes the movie " The Lord of the Rings," the RS could suggest other movies that are popular among users who also liked " The Lord of the Rings" series, such as " The Hobbit" " Avatar," or " Star Wars.".

Definition 3 *Relation r_1 is composed of relations r_2 and r_3 if*

$$\forall x, y, z : r_2(x, y) \wedge r_3(y, z) \Rightarrow r_1(x, z) \tag{1.0.3}$$

The " movies by the same director watched by users who have enjoyed similar movies" is a composition relation. For example, if user A enjoyed the movie " The Grand Budapest Hotel" directed by Wes Anderson, the RS could suggest other movies based on the behavior of other users who have also enjoyed " The Grand Budapest Hotel" and other movies directed by Wes Anderson. This is considered a composition relation because it involves combining multiple relationships to make a more accurate recommendation. Specifically, it combines the relationship between user A and the movie “ The Grand Budapest Hotel,” the relationship between the movie “ The Grand Budapest Hotel” and its director Wes Anderson, and the relationship between users who have enjoyed “ The Grand Budapest Hotel” and other movies directed by Wes Anderson.

1.0.6.2 Relationship Multi-folds

Here, we provide some informal definitions for common relation multi-folds. These definitions are based on the definitions in [24]

One-to-many: A type of relationship cardinality where an entity is related to multiple other entities, but each of those other entities are not related back to the original entity in a one-to-many relation. An example of a one-to-many relationship is the relationship between a director and the movies they directed. A director can direct multiple movies. However, each one of these movies can not relate back to the director in a one-to-many relation.

Many-to-one: A type of relationship cardinality, where multiple entities are related to a single entity, but the single entity is not related back to all of those other entities. An example of a many-to-one relation is the relationship between movies and a movie production company. A production company can produce multiple movies, but each movie is only produced by one production company.

One-to-one: A type of relationship cardinality, where a single entity is related to exactly one other entity, and vice versa. For example, a movie and its Internet Movie Database (IMDb) identification number. Each movie is assigned a unique IMDb ID, and each IMDb ID corresponds to exactly one movie. This is a one-to-one relationship because each movie is uniquely identified by one IMDb ID, and each IMDb ID corresponds to only one movie.

Many-to-many: A type of relationship cardinality, where multiple entities can relate to multiple entities. For example, multiple movies can have the same multiple actors, and multiple actors can be in the same multiple movies.

1.1 Problem Definition

Given a knowledge graph $G = (E, R, T)$, where E is the set of entities, R is the set of relations, and T is the set of triples in G.

Additionally, we have a list of user-item interactions denoted by $Y = \{(u, i)\}$. Here $u \in U$ denotes the set of all users and $i \in I$ denotes the set of all items. We know that $I \cap E \neq \varnothing$, which means that there exists at least one item that is common to both sets I and E. The value of $Y_{ui} \in \{0, 1\}$, depending on whether the user u engaged with item i. Specifically, if $Y_{ui} = 1$, it means that the user u interacted with item i, whereas if $Y_{ui} = 0$, it means that the user did not interact with the item.

Our goal is to learn more expressive vector representations of entities (E) and relations (R) in a KG that is part of a multi-task learning (MTL) model. The MTL model jointly learns to recommend the top-N items for a user and complete G through finding the missing set of triplets T' = $\{(e_s, r, e_t) \mid e_s \in E,\ r \in R,\ e_t \in E,\ (e_s, r, e_t) \notin T\}$, where e_s, r and e_t denote the source entity, the relation and the head entity, respectively.

The notations used in this section are commonly used in research related to knowledge representation and data recommnder systems such as [68, 60].

1.2 Thesis Motivation

Knowledge graphs are computable databases that store structural knowledge in a graph, with nodes representing entities and edges representing relations [43]. In recent years, KGs have attracted considerable interest due to their unique capability of displaying and managing vast amounts of nonlinear information in an interpretable manner [59]. Thus, KGs have improved many downstream tasks, including recommendations [27].

Recommender systems help provide the best recommendations by suggesting products and items most relevant to the user's preference. Most classical recommendation tasks rely on a rating structure in which the user explicitly or implicitly rates an item by watching, clicking, or purchasing [68]. Classical similarity-based approaches, such as collaborative filtering (CF) [56] and content-based filtering, suffer from data sparsity, and cold start problems [65] leading to poor RS performance. Combining the KG and RS training can resolve these issues and improve the recommendation's performance and explainability by giving comprehensive details about the recommended items [27].

While most RSs that rely on KGs assume that the KGs are complete [27], in reality, KGs are inherently incomplete [43]. Therefore, it is crucial to consider this aspect when designing RS. Several Multi-task learning (MTL) models that jointly learn the knowledge graph completion (KGC) and recommendation tasks have been proposed to address this issue. Some of the popular MTL models include [69, 41, 11]. However, the majority of these models, which employ translation-based knowledge graph embedding (KGE) techniques like TransH [71] and TransR [45], fail to capture various relation patterns, such as composition relations, which are common in real-world KGs [32]. As a result, their performance may be negatively affected when dealing with missing composition relations.

The relationships between entities in the KG can reflect the user's preferences. However, the incompleteness of KGs can lead to some relationships being missed, which can affect the representation of user preferences [11]. Some KGE methods can learn to better represent entities and relations in the KG [58, 64]. We suspect that KGE models that can infer a wider range of relation patterns may provide a more accurate representation of user preferences, which could ultimately lead to improved performance of the RS.

Fig 1.2.1 illustrates the significance of capturing compositional relations during the joint learning of the KGC and the recommendation tasks for personalized recommendations. The red dotted arrow between *Dan Aykroyd* and *Canada* represents the missing relation *isFrom*. Assuming that the reason for the user's choices is his preference for movies starring Canadian actors, we can infer that the relation *isFrom* represents the user's preference.

In this case, although the user's preference was determined, the RS fails to recommend *Ghostbusters II*, which is one of the user's preferences. The fact that *Dan Aykroyd* is Canadian was overlooked due to the missing relation in the KG (the red dotted arrow), which prevented the recommendation of *Ghostbusters II*.

The relation *isFrom* is a composition relation. Composition relations are defined in [64] as follows: relation r_1 is composed of relations r_2 and relation r_3 *if*

$$\forall x, y, z : r_2(x, y) \wedge r_3(y, z) \Rightarrow r_1(x, z) \tag{1.2.1}$$

In our example, x = Dan Aykroyd, y = Ottawa, ON, z = Canada, r_2 = bornIn, r_3 = locatedIn, and r_1 = isFrom. Thus, $r_2(x, y)$ = Dan Aykroyd bornIN Ottawa, ON $\wedge$ $r_3(y, z)$ = Ottawa, ON locatedIn Canada $\Rightarrow$, and $r_1(x, z)$ = Dan Aykroyd isFrom Canada.

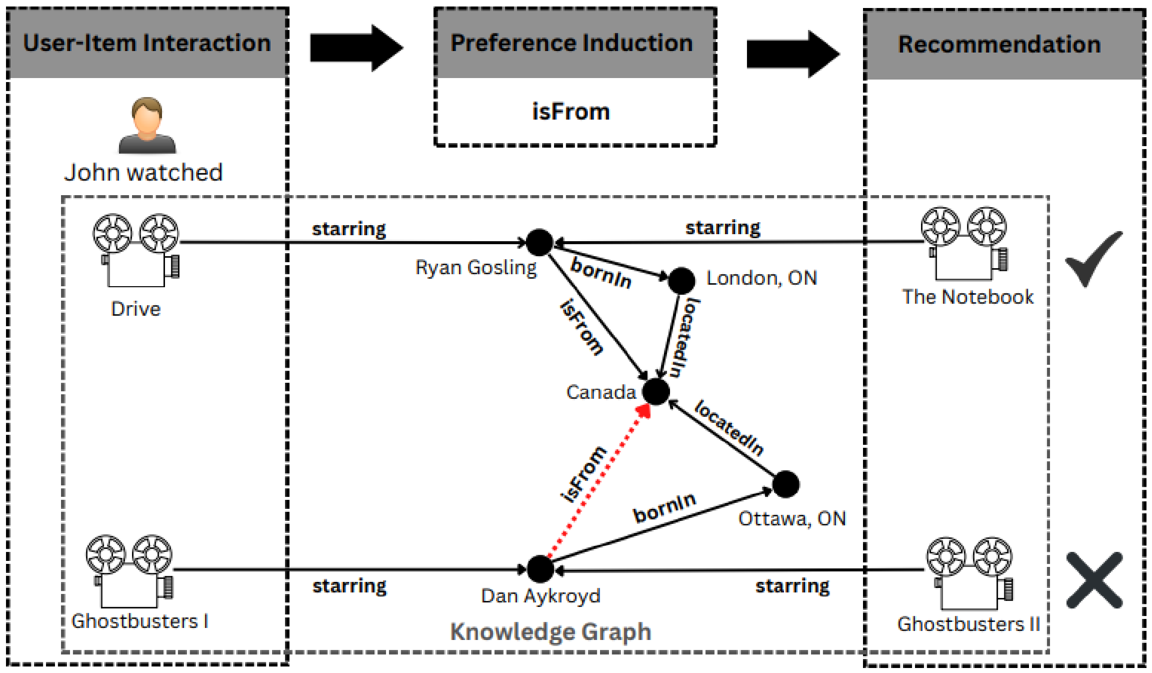

Fig. 1.2.1: An illustration of the significance of predicting missing composition patterns for better recommendations.

As real-life KGs are full of composition relations [32], we can see the importance of utilizing a KGC method capable of capturing various relation patterns to improve upon the KG and enhance the RS's performance. Our proposed approach explores the effect of capturing a wide range of relation patterns, including composition relations, on the performance of the RS by leveraging a multi-task learning (MTL) model called rotational knowledge-enhanced translation-based user preference (RKTUP). We aim to investigate whether incorporating rotational-based KGE methods, such as RotatE or HRotatE, in the MTL model would improve the RS's performance compared to a state-of-the-art approach that uses TransE, TransR and TransH for the KGC task and outperforms other baselines.

1.3 Thesis Statement

The primary objective of this research is to create an MTL model in order to boost the performance of the recommender system. The aim is to enhance an existing MTL model by using a stronger KGC approach that can infer and predict a broader range of relationship patterns leading to higher precision and more relevant recommendations.

Our proposed approach (RKTUP) is based on KTUP [11] and inherits all of its features and operations, including interpretability. However, RKTUP utilizes RotatE [64] instead of TransH [71] for the knowledge graph completion task. This provides an improvement over KTUP, as RotatE's ability to model and infer a wider range of relation patterns enhances the performance of the recommender system.

TransH is utilized by the KTUP model for the KGC task. Nevertheless, TransH is incapable of capturing all related patterns, such as composition relations, which are prevalent in real-world KGs. Therefore, the performance of TransH is limited on KGC [32].

The second objective of this thesis is to improve the efficiency of the RKTUP model by adopting HRotatE [58] for the KGC task instead of RotatE. By doing so, we hope to reduce the number of training steps while keeping the same performance level. We shall illustrate the efficiency of our proposed methodology through experimentation and analysis.

1.4 Thesis Contributions

This research focuses on enhancing the performance of an MTL recommender system by introducing a new MTL model called RKTUP. This model builds upon an existing MTL model, KTUP, and improves its ability to learn more expressive representations of users, items, entities and relations. By using RKTUP, the recommender system can better understand the user's preferences and provide more personalized recommendations. The model takes in a knowledge graph and a user-item interaction list as inputs and uses rotation-based KGE approaches such as RotatE or HRotatE. The main contributions of this thesis are:

1. We propose a new MTL model that utilizes rotational-based KGE techniques to capture a broad range of relation patterns in the KGC task.
2. We demonstrate that using RotatE for the KGC task in an MTL model improves the performance of the RS.
3. We show that HRotatE can increase the efficiency of RKTUP model by reducing the number of training steps while achieving the same results as RotatE.
4. Our model outperforms various state-of-the-art models for recommendation and KGC tasks using two popular benchmark datasets.
5. We reaffirm the significance of jointly learning item recommendations and KGC to enhance recommendations.

1.5 Thesis Organization

The following outline constitutes this thesis's structure:

In Chapter 2, we present a comprehensive review of prior research in the areas of classical recommender systems, knowledge graph-based recommender systems, as well as, translation-based knowledge graph embeddings.

In Chapter 3, we discuss our framework, which is referred to as RKTUP. In this chapter, we present a full overview of the step-by-step methodology that underpins our approach.

In Chapter 4, we describe the experimental setup and analysis that we carried out. This includes specifics of the datasets that were utilized, the hyper-parameters that were used for training and the assessment metrics that were used to determine how well our model performed.

In Chapter 5, we discuss how our findings compare to several models that are considered to be state-of-the-art. Experiments were conducted using two benchmark datasets.

Lastly, in Chapter 6, we summarize our findings and propose suggestions for future research based on what we observed during this study.

CHAPTER 2

Related Works

Despite significant advancements, modern recommender systems still have significant limitations. The cold start problem, data sparsity and performance are the main issues that most studies are attempting to overcome. Knowledge graphs (KG) have been used by many researchers to improve the performance of recommender systems. Some of them demonstrated a variety of intriguing methods for incorporating the KG into the recommender system. This section will go over these models as well as the relevant knowledge graph embedding (KGE) models. We divided the literature review into three sections: classical recommender system approaches, KG-based recommender system approaches and KGE models for KG completion.

2.1 Classical Recommender Systems

The rating and similarity structure is used in the majority of classical recommendation problems. The recommender system in an application suggests to users what they might like to watch, buy or read based on similarities between users and items[39]. Classical recommender system approaches are classified as follows:

Collaborative Recommendations: This approach is based on the assumption that similar users like similar things. Items that have been preferred previously by users with similar tastes will be recommended [54]. One of the most popular techniques for collaborative recommendations is collaborative filtering (CF) [56]. The most common form of collaborative recommendation is user-based collaborative filtering. This method involves looking at the past behaviour of users who are similar to the active user and making recommendations based on what those similar users liked in the past [54]. Another form of collaborative recommendation is item-based collaborative filtering. This method involves looking at the past behaviour of users who liked similar items to the active user and making recommendations based on what those users liked in the past [54].

One of the techniques that is widely used in collaborative recommendations is matrix factorization [37]. This technique aims to factorize the user-item interaction matrix into two low-rank matrices, one representing users and the other representing items. The most popular matrix factorization technique is Singular Value Decomposition (SVD) [2], which is based on the idea that the user-item interaction matrix can be approximated by the product of two low-rank matrices. Another popular technique is factorization machines (FM) [52] which is a generalization of matrix factorization and can handle both categorical and numerical features.

BPRMF (Bayesian Personalized Ranking Matrix Factorization) [53] is another approach in this category, which is an extension of matrix factorization that aims to optimize the ranking of items for each user. BPRMF is based on the assumption that users only interact with a small subset of items and that the ranking of items is more important than their absolute values.

Despite the effectiveness of collaborative recommendations, some challenges need to be addressed. One of the main challenges is the sparsity of the user-item interaction data, making it difficult to extract meaningful patterns and make accurate recommendations since these systems rely on feature extraction to express numerical similarities between users and items [54].

