A Systematic Review of Deep Knowledge Graph-Based Recommender Systems, with Focus on Explainable Embeddings

Ronky Francis Doh 1,*, Conghua Zhou 1,*, John Kingsley Arthur 1, Isaac Tawiah 2,* and Benjamin Doh 3

1 Department of Computer Science, Jiangsu University, Zhenjiang 210000, China; jkarthur@vvu.edu.gh
2 Department of Automation, Shanghai Jiao Tong University, Shanghai 200240, China
3 School of Electrical and Information Engineering, Jiangsu University, Zhenjiang 210000, China; dextlin247@gmail.com
* Correspondence: rdoh@vvu.edu.gh (R.F.D.); chzhou@ujs.edu.cn (C.Z.); isaac_k_tawiah@sjtu.edu.cn (I.T.)

Abstract: Recommender systems (RS) have been developed to make personalized suggestions and enrich users’ preferences in various online applications to address the information explosion problems. However, traditional recommender-based systems act as black boxes, not presenting the user with insights into the system logic or reasons for recommendations. Recently, generating explainable recommendations with deep knowledge graphs (DKG) has attracted significant attention. DKG is a subset of explainable artificial intelligence (XAI) that utilizes the strengths of deep learning (DL) algorithms to learn, provide high-quality predictions, and complement the weaknesses of knowledge graphs (KGs) in the explainability of recommendations. DKG-based models can provide more meaningful, insightful, and trustworthy justifications for recommended items and alleviate the information explosion problems. Although several studies have been carried out on RS, only a few papers have been published on DKG-based methodologies, and a review in this new research direction is still insufficiently explored. To fill this literature gap, this paper uses a systematic literature review framework to survey the recently published papers from 2018 to 2022 in the landscape of DKG and XAI. We analyze how the methods produced in these papers extract essential information from graph-based representations to improve recommendations’ accuracy, explainability, and reliability. From the perspective of the leveraged knowledge-graph related information and how the knowledge-graph or path embeddings are learned and integrated with the DL methods, we carefully select and classify these published works into four main categories: the Two-stage explainable learning methods, the Joint-stage explainable learning methods, the Path-embedding explainable learning methods, and the Propagation explainable learning methods. We further summarize these works according to the characteristics of the approaches and the recommendation scenarios to facilitate the ease of checking the literature. We finally conclude by discussing some open challenges left for future research in this vibrant field.

Keywords: deep neural network embeddings; explainable artificial intelligence; knowledge graph embeddings; relational learning; recommender systems

1. Introduction

The explosive growth and variety of information accessible on the internet overwhelmed users all too often, making it difficult to pick out what interests them the most among a variety of choices. To improve personalized recommendations and make quality predictions, RSs have proven to be a valuable tool in assisting users to overcome information overload [1]. An RS-based algorithm, explicitly or latently, collects valuable information on its users’ preferences for a set of items (e.g., books [2], movies [3], music [4], clicked news [5], gadgets [6], and so forth) to balance factors like dispersity, accuracy, and stability in the recommendations. These algorithms can be categorized into collaborative filtering (CF)-based RSs [7], content-based RSs [8], and hybrid-based RSs [9]. We refer the reader...
to the cited papers for in-depth discussions on the types of RSs. Despite their ubiquity in recommending items, traditional RSs act as black boxes, failing to provide justifications to recommended items, thus, invoking distrust and failure of acceptance within its user base. Overall, providing explanations for recommending items is vital if the results are to be appropriately trusted, effectively managed, and understood by users.

Regardless of the scenario or the algorithm being used, generating explanations for recommendations is not an easy task. The introduction of knowledge graphs (KGs) into RSs has made it feasible to extract deep logical connections among items and users. A KG is a heterogeneous graph composed of nodes denoted as entities and edges representing relations between entities [10]. Each edge constitutes an entity-relation-entity triple (head, relation, tail), signifying that a specific relation connects two entities. Figure 1 is an example of a KG-based recommendation. Here, users, movies, actors, writers, and genres are entities, while friendship, acting, belonging, and writing denote the relations between entities. In addition, (George, Watch, Martian) represents a triple, where “George” and “Martian” are the entities, and “Watch” is the relation. Thus, items and their attributes can be mapped into the KG, making it possible to understand deep relations among user preferences and make personalized recommendations [11]. Notice that in the same Figure 1, the movies “Spectral” and “Contagion” have been recommended to “Alice”, a friend of “George”. Such machine-readable symbolic representation in the form of graphs allows to formalize and capture knowledge within specific domains and across domains and enables machines to discover knowledge in a structured but also in a serendipitous way [12]. In the literature, several academic KGs have been proposed, such as NELL [13] and DBPedia [14], as well as commercial ones such as Google KG [15], and Microsoft Satori [16].

Figure 1. Illustration of a KG-based recommendation. Here, users, movies, actors, writers, and genres are entities, while friendship, acting, belonging, and writing denotes the relations between entities.

Although the information provided in the KG can be leveraged for explainable recommendations by following the relation sequences, the underlying symbolic nature of such triples renders KGs difficult to manipulate. In addition, even though KGs have proven to be an effective approach to representing large-scale heterogeneous data, they still suffer from high computational costs when searching and matching entities in such a symbolic space. A new research direction, known as knowledge graph embedding (KGE), which resolves the above-stated issue, has gained traction in the research community [17–20]. KGEs map essential elements of a KG, including entities and relations, into a continuous vector representation, thus, simplifying the manipulations while preserving the structural information of the original KG. Given all the triples in the KG, a low-dimensional representation vector for each entity and relation can then be learned by the KGE. The usefulness of these embeddings is so immense and can be beneficial in all kinds of tasks such as relation extraction [21], entity classification [22], and entity resolution [23].

For most of the currently available KGE approaches presented in the literature, the embedding task is exclusively accomplished based on the observed triples. Thus, for a given KG, such approaches first denote entities and relations in a continuous vector space.
Then a scoring function is defined on each triple to evaluate its likelihood. The entity and relation embeddings can then be obtained by maximizing the total likelihood of the observed triples. Due to the variability of user-item attributes, providing explanations for recommended items based solely on KGEs might not be effective enough for downstream tasks [24,25], as the learned embeddings are only compatible on individual triples. Besides, the approach lacks explainability because of its disregard for semantic relations of entities that are connected by paths. Thus, path embedding methods, which employ semantic paths of entities and relations in the KG for the construction of recommendation systems, have been considered in the literature [26,27].