Content-based Recommendations: This technique bases the recommendations on the user's behaviour [65]. The system suggests items based on their similarity to previously consumed items. The algorithm uses the features of the items to identify the similarity between them [65]. For example, if a user has previously watched a particular type of movie, the algorithm will recommend other movies that are similar in genre. These approaches are highly dependent on data. Thus, recent techniques tried to gather additional information from external sources, such as Liked

Open Data Clouds like DBpedia [40]. This additional information can help to improve the accuracy of recommendations [65].

The co-Factorization model (CoFM) [51] is a leading approach in this category. CoFM relies on item-entity alignments which is a method to link items to their corresponding entities in a KG. The CoFM model uses two methods, share and regularize, to align the item and entity latent factors. The share approach assumes that linked item and entity latent factors are the same, while the regularize approach reduces the distance between latent item and entity representations. However, this approach may lead to information loss. Additionally, CoFM fails to transfer relation representations and only focuses on entity vector information. Thus, it fails to offer interpretability.

Other notable approaches in content-based recommendations include the item-based k-nearest neighbors (k-NN) method [56] and the Latent Semantic Analysis (LSA) [18]. The k-NN method uses the similarity between items to make recommendations, while LSA is a technique that is used to extract latent semantic information from text data.

Lastly, in recent years, there have been many efforts to combine content-based recommendations with other types of recommendation techniques, such as collaborative filtering (CF) and hybrid recommendation systems (HRS) [8]. These approaches aim to exploit the strengths of both approaches by combining the information from both techniques to make more accurate recommendations.

Hybrid Approaches: These models combine elements of neural network, content-based and collaborative filtering approaches to make better recommendations.

One of the leading models in this category is the Multilayer Perception (MLP) model [7], which combines both content-based and collaborative filtering information in a neural network architecture. Another popular model is Autoencoders [57] which uses an unsupervised learning technique to extract features from the user-item interactions then uses these features to make recommendations.

The Neural Network Matrix Factorization (NNMF) [22] model is another popular approach in this category which combines matrix factorization techniques with neural networks to make recommendations. Similarly, Neural Collaborative Filtering (NCF)

[30] uses neural networks to model the user-item interactions for collaborative filtering. The Factorization-Machine based Neural Network (DeepFM) [26] is another hybrid model that combines the strengths of factorization machines and neural networks.

While these models have shown promising results, one of the main concerns with hybrid approaches is that they are often not interpretable, meaning it is difficult to understand how the system is making recommendations. This lack of interpretability can make it challenging for practitioners to understand and trust the recommendations made by these systems [47].

2.2 Knowledge Graph-based Recommender Systems

With classical methods a lot of challenges and issues exist. Some of the challenges faced in the previously mentioned methods are the cold start problem, sparsity in ratings, scalability of the approach, the performance of the RS, and the interpretability of the recommendations. In an attempt to solve these issues, KGs have been used as a source of side information. These attempts are classified into three groups: connection-based, propagation-based, and graph embedding-based models [27]. In this study, we focus on graph embedding-based models. These models aim to enhance the users' and items' vector representation. There are three approaches for models in this category:

Two-Stage Learning Methods (TSL.): This approach is a sequential training process for the graph embedding and recommendation models. Initially, the KGE algorithms are utilized to learn representations of entities and relations in the first phase. Subsequently, the pre-trained KG-related entity embeddings are inputted into the recommendation model, along with other user and item features, to make predictions [27]. One of the leading models in this category is the deep knowledge-aware network (DKN) [67]. It is a model for news recommendation that utilizes a two-stage approach. In the first stage, entities in news titles are matched to a KG to extract knowledge-level relationships between news. The news representation is then created by combining the sentence-level textual embedding learned through CNN. The second stage is training the recommendation model. The RS then uses the KG embeddings of the news items to enhance the accuracy of the recommendation model.

Another model in this category is the KSR [31]. The authors first propose a method to construct a KG, called KB-Rec, which is built based on the items' attributes and the relations between them. Then, they use the TransE model to learn the embeddings of the entities and relations in the KG. These embeddings are then inputted into a gated recurrent unit (GRU) network to capture the users' short-term and long-term interests. A key contribution of this work is the integration of a GRU network with a KG embedding model, enabling the model to effectively capture the users' short-term and long-term interests and the inter-dependencies between the items.

The two-stage learning (TSL) models have certain benefits. They are comparatively easy to implement, as the embeddings of the KG are usually considered extra features for the recommendation model [27]. Moreover, these embeddings can be learned without relying on interaction data, making it possible to work with massive datasets without increasing computational complexity. Also, the embeddings typically do not need to be updated frequently once learned, as the KG remains stable[27].

Although the TSL methods have shown promise, they do have some limitations. One such limitation is that they tend to rely solely on the TransE model to learn the embeddings of the entities and relations within the KG. However, other models such as TransH, TransR, TransD, RotatE and HRotatE can be utilized to learn the embeddings of entities and relations in the KG which may lead to better performance of the recommender system. Furthermore, since the KGE model and the recommendation model are loosely coupled, the learned embeddings may not be suitable for recommendation tasks [27].

Joint Learning Methods (JL.): In this approach, the embedding model and the recommender system are jointly trained. The main purpose of the KG is to store the user's side information, such as their social media profiles.

One of the popular approaches in this category is Collaborative knowledge-base embedding (CKE) [76]. It was introduced in 2016. CKE is a joint learning method that uses a KG to store the side information of users, such as their social media profiles.

The CKE algorithm consists of two main components: a KG embedding model and a recommendation model. TransE [9] is used for the KG embedding model.

The recommendation model is based on a pairwise ranking objective which maximizes the probability of a positive item being ranked higher than a negative item for each user. The CKE algorithm learns the representations of entities and relationships in the KG and uses them to make recommendations to users.

CKE learns the representations of entities and relationships in the KG then uses them to make recommendations to users. By jointly learning the side information from a KG and the recommendation task, CKE aims to improve the performance of the recommendation system.

However, CKE has a limitation. It assumes that the KG is complete, which means that it assumes the KG contains all the information that is necessary to make recommendations. This assumption may not hold in practice, and it is a limitation that researchers are trying to overcome in recent works.

Another JL method that is similar to CKE is Collaborative filtering with KG model (CFKG) [78]. The CFKG model is an extension of the collaborative filtering (CF) approach that incorporates KG representations to enhance the performance of CF.

The CFKG model has two main components: a KG embedding model and a recommendation model. The KG embedding model is based on the TransE model. The recommendation model is a matrix factorization-based approach that uses the embeddings of entities in the KG to learn the representations of users and items.

CFKG outperforms classical CF methods and effectively utilizes the side information stored in the KG to make more accurate recommendations. However, it also assumes that the KG is complete which may limit its applicability in real-world settings.

Multi-Task Learning Methods (MTL.): Recent research uses multitask learning to train the recommendation task with the guidance of the KG-related task. The concept is that items in the list of user-item interactions and their corresponding KG entities have comparable structures. The findings demonstrate that transferring low-level features between items and entities enhances recommender systems' performance [27].

One of the early models in this category is the Multi-task Feature Learning approach for KG enhanced recommendation (MKR) [69]. This model is an end-to-end deep framework that utilizes KG entity embeddings to assist in recommendation tasks. MKR combines joint learning and alternating learning approaches for the KG and RS. Its framework consists of three main components: the recommendation model, the KGE model and cross and compress units. The recommendation model receives a user and item as inputs, extracting features for each through an MLP and cross and compress units. The KGE model uses another MLP to extract features from the head and relation of a KG triple, outputting a predicted tail representation using a score function. The two models are bridged by cross and compress units designed to learn high-order feature interactions between items and entities. A drawback of this model is the lack of interpretability of its recommendations.

Another approach in the KGE-based recommendation models is the Robustly Co-Learning Model (RCoLM) [41]. This approach aims to improve both the recommendation and KG completion tasks by transferring entities and items' embeddings between them. The model uses facts in the KG to enhance item recommendations and uses user-item interactions in the recommendation task to fill in missing information in the KG. However, RCoLM has limitations. For example, it uses TransR [45] which cannot infer all types of relationship patterns, limiting its performance. Furthermore, RCoLM does not consider the interpretability of recommendations by transferring relation embedding from the KG to the RS.

One of the leading approaches in the KGE-based recommendation category is the Knowledge-enhanced Translation-based User Preference (KTUP) model [11] which jointly learns the recommendation and KG completion tasks. Unlike other models in this category, KTUP uses KG embeddings to enhance the user's preferences and item representations. This is a very important feature for two reasons. First, the enhanced item embeddings capture the relational knowledge among items and complement user-item interactions, thereby improving item recommendations. Second, combining the relations and preferences embeddings provides explicit interpretation and explainability of the recommendations.

KTUP transfers knowledge from related entities and relations in the KG to the item and preference embeddings. This approach captures the user's preferences for specific aspects of an item represented by related entities in the KG.

Although KTUP achieves state-of-the-art results, it has a disadvantage in using TransH for the KGC task. The performance of TransH in KG completion is limited, as it cannot infer all types of relation patterns, such as composition.

To overcome this limitation, our model adapts the TUP approach in KTUP, and enhances it with rotational-based KGE methods for the KGC task. The item embeddings are enhanced by aligned entity embeddings to improve the RS's knowledge of the item, while the preference embeddings are enhanced by relation embeddings to increase the recommendation's interpretability. Our approach achieves better performance than KTUP and other state-of-the-art models, demonstrating the importance of embedding enhancement of items and preferences during joint training of KGC and RS for improved performance and interoperability.

So far in the related work section, we discussed various recommender systems and Fig. 2.2.1 illustrates their hierarchical branching. The Multi-task Learning model, KTUP, which is the focus of this thesis, and whose performance we aim to enhance, is highlighted in light blue.

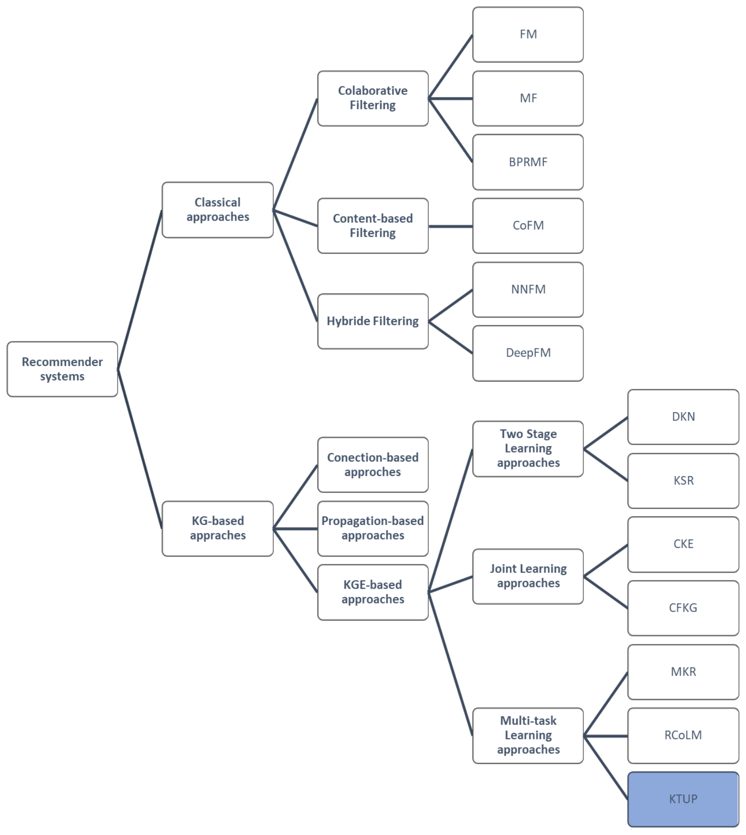

Fig. 2.2.1: Recommender Systems' hierarchical branching

Table 2.2.1 provides a summary of our literature review for the state-of-the-art models that will be compared to our proposed approach in chapter 5. The table includes information about various RSs, their categories, contributions, and challenges. We have mainly focused on the points that are relevant to this thesis. In the table, JL. refers to joint learning models, MTL. refers to multi-task learning models and KG Emb. refers to KG embedding methods. Additionally, "improper embedding" means that the model used a KGE method incapable of capturing one or more of the common relation patterns and multi-folds described in Chapter 1, Section 1.0.6.1.

Table 2.2.1: Comparison of recommender systems in the literature

Authors	Category	Year	Model	KG Emb.	Contribution	Challenges
Rendle et al.	Classical	2010	FM	N/A	Similarity-based Factorization Machine	Data Sparsity and cold start
Rendle et al.	Classical	2012	BPRMF	N/A	Bayesian implicit feedback ranking	Data Sparsity and cold start
Zhang et al.	JL.	2016	CKE	TranE	Simple implementation and used KG to store side info.	Improper embeddings and assumes KG is complete
Zhang et al.	JL.	2018	CFKG	TransE	Outperforms classical CF methods and used entity embeddings to improve the item embeddings	Only shares entity embedding and assumed the KG is complete
Li et al.	MTL.	2019	RCoLM	TransR	Used User-Item interaction to improve KGC	Improper embeddings and doesn't offer interpretability
Cao et al.	MTL.	2019	KTUP	TransH	Modeled the user preference by relations in KG and offer interpretability	Improper embeddings (can't capture composition)

2.3 Knowledge Graph Embedding for Knowledge Graph Completion

One of the most common methods for KG completion is KG embedding. These models learn the representations of entities and relationships (head, relation, tail) [71]. Then, the correctness of the predicted facts is validated using a scoring function. The most widely used KGE approaches are bilinear-based, neural-network-based and translation-based. Because of their simplicity and interpretability, translation-based models are widely used for KGC [70]. In translation-based models, the distance between two entities is used to build the scoring function, and the entities are linked using relations embedding[70]. In this section we will go over the translation-based approaches that fall within the scope of this thesis.

TransE [9]: It is one of the earliest and most well-known models in the translation-based category. It was introduced by Bordes *et al.* in 2013, and since then, it has been widely used and extended to various tasks such as link prediction, KGC and triple classification. In TransE, entities and relationships are represented as vectors in a low-dimensional space.

Given a triple (h, r, t) where h is the head entity, r is the relationship and t is the tail entity, TransE assumes that the relationship is a translation from the head entity to the tail entity. This assumption is modeled as a simple vector addition operation, where the tail entity vector is the sum of the head entity vector and the relationship vector. The score for (h, r, t) is calculated by:

$$||h + r = t|| \tag{2.3.1}$$

where ||.|| denote the L1-norm distance function [9].

The embeddings of entities and relationships are learned by minimizing the error between the predicted tail entity and the actual tail entity. This is done by minimizing a loss function which penalizes the model if the embedding of a correct triple (h, r, t) is ranked lower than that of an incorrect triple (h, r, t'). The ranking loss function can be defined as:

$$L = \sum_{(h,r,t)\in S} \sum_{(h',r,t')\in S'} ([\gamma + d(h,r,t) - d(h',r,t')]_+), \tag{2.3.2}$$

where S is the set of all triples in the KG, $[x]_+$ denotes the positive part of x, $\gamma > 0$ is a margin (a positive constant) and S' is the set of corrupted triples.

$$S'_{(h,r,t)} = \{(h',r,t)|h' \in E\} \cup \{(h,r,t')|t' \in E\} \tag{2.3.3}$$

The corrupted triplets, which are generated according to equation (2.3.3), consist of training triplets where either the head or the tail has been substituted with a random entity, but not both at once.