Indeed, due to the spectacular success of machine learning and DL algorithms, more and more RSs are incorporating these algorithms for improved performance of their predictions. A comprehensive survey of the use of DL in RSs can be found in [28,29]. These algorithms can successfully capture the non-linear and significant user-item relationships to allow the codification of more complex abstractions [30]. DL algorithms capture the complex relationships inside the data from available data sources such as textual, visual, and contextual information. However, a significant drawback of DL in RSs is their opaqueness and inability to provide explanations that are easier to understand by humans, urging for the need to improve interpretability and trustworthiness.

In recent years, there has been a growing interest in the literature about integrating symbolic knowledge-driven artificial intelligence (KG representations) and sub-symbolic data-driven algorithms (machine learning and DL) to produce hybrid, more intelligent systems to support RSs to be more explainable, transparent and trustworthy.

In this paper, we focus on the role of DKGs in the context of XAI for RSs. Typically, DKG models integrate KGs and KGEs with DL for intuitive explainable learning in RSs. The models combine either or both of KG-based models and KGEs with DL internally and provide explanations and interpretations in a form that is easier for humans to understand. This paper investigates how KGs, KGEs, and DL are merged to deliver more meaningful, insightful, and trustworthy explanations. Thus, our primary research question is how can KGs or KGEs be integrated with DL algorithms to provide more accurate predictions and trustworthy explanations in RSs? We seek to answer the above question by analyzing the literature using a systematic review framework, by providing a fine-grained hierarchical classification methodology. In the first stage, we categorize the papers into four main groups based on the extracted KG-related information and how the KGEs or path embeddings are learned and integrated with the DL algorithms. The four categories are the Two-stage explainable learning methods, the Joint-stage explainable learning methods, the Path-embedding explainable learning methods, and the Propagation explainable learning methods. In the second contribution, we further subdivide these papers according to the distinctive approaches and the scenarios for which the algorithms explain the recommendations. Finally, we provide the most relevant open challenges to be tackled to pave the way for future researchers to define a roadmap for the next wave of explainable systems, integrating KGs and DL reasoning fully. In particular, the following research contributions are presented:

- An analytical framework based on a fine-grained hierarchical taxonomy to systematically categorize recently published papers in the field of DKG-based RSs is presented;
- According to the above, we identify and discuss the strengths and limitations of using DKG in the context of explainable recommendations;
- Eventually, we analyze the relevant challenges to be pursued to promote the design of the next wave of explainable RSs.

The remainder of the paper is organized as follows: to lay down the necessary ground for our study, we present the preliminary steps and related research in Section 2 to establish the background and the working definition for XAI and provide the primary notion regarding KGs and KGEs. In Section 3, we apply our analytical survey approach to the leading current research on DKG-based RSs. Section 4 provides a summary of the systematic review process employed in this paper. We then present the most relevant open challenges
and potential research directions in this vibrant field in Section 5. Finally, the concluding remarks are given in Section 6.

2. Background and Related Work

Recently, DKG-based models as a subset of XAI have attracted significant attention in the research community. These models utilize the strengths of DL algorithms to learn, provide high-quality predictions, and complement the weaknesses of KGs to enhance the explainability of recommendations. Despite their ability to render more meaningful and trustworthy recommendations, researchers have not extensively explored DKG methodologies, and systematic reviews in this research direction is still insufficiently explored. In [31], a survey of recommendation methods based on heterogeneous information networks was presented; nonetheless, the recent development of DL-based methodologies are not covered. In [28], Mu provided a comprehensive review of the related research contents of deep learning-based RSs; however, KG-based and XAI algorithms were not considered. In [32], the authors demonstrated how KGs present interpretability to recommender systems; however, only a few works were illustrated. The paper [33], provides an overview of the use of knowledge graphs in the context of Explainable Machine Learning; nonetheless, some representative works are missing. The authors of [34] listed some representative methods; however, the inherent relations between algorithms were not introduced. Compared with the few survey papers presented in the literature, our survey paper goes deeper into the mathematical algorithms used by the DKG-based models for explainable recommendations.

In the next section, we briefly introduce the basic knowledge and summarize related works in XAI and KG-based recommendations. Moreover, before delving deeper into the state-of-the-art techniques exploiting DKGs for explainable recommendations, we first introduce the literature search methodology to lay the foundation for our survey.

2.1. Structured Knowledge in the Form of Graphs

The concept of an intelligent model that encodes real-world “things” and their inter-relationships was already available in the literature since 1980 [35]. However, the term “knowledge graph” gained popularity after Google announced its KG in 2012. Generally, a KG can be considered as a data structure representing entities and their relationships through a directed, edge-labeled graph covering several topics and often arranged in an ontological schema [10]. Reasoning over KGs can be made by employing standard knowledge representation formalisms, allowing one to define and label entities and their relationships. A popular way of describing a KG is the format provided by the Resource Description Framework (RDF) standard [36], in which nodes signify entities, whereas edges denote the exact relationship between the head entity and tail entity. The concept of entities and edges has already been introduced in Section 1. By following the “semantic web standards” [37], KGs can also be referred to as Linked Data, where high-level relations of entities can be recognized through these relational links.

For mathematical definition, let $\mathcal{E} = \{e_1, e_2, \ldots, e_k\}$ be the set of entities, and $\mathcal{R} = \{r_1, r_2, \ldots, r_k\}$ denote the set of entity relations. Then KG can be defined as a directed graph $\mathcal{G} = (\mathcal{E}, \mathcal{L})$ with entity type mapping function $\psi : \mathcal{E} \rightarrow \mathcal{B}$ and a link type mapping function $\phi : \mathcal{L} \rightarrow \mathcal{R}$, where each entity $e \in \mathcal{E}$ belongs to an entity type $\psi(e) \in \mathcal{B}$, and each link $l \in \mathcal{L}$ belonging to a relation $\phi(l) \in \mathcal{R}$.

An essential advantage of structuring knowledge in graphs as an alternative to “relational settings” is the flexibility of defining and changing data at a later stage, allowing more freedom for data evolution.

Several KGs have been made accessible due to the different standards and practices for data presentation. Table 1 presents the most commonly adopted KGs in the literature. These graphs have been classified as either “common-sense KGs” (which comprises knowledge about the everyday world), “domain-specific KGs,” (that contains knowledge peculiar to an area of expertise), “factual KGs,” (encompasses knowledge about facts from various domains), and “Proprietary KGs,” (sometimes called Enterprise Knowledge Graphs).
Table 1. Summary of KGs from the literature.