One of the limitations of TransE is that it cannot handle symmetric relationships. For example, in a symmetric relationship such as " is_friend_of," both $h + r = t$ and $t + r = h$ should be true, but this is not the case in TransE. In TransE, only one direction of the relationship is encoded.

Let's consider the symmetric relationship " is_friend_of" between two entities " John" and " Sam". We have the following triples: (John, is_friend_of, Sam) and (Sam, is_friend_of, John).

Let e_j, e_s represent the embeddings of entities " John" and " Sam", respectively, and e_r represent the embedding of the relationship " is_friend_of". In TransE the relationship " is_friend_of" is a translation from " John" to " Sam", so we have the following equation: $e_j + e_r = e_s$. This does not hold true for the reverse relationship from " Sam" to " John" ($e_s + e_r = e_j$). It can only hold true if $e_r = 0$ and $e_s = e_j$ which is not possible since Sam and John are different entities. Thus, Trans E cannot model symmetric relationships that are equal in both directions.

Although TransE is a simple and effective method, it is incapable of capturing symmetric relationships. Furthermore, it cannot capture many-to-many, one-to-many or many-to-one expressions. To overcome these limitations, researchers have proposed many variants of TransE. These variants resolved some of these issues but not all of them.

TransH [71]: It is a variant of TransE that was introduced in 2014 by Wang *et al.* The main difference between TransE and TransH is that TransH uses a hyperplane to represent relationships between entities in the KG, rather than a simple vector addition as in TransE. In TransH, each relationship has its own hyperplane, and entities are represented by vectors that are projected onto the hyperplanes. The translation between head and tail entities is valid only if they are projected onto the same hyperplane.

The hyperplane is a subspace of a higher-dimensional space, and it is defined by a normal vector (w_r).To project an entity vector into the hyperplane, the TransH model subtracts the component of the entity vector along the normal vector from the entity vector. The resulting vector is the orthogonal projection of h onto the hyperplane which is represented by $\perp$.

The transformed head and tail entities are then used in the scoring function to calculate the likelihood of a relationship existing between the head and tail entities.

Let h, r and t be the head entity, relation and tail entity, respectively. The scoring function calculates a score that represents the likelihood of a triplet (h, r, t) being true is calculated by:

$$||(h^{\perp} + r - t^{\perp})|| \tag{2.3.4}$$

where $h^{\perp}$ and $t^{\perp}$ are the projected entity vectors such that:

$$h^{\perp} = h - w_r^T h w_r \tag{2.3.5}$$

$$t^{\perp} = t - w_r^T t w_r \tag{2.3.6}$$

where w_r, $||.||$, h, and t are the relation-specific hyperplane (normal vector), the L1-norm distance function and the entity embeddings, respectively.

Then scoring function f_r is:

$$f_r(h, t) = ||(h - w_r^T h w_r) + d_r - (t - w_r^T t w_r)|| \tag{2.3.7}$$

where d_r is the translation vector. This is a vector in the hyperplane that is used to translate an entity vector to another point in the hyperplane. The purpose of the translation vector is to allow entities with different relations to the same tail entity to have different representations, even if they have the same head entity.

To train the TransH model, the margin-based ranking loss function aims to minimize the difference between the predicted score of a triplet (h, r, t) and the actual truth value of that triplet. The truth value is represented as 1 for a true triplet and 0 for a false triplet. The loss is calculated as follows:

$$L = \sum_{(h,r,t)\in S} \sum_{(h',r',t')\in S'_{(h,r,t)}} [f_r(h,t) + \gamma - f_{r'}(h',t')]^+ \tag{2.3.8}$$

where S is the set of positive (golden) triplets, $[x]_+ = max(0, x)$, $\gamma > 0$ is a margin (a positive constant) separating positive and negative triplets and $S'_{(h,r,t)}$ is the set of corrupted triples constructed by corrupting (h, r, t).

The loss function encourages the representations of positive triplets to be closer to each other than the representations of negative triplets.

TransH addressed some of the limitations of TransE in modeling complex relationships between entities, including one-to-many, many-to-one and many-to-many. However, it fails to capture composition relations commonly present in real-world KGs which restricts its performance [32].

TransH's failure to represent composition relations is due to its design limitations. TransH projects entity vectors into a hyperplane which is defined by a normal vector and a translation vector. However, the normal vector is fixed for all entities and cannot vary based on the relationship being modeled. This means that the same normal vector is used to model all relationships, regardless of whether they are composition relationships or not. As a result, TransH cannot accurately capture the compositionality of relationships which involves combining multiple entity vectors to form a new representation for the relationship. This limitation has been reported in the literature [32, 64, 70], and research has shown that other models, such as RotatE perform better in capturing compositionality.

RotatE [64]: It was proposed in 2019 by Sun *et al.* The main goal of RotatE is to overcome some of the limitations of existing models such as TransE and TransH.

Inspired by Euler's identity, RotatE maps the entities and relations into a complex space. Then, it views the relations as the result of the rotation from the head entity to the tail entity.

Euler's identity is an important concept in complex mathematics [63] and it states that:

$$e^{(i\theta)} = cos(\theta) + i * sin(\theta) \tag{2.3.9}$$

where e is the base of the natural logarithm and i is the imaginary number ($i^2 = -1$)

In RotatE, θ represents the rotation angle for relation r. The real and imaginary parts of $e^{(i\theta)}$ are used to define the real-valued vectors representing the head and tail entities. The real part of the complex number represents the cosine of the angle, and the imaginary part represents the sine of the angle. These vectors are then used in the scoring function to determine the likelihood of a given triple being true or not.

By defining each relation as a rotation from the head entity to the tail entity in the complex space, RotatE can model different types of relationships between entities in a KG, including one-to-one, many-to-one, one-to-many and many-to-many, symmetric/antisymmetric, inversion and composition relationships.

For example, let's say we have three entities, A, B, and C. The relationship between A and B is represented by the vector $r_1 = e^{i\theta_1}$, and the relationship between B and C is represented by the vector $r_2 = e^{i\theta_2}$. If the relationship between A and C is a combination of the relationships between A and B and between B and C, then this relationship can be represented by the vector $r_3 = e^{i\theta_3}$, where $\theta_3 = \theta_1 + \theta_2$. This means that $r_3 = r_1 \circ r_2$.

To model a relationship between entities h and t, the scoring function in RotatE measures the proximity between the rotated entity vectors and calculates a score for the relation between the entities as follows:

$$||h \circ r = t|| \tag{2.3.10}$$

where h, r, $t \in C^d$, $||.||$ denotes the L1-norm distance function and $\circ$ represents the element-wise (Hadamard) product of two complex vectors. The magnitude of each dimension of r is constrained to be 1, which represents the relationship being a rotation in the complex plane.

Specifically, for each dimension in the complex space, RotatE expects that:

$$t_i = h_i r_i \tag{2.3.11}$$

where $h_i, r_i, t_i \in C$ and $|r_i| = 1$.

In training RotatE, the goal is to minimize the distance between the modelled triples and the actual triples in the dataset. The model is optimized using a self-adversarial negative sampling loss. The negative triples are sampled from a distribution which is represented by the following equation:

$$p(h'_j, r, t'_j | (h_i, r_i, t_i)) = \frac{exp \alpha f_r(h'_j, t'_j)}{\sum_i exp \alpha f_r(h'_i, t'_i)} \tag{2.3.12}$$

where α is the temperature of the sampling. The self-adversarial negative sampling formula used to update the embeddings during training is defined as follows:

$$Loss = log \sigma(\gamma d_r(h, t)) - \sum_{i=1}^{n} p(h'_i, r, t'_i) log \sigma(d_r(h'_i, t'_i) - \gamma), \tag{2.3.13}$$

where γ is a fixed margin, σ is the sigmoid function, and (h'_i, r, t'_i) is the i-th negative triplet.

Algorithm (2.3.1) shows the steps followed to train RotatE .

Algorithm 2.3.1 Learning RotatE

```
Input: Knowledge graph $G = (E, R, T)$, embedding size $k$, learning rate $\alpha$, margin
$\gamma$, number of negative samples $n$
Output: Entity and relation embeddings
Initialize entity and relation embeddings $e \in \mathbb{R}^{|E| \times k}$ and $r \in \mathbb{R}^{|R| \times k}$ randomly;
while terminal condition not met (i.e. maximum training steps not reached) do
    Sample a batch of triples $B$ from $G$;
    Generate negative samples using self-adversarial generation;
    for each $(h, r, t) \in B$ and $n$ negative samples $(h', r, t')$ do
        Compute the energy score $||h \circ r - t||^2$ for the positive triple and negative
        samples;
        Update the embeddings w.r.t. the loss:
        $Loss = -\log \sigma(\gamma - d_r(h, t)) - \sum_{i=1}^{n} p(h'_i, r, t'_i) \log \sigma(d_r(h'_i, t'_i) - \gamma)$
    end
end
return Entity and relation embeddings for downstream tasks such as link prediction
```

Despite its contributions, RotatE also has some limitations. One of the main limitations of RotatE is that it is computationally expensive, as it requires a large amount of memory to store the real and imaginary parts of each vector.

HRotatE [58]: HRotatE was introduced in 2021 by Shah *et al.* The main goal of HRotatE is to overcome some of the limitations of RotatE. HRotatE combines elements of two existing models: RotatE [64] and SimplE [35].

RotatE is a technique that views each relationship in a KG as a rotation from the source entity (head) to the target entity (tail) in a complex vector space. However, in RotatE, the head and tail entities are generated using a single set of rules. The process of generating embeddings for both the head and tail entities using a single set of rules is known as having a single embedding-generation class. Having a single embedding-generation class allows for a simpler model architecture but can result in lower prediction scores compared to other methods that have separate rules for the head and tail entities.

SimplE is an approach that is primarily based on a Canonical Polyadic (CP) decomposition. The Canonical Polyadic (CP) decomposition is a method for representing multi-dimensional arrays, also known as tensors, as a sum of rank-1 tensors. In the CP approach, each rank-1 tensor is represented by a vector and the overall tensor is represented as the sum of these vectors. However, this method does not allow for the modelling of inverse relationships in the data, as it only considers direct relationships.

To address this limitation, SimplE enhances the CP approach by adding the inverse relation. This is done by representing each relation as a pair of vectors, one for the direct relationship and one for the inverse relationship, where the head embedding and tail embedding are taken from different embedding-generation classes. This allows SimplE to model both direct and inverse relationships in the data, resulting in a more comprehensive and expressive representation of the KG. However, SimplE is not able to predict composition patterns.

The HRotatE approach inherits all the characteristics of RotatE such as the loss function and the self-adversarial negative sampling technique. Although it aims to improve the prediction scores of RotatE by using the principle of inverse embedding from the SimplE model, HRotatE uses different embedding-generation classes to generate embedding vectors for the head entity and the tail entity. The main difference between HRotatE and RotatE is in the scoring function. The scoring function of HRotatE is as follows:

$$d_r(h,t) = \frac{1}{2}(||(h_{e_i} \circ v_r) - t_{e_j}|| + ||(h_{e_j} \circ v_r^{-1}) - t_{e_i}||) \tag{2.3.14}$$

In HRotatE, each entity, e, is represented by the two vectors, $h_e, t_e \in C^d$ and each relation, r, is also represented by two vectors $v_r, v_r^{-1} \in C^d$.

The scoring function of HRotatE is used to predict the relationship between two entities (head and tail) in a KG. The equation calculates the similarity between two entities with respect to a particular relation.

The equation takes into consideration two factors:

1. The distance between the head entity vector and the target entity vector after the head entity vector is transformed by the relation vector v_r. This distance is measured using the L1-norm (represented by $||\cdot||$).

2. The distance between the head entity vector and the target entity vector after

the head entity vector is transformed by the inverse of the relation vector v_r^{-1}.

The final result is obtained by taking the average of the two distances calculated above and dividing it by 2. The resulting score reflects the similarity between the head and tail entities with respect to a particular relation and can be used to predict the presence of a missing relationship in a KG.

Algorithm (2.3.2) shows the steps used to train HRotatE.

Algorithm 2.3.2 Learning HRotatE

```
Input: Training Set $S = (h, r, t)$
Initialize: Hyperparameters $\gamma, \alpha, d$, learning rate, hidden dimension and generating
negative samples, generating embedding class for $h, r, t, r^{-1}$
while terminal condition not met (i.e., maximum training steps not reached) do
    $S_{batch} \leftarrow$ a sample of $(h, r, t)$ triples from $S$ of size $b$
    $h_e, v_r, t_e, v_{r^{-1}} \leftarrow$ generating embedding vectors for $h, r, t, r^{-1}$
    for $(h_i, r_i, t_j) \in S_{batch}$ do
        $score = \frac{1}{2}(||(h_{e_i} \circ v_r) - t_{e_j}|| + ||(h_{e_j} \circ v_r^{-1}) - t_{e_i}||)$
        Update embeddings w.r.t.
        $-\log \sigma(\gamma - d_r(h, t)) - \sum_{i=1}^{n} p(h'_i, r, t'_i) \log \sigma(d_r(h'_i, t'_i) - \gamma)$
```

The results of the HRotatE approach are shown to be better than the native RotatE, and it outperforms several state-of-the-art models on different datasets. The HRotatE approach is also relatively efficient, as it utilizes half the number of training steps required by RotatE and generates approximately the same result as RotatE.

Table 2.3.1 shows some relationship patterns that can be captured by selected translation-based KGE models that are within the scope of this study.

Table 2.3.1: Patterns captured by selected translational KGE models [64, 58]

Model	Year	Symmetry	Asymmetry	Inversion	Composition
TransE	2013	×	✓	✓	✓
TransH	2014	✓	✓	✓	×
TransR	2015	✓	✓	✓	×
RotatE	2019	✓	✓	✓	✓
HRotatE	2021	✓	✓	✓	✓

CHAPTER 3

Proposed Approach

3.1 Introduction

In this section, we introduce our proposed approach, the Rotational Knowledge-Enhanced Translation-Based User Preference (RKTUP) model, which combines a recommender system (TUP) and a rotational-based knowledge graph embedding model (RotatE or HRotatE) for knowledge graph completion within a multi-task learning framework. We begin by presenting TUP and then providing a brief overview of RotatE and HRotatE. Finally, we offer a comprehensive explanation of RKTUP.

RKTUP is a novel approach that improves upon the Translation-Based User Preference (TUP) recommender system introduced in KTUP[11]. What sets RKTUP apart from previous models is that it incorporates most relation patterns, including composition relations, from the KG into an interpretable RS. Using rotational-based KGE models enables RKTUP to effectively capture complex relationship patterns, resulting in improved precision of personalized recommendations. Moreover, adopting HRotatE enhances the model's efficiency by achieving comparable results to RotatE but with fewer training steps.

A KG contains rich information about entities and their relationships which can provide valuable insights into the preferences of users and the characteristics of items. Thus, capturing more relationship patterns by the KGC method can greatly improve the recommender system's ability to produce more relevant recommendations.

For example, consider a movie recommendation system with a KG of movie entities and their relationships with actors, directors and genres. In this scenario, TransH [71] may not accurately capture the composition relationships between movies and their associated entities (e.g., the relationship between a movie and its director). However, with the use of rotation-based KGE methods, such as RotatE or HRotatE, the RK-TUP algorithm can better understand these relationships and use this information to make more relevant recommendations.

3.2 The Relation between Recommender System and Knowledge Graph Completion

The recommendation and the KGC tasks share the common goal of ranking candidates based on a query. The RS aims to suggest the top-N items for a particular user. Meanwhile, in the KGC task, given a (head entity, target entity) pair, the objective is to recommend the top-N items from a pool of candidate relations. Combining both tasks into a single model leverages their interplay and enhances their performance [11, 41, 51].

3.3 Translation-Based User Preference (TUP) for Item Recommendation

We use TUP for item recommendation. It is a state-of-the-art recommender system introduced in [11]. TUP aims to model user preferences as translational relationships between users and items by projecting the user and item vector representations to the preference hyperplane and employing a score function similar to that used in TransH [71].

Given a list of user-item pairs Y, TUP takes as input a user u, an item i and a preference p, then outputs a relevance score indicating the likelihood that user u likes item i. The model learns the embeddings of preference, user and item that satisfy the relationship $u + p \approx i$. The set of preferences P is a predefined hyperparameter, where the number of preferences corresponds to the number of relations in the KG.

For each user-item pair, the model generates a preference that serves as a relationship between the user and the item.

TUP has two main components: preference prediction and hyperplane-based translation.

1. **User's preference prediction:** The preference prediction task is to predict the user's preference from a set of latent features P given a user-item pair (u, i). The set P contains features shared between all users and each $p \in P$ denotes a different preference. The latent features include underlying characteristics or attributes of users and items that influence their preferences. For example, latent factors might include the genre of a movie.

   Two strategies are employed to handle the user's implicit and varied preferences: a hard approach that selects one preference out of the set of P preferences and a soft approach that combines all preferences with attention. In the hard strategy, the Straight-Through (ST) Gumbel SoftMax technique is used to sample a user's preference for an item, while in the soft strategy, the user's item selection is based on more than one preference and combines them using an attention mechanism.

   (a) **Hard strategy:** This method assumes that when a user chooses an item only a single preference has an effect on the user's decision.

   To achieve this, the Straight-Through (ST) Gumbel SoftMax [33] is used to sample a user's preference for an item. It works by approximating a one-hot vector, which represents the preference, from a multi-classification distribution.

   The log softmax function is used to calculate the unnormalized output of the score function which represents the probability of being a member of class p in a P-way distribution:

$$\Phi(p) = \frac{exp(log(\pi_p))}{\sum_{j=1}^{p} exp(log(\pi_j))} \tag{3.3.1}$$

where π_p is the scoring function's unnormalized output.

After that, a one-hot vector is sampled from the above distribution as follows:

$$z_p = \begin{cases} 1, & \text{if } p = \arg\max_j(\log(\pi_j) + g_j) \\ 0, & \text{otherwise} \end{cases}$$

where the Gumbel noise is $g = -log(-log(u))$, and u is generated by a specific noise distribution [11]. Adding the random Gumbel noise to the log-softmax output makes the process equivalent to drawing a sample from a continuous probability distribution.

For example, let's say we have a user-item pair (u, i) and a set of preferences $P = \{p1, p2, p3\}$. Using the score function $f_r(u, i, p)$, we calculate the similarity between $(u + i)$ and each preference in P. The unnormalized output is then calculated using the log softmax function. The one-hot vector z_p is sampled from the Gumbel-Softmax distribution using the ST Gumbel SoftMax technique. The resulting one-hot vector z represents the preference that has the highest similarity with the user-item pair (u, i).

(b) **Soft strategy:** Here, the user's item selection is based on more than one preference. This strategy combines the user's preferences using the attention mechanism

$$P = \sum_{p' \in P} \alpha p' P' \tag{3.3.2}$$

where $\alpha p'$ is the attention weight of preference P'. Additionally, $\alpha p'$ is proportional to the similarity score:

$$\alpha p' \propto \phi(u, i, P') \tag{3.3.3}$$

2. **Hyperplane-based translation:** To deal with the common N-to-N problem where a set of users might have the same preference for different items, TUP adopts the hyperplane translation strategy used by TransH [71]. In TUP, each

preference has its own hyperplane and the item and user embeddings are projected onto the hyperplane. The projection helps to maintain the preference-specific information for each user-item pair.

TUP's scoring function is defined as follows:

$$g(u, i; p) = ||(u^{\perp} + p - i^{\perp})|| \tag{3.3.4}$$

where p, $u^{\perp}$ and $i^{\perp}$ are the inferred preference and projected vectors of the user and item, such that

$$u^{\perp} = u - W_p^T u W_p \tag{3.3.5}$$

$$i^{\perp} = i - W_p^T i W_p \tag{3.3.6}$$

Here, W_p denotes the projection vector of the corresponding hyperplane, p is the translation vector and $||.||$ denotes the L1-norm distance function [11].

To train the model, the loss function aims to minimize the difference between the predicted score of a triplet (u, p, i) and the actual truth value of that triplet. The loss is calculated as follows:

$$L_p = \sum_{(u,i) \in Y} \sum_{(u,i') \in Y'} - \log \sigma \left[g(u, i'; p') - g(u, i; p) \right] \tag{3.3.7}$$

where Y is the set of positive triplets and Y' is the set of corrupted triples constructed by randomly corrupting an interacted item to a non-interacted one for each user.

Algorithm 3.3.1 Learning TUP

```
Input: User-Item Interaction List $Y = (u, i)$, set of latent features $P$
initialize users, items and preferences embeddings with random values;
while terminal condition not met (i.e. maximum training steps not reached) do
    $S_{batch} \leftarrow$ a sample of (u,i) pairs from Y;
    getPreferences (u, i, P, preference induction approach) for each pair in $S_{batch}$;
    Generate negative samples for each $(u, p, i)$ triple;
    for each $(u, p, i)$ triple do
        Calculate $g(u, i; p) = ||(u^{\perp} + p - i^{\perp})||$;
        Update embeddings w.r.t.
        $L_p = \sum_{(u,i) \in Y} \sum_{(u,i') \in Y'} - \log \sigma \left[ g(u, i'; p') - g(u, i; p) \right]$;
    end
end
```

3.4 RotatE (or HRotatE) for Knowledge Graph Completion

RotatE [64] addresses the limitations of previous KGC models such as TransE and TransH. Unlike these models, RotatE maps entities and relations into a complex space. Then, it views the relations as the result of the rotation from the head entity to the tail entity. To accomplish this for any triplet (h, r, t), RotatE does element-wise multiplication as follows:

$$||h \circ r = t|| \tag{3.4.1}$$

where $h, r, t \in C^d$ and $||.||$ denote the L1-norm distance function.

During training, RotatE uses a self-adversarial negative sampling loss to optimize the model. The loss function is defined as:

$$L = -\log \sigma(\gamma - d_r(h, t)) - \sum_{i=1}^{n} p(h'_i, r, t'_i) \log \sigma(d_r(h'_i, t'_i) - \gamma) \tag{3.4.2}$$

where γ is a fixed margin, σ is the sigmoid function and (h'_i, r, t'_i) is the i-th negative triplet. The training goal is to minimize the distance between the modelled triples and the actual triples in the dataset.

While RotatE has proven to be effective, one of its main limitations is its high computational cost. In response to this, HRotatE [58] addressed some of the limitations of RotatE. HRotatE combines elements of two existing models, RotatE and SimplE [35]. While HRotatE inherits the loss function and self-adversarial negative sampling technique from RotatE, it aims to improve prediction scores by using the principle of inverse embedding from SimplE. HRotatE uses different embedding-generation classes to generate embedding vectors for the head and tail entities. Thus, it learns more efficiently. The scoring function of HRotatE is as follows:

$$d_r(h,t) = \frac{1}{2}(||(h_{e_i} \circ v_r) - t_{e_j}|| + ||(h_{e_j} \circ v_r^{-1}) - t_{e_i}||) \tag{3.4.3}$$

In HRotatE, each entity, e, is represented by the two vectors, $h_e, t_e \in C^d$, and each relation, r, is also represented by two vectors $v_r, v_r^{-1} \in C^d$.

RotatE and HRotatE define each relation as a rotation from the head entity to the tail entity in the complex space. This unique approach enables the model to effectively capture a diverse range of relationships between entities in a knowledge graph, including but not limited to one-to-one, many-to-one, one-to-many and many-to-many, as well as symmetric/antisymmetric, inversion and composition relationships.

3.5 Shared Embeddings and Explainability

In most KG-based recommender systems, shared embeddings are used to represent both items and entities, reducing the number of parameters in the model and improving scalability [51]. However, this approach can limit the ability to differentiate between items and entities. As in [11], the RKTUP model uses separate embeddings for entities and items and associates each item with its corresponding entity. This enhances item embeddings with structural knowledge from the knowledge graph and improves recommendations by complementing user-item interactions. Additionally, using one-to-one mapping for relations and preferences enhances the explainability of the model. In RKTUP, each preference is associated with a specific relation label, which reveals its meaning. For example, a relation labeled " isDirectedBy" could reveal a preference for certain directors, while a relation labeled " hasGenre" indicates a preference for a specific genre.

In RKTUP the ultimate goal of the final loss function is to ensure a strong alignment between entity and item embeddings, as well as relation and preference embeddings. This alignment is crucial because the same entity may appear in both the knowledge graph and recommendation tasks, and a lack of alignment between these embeddings could have a negative impact on the overall performance of the model.

For example, the entity " Tom Hanks" may appear in the knowledge graph as an actor who acted in a certain movie, and also in the recommendation task as a user who likes to watch movies featuring Tom Hanks. If the entity embedding for " Tom Hanks" in the knowledge graph task is very different from the entity embedding for " Tom Hanks" in the recommendation task, then the performance of the overall model may suffer.

3.6 Enhanced Learning via RKTUP

This thesis aims to explore how modelling the vast majority of relation patterns impacts the performance of the RS in an MTL framework. To achieve this, we propose a new approach called RKTUP that jointly learns the task of recommendation and rotational-based KGC. To the best of our knowledge, there is no other model that specifically focuses on incorporating most of the relation patterns, including compositional relations from the knowledge graph, into the recommendation system.

The main goal of RKTUP is to enhance the RS's knowledge of user preferences and items by leveraging related entity and relation embeddings produced by the rotational-based KGC method.

RKTUP takes as inputs a KG, a user-item interaction list $Y = \{(u, i)\}$ and a set of item-entity alignments $A = (i, e)|i \in I, e \in E$, where each (i, e) means that i can be mapped to an entity e in the given KG. It outputs $g(u, i; p)$, which is the score that measures the likelihood of a user u interacting with an item i with a preference p and also a score $f(e_h, r, e_t)$ that indicates the plausibility of the given fact. This is based on the jointly learned embeddings of users, items, preferences, entities and relations.

The knowledge enhancement of TUP with the guidance of the rotational-based KGC methods is done as follows::

Step 1. Calculate the enhanced item embedding $\hat{i}$ as $i + e$ where $(i, e) \in A$.

$$\hat{i} = i + e, (i, e) \in A \tag{3.6.1}$$

Step 2. Calculate the translation vector $\hat{p}$ and the projection vector $\hat{W}_p$ enhanced by the corresponding relation embedding.

$$\hat{p} = p + r \tag{3.6.2}$$

$$\hat{W}_p = W_p + W_r \tag{3.6.3}$$

Step 3. Project the enhanced item embedding $\hat{i}$ onto the enhanced preference hyperplane.

$$\hat{i}^{\perp} = \hat{i} - \hat{W}_p^T \hat{i} \hat{W}_p \tag{3.6.4}$$

Step 4. Calculate the enhanced score of recommendation $g(u, i; p)$.

$$g(u, i; p) = ||(u^{\perp} + \hat{p} - \hat{i}^{\perp})|| \tag{3.6.5}$$

Step 5. Calculate the final loss as a weighted sum of the RS and KGC losses with a hyperparameter λ used to adjust their hyperparameters.

$$L = \lambda L_p + (1 - \lambda) L_R \tag{3.6.6}$$

An overview of the algorithm to optimize RKTUP is presented in algorithm 3.6.1, where the target task is item recommendations.

Algorithm 3.6.1 Main components of the algorithm to learn RKTUP

```
Input: KG, $Y = \{(u,i)\}$, $A = (i,e) | i \in I, e \in E$, set of latent features $P$
Output: $g(u,i;p)$: score of recommendation, $f(e_h,r,e_t)$: plausibility of the given fact. Initialize: initialize models' hyperparameters and $\lambda$;
while terminal condition not met (i.e. converged or maximum training steps not reached) do
    for trainingStep = 1 to N do
        if trainingStep is even then
            $S_{batch_R} \leftarrow$ a sample of (u,i) pairs from Y;
            entityID, itemID $\leftarrow$ getMappedEntities(A); for each $(u,i) \in S_{batch_R}$
            Optimize Opt(TUP) using algorithm 3.3.1;
        end
        else
            $S_{batch_K} \leftarrow$ a sample of triples from KG;
            entityID, itemID = getMappedItems(A); for each $(e_h,r,e_t) \in S_{batch_K}$
            Optimize Opt (RotatE) using algorithm 2.3.1 or Opt(HRotatE) using
            algorithm (2.3.2);
        end
        for each $(u,p,i)$ and its corresponding $(e_h,r,e_t)$ triple do
            $\hat{i} \leftarrow i + e$;
            $\hat{p} \leftarrow p + r$;
            Project $\hat{i}$ onto $\hat{p}$ hyperplane using equations (3.6.3) and (3.6.4);
            $g(u,i,p) \leftarrow ||(u^{\perp} + \hat{p} - \hat{i}^{\perp})||$;
            Update embeddings w.r.t.
            $L = \lambda L_p + (1-\lambda) L_R$;
            $f(e_h,r,e_t) \leftarrow$ likelihood score of $(e_h,r,e_t)$ being true from KGE method;
        end
    end
end
return $f(e_h,r,e_t)$ and $g(u,i;p)$
```

An illustration of the RKTUP model is shown in Fig 3.6.1. The input is on the left: user-item interactions, knowledge graph and the alignments between items and entities. The TUP model for item recommendation is at the top-right corner, while the RotatE (or HRotatE) model for knowledge graph completion is at the bottom-right corner. The RKTUP model jointly learns both tasks by enhancing the embeddings of items and preferences with those of entities and relations.