<table>
<thead>
<tr>
<th>Name of KG</th>
<th>Domain</th>
<th>Graph Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OpenCyc [38]</td>
<td>Common-Sense</td>
<td>A curated common sense-based KG realized from the Cyc knowledge base</td>
</tr>
<tr>
<td>ConceptNet [39]</td>
<td>Common-Sense</td>
<td>A crowdsourcing project which include data from WordNet, DBPedia, OpenCyc, and Wikimedia.</td>
</tr>
<tr>
<td>Freebase [40]</td>
<td>Factual</td>
<td>A crowdsourced KG that utilizes structured data from various sources, including Wikipedia schemes</td>
</tr>
<tr>
<td>Wikidata [41]</td>
<td>Factual</td>
<td>Free, collaborative database that collects structured data to support wikipedia, wikimedia, and to anyone worldwide</td>
</tr>
<tr>
<td>DBpedia [42]</td>
<td>Factual</td>
<td>A KG built from the extraction of structured content from the information created in the Wikipedia project</td>
</tr>
<tr>
<td>Yago3 [43]</td>
<td>Factual</td>
<td>An open-sourced KG built from information extracted from Wikipedia, WordNet, and GeoNames, and further linked to the DBpedia ontology</td>
</tr>
<tr>
<td>Google’s KG [44]</td>
<td>Proprietary</td>
<td>The KG used by Google and its services to enrich the results provided by its search engine.</td>
</tr>
<tr>
<td>Microsoft’s Satori [16]</td>
<td>Proprietary</td>
<td>A graph-based dataset that emerged from Microsoft Research’s Trinity and uses the RDF standard</td>
</tr>
<tr>
<td>Facebook’s KG [45,46]</td>
<td>Proprietary</td>
<td>A semantic-based dataset used by Facebook and its services to give answers to user’s natural language queries, with support from Microsoft’s Bing</td>
</tr>
<tr>
<td>WordNet [47]</td>
<td>Domain-Specific</td>
<td>A curated lexical datasets of relationships between concepts</td>
</tr>
</tbody>
</table>

2.2. Knowledge Graph Embeddings

There has been a rapid development of KGE in RS in recent years. The key notion behind KGE is to transform the entities and relations of a KG into a continuous vector space, the so-called embedding. The embeddings can then be utilized for various KG downstream tasks, such as entity classification [22], relation extraction [21], and entity resolution [23]. To perform various downstream tasks, KGEs typically follow three main steps. The first instance is to represent the entities and relations. Entities are usually represented in continuous or deterministic points in a vector space, whereas relations are often represented as operations in the vector space. A scoring function is then designed in the second instance for each triple to evaluate its plausibility. Then finally, the entity and relation embeddings are learned by solving an optimization problem that maximizes the total likelihood of an observed triple [34]. Various learning models and versions of KGEs have been proposed over the last years. In [48], Nickel et al. proposed RESCAL, the first relation learning method based on tensor decomposition that respectively encodes relations and entities into two-dimensional matrices and vectors. An improved version of RESCAL was later proposed by Yang et al. [49]. The authors of [50] proposed the “ComplEx method” to model asymmetric relations by embedding entities and relations into the complex space. Translational-based embeddings have also been proposed to deal with more complex relations. These models exploit distance-based scoring functions to measure the plausibility of a triple as the distance between two entities. TransE, which was proposed in [17], regards the relation as a translation between the head and tail entities. TransH [18] believes that an entity should possess different representations under different relations by employing relation-specific hyperplanes. TransR [19] shares a similar idea with TransH, but proposes that different relations should give attention to different attributes of entities. TransR develops relation-specific spaces rather than hyperplanes. TransD [51] simplifies TransR by breaking down the projection matrix into a product of two vectors and establishes that a relation may represent multiple semantics. Other distance models such as TranSparse [20], TransF [52], TransA [53], etc., have been proposed.
2.3. Explainable and Deep Learning Recommendations in Brief

Explainable recommendations apply to personalized recommendation algorithms that address the problem of “why” by providing users and system designers with explanations to recommendation results to clarify the reasons for recommending such results. The algorithms behind explainable recommendations can either be model-intrinsic or model-agnostic. The model-intrinsic algorithms provide interpretable models with more transparent decision mechanisms, thus, making it possible to provide explanations for the model’s decision naturally[54]. The model-agnostic approach, sometimes called post-hoc explanation algorithms, is a black box decision mechanism that develops explanation algorithms to generate explanations only after a decision has been made. This work focuses on recommendation algorithms that provide either transparent explanations or justifications for recommended items. We provide a clear distinction between transparency and justification as used in this paper. Transparency prevails if the explanations to recommended items make users aware of how the recommendation algorithm works, thus explaining somehow the underlying model behind the proposed suggestions. On the other hand, justification implies explanations that can be generated more freely and are not directly related to the recommendation algorithm. Because of the difficulty associated with algorithm explanations, justifications are preferred when it comes to user-item explainable recommendations. In summary, providing explanations to recommendations facilitates transparency by showing the user how the RS works. Explanations further assist the user in making informed and effective decisions on the choice of items. By explaining the recommended items, the user can also make rapid and efficient decisions. It also helps to improve persuasiveness, trustworthiness, and satisfaction of the recommendation systems. For system designers, explainable recommendations help diagnose, debug, and refine the recommendation algorithm quickly.

Moreover, with the rapid growth of DL models, the architectures behind modern recommendation algorithms have seen a dramatic revolution. DL algorithms learn deep representations, including learning multiple levels of abstractions and representations from data. The concept was proposed by Hinton et al. [55] in 2006, wherein an unsupervised greedy layer-by-layer training algorithm based on a deep belief network (DBN) was proposed. Since then, the concept has been applied in many Recommender-based systems, especially in industrial applications. In [30], Covington et al. proposed a deep learning-based algorithm for video recommendation on YouTube. Cheng et al. [56] presented an App-based RS for Google Play with a wide & deep model. Okura et al. [57] developed a reinforcement neural networks-based news RS for Yahoo News. Neural Networks in the form of Autoencoders have also found many used cases in RS. Autoencoders are unsupervised learning-based Neural Networks that learn a function to reconstruct the original data available at the output layer. In the training phase, autoencoders learn how to rebuild input vectors through a latent representation encoded in the hidden layers. For instance, in Semantic-Aware Autoencoder [58,59], the hidden layers of the neural networks are replaced with latent representation from a KG, thus having an explicit representation of the semantics associated with the hidden neurons and their connections. All of the above-mentioned algorithms have demonstrated significant advancement over traditional RS models.

2.4. Method

To ensure a systematic review process, we adopt the approach used in the PRISMA guidelines [60]. Figure 2 illustrates the steps utilized in our review process. It involves four major steps: Research Questions Formulation; Search Process; Literature Review and Analysis; and Data Analysis and Report Findings.
2.4.1. Research Questions

Using the PRISMA guidelines, the following research questions are formulated:

- What are the cutting-edge DKG-based techniques applied in recently published RSs?
- How do these techniques learn and integrate knowledge-graph or path embeddings with the DL methods?
- What type of explanations do the DKG-based RSs deal with? In what form are they communicated; are they human-understandable explanations or integrated interpretability based on attention networks?
- What type of data sources have been used to evaluate the techniques in the recently published works?

2.4.2. Search Process

A comprehensive approach to thoroughly search articles on Explainable KG-based and DL-based RSs from the leading academic databases, MDPI, ACM Digital Library, ScienceDirect, and Google Scholar, was performed. We keyword-searched for papers containing a combination of the keywords summarized in Table 2. We began our search on Google Scholar. We used the advanced search option “where my words occur” on Google Scholar and selected “in the title of the article” option. Our main focus was in recently published papers within the years 2018 and 2022. The Boolean operation “OR” was used for the key-word combination, and the keyword, “Knowledge graph recommender systems” was used as the “pivot” keyword. For example, using the search string combination “Knowledge graph recommender systems” OR “Explainable recommendation” returned 134 results. Searching for all the key-word combinations in Google Scholar led to a total of 178 articles. The same keyword combinations were applied in the remaining databases. We then apply the “inclusion” and “exclusion” criteria highlighted in Section 2.4.3 to guarantee a more focused and manageable search process. As shown in Figure 3, a total of 18 papers out of the 38 eligible results integrate KG or KGE with DL for explainable recommendations.

Table 2. Summary of the keywords used in the literature search.

<table>
<thead>
<tr>
<th>Keywords Combination</th>
<th>Google Scholar</th>
<th>MDPI</th>
<th>ACM</th>
<th>Science Direct</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1) Knowledge graph recommender systems</td>
<td>(1) OR (2)</td>
<td>134</td>
<td>4</td>
<td>28</td>
</tr>
<tr>
<td>(3) Deep Learning recommender systems</td>
<td>(1) OR (3)</td>
<td>13</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>(4) Explainable embeddings</td>
<td>(1) OR (4)</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>(5) Path embeddings</td>
<td>(1) OR (5)</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>(6) Knowledge graph attention network</td>
<td>(1) OR (6)</td>
<td>27</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>Total results</td>
<td></td>
<td>178</td>
<td>12</td>
<td>32</td>
</tr>
</tbody>
</table>
2.4.3. Literature Review and Analysis

The next stage is to assess the quality of the selected published articles. The primary sources were reviewed according to the concepts highlighted in the abstract and title of this paper, and the “inclusion” and “exclusion” criteria given below are enforced. Since we are interested in only Explainable DKG-based RSs, we exclude articles with other RSs. Furthermore, based on how the KG or KGEs are learned and integrated with the DL model, we divide the selected papers into four main groups, as explained in Section 1. We finally summarize them in a tabular form. Besides, only a few papers have been published within this “year range” in this particular research niche.

Inclusion criteria:
The following criteria were used to include published articles in the collected papers.

- Publications from 2018 until January 2022.
- Journals and conference proceedings.
- Research focusing on explainable knowledge graph recommender systems.
- Research works that reveal well-defined methodologies and evaluations.
- For a paper published in more than one journal, only the most complete version of the papers is chosen for inclusion.

Exclusion criteria:
The following criteria were used to exclude published articles in the collected papers.

- All publications done before 2018.
- Exclude all papers that do not offer explainability pertinent to knowledge graph recommender system.
- Exclude articles in books, standards, textbooks, and courses.
- Discard all articles written in other languages other than the English language.
- Exclude all publications that are not peer-reviewed.
- Exclude all articles with only abstracts, or without full paper.

2.4.4. Data Analysis and Report Findings

Figure 3 describes the different stages of the document selection process, which is developed according to the PRISMA guidelines [60]. Using the data from Table 2, all the 288 published results from the 4 databases (Google Scholar, MDPI, ACM, and Science Direct) were checked for duplicate records, of which 185 duplicates were removed. 39 papers from the remaining 103 results were excluded based on the “Exclusion” criteria highlighted.
in Section 2.4.3. 26 more papers were removed after reading the titles and abstracts of these papers. In all, 38 papers were selected after “screening” the records. After conducting a detail review of the 38 papers, only 18 were identified as papers that satisfy our selection criteria highlighted in the abstract for explainable DKG-based RSs (5 papers from Google Scholar, 9 papers from ACM digital library, 1 paper from MDPI, and 3 papers from Science Direct). Section 3 presents a detailed survey of our findings from the systematic review.

3. Methods of RSs with DKGs

This section presents the collected papers related to DKG-based RSs. Depending on how these papers integrate the KGs or KGEs with DL models, we categorize them into four main groups: two-stage explainable learning methods, joint-stage explainable learning methods, Path-embedding explainable learning, and the Propagation explainable methods. As stated earlier, our survey goes deeper to present the mathematical modules and algorithms employed in those collected papers. For each published paper, we highlight the drawback associated with the algorithm used, if necessary. A detailed survey of these papers is given in the subsequent sections. For ease of checking the literature, we summarize them in a tabular form. Table 3 presents a summary of collected papers. In the table, under “Mode of Explainability”, “WHUE” means “Written Human-Understandable Explanations”, “IANM”, means “Interpretability based on Attention Network Mechanism”, “TSELM” stand for “Two-Stage Explainable Learning Method”, “JSELM” means “Joint-Stage Explainable Learning Methods”, “PEELM” stands for the Path-Embedding Explainable Learning Methods, and “PEM” means the Propagation Explainable methods.

Table 3. Summary of publications reviewed.