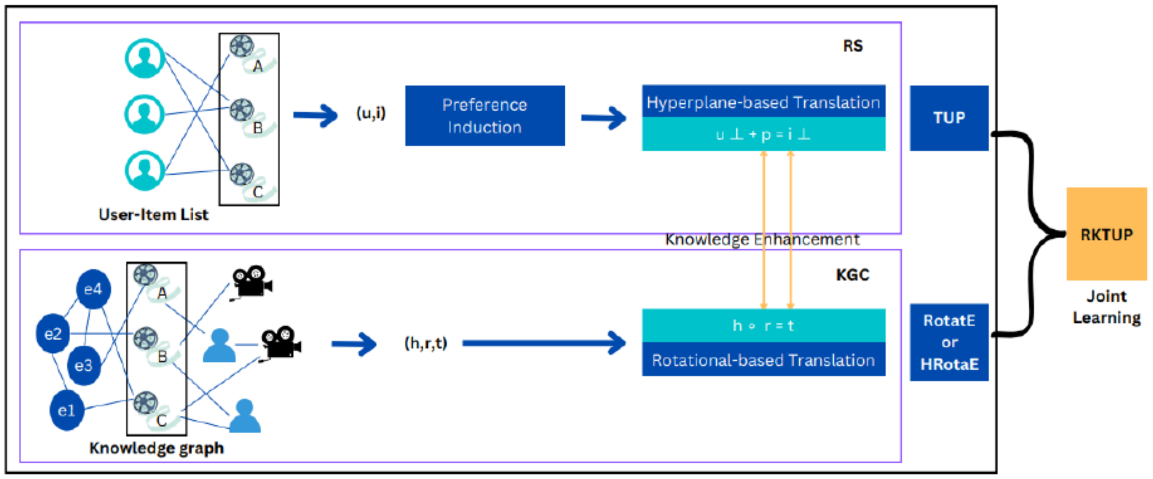

Fig. 3.6.1: An illustration of RKTUP model

In this chapter, we presented an MTL approach for integrating a wide range of relationship types from KG into RS. To achieve this, we employed two different rotational-based KGE models for the KGC task. Specifically, we utilized RotatE to capture complex and compositional relationships that are often present in real-world recommender systems. Additionally, we leveraged the efficiency of the HRotatE model to improve the overall effectiveness of the KGC task. In the next chapter, we will dive into the details of our experimental setup and evaluation methodology.

CHAPTER 4

Experimental Evaluation

In this chapter, we provide a comprehensive overview of our experimental setup, which encompasses detailed information regarding the datasets used, the hyperparameters utilized during training and the evaluation metrics employed to assess the performance of our model.

4.1 Datasets

Our experiments were conducted on two widely-used datasets in the field namely, MovieLens-1m [50] and DBbook2014 [1]. Notably, numerous state-of-the-art recommender systems rely exclusively on these two datasets for model evaluation, as evidenced by prior research such as [51, 11].

MovieLens-1m was compiled by the University of Minnesota Grouplens research project [28]. The dataset consists of user ratings on movies. The ratings are on a scale of 1-5, and there are a total of 1 million ratings provided by 6,040 users on 3,952 movies. The dataset also includes additional information such as movie titles, genres and release year [7].

The DBbook2014 dataset collects book reviews, ratings, and metadata. The dataset was released in 2014. Each review in the DBbook2014 dataset includes ratings on a scale of 1 to 5. The metadata associated with each book includes its title, author and genre [46].

To enrich the datasets, they were refined for linked open data-based RS (LODRecSys) by mapping entities to DBpedia KG [40] where mappings were available.

This allows for incorporating additional information about the movies and books and related entities in the recommendation process.

To build the knowledge graphs, both datasets' items are first mapped into their corresponding DBpedia Uniform Resource Identifiers (URIs) if a mapping is available. Then, SPARQL queries retrieve Resource Description Framework (RDF) triples from DBpedia for each mapped item, regardless of whether they are represented as head or tail entities. Afterwards, the query results are filtered to remove any non-English literals. This extracted information is subsequently used as input to construct the knowledge graphs. This process ensures the intersection between the set of items I in the user-item interaction list and the entity set E in the knowledge graph is not empty (i.e. $I \cap E \neq \varnothing$).

RKTUP is an enhanced variant of KTUP. To ensure a fair comparison, we maintained consistency with the data pre-processing methodology outlined in [11] as follows:

1. Users who rate less than 10 movies in MovieLens-1m and less than 5 books in DBbook2014 were dropped.
2. Items that occurred less than 10 times were dropped from both datasets.
3. Existing ratings were treated as positive samples, and negative samples were randomly generated.
4. Unrelated relations were dropped, and similar relations were merged manually.

Table 4.1.1 shows a statistical overview of MovieLens-1m and DBbook2014 after preprocessing.

Table 4.1.1: Datasets characteristics

Attribute	**MovieLens-1m**	**DBbook2014**
Users	6,040	5,576
Items	3,240	2,680
Ratings	998,539	65,961
Avg. ratings	165	12
Positive ratings	56 %	45.8%
Sparsity	94.9 %	99.6 %
Entity	14,708	13,882
Relation	20	13
Triple	434,189	334,511
I-E Mapping	2,934	2,534

The preprocessed MovieLens-1m dataset contains 6,040 users and 3,230 items with a total of 998,539 ratings and an average of 165 ratings per user. The dataset is highly sparse with a sparsity rate of 94.9 percent. The sparsity rate is even higher in the DBbook2014 dataset, which contains 5,576 users and 2,680 items with 65,961 ratings and an average of 12 ratings per user. The sparsity rate in this dataset is 99.6 percent. MovieLens-1m has approximately 100,000 more triplets than DBbook2014. Specifically, MovieLens-1m contains 14,708 entities and 20 relations, whereas DBbook2014 comprises 13,882 entities and 13 relations.

Furthermore, we randomly divided the datasets into training, validation and testing subsets using a 70:10:20 ratio and confirmed that each user had at least one item in the test set.

4.2 Hyperparameter Tuning

Hyperparameter tuning is the process of selecting the optimal set of hyperparameters for a machine-learning model. In this section, we first define the hyperparameters utilized in this thesis, then we discuss the values used to generate the results discussed in chapter 5. In this thesis, we consider the following hyper-parameters:

- **Learning rate (η):** It determines how quickly or slowly the model parameters converge to the optimal values during the training process. Basically, it controls the step size. The step size is the size of the update that the optimizer makes to adjust the model parameters during each iteration of the training process while moving toward a minimum of the loss function.

- **Embedding size (k):** It is the hyperparameter that controls the dimensionality of the learned embeddings. Increasing the embedding dimension may enhance the model's expressiveness, but it could also result in longer training periods.

- **Batch size (b):** It controls the number of examples sampled from the training dataset to be used in each training iteration. While utilizing larger batch sizes can expedite convergence, it may also result in memory limitations and slower training durations.

- **Joint hyperparameter (λ):** It is used in models that perform joint learning of two tasks, and it controls the trade-off between the two models' hyperparameter.

- **Self-adversarial sampling temperature (α):** This is the hyperparameter used in RotatE and HRotatE to control the degree of randomness introduced in the negative sampling process. Opting for a lower value results in more cautious sampling, whereas selecting a higher value leads to more daring sampling.

- **Fixed margin (γ):** It is the hyperparameter used in the loss function of RotatE and HRotatE to control the minimum distance between positive and negative samples.

- **The L2 coefficient:** This is a regularization hyperparameter that controls the amount of L2 regularization applied to the model's weights during training.
- **Adam, Adagrad, and SGD:** These are optimization algorithms used to update the model's parameters during training.
- **The number of preferences:** This is a predefined hyperparameter that determines the number of preferences in the set of latent features P used for the preference induction process in TUP.
- **Maximum Step:** This is a hyperparameter that determines the maximum number of iterations the model undergoes during training.

We conducted a grid search to determine the optimal hyperparameters for both the recommendation and KGC tasks. For the learning rate η, we searched in {0.0005, 0.005, 0.001, 0.05, 0.01} and ultimately set it to 0.001. The L2 coefficient value was searched in $\{10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 0\}$ and set to 10^{-5}. The self-adversarial sampling temperature α was searched in the values of {0.5, 1.0} and we set it to 1.0. The fixed margin γ was searched in {3, 6, 9, 12, 18, 24, 30} and set to 24. The joint hyperparameter λ was used to balance the parameters of the two tasks during joint learning and was set to 0.5 in MovieLens and 0.7 in DBook after searching in {0.7, 0.5, 0.3}. The embedding and batch sizes were empirically set to 100 and 256, respectively. In both MovieLens-1m and DBook2014 datasets, the number of preferences was predefine as the same number of distinct relations in the triplets, which amounts to 20 and 13 different preferences, respectively. Adam [36] was selected as the optimizer out of Adagrad and SGD. Moreover, we set the maximum number of steps to 140,000 for RotatE. However, to demonstrate the effectiveness of HRotatE, we set the maximum number of training steps to 70,000.

To ensure a fair comparison, we trained RKTUP and all baseline models in the same environment and with the same hyperparameters.

We list the values of the hyperparameters for all models in Table 4.2.1

Table 4.2.1: Models' hyperparameters

Search method	Hyperparameter	Value
Grid search	η	0.001
	L2 coefficient	10^{-5}
	α	1.0
	γ	24
	λ	0.5 (MovieLens) and 0.7 (DBbook)
	Optimizer	Adam
Empirically	k	100
	b	256
Predefined	preferences	20 (MovieLens) and 13 (DBbook)

4.3 Evaluation Metrics

Evolution metrics are commonly utilized in machine learning to assess a model's performance and efficacy, as it is trained on more data or as new models are developed. These metrics are crucial for evaluating a model's capability to learn from data, make precise predictions and enhance its performance. Therefore, it is vital to choose the appropriate metric for learning and evaluating the machine learning model. While accuracy is a popular metric, it is not always suitable for evaluating recommender or link prediction systems, as these tasks are rank-based.

For example, consider a movie recommendation system that has a database of 10,000 movies and a user has only watched 50 of them. In this case, the vast majority of movies are not relevant to the user, and simply predicting that they will dislike every movie will still result in a high accuracy score, but an ineffective recommendation.

Several decision support metrics are commonly used to evaluate the performance of recommender systems, depending on the specific needs and goals of the task. For the Recommendation Task, we measured the performance of our approach using five metrics. These are Precision, Recall, F1 score, Hit ratio and Normalized Discounted Cumulative Gain (nDCG).

As for the knowledge graph completion task we used two of the common metrics that have been widely used in the literature (Hit ratio and Mean Rank).

Regarding our experiments, the mean value of Precision, Recall and Hit ratio for all users is considered as the final result, where the value of N for Top-N recommendation is set to 10 in both recommendation and KGC tasks. Additionally, in the KGC task, the final result of the Hit ratio is computed as the mean of all triples.

We discuss these matrices in the following sections.

4.3.1 Precision

Precision measures the proportion of recommended items that are relevant to the user, i.e., out of the top N recommendations, what proportion of items the user actually liked or found useful? It is also denoted as Precision@N.

Precision is computed using equation (4.3.1).

$$Precision = \frac{\text{True Positive}}{\text{True Positive + False Positive}} \tag{4.3.1}$$

where " True Positive" is the number of items in the recommendation list that the user found relevant or useful, and " True Positive + False Positive" is the total number of items in the recommendation list.

For example, if a recommender system suggests ten items to a user, and the user finds four of them relevant, the precision of the system for that user is 0.4 or 40%. The higher the precision, the better the system is at recommending items that the user likes or finds useful.

4.3.2 Recall

Recall measures The fraction of items relevant to the user that were successfully recommended to the user among the top N recommendations. It is also denoted as Recall@N.

Recall is computed using equation (4.3.2).

$$Recall = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negatives}} \tag{4.3.2}$$

True Positive represents the number of relevant items that were correctly recommended, and False Negatives represents the number of relevant items that were not recommended by the system.

Recall is an important metric when the goal is to ensure that no relevant items are missed, as it penalizes the system more for missing relevant items than for recommending irrelevant items. A higher recall indicates that the recommender system is recommending more of the relevant items to the user.

However, the high recall does not necessarily guarantee high precision. A system with high recall may also have a lot of false positives, which are irrelevant items that were also recommended. Therefore, a good recommender system should aim for a balance between high recall and high precision. Thus we need to use a metric like The F1 score.

4.3.3 F1 score

The F1 score is a measure of a model's performance that combines both precisions and recall into a single value. It provides a way to balance the tradeoff between precision and recall. The F1 score is calculated using the harmonic mean of precision and recall as shown in equation (4.3.3).

$$F1\ score = \frac{2 \times precision \times recall}{precision + recall} \tag{4.3.3}$$

The F1 score ranges between 0 and 1, with a higher score indicating better performance. A perfect F1 score of 1 indicates that the model has perfect precision and recall.

4.3.4 nDCG

Normalized Discounted Cumulative Gain (nDCG) is a measure of ranking quality. It measures how well a ranking algorithm ranks a set of items based on their relevance to a user's preferences. The further down the list of results you go, the more the metric " discounts" the result, so it pays to have the best results up top. This metric is also denoted as nDCG@N, where N represents the length of the recommendation list. It is computed using equation (4.3.4).

$$NDCG_p = \frac{DCG_p}{IDCG_p} \tag{4.3.4}$$

where p is the position of the last item in the ranking that is considered, DCG_p is the discounted cumulative gain at position p, and $IDCG_p$ is the ideal discounted cumulative gain at position p.

Discounted Cumulative Gain (DCG) at position p is computed as:

$$DCG_p = \sum_{i=1}^{p} \frac{2^{rel_i} - 1}{\log_2(i+1)} \tag{4.3.5}$$

where rel_i is the relevance of the item at position i in the ranking. The relevance is typically a binary or ordinal value that reflects how relevant the item is to the user's query or preferences.

The Ideal Discounted Cumulative Gain (IDCG) at position p is the DCG score that would be obtained if the ranking contained only relevant items, and they were perfectly ranked according to their relevance. IDCG is computed as:

$$IDCG_p = \sum_{i=1}^{min(p,|R|)} \frac{2^{rel_i} - 1}{\log_2(i+1)} \tag{4.3.6}$$

where $|R|$ is the total number of relevant items in the dataset.

NDCG ranges between 0 and 1 with higher values indicating better ranking quality.

4.3.5 Hit Ratio

For the recommendation task, the hit ratio is a way to test how good the top-N recommendations are. It measures how many relevant items made it to the top N recommendation list. It is also denoted as Hit@N. It is calculated using equation (4.3.7).

$$hit\ rate = \frac{hits}{total\ recommendations} \tag{4.3.7}$$

where hits represent the number of recommended items that the user actually interacted with, and total recommendations represent the total number of items recommended to the user.

Hit ratio can also be used to assess the knowledge graph completion model's ability to predict the correct head or tail entity for a given relation among the top N-ranked candidate entities. Equation (4.3.8) is used to compute Hit Ratio@N in the KGC task.

$$\text{Hit@N} = \frac{1}{|\mathcal{S}|} \sum_{(h,r,t)\in\mathcal{S}} \delta(t \in \text{Top-N}_h^r) \tag{4.3.8}$$

where $\mathcal{S}$ is the set of test triples, (h, r, t) represents a triple with head entity h, relation r and tail entity t. Top-N_h^r denotes the top-N ranked candidate entities for the triple $(h, r, ?)$ where " ?" denotes a missing entity (either head or tail). δ is the indicator function, which returns 1 if the true tail entity t is present in the set of top-N ranked candidate entities and 0 otherwise.

The higher the Hit@N, the better the model is at predicting missing entities in the knowledge graph and relevant items to the user in the recommendation list.