<table>
<thead>
<tr>
<th>Method</th>
<th>Model</th>
<th>KG/KGE Used</th>
<th>DL Used</th>
<th>Mode of Explainability</th>
<th>RS Scenario</th>
</tr>
</thead>
<tbody>
<tr>
<td>TSELM</td>
<td>DEAML [61]</td>
<td>Microsoft KG</td>
<td>GRAM [62]</td>
<td>WHUE</td>
<td>Toys, Games, &amp; Musics</td>
</tr>
<tr>
<td></td>
<td>DKN [63]</td>
<td>Microsoft KG &amp; TransD KGE</td>
<td>KCNN</td>
<td>IANM</td>
<td>News</td>
</tr>
<tr>
<td></td>
<td>KSR [64]</td>
<td>Freebase KG &amp; TransE</td>
<td>RNN (GRU)</td>
<td>IANM</td>
<td>Movies</td>
</tr>
<tr>
<td></td>
<td>JKP [65]</td>
<td>MovieLens, &amp; LastFM</td>
<td>MLP</td>
<td>IANM</td>
<td>Movies &amp; musics</td>
</tr>
<tr>
<td>JSELM</td>
<td>ECFKG [66]</td>
<td>Freebase &amp; TransE</td>
<td>Fuzzy-SMA</td>
<td>WHUE</td>
<td>Amazon product review</td>
</tr>
<tr>
<td></td>
<td>ESemAuto [58]</td>
<td>DBpedia KG, &amp; Autoencoder</td>
<td>SemAuto</td>
<td>WHUE</td>
<td>Movies</td>
</tr>
<tr>
<td>PEELM</td>
<td>RuleRec [67]</td>
<td>Freebase KG</td>
<td>BPRMF &amp; NCF</td>
<td>IANM</td>
<td>Amazon product review</td>
</tr>
<tr>
<td></td>
<td>EKAR [68]</td>
<td>MovieLens, DBpedia KG &amp; ConvE KGE</td>
<td>MDP, BPR &amp; ReLU</td>
<td>IANM</td>
<td>Movies &amp; Books</td>
</tr>
<tr>
<td></td>
<td>EIUM [27]</td>
<td>MovieLens/ Freebase KG</td>
<td>WPL, FastText &amp; AlexNet</td>
<td>WHUE</td>
<td>Movies</td>
</tr>
<tr>
<td></td>
<td>KPRN [69]</td>
<td>MovieLens, KKBox datasets, &amp; IMDb KG</td>
<td>RNN (LSTM)</td>
<td>WHUE</td>
<td>Movies &amp; Music</td>
</tr>
<tr>
<td></td>
<td>RKGE [70]</td>
<td>Freebase, MovieLens</td>
<td>RNN, &amp; BCELoss</td>
<td>IANM</td>
<td>Movies, &amp; local businesses</td>
</tr>
<tr>
<td></td>
<td>DHKGE [71]</td>
<td>MovieLens, Freebase</td>
<td>CNN, LSTM, &amp; (SGD) BCELoss</td>
<td>IANM</td>
<td>Movies, &amp; local businesses</td>
</tr>
</tbody>
</table>
Table 3. Cont.

<table>
<thead>
<tr>
<th>Method</th>
<th>Model</th>
<th>KG/KGE Used</th>
<th>DL Used</th>
<th>Mode of Explainability</th>
<th>RS Scenario</th>
</tr>
</thead>
<tbody>
<tr>
<td>HAGERec [72]</td>
<td>MovieLens, Book-Crossing, &amp; Last.fm</td>
<td>GCN &amp; BPR</td>
<td>IANM</td>
<td>Movies, Music &amp; Books</td>
<td></td>
</tr>
<tr>
<td>KGAT [73]</td>
<td>Last.FM, Yelp dataset &amp; TransR</td>
<td>GCN, &amp; BPR</td>
<td>IANM</td>
<td>Books, Music, restaurants &amp; bars</td>
<td></td>
</tr>
<tr>
<td>KGAT+ [74]</td>
<td>Freebase KG &amp; TransR</td>
<td>ReLU &amp; BPR</td>
<td>IANM</td>
<td>Real world services</td>
<td></td>
</tr>
<tr>
<td>AKUPM [76]</td>
<td>Microsoft KG &amp; TransR</td>
<td>SoftMax</td>
<td>IANM</td>
<td>Movies &amp; Books</td>
<td></td>
</tr>
<tr>
<td>AKGE [77]</td>
<td>Freebase, TransR</td>
<td>RNN</td>
<td>IANM</td>
<td>Music</td>
<td></td>
</tr>
</tbody>
</table>

3.1. The Two-Stage DKG-Based Learning Methods

In the two-stage explainable learning method, the KG or KGE modules and the recommendation module are trained separately. In the first instance, KG or KGE algorithms are integrated with DL methods and are used to learn how the entities and relations are represented. Then, along with other attributes, the information from the trained graphs is fed into the prediction or recommendation module for accurate predictions and explanations. Table 4 provides a summary of the collected papers under this method.

As mentioned earlier, accurate predictions and explainability are the two main factors to consider when evaluating a recommendation model and have become one of the basic trade-offs in XAI. In [61] Gao et al. proposed a Deep Explicit Attentive Multi-View Learning Model (DEAML) to mitigate the trade-off between accuracy and explainability by developing a deep explainable model for item recommendation (Toys and Games, and Digital Music). First, the authors leverage the Microsoft Concept KG to build a tree-like explicit feature hierarchy \( Y \), where each node \( F_l \) in \( Y \) represent an explicit feature of an item. The model consists of two steps: the first step involves item prediction with attentive multi-view learning, while the second step generates personalized explanations. In the first instance, the embeddings of user \( i \) and item \( j \) are generated and used as inputs to the DEAML model. Following the method proposed in [62], the user-item feature embeddings are then trained to capture both semantic and hierarchical information to represent multiple user views. Given the user \( i \) with embedding \( u_h^i \) for view \( h \), and item \( v \) with embedding \( v_h^j \) at level \( h \), the authors proposed the rating prediction \( \hat{r}_{ij} \) by adding rating bias \( \mu \), user bias \( o_i \) and item bias \( o_j \) to the rating formula of [78] to give

\[
\hat{r}_{ij}^h = u_h^i v_h^j + o_i + o_j + \mu
\]

Multiple views are then combined using the weighted sum of predictions \( \hat{r}_{ij}^h \) in each view as

\[
\hat{r}_{ij} = \sum_{h=1}^{H} w_h \hat{r}_{ij}^h
\]

where \( w_h \) is the weight of view \( h \). Finally, the Adam optimizer is employed to automatically adjust the learning rate by optimizing the following objective function:

\[
\mathcal{L} = \sum_{i,j \in \Omega} (\hat{r}_{ij} - r_{ij})^2 + \sum_{h=1}^{H} \mathcal{L}_h + \lambda_\nu \mathcal{L}_\nu + \lambda_r \|\Theta\|^2,
\]

where \( \mathcal{L}_h \) is the loss in view \( h \), \( \mathcal{L}_\nu \) is the co-regularization loss, \( \lambda_\nu, \lambda_r \) are weights parameters, and \( \|\Theta\|^2 \) is the L2 norm of all parameters in the model. The explanations presented to the
user are in the form: “You might be interested in (Features of $E$) on which the item performs well”.