4.3.6 Mean Rank

This metric measures How good the knowledge graph completion model is at guessing the missing facts in a KG. It measures the average rank of the true tail entity among all possible entities in the knowledge graph, given a head entity and a relation.

The Mean Rank is computed using equation (4.3.9)

$$\text{Mean Rank} = \frac{1}{|\mathcal{S}|} \sum_{(h,r,t)\in\mathcal{S}} \text{rank}(h, r, t) \tag{4.3.9}$$

where $\mathcal{S}$ is the set of triples in the test set, and $\text{rank}(h, r, t)$ is the rank of the true tail entity t among all possible entities in the knowledge graph, according to the score assigned by the model for the triple (h, r, t). A lower Mean Rank indicates better performance, as it means that the true tail entity is ranked higher on average.

In this chapter, we discussed the implementation requirements, provided details about the dataset used, and the evaluation matrices and described the hyperparameters that were set to achieve optimal results. The next chapter will focus on presenting a detailed analysis of our findings.

CHAPTER 5

Results, Analysis, and Discussion

In this chapter, we evaluate our model's performance by comparing it with several state-of-the-art models for both the recommendation and knowledge graph completion tasks. Specifically, for the recommendation task, we compare our model against classical recommender systems like FM [52] and BPRMF [53], as well as three joint-learning models (CFKG [78], CKE [76], and CoFM [51]) and a multi-task learning model (KTUP [11]). For the knowledge graph completion task, we compare our model against TransE [9], TransH [71], and TransR [44].

To conduct our experiments, we used two widely-used benchmark datasets. The entities in both datasets were mapped to DBpedia when mappings were available. Chapter 4 provides details on the experimental setup and hyperparameters used.

To further improve the efficacy of our model, we tested it using HRotatE instead of RotatE for the KGC task. When using HRotatE, we trained RKTUP with half the number of training steps required when using RotatE.

Our comparative results demonstrate the importance of incorporating the vast majority of relation patterns from the KG into the RS, including the compositional relations that were previously overlooked in previous studies. Additionally, we found that RKTUP converged to the optimum solution much faster when using HRotatE for the KGC task.

5.1 Result Analysis and Discussion

5.1.1 Recommender System's Performance

Table 5.1.1: Recommender systems' experimental results on two datasets.

Models	MovieLens-1M (@10, %)					DBook2014 (@10, %)				
	Percis.	Recall	F1	Hit	NDCG	Percis.	Recall	F1	Hit	NDCG
FM	28.91	12.07	12.34	80.01	58.74	4.19	20.89	4.99	31.28	19.87
BPRMF	31.07	13.10	12.98	82.76	60.56	4.59	21.03	6.14	30.67	22.38
CKE	37.54	15.87	18.68	87.32	66.98	4.01	22.73	6.60	34.09	27.35
CFKG	30.14	11.82	13.76	81.93	57.86	4.50	20.85	4.28	30.01	18.94
CoFM(share)	31.78	12.53	15.62	82.98	57.19	3.27	21.04	6.31	29.80	20.87
CoFM(reg)	30.87	11.98	13.05	81.32	57.45	4.05	20.97	5.13	29.32	20.98
TUP(hard)	36.67	16.71	18.83	88.12	66.81	3.98	22.08	4.99	30.14	20.35
TUP(soft)	36.03	15.94	18.41	87.95	66.17	3.86	22.15	7.08	30.59	22.67
KTUP(hard)	40.90	17.11	18.72	87.88	69.04	4.86	25.06	7.50	34.38	28.16
KTUP(soft)	41.06	17.64	18.75	87.94	69.31	4.89	25.12	7.53	34.50	28.25
RKTUP$(hard)_R$	46.85	19.24	21.15	89.72	74.96	5.78	29.61	8.62	38.20	30.79
RKTUP$(soft)_R$	47.26	19.84	21.18	90.34	75.08	5.81	29.80	8.70	38.64	31.07
RKTUP$(hard)_H$	46.91	19.31	21.20	89.87	75.12	5.81	29.73	8.63	38.29	30.86
RKTUP$(soft)_H$	**47.28**	**19.89**	**21.23**	**90.42**	**75.23**	**5.85**	**29.82**	**8.73**	**38.70**	**31.16**

In table 5.1.1 Precis. refers to precision, F1 refers to the F1 score and Hit refers to the Hit Ratio. Additionally, we use abbreviations to indicate the specific models and strategies used. CoFM(share) indicates the CoFM model used with its shared entity strategy, while CoFM(reg) indicates CoFM used with its entity regularization strategy. TUP(hard) refers to training the TUP recommender system without the KGC task, using the hard strategy for preference induction, while TUP(soft) uses the soft strategy. Similarly, KTUP(hard) and KTUP(soft) indicate the hard or soft strategies used for preference induction in TUP. Lastly, the R in RKTUP$(hard)_R$ and RKTUP$(soft)_R$ indicates the use of RotatE for the KGC task, while the H in RKTUP$(hard)_H$ and RKTUP$(soft)_H$ indicates the use of HRotatE. The same notations are used in table 5.1.2. The results presented in both tables for RKTUP model strategies represent an average of five runs for each strategy.

The experimental results of several recommender systems, including our model RKTUP, on two datasets, MovieLens-1M and DBook2014 are presented in Table 5.1.1. Looking at the results, we can see that RKTUP outperformed several state-of-the-art models, such as FM, BPRMF, CKE, CFKG, CoFM and KTUP on both datasets based on precision, recall, F1 score, hit ratio and NDCG metrics.

Specifically, RKTUP$(soft)_H$ outperforms the state-of-the-art models and shows comparable results to RKTUP$(soft)_R$ but with only half the number of required training steps (i.e. 70,000 v.s. 140,000). This is due to RotatE and HRotatE's ability to detect various relationship patterns, including symmetry, inversion and composition. HRotatE also converges faster than RotatE.

Furthermore, RKTUP demonstrated more improvements on DBbook2014 than MovieLens-1m (i.e., 13.7% vs. 11.6% gains in F1). This suggests that integrating more knowledge is particularly helpful for sparse data.

However, RKTUP performs better on MovieLens-1m compared to DBbook2014, which may be caused by the higher level of data sparsity in DBbook2014. The data sparsity in DBbook2014 is 99.6%. This led to several items not having corresponding entities in the KG, thus restricting the performance of all the recommender systems on DBbook2014.

The results highlight the effectiveness of integrating compositional relations, and various other relation patterns, from the KG into the RS to enhance recommendation performance. The enriched embeddings with additional knowledge led to a noticeable improvement in recommendation performance. This enhancement was achieved through the rotational-based KGE method's ability to handle complex relationships. Moreover, the superiority of RKTUP over classical and joint learning models further accentuates the significance of joint learning of item recommendations and KG completion to enhance recommendation systems' performance. Figures 5.1.1 and 5.1.2 are visualization of the results.

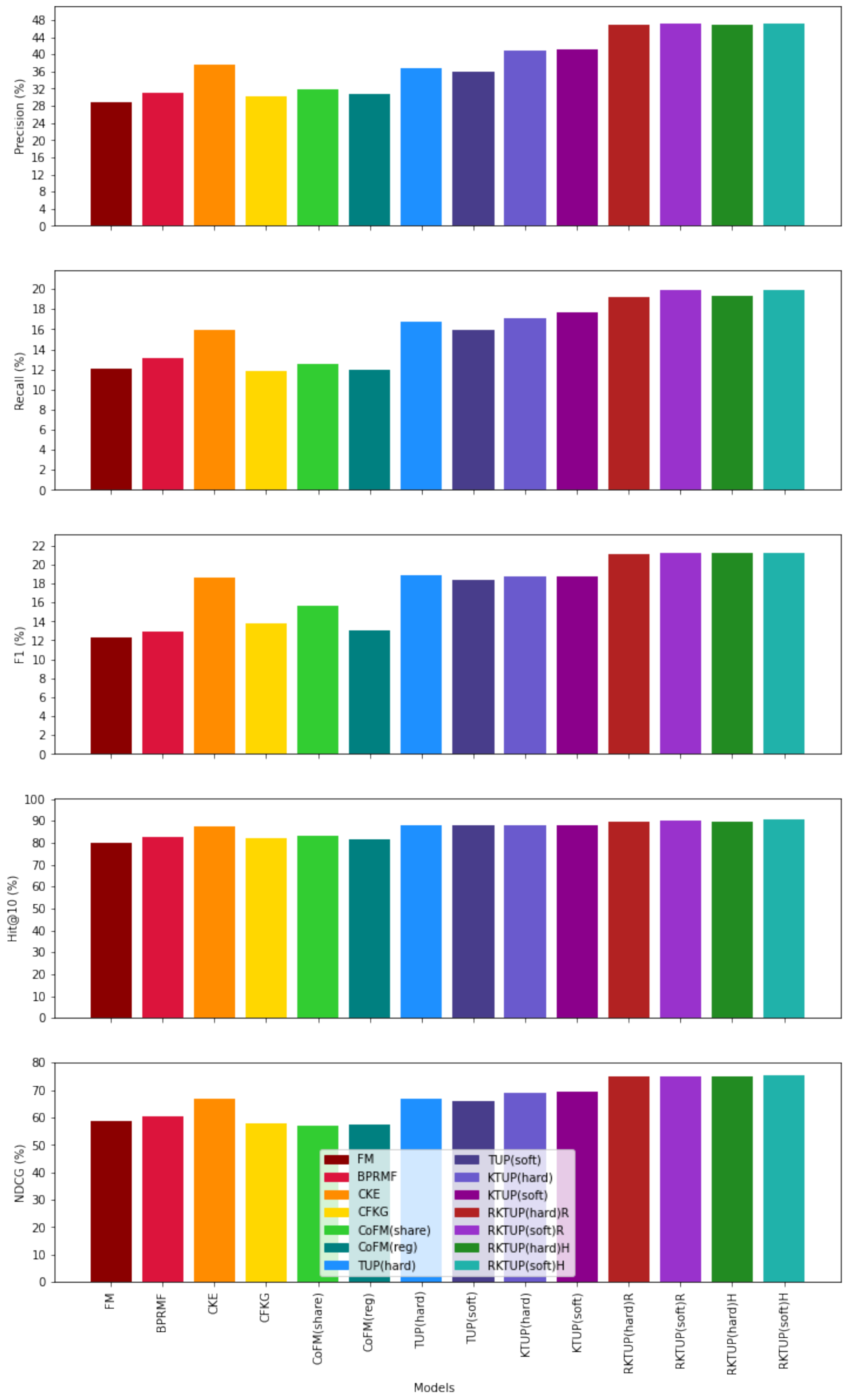

Fig. 5.1.1: Comparing RS Models on MovieLens-1M

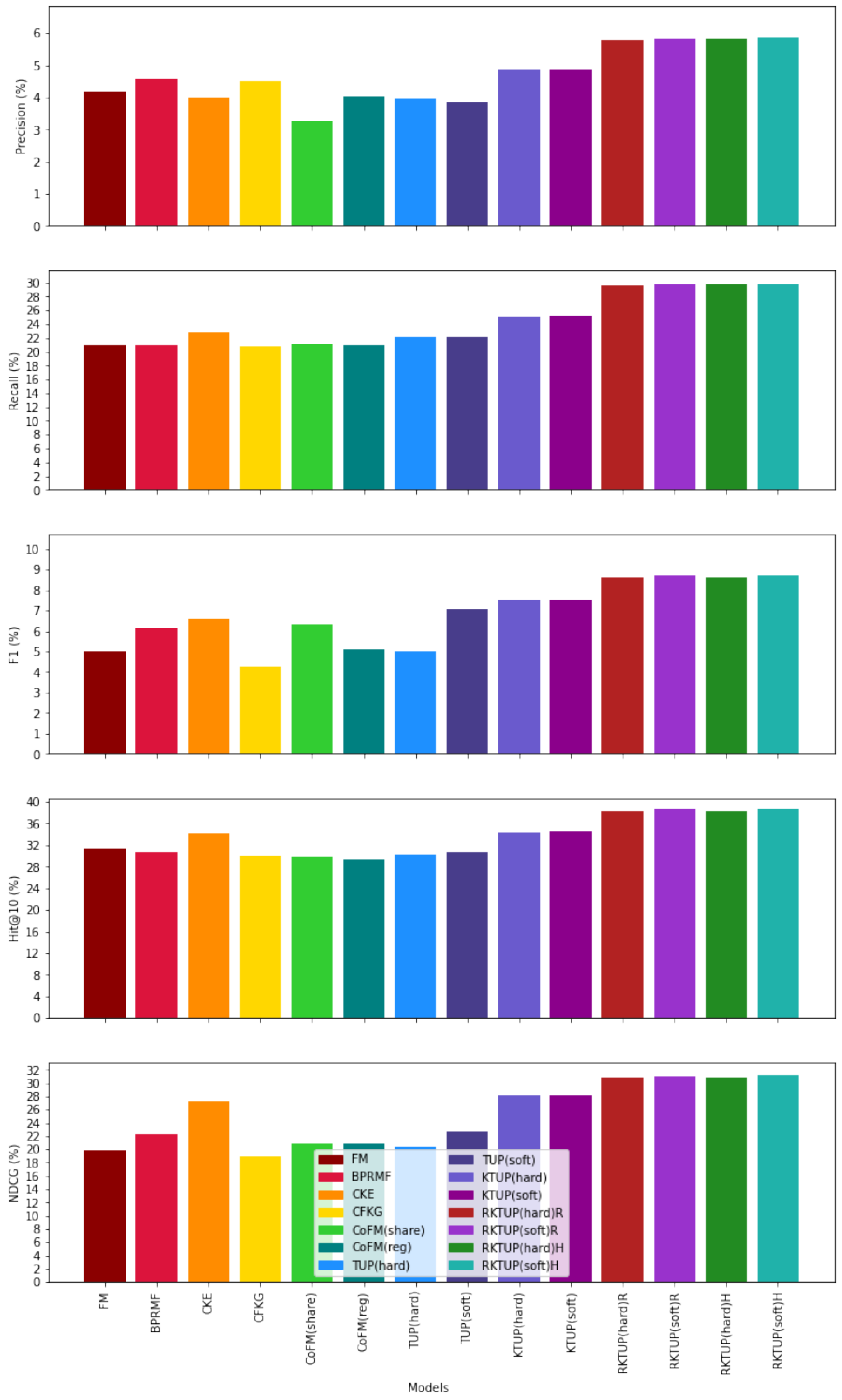

Fig. 5.1.2: Comparing RS Models on DBbook2014

Table 5.1.2: Knowledge graph completion experimental results on two datasets.

Model	**MovieLens-1M**		**DBook2014**	
	Hit @10 (%)	**MR**	**Hit @10(%)**	**MR**
TransE	47.50	536	59.87	532
TransH	46.85	536	61.02	554
TransR	40.29	608	55.23	565
CKE	35.26	584	55.02	592
CFKG	42.16	522	58.93	547
CoFM(share)	47.65	513	58.01	529
CoFM(reg)	47.38	505	61.09	521
KTUP(hard)	49.00	525	59.94	502
KTUP(soft)	49.68	526	60.37	499
RKTUP$(hard)_R$	56.19	499	67.66	474
RKTUP$(soft)_R$	56.42	494	68.07	466
RKTUP$(hard)_H$	56.30	498	67.71	473
RKTUP$(soft)_H$	**56.47**	**493**	**68.13**	**465**

Table 5.1.2 presents the overall performance of the KGC task by several models, and it can be observed that RKTUP$(soft)_H$ outperforms all other models on both datasets. In comparison to KTUP(soft), RKTUP$(soft)_H$ achieves a higher Hit Ratio for MovieLens-1m as compared to that for DBbook2014 (13.6% v.s. 12.8%) which can be due to the fact that MovieLens-1m has more connectivities between users and items that can help in modeling structural knowledge between entities.

Furthermore, the results demonstrate that RKTUP$(soft)_H$ performs similarly to RKTUP$(soft)_R$ while requiring only half of the training steps needed for the KGC task by RKTUP$(soft)_R$, i.e. 70,000 steps v.s. 140,000 steps. These findings provide evidence to support our initial hypothesis that integrating rotational-based KGE methods into an MTL model can lead to better performance. Moreover, the results suggest that using HRotatE enhances the efficiency of RKTUP by using half the number of training steps required by RotatE to achieve comparable results. Figures 5.1.3 and 5.1.4 visualize the results.

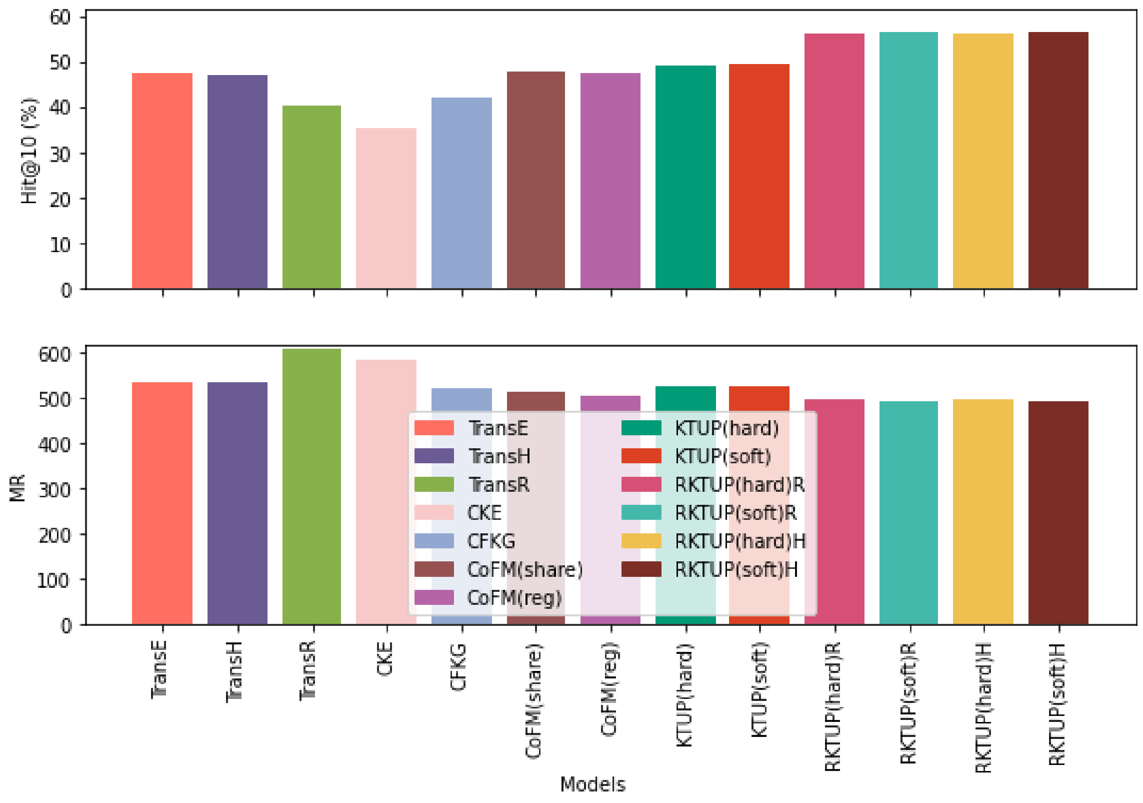

Fig. 5.1.3: Comparing KGC Models on MovieLens-1M

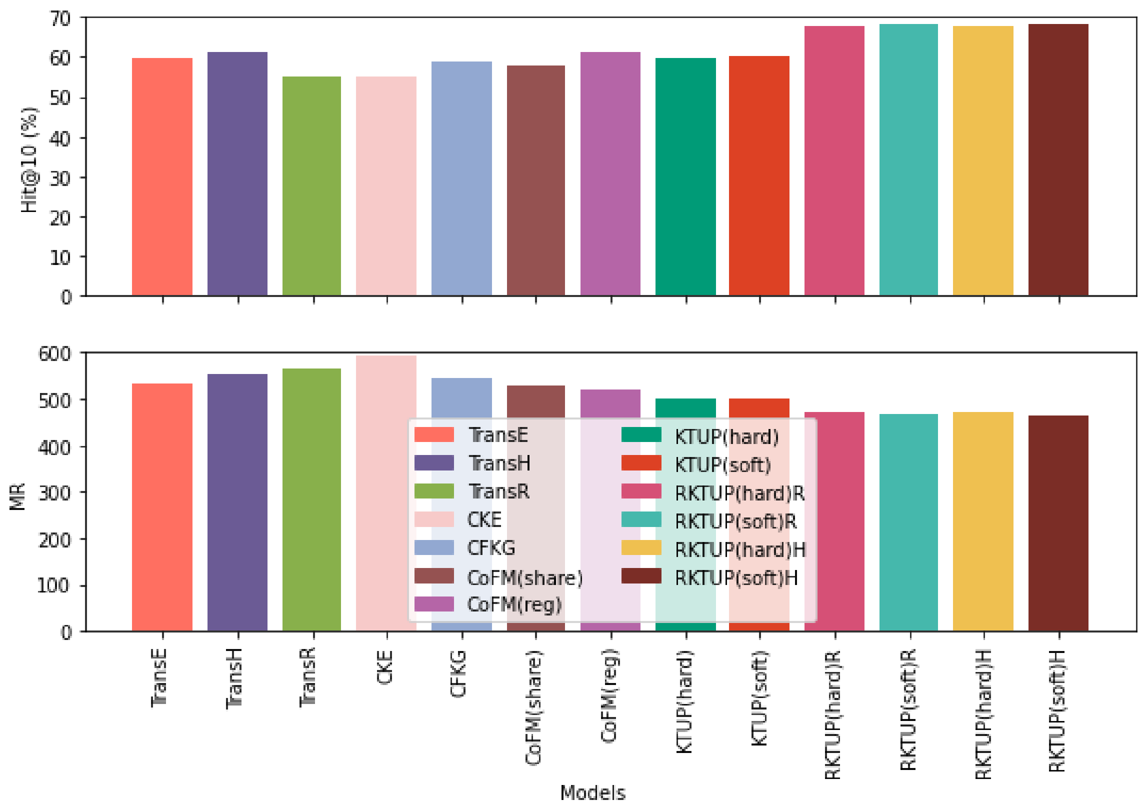

Fig. 5.1.4: Comparing KGC Models on DBbook2014

5.1.2 Assumptions

The RKTUP model is designed with certain assumptions about the datasets used in the study and the factors that can influence a user's preference. The model assumes that both datasets have an adequate number of composition patterns. To ensure a fair comparison with the KTUP model, RKTUP adopted the same assumptions. These assumptions dictate that a user's preference is solely determined by their interactions with the items and knowledge graph entities used in the study and do not take into account external factors such as social or cultural influences. Lastly, all other models used in the study were trained under the same set of assumptions as RKTUP.

5.1.3 Test Case of RKTUP Model

In this section, we demonstrate a test case of generating personalized recommendations for UserID: 1920 from the MovieLens 1M dataset using RKTUP$(soft)_H$ and

KTUP(soft) models. This test case serves as an example to compare the performance of the RKTUP$(soft)_H$ and KTUP(soft) models.

The user 1920 is a female aged between 35-44 who works in an executive/managerial position. She rated 601 movies in the MovieLens 1M dataset, with 252 movies from the Comedy/Romance genre receiving high ratings. Additionally, she rated 210 movies from the Drama genre, while the remaining 139 ratings were divided among other genres such as Action, Adventure, Children's, Sci-Fi, Thriller, Musical, War, Animation, Western, Crime, Horror, Mystery, and Documentary. TUP's preference induction with the soft strategy revealed that user 1920 prefers movies from the Comedy/Romance, movies starring Tom Hanks, movies directed by Nora Ephron, and movies from drama genres, in this order based on the weights assigned to them by the attention mechanism.

Table 5.1.3 displays the top 10 movie recommendations generated for user 1920 by RKTUP$(soft)_H$ and KTUP(soft). The recommended movies are listed in the same order as their ranking based on the recommendation and the plausibility scores produced by the models. For example, in RKTUP$(soft)_H$ "Sleepless in Seattle" and "You've Got Mail" were ranked higher than "That Thing You Do!" even though all three movies are in the Comedy/Romance genre. This is because the first two movies starred Tom Hanks and were directed by Nora Ephron, which is more relevant to the user's preference.

Furthermore, RKTUP provided better recommendations for this user compared to KTUP. For example, let's compare the second recommendation generated by both models. KTUP recommended "Forrest Gump", which is a Comedy/Romance that starred Tom Hanks but did not fit the user's preferences, as well as, RKTUP$(soft)_H$'s recommendation. In contrast, RKTUP$(soft)_H$ recommended "You've Got Mail", which is a Comedy/Romance that starred Tom Hanks and was directed by Nora Ephron, making it a better fit for the user's preferences. Therefore, RKTUP$(soft)_H$'s recommendations were more relevant to the user's interests.

Lastly, in this example, RKTUP$(soft)_H$ method outperformed the KTUP(soft) method with a 15.7% increase in F1 score. Specifically, RKTUP$(soft)_H$ achieved a precision of 0.67, a recall of 0.86, and an F1 score of 0.75, while KTUP(soft) achieved a precision of 0.62, a recall of 0.71, and an F1 score of 0.66. It should be noted that the results presented in Table 5.1.1 represent the average performance across all users.

Table 5.1.3: Test Case of RKTUP Model

RKTUP$(soft)_H$		**KTUP(soft)**	
Movie	**Genre**	**Movie**	**Genre**
Sleepless in Seattle (1993)	Comedy/Romance	Sleepless in Seattle (1993)	Comedy/Romance
You've Got Mail (1998)	Comedy/Romance	Forrest Gump (1994)	Comedy/Romance
That Thing You Do! (1996)	Comedy/Romance	Animal House (1978)	Comedy
Notting Hill (1999)	Comedy/Romance	Young Guns (1988)	Action/Comedy
Remember the Titans (2000)	Drama	Driving Miss Daisy (1989)	Drama
King of Masks, The (Bian Lian)	Drama	Highlander (1986)	Action/Adventure
Gladiator (2000)	Action/Drama	The Deer Hunter (1978)	Drama/War
The Patriot (2000)	Action/Drama	Remember the Titans (2000)	Drama
Sanjuro (1962)	Action/Adventure	Dr. No (1962)	Action
Double Jeopardy (1999)	Action/Thriller	Sleeping Beauty (1959)	Children's

This chapter examines the experimental results and compares them with state-of-the-art models. To conduct our experiments, we used two widely-used benchmark datasets in the field. These datasets were refined for LODRecSys, and in cases where mappings were available, the entities were mapped to DBpedia.

To further improve the efficacy of our model, we tested it using HRotatE instead of RotatE for the KGC task. When using HRotatE, we trained RKTUP with half the number of training steps required when using RotatE.

Our comparative results empirically demonstrate the importance of modeling and inferring different relation patterns to enhance the performance of recommendation and KG completion. Additionally, we found that RKTUP converged to the optimum solution much faster when using HRotatE for the KGC task.

CHAPTER 6

Conclusion and Future Work

6.1 Conclusion

This thesis focused on enhancing the performance of MTL recommendation systems by introducing a new MTL model called RKTUP. This model builds upon an existing MTL model, KTUP, and improves its ability to learn the representations of users, items, entities and relations. By using RKTUP, the recommender system can gain a better understanding of a user's preferences, thereby providing more personalized recommendations. This is evidenced by the increase in F1 score that RKTUP achieved compared to the KTUP model on the DBbook2014 and MovieLens-1m datasets, with improvements of 13.7% and 11.6%, respectively.

The model uses a knowledge graph and a user-item interaction list as inputs and rotation-based KGE approaches such as RotatE or HRotatE.

Knowledge graph Embedding-based recommendation systems (KGE-based RS) have been shown to be effective in providing personalized recommendations to users. However, these systems rely heavily on the quality of entity and relation representations in the knowledge graph. Inaccurate or incomplete representations can lead to poor recommendation performance. Although numerous KGE-based RS are available, they either overlook the inherent incompleteness of knowledge graphs or employ KGE models that cannot adequately capture complex relation patterns, such as composition.

Therefore, we have proposed a new Multi-Task Learning (MTL) model that utilizes Rotation-based Knowledge Graph Embedding (KGE) techniques like RotatE or

HRotatE to infer complex relationship patterns for the Knowledge Graph Completion (KGC) task to overcome the limitations of TransH in KTUP.

We evaluated the effectiveness of RKTUP using two different rotational-based KGE models, RotatE and HRotatE, on two benchmark datasets that are widely used. The experimental results revealed that RKTUP outperforms several state-of-the-art methods (including classical models and several KGE-based RS models) in recommendation and KG completion tasks when it uses RotatE or HRotatE.

Specifically, we showed that RKTUP can capture relation patterns better, leading to improved user preference representation and recommendation performance. Additionally, using HRotatE in RKTUP improved its efficiency by reducing training steps by half while achieving comparable results to RotatE. Overall, the results of this thesis contribute to the development of KGE-based RS by identifying the importance of multi-task learning models that consider KG incompleteness and the potential of advanced KGE methods to improve the recommendation's performance.

6.2 Limitations and Future Research

We identified some limitations of our work and potential directions for future research. First, our proposed model uses a rotational-based embedding approaches that fail to capture hierarchical relations. RKTUP can be extended further by investigating advanced KGE models capable of capturing and inferring more complex relation patterns, such as hierarchical relations.

Second, RKTUP employs static KGE models that do not account for changes in entities and relations over time. In contrast, dynamic KGE methods capture temporal dynamics by adapting to evolving relationships in the graph. Therefore, exploring how RKTUP's performance might be enhanced by integrating dynamic KGE methods that can capture changes in the graph over time would be a promising avenue for future research.

Finally, the main objective of this thesis was to improve the performance of MTL models, but there are other challenges that recommender systems commonly face, such as the cold start problem, data sparsity, interpretability and the long-tail problem. Therefore, future investigations could explore the effectiveness of RKTUP in addressing these challenges.

REFERENCES

[1] (2014). Linked open data-enabled recommender systems challenge. `https://2014.eswc-conferences.org/important-dates/call-RecSys.html`.