Table 4. A summary of the Two-stage DKG-based learning methods.

<table>
<thead>
<tr>
<th>Model</th>
<th>Key Contribution/Drawbacks</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEAML [61]</td>
<td>Develops a deep explainable model for item recommendation to mitigate the trade-off between accuracy and explainability.</td>
</tr>
<tr>
<td>DKN [63]</td>
<td>Uses an attention network mechanism to develop user’s diverse interests, which performs similarity matching between the user’s clicked news history and potential candidate news. The performance and accuracy of prediction increase if there is enough clicked news history. The main drawback is that, in most cases, the user may not have clicked news history, or if available, not enough for the attention network to provide dynamic interest for the user. In addition, the model is restrictive as it only applies to news recommendations.</td>
</tr>
<tr>
<td>KSR [64]</td>
<td>Integrates RNN-based algorithm with TransE embeddings to provide a post-hoc explanation for selected recommendations, by conducting empirical similarity matching between user-item embeddings. Nonetheless, an intelligent RS should have the capacity to carry out distinct and exact reasoning over KGs for decision-making instead of merely embedding the graph as latent vectors for similarity matching.</td>
</tr>
<tr>
<td>JKP [65]</td>
<td>Uses MLP for both representation of embeddings and item recommendation. JKP outperforms DKN [63] in terms of AUC. However, the recommendations may not be trustworthy. This is because both entity representation and item recommendation are jointly trained together with the same MLP algorithm, and this “MLP joint-training” process can negatively influence the prediction of the user’s preference of candidate items.</td>
</tr>
</tbody>
</table>

As stated in [79], typical explainable RSs include one or two layers of attention networks. Being aware of this, a novel multi-channel and word-entity-aligned algorithm, known as Deep Knowledge-aware Network (DKN), was proposed in [63]. DKN combines word semantics, TransD knowledge embedding method, and knowledge level information from Microsoft Satori KG to learn the representation of entities. Instead of concatenating the word and entity embeddings of news titles as proposed in [80], two additional Knowledge-aware CNN (KCNN) embeddings, namely “transformed entity embeddings” and “transformed context embeddings”, are used together with the word embeddings. Here, the transformed embeddings aid in alleviating the heterogeneity between word and entity spaces by self-learning. The “transformed context embeddings” aid in identifying the position of each entity in the KG. Similar to Kim CNN [81], the three embedding matrices are aligned and stacked together as a multi-channel. Multi-filters are then applied to the multi-channel input to extract specific local features, which are then concatenated together and used as the final representation of the input news title. To characterize the dynamic interests of the user, a DL attention-based network is used to model the user’s clicked news title on the candidate news. Mathematically, for a user $u_i$ with clicked news-history $\{h_{i1}, h_{i2}, \ldots, h_{IN_i}\}$, the attention-based mechanism is used to generate its embedding $U_i$ with respect to the candidate news $c_j$ as

$$U_i = \sum_{l=1}^{N_i} s_{c_u h_l} E(h_l),$$

where $E(h_1), E(h_2), \ldots, E(h_{N_i})$ is the embeddings of user $u_i$’s clicked history, and $s_{c_u h_l}$ is the softmax function which measures the degree of similarities between the candidate news $c_j$ and the clicked news history $h_l$. Finally, based on the calculated $U_i$, another DL algorithm is employed for the prediction, $p_{i,c_j}$ of user $u_i$’s preference. Mathematically, for a given DL algorithm denoted as $Dnn$, the authors evaluate the prediction as

$$p_{i,c_j} = Dnn(u_i, E(c_j)).$$
where $E(c_j)$ is the candidate news embeddings. In summary, the superiority of DKN lies in its two main properties: (1) the use of the word-entity-aligned KCNN for sentence representation learning, which greatly preserves the association between words and entities; (2) and the use of an attention-based network to treat users’ click history discriminatively, to better capture users’ diverse reading interests.

In [64], Huang et al. proposed the Knowledge-enhance Sequential Recommender (KSR) model for a sequential recommendation. KSR integrates Recurrent Neural Networks (RNN)-based algorithm structured as Gated Recurrent Unit (GRU) with TransE embedding and information from the Freebase KG for item recommendation and interpretability. The model starts with a GRU-based sequential recommender, which is then augmented with a Key-Value Memory Networks (KV-MN), using entity attribute information from the Freebase KG. The KV-MN is an external memory consisting of a larger array of slots explicitly storing and memorizing information from the Freebase KG. The KV-MN splits the array slots into a key vector and a value vector, which are then stacked together in a memory slot to capture fine-grained user preference to improve interpretability. Here, the TransE embedding is used to learn both the embeddings for the entities $E$ and relations $R$ from the Freebase KG, of which the relation embedding is taken as the attribute key vector in the KV-MN model. In mathematical detail, given the interaction sequence $\{i_1, i_2, \ldots, i_t\}$ of a user $u$ at time $t$, the GRU first computes the current hidden state vector $h^{t}_{u}$ (also called the sequential preference representation of user $u$), conditioned on the previous hidden state $h^{t-1}_{u}$ as

$$h^{t}_{u} = \text{GRU}(h^{t-1}_{u}, q_{i_t}).$$

Using $\tilde{h}^{t}_{u}$ as the query to read the KV-MN, an attribute-based preference representation vector $m^{t}_{u}$ is then designed with attention mechanism on vectors of item attributes. A vector concatenation is then used to combine the representation of both user preference $p^{t}_{u}$, and item embedding $\tilde{q}_{i}$ into single vectors $p^{t}_{u} = h^{t}_{u} \oplus m^{t}_{u}$, and $\tilde{q}_{i} = q_{i} \oplus E_{i}$, respectively. After transforming $p^{t}_{u}$ and $\tilde{q}_{i}$ to the same dimension, the user’s preference for items is computed as a ranking score given as

$$S_{u,j,t} = \text{MLP}(p^{t}_{u}) \cdot \text{MLP}(\tilde{q}_{i}).$$

The KSR recommendation model is also highly interpretable by checking the user’s attention weight over explicit attributes. For instance, the “actor’s” attribute dominating the attention weight for a recommended movie indicates the recommendation is produced based on that feature. Thus, such feature-level attention weight reflects the user’s explicit preference. In [65], Wang et al. proposed the joint KG and user preference model (JKP) for an explainable recommendation. The authors employ MLP for both the representation of embeddings and item recommendation. According to the data presented by the authors, JKP outperforms DKN in terms of the AUC. Although this claim may be valid, the recommendations may not be trustworthy. This is because both entity representation and item recommendation are jointly trained together with the same MLP algorithm, and this “MLP joint-training” process can negatively influence the prediction of the user’s preference of candidate items.

Overall, the two-stage DKG learning methods are straightforward to execute as the KG embeddings can quickly be learned without necessarily interacting with the data. The KG embeddings are usually treated as extra attributes that can be used for the subsequent recommendation module. Thus, large-scale interactive datasets can be learned separately for the recommendation, thereby reducing the computational cost. In addition, it becomes
unnecessary to repeat the learning process or update embeddings once they are learned. In all, it is easy to implement this method, and highly scalable. However, this method suffers from improper embeddings, and is more applicable to in-graph applications such as link predictions than recommendation tasks and mostly lack an end-to-end manner.

3.2. The Joint-Stage DKG-Based Learning Methods

In this trend, information from multiple sources, including side information, can be learned and aggregated to produce the final recommendation. Thus, this method leverages multi-modal information, including images, text, and ratings, for top N recommendations. We provide a summary of reviewed publications under this method in Table 5.

In [66], Ai et al. designed an explainable RS (ECFKG) by integrating the traditional CF framework with the learning of a KG for amazon product recommendations. First, the model is embedded with an automatically-built graph containing entities and user behaviors and a set of minimal features (e.g., produced_by, Bought_together, also_viewed). Then the design philosophy of CF and the TransE KGE is employed to learn over the KG for a personalized recommendation. Based on a fuzzy soft matching algorithm (Fuzzy-SMA), the authors then conduct fuzzy reasoning over the paths in the product KG to generate personalized explanations. In all, two main objectives are identified: for each user, \( u \), a set of items \( i \) that user \( u \) is most likely to purchase is identified; and for each retrieved user-item pair, a natural language sentence is developed to explain why the user should buy the item. In mathematical detail, the product KG is first constructed as a set of triples \( S = \{(e_h, e_t, r)\} \), where \( e_h \) and \( e_t \) are head and tail entities respectively, and \( r \) is the relationship between \( e_h \) and \( e_t \). Each entity is then projected to a low-dimensional latent space, and each relation is treated as a translational function for entity conversion. Latent vectors \( \mathcal{E}_h \) and \( \mathcal{E}_t \) are constructed for the head and tail entities, respectively, and the relation vector \( \mathcal{R} \) is modeled as a linear projection of the form \( \mathcal{E}_t = \text{trans}(e_t, r) = \mathcal{E}_h + \mathcal{R} \). The entity embeddings are then learned by joining entities in the product KG with the translational function in the latent space. Meanwhile, the translation model \( \text{trans}(e_u, r) \) is learned by optimizing the generative probability of \( e_t \), using the negative sampling model proposed in [83], and the log-likelihood approximation of \( \log P(e_u, c_t, r) \). The second stage is to conduct soft matching between the tail entities and the translation model to explain recommended items. This is done by constructing an explanation path between the user and the item in the latent space. The algorithm first conducts breadth-first search (BFS) of maximum depth \( z \) from the user \( e_u \) and item \( e_t \) entities to search for an explanation path that can potentially exist between them. Thus, given an intermediary entity, \( e_s \), the paths between \( e_u \) and \( e_t \) are then memorized, and the path probability is computed using the soft matching equation as

\[
P(e_s|e_u, R_{a}, R_{b}, R_{\beta}) = \frac{P(e_s|\text{trans}(e_u, R_{a}))P(e_s|\text{trans}(e_t, R_{\beta}))}{P(e_s|\text{trans}(e_u, R_{a}))}
\]

where

\[
P(e_s|\text{trans}(e_u, R_{a})) = \frac{\exp(\mathcal{E}_s \cdot \text{trans}(e_u, R_{a}))}{\sum_{e_s \in E^m_{r_{m}}} \exp(\mathcal{E}_s \cdot \text{trans}(e_u, R_{a}))},
\]

\( E^m_{r_{m}} \) denotes the set of tail entities for \( r_{m} \) relations, \( R_{a} \) and \( R_{\beta} \) are two sets of relations for the user \( e_u \) and item \( e_t \) respectively, \( \mathcal{E} \) denotes the set of entity vector, and \( a, \beta, m \) are integers. The best explanation for \( (e_u, e_t) \) can then be obtained by ranking all the path probabilities, and the best one is selected for natural language explanation.

In [58], the authors exploit the potentials of Semantic-Aware Autoencoder (SemAuto) [39] to develop content-based explainable (ESEmAuto) recommendations for movies. In their algorithm, the structure of DBpedia KG is first combined with an Autoencoder Neural Networks (Auto-NN), whose structure is constructed to mimic the existing connections in the DBpedia KG. User ratings are then fed into the Auto-NN to extract the weights associated with the hidden neurons and then employed to compute recommendations. Mathematically, for those neurons that tend to be not connected in the KG, their autoencoders are trained via
feedforward and backpropagation using a masking multiplier matrix $M$, where the rows and columns represent items and features, respectively. Hidden layers ($h$) and output layers ($o$) are computed as

$$h = g(X \times (W_1 \circ M)),$$

$$o = g(h \times (W_2 \circ M^T)).$$

A modified stochastic gradient descent (SGD) is then used to learn the hidden neuron weights $W_1$ and $W_2$ as

$$W_1 = (W_1 \circ M) - r \cdot \frac{\partial E}{\partial W_1},$$

$$W_2 = (W_2 \circ M^T) - r \cdot \frac{\partial E}{\partial W_1},$$

with respect to the mean square error $E$. As each hidden neuron constitutes an entity in the DBpedia KG, the pre-trained weights $W_1, W_2$ are used as indicators, representing the significance of the corresponding entity for the user. After training the autoencoder for each user, the semantic autoencoder algorithm proposed initially in [59] is adopted to provide explanations for the top $N$ movie recommendations. To describe movies, the authors used a set of predicates of the form (dct:subject, dct:starring, dct:director, dct:writer), then relying on the weights $W_1, W_2$ associated with features in the user’s profile, a human-understandable explanation is constructed. The explanations are presented in three forms, popularity-based: (“We suggest these items since they are very popular among people who like the same movies as you”), point-wise personalization: (“We guess you would like to watch X and Y since they are about P”), pair-wise personalization: (“We guess you would like to watch X more than Y because you may prefer R”).