[2] Abdi, H. (2007). Singular value decomposition (svd) and generalized singular value decomposition. *Encyclopedia of measurement and statistics*, 907:912.

[3] Abdollahpouri, H., Burke, R., and Mobasher, B. (2019). Managing popularity bias in recommender systems with personalized re-ranking. In *The thirty-second international flairs conference*.

[4] Amer, A. A., Abdalla, H. I., and Nguyen, L. (2021). Enhancing recommendation systems performance using highly-effective similarity measures. *Knowledge-Based Systems*, 217:106842.

[5] Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., and Ives, Z. (2007). Dbpedia: A nucleus for a web of open data. In *The semantic web*, pages 722–735. Springer.

[6] Basilico, J. and Hofmann, T. (2004). Unifying collaborative and content-based filtering. In *Proceedings of the twenty-first international conference on Machine learning*, page 9.

[7] Bi, L., Wang, Y., Qu, D., and Guan, B. (2018). A hybrid recommendation method with multilayer perception applied on real world data. In *Proceedings of the 2018 International Conference on Algorithms, Computing and Artificial Intelligence*, pages 1–7.

[8] Bobadilla, J., Ortega, F., Hernando, A., and Gutiérrez, A. (2013). Recommender systems survey. *Knowledge-based systems*, 46:109–132.

[9] Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., and Yakhnenko, O. (2013). Translating embeddings for modeling multi-relational data. *Advances in neural information processing systems*, 26.

[10] Burke, R. (2002). Hybrid recommender systems: Survey and experiments. *User modeling and user-adapted interaction*, 12(4):331–370.

[11] Cao, Y., Wang, X., He, X., Hu, Z., and Chua, T.-S. (2019). Unifying knowledge graph learning and recommendation: Towards a better understanding of user preferences. In *The world wide web conference*, pages 151–161.

[12] Chen, W., Cao, Y., Feng, F., He, X., and Zhang, Y. (2022). Explainable sparse knowledge graph completion via high-order graph reasoning network. *arXiv preprint arXiv:2207.07503*.

[13] Chen, Z., Wang, Y., Zhao, B., Cheng, J., Zhao, X., and Duan, Z. (2020a). Knowledge graph completion: A review. *Ieee Access*, 8:192435–192456.

[14] Chen, Z., Wang, Y., Zhao, B., Cheng, J., Zhao, X., and Duan, Z. (2020b). Knowledge graph completion: A review. *IEEE Access*, 8:192435–192456.

[15] Chen, Z., Wang, Y., Zhao, B., Cheng, J., Zhao, X., and Duan, Z. (2020c). Knowledge graph completion: A review. *Ieee Access*, 8:192435–192456.

[16] Costabello, L., Pai, S., McCarthy, N., and Janik, A. (2020). Knowledge graph embeddings tutorial: From theory to practice. https://kge-tutorial-ecai2020.github.io/.

[17] DBpedia (accessed February 14, 2023). Dbpedia dataset releases. `https://www.dbpedia.org/dataset-releases/`.

[18] Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., and Harshman, R. (1990). Indexing by latent semantic analysis. *Journal of the American society for information science*, 41(6):391–407.

[19] Dettmers, T., Minervini, P., Stenetorp, P., and Riedel, S. (2018). Convolutional 2d knowledge graph embeddings. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32.

[20] Dong, Z., Wang, Z., Xu, J., Tang, R., and Wen, J. (2022). A brief history of recommender systems. *arXiv preprint arXiv:2209.01860*.

[21] Du, H. and Tang, Y. (2021). Symbiosis: A novel framework for integrating hierarchies from knowledge graph into recommendation system. In *Knowledge Science, Engineering and Management: 14th International Conference, KSEM 2021, Tokyo, Japan, August 14–16, 2021, Proceedings, Part I 14*, pages 242–254. Springer.

[22] Dziugaite, G. K. and Roy, D. M. (2015). Neural network matrix factorization. *arXiv preprint arXiv:1511.06443*.

[23] Ehrlinger, L. and Wöß, W. (2016). Towards a definition of knowledge graphs. *SEMANTiCS (Posters, Demos, SuCCESS)*, 48(1-4):2.

[24] Elmasri, R. and Navathe, S. (1989). *Fundamentals of Database Systems*. Benjamin/Cummings.

[25] Erdt, M., Fernandez, A., and Rensing, C. (2015). Evaluating recommender systems for technology enhanced learning: a quantitative survey. *IEEE Transactions on Learning Technologies*, 8(4):326–344.

[26] Guo, H., Tang, R., Ye, Y., Li, Z., and He, X. (2017). Deepfm: a factorization-machine based neural network for ctr prediction. *arXiv preprint arXiv:1703.04247*.

[27] Guo, Q., Zhuang, F., Qin, C., Zhu, H., Xie, X., Xiong, H., and He, Q. (2022). A survey on knowledge graph-based recommender systems. *IEEE Transactions on Knowledge and Data Engineering*, 34(8):3549–3568.

[28] Harper, F. M. and Konstan, J. A. (2015). The movielens datasets: History and context. *Acm transactions on interactive intelligent systems (tiis)*, 5(4):1–19.

[29] Harper, K. (2016). Harper fm, konstan ja. *The movielens datasets: History and context, ACM Transactions on Interactive Intelligent Systems (TIIS)*, 5(4).

[30] He, X., Liao, L., Zhang, H., Nie, L., Hu, X., and Chua, T.-S. (2017). Neural collaborative filtering. In *Proceedings of the 26th international conference on world wide web*, pages 173–182.

[31] Huang, J., Zhao, W. X., Dou, H., Wen, J.-R., and Chang, E. Y. (2018). Improving sequential recommendation with knowledge-enhanced memory networks. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 505–514.

[32] Huang, X., Tang, J., Tan, Z., Zeng, W., Wang, J., and Zhao, X. (2021). Knowledge graph embedding by relational and entity rotation. *Knowledge-Based Systems*, 229:107310.

[33] Jang, E., Gu, S., and Poole, B. (2017). Categorical reparameterization with gumbel-softmax. arxiv: 161101144 [cs, stat].

[34] Jannach, D. and Jugovac, M. (2019). Measuring the business value of recommender systems. *ACM Transactions on Management Information Systems (TMIS)*, 10(4):1–23.

[35] Kazemi, S. M. and Poole, D. (2018). Simple embedding for link prediction in knowledge graphs. *Advances in neural information processing systems*, 31.

[36] Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

[37] Koren, Y., Bell, R., and Volinsky, C. (2009). Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37.

[38] Kouadria, A., Nouali, O., and Al-Shamri, M. Y. H. (2020). A multi-criteria collaborative filtering recommender system using learning-to-rank and rank aggregation. *Arabian Journal for Science and Engineering*, 45:2835–2845.

[39] Lakshmi, S. S. and Lakshmi, T. A. (2014). Recommendation systems: Issues and challenges. *International Journal of Computer Science and Information Technologies*, 5(4):5771–5772.

[40] Lehmann, J., Isele, R., Jakob, M., Jentzsch, A., Kontokostas, D., Mendes, P. N., Hellmann, S., Morsey, M., Van Kleef, P., Auer, S., et al. (2015). Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic web*, 6(2):167–195.

[41] Li, Q., Tang, X., Wang, T., Yang, H., and Song, H. (2019). Unifying task-oriented knowledge graph learning and recommendation. *IEEE Access*, 7:115816–115828.

[42] Lika, B., Kolomvatsos, K., and Hadjiefthymiades, S. (2014). Facing the cold start problem in recommender systems. *Expert systems with applications*, 41(4):2065–2073.

[43] Lin, X. V., Socher, R., and Xiong, C. (2018). Multi-hop knowledge graph reasoning with reward shaping. *arXiv preprint arXiv:1808.10568*.

[44] Lin, Y., Liu, Z., Luan, H., Sun, M., Rao, S., and Liu, S. (2015a). Modeling relation paths for representation learning of knowledge bases. *arXiv preprint arXiv:1506.00379*.

[45] Lin, Y., Liu, Z., Sun, M., Liu, Y., and Zhu, X. (2015b). Learning entity and relation embeddings for knowledge graph completion. In *Twenty-ninth AAAI conference on artificial intelligence*.

[46] Lin, Y., Xu, B., Feng, J., Lin, H., and Xu, K. (2021). Knowledge-enhanced recommendation using item embedding and path attention. *Knowledge-Based Systems*, 233:107484.

[47] Liu, J. and Wu, C. (2017). Deep learning based recommendation: A survey. In *Information Science and Applications 2017: ICISA 2017 8*, pages 451–458. Springer.

[48] Luo, C., Pang, W., Wang, Z., and Lin, C. (2014). Hete-cf: Social-based collaborative filtering recommendation using heterogeneous relations. In *2014 IEEE International Conference on Data Mining*, pages 917–922. IEEE.

[49] Natarajan, S., Vairavasundaram, S., Natarajan, S., and Gandomi, A. H. (2020). Resolving data sparsity and cold start problem in collaborative filtering recommender system using linked open data. *Expert Systems with Applications*, 149:113248.

[50] Noia, T. D., Ostuni, V. C., Tomeo, P., and Sciascio, E. D. (2016). Sprank: Semantic path-based ranking for top-n recommendations using linked open data. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 8(1):1–34.

[51] Piao, G. and Breslin, J. G. (2018). Transfer learning for item recommendations and knowledge graph completion in item related domains via a co-factorization model. In *European Semantic Web Conference*, pages 496–511. Springer.

[52] Rendle, S. (2010). Factorization machines. In *2010 IEEE International conference on data mining*, pages 995–1000. IEEE.

[53] Rendle, S., Freudenthaler, C., Gantner, Z., and Schmidt-Thieme, L. (2012). Bpr: Bayesian personalized ranking from implicit feedback. *arXiv preprint arXiv:1205.2618*.

[54] Sandvig, J. J., Mobasher, B., and Burke, R. D. (2008). A survey of collaborative recommendation and the robustness of model-based algorithms. *IEEE Data Eng. Bull.*, 31(2):3–13.

[55] Sar Shalom, O., Koenigstein, N., Paquet, U., and Vanchinathan, H. P. (2016). Beyond collaborative filtering: The list recommendation problem. In *Proceedings of the 25th international conference on world wide web*, pages 63–72.

[56] Sarwar, B., Karypis, G., Konstan, J., and Riedl, J. (2001). Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*, pages 285–295.

[57] Sedhain, S., Menon, A. K., Sanner, S., and Xie, L. (2015). Autorec: Autoencoders meet collaborative filtering. In *Proceedings of the 24th international conference on World Wide Web*, pages 111–112.

[58] Shah, A., Molokwu, B., and Kobti, Z. (2021). Hrotate: Hybrid relational rotation embedding for knowledge graph. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE.

[59] Shen, T., Zhang, F., and Cheng, J. (2022). A comprehensive overview of knowledge graph completion. *Knowledge-Based Systems*, page 109597.

[60] Shi, B. and Weninger, T. (2018). Open-world knowledge graph completion. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32.

[61] Socher, R., Chen, D., Manning, C. D., and Ng, A. (2013). Reasoning with neural tensor networks for knowledge base completion. *Advances in neural information processing systems*, 26.

[62] Steffen, R. (2010). Factorization machines. In *IEEE 10th International Conference on Data Mining (ICDM)*, pages 995–1000.

[63] Stipp, D. (2017). *A Most Elegant Equation: Euler's Formula and the Beauty of Mathematics*. Hachette UK.

[64] Sun, Z., Deng, Z.-H., Nie, J.-Y., and Tang, J. (2019). Rotate: Knowledge graph embedding by relational rotation in complex space. *arXiv preprint arXiv:1902.10197*.

[65] Thorat, P. B., Goudar, R. M., and Barve, S. (2015). Survey on collaborative filtering, content-based filtering and hybrid recommendation system. *International Journal of Computer Applications*, 110(4):31–36.

[66] Trouillon, T., Welbl, J., Riedel, S., Gaussier, É., and Bouchard, G. (2016). Complex embeddings for simple link prediction. In *International conference on machine learning*, pages 2071–2080. PMLR.

[67] Wang, H., Zhang, F., Xie, X., and Guo, M. (2018). Dkn: Deep knowledge-aware network for news recommendation. In *Proceedings of the 2018 world wide web conference*, pages 1835–1844.

[68] Wang, H., Zhang, F., Zhang, M., Leskovec, J., Zhao, M., Li, W., and Wang, Z. (2019a). Knowledge-aware graph neural networks with label smoothness regularization for recommender systems. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 968–977.

[69] Wang, H., Zhang, F., Zhao, M., Li, W., Xie, X., and Guo, M. (2019b). Multi-task feature learning for knowledge graph enhanced recommendation. In *The world wide web conference*, pages 2000–2010.

[70] Wang, Q., Mao, Z., Wang, B., and Guo, L. (2017). Knowledge graph embedding: A survey of approaches and applications. *IEEE Transactions on Knowledge and Data Engineering*, 29(12):2724–2743.

[71] Wang, Z., Zhang, J., Feng, J., and Chen, Z. (2014). Knowledge graph embedding by translating on hyperplanes. In *Proceedings of the AAAI conference on artificial intelligence*, volume 28.

[72] Wen, B., Deng, S., and Chen, H. (2021). Knowledge-enhanced collaborative meta learner for long-tail recommendation. In *Knowledge Graph and Semantic Computing: Knowledge Graph and Cognitive Intelligence: 5th China Conference, CCKS 2020, Nanchang, China, November 12–15, 2020, Revised Selected Papers*, pages 322–333. Springer.

[73] West, D. B. et al. (2001). *Introduction to graph theory*, volume 2. Prentice hall Upper Saddle River.

[74] Yang, B., Yih, W.-t., He, X., Gao, J., and Deng, L. (2014). Embedding entities and relations for learning and inference in knowledge bases. *arXiv preprint arXiv:1412.6575*.

[75] Yu, X., Ren, X., Gu, Q., Sun, Y., and Han, J. (2013). Collaborative filtering with entity similarity regularization in heterogeneous information networks. *IJCAI HINA*, 27.

[76] Zhang, F., Yuan, N. J., Lian, D., Xie, X., and Ma, W.-Y. (2016). Collaborative knowledge base embedding for recommender systems. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 353–362.

[77] Zhang, S., Yao, L., Sun, A., and Tay, Y. (2019). Deep learning based recommender system: A survey and new perspectives. *ACM Computing Surveys (CSUR)*, 52(1):1–38.

[78] Zhang, Y., Ai, Q., Chen, X., and Wang, P. (2018). Learning over knowledge-base embeddings for recommendation. *arXiv preprint arXiv:1803.06540*.

[79] Zhu, R., Cai, L., Mai, G., Shimizu, C., Fisher, C. K., Janowicz, K., Lopez-Carr, A., Schroeder, A., Schildhauer, M., Tian, Y., et al. (2021). Providing humanitarian relief support through knowledge graphs. In *Proceedings of the 11th on Knowledge Capture Conference*, pages 285–288.

VITA AUCTORIS

NAME:	Lama Khalil
EDUCATION:	University of Windsor, B.Sc. in Computer Science, Windsor, ON, 2021
	University of Windsor, M.Sc. in Computer Science – AI Specialization, Windsor, ON, 2023