Contrary to the two-stage methods, the joint-stage DKG-based methods can be trained in an end-to-end manner. In this way, the module used for the recommendation can also be used to guide the attribute learning process of the graph embedding module. However, these methods require efforts to aggregate different models under one framework. For instance, weights from different objective functions from the KG are often combined and fine-tuned, and used to regularize the RS, which can introduce bias due to regularization.

**Table 5.** A summary of the Joint-stage DKG-based learning methods.

<table>
<thead>
<tr>
<th>Model</th>
<th>Key Contribution/Drawbacks</th>
</tr>
</thead>
<tbody>
<tr>
<td>ECFKG [66]</td>
<td>Uses the design philosophy of CF and TransE to learn over the KG for a personalized recommendation. The algorithm conducts BFS of maximum depth from the user-item entities to search for an explanation path that can potentially exist between users and items. The main drawback is that sampling paths via BFS is very inefficient and may miss meaningful paths.</td>
</tr>
<tr>
<td>SemAuto [58]</td>
<td>Feeds user ratings into an auto-encoder neural network to extract the weights associated with the hidden neurons, which are then employed to compute recommendations.</td>
</tr>
</tbody>
</table>

### 3.3. The Path-Embedding DKG-Based Learning Methods

In this method, explicit embeddings of various patterns of user-item or item-item connections in the KG are explored to provide additional guidance for the recommendation. A summary of the reviewed publications under this method is given in Table 6.

In [67], the authors integrated two rule-based modules (induction rules and recommendation rules modules) to generate an effective and explainable recommendations model (RuleRec). The induction rules, mined from the Freebase KG, utilize multi-hop relational patterns to infer associations among different items for model predictions and explanations. Mathematically, given a rule $R = r_1, \ldots, r_k$ between item pairs $(a, b)$, the authors defined a probabilistic term $P$ to find specific paths between $a$ and $b$ as

$$P(b|a, R) = \sum_{e \in N(a, R')} P(e|a, R') \cdot P(b|e, r_k),$$
\[ x_{(a,b)} = [P(b|a, R_1), \ldots, P(b|a, R_n)]^T, \]

where \( R' = r_1, \ldots, r_{k-1} \) and \( P(b|e, r_k) \) is the probability of reaching node \( b \) from \( e \) with a one-hop random walk, given the relation \( r_k \), and \( N(a, R') \) is a reachable node set with \( R' \) from node \( a \). Given the \( i \)-th entry \( x^{(i)}_{(a,b)} \) of \( x_{(a,b)} \) and a target vector \( y_{a,b|A} \) of a stochastic variable \( A \), a Chi-square objective function \( O_l \) is employed to choose the top \( n \) useful rules from the derived rules, given by

\[ O_l = \sum_{a \in \text{pair} \in A} \sum_{i=0}^{n} \frac{\left| x^{(i)}_{(a,b)} \right|}{w_i} \cdot \left( x^{(i)}_{(a,b)} + b - y_{a,b|A} \right)^2, \]

where \( w_i \) is the \( i \)-th weight of the weight vector \( w \), representing the importance of each rule. After generating the correct paths, the induced rules are augmented with the recommendation module to provide better generalization capacity. In detail, each user/item is represented with a vector of latent features by separately using a modified BPR Matrix Factorization (BPRMF) model [82], and a refined version of the Neural Collaborative Filtering (NCF) [84] algorithm for item prediction. Here, the goal is to predict an item list from a set of items \( I_u \) for user \( u \) based on a preference score \( S \) given by

\[ S'_{u,t} = f_w(S_{u,t}, F_{i,I_u,R}) = S_{u,t} + \alpha \cdot w^T F_{i,I_u,R}, \]

where \( S_{u,t} \) can be replaced by the BPRMF or the NCF predictive functions, thus obtaining two separate predictive models for performance verification. \( f_w \) is a parameterized function of weights \( w \), and \( F_{i,I_u,R} \) is an indicator function of at least a fraction \( \alpha \) of the training item pairs. The objective function for the recommendation module is then defined as

\[ O_r = \sum_{u \in U} \sum_{p \in I_{u} \setminus I_{u}} (S'_{u,p} - S'_{u,n}) \]

Finally, a joint optimization framework that combines the rule-based learning module and the recommendation module is then employed by the authors to conduct joint learning, as \( O = O_r + O_l \). The authors then show how the proposed model provides explainability to the recommended item.

In [68], Weiping et al. proposed a knowledge-aware model (EKAR) for a path-based explainable recommendation for movies and books. The authors utilized deep reinforcement learning and Markov Decision Process (MDP) on the user-item-entity graph to generate explainable paths. Treating the user-item-entity graph as a state \( s_t \) of a sequence of visited nodes and edges, a policy network consisting of two fully connected layers is developed to produce probability distribution over possible action space \( A_t \). A success reward is then defined to depict an agent’s success in finding those items consumed by the target user \( u \) in history. The rewards are finally augmented with a pre-trained KGE to stabilize training and explore diverse recommendation paths. In mathematical detail, choosing each action \( a = (r, e) \) as the concatenation of relation and entity embeddings, the state \( s_t = (r_0, e_0, \ldots, r_t, e_t) \) is first encoded. Then based on the parameterized state \( S_t \) and parameterized action \( a_t \), the probability distribution over possible action space \( A_t \) is computed as

\[ y_t = W_2 \text{ReLU}(W_1 S_t + b_1 + b_2), \]

\[ \pi_\theta(a'|s_t) = \frac{\exp(a'^T y_t)}{\sum_{a \in A_t} \exp(a'^T y_t)}, \]

where \( \{W_1, W_2\} \) and \( \{b_1, b_2\} \) are weights matrices and weight vectors of the neural network, \( \text{ReLU}(m) = \max(0, m) \) is a nonlinear activation function, and \( \pi_\theta(a'|s_t) \) is the probability of action \( a' \) given state \( s_t \). To accelerate the training process, the success reward \( R_f \) [85] is generated as follows to encourage the agent to explore items that have not been purchased:
RuleRec [67] Integrates two rule-based modules (induction rules and recommendation rules modules) to generate an effective and explainable recommendations. The induction rules, mined from the Freebase KG, utilize multi-hop relational patterns to infer associations among different items for model predictions and explanations. After generating the correct paths, the induced rules are augmented with the recommendation module to provide better generalization capacity.

EKAR [68] Uses deep reinforcement learning and Markov Decision Process (MDP) on a user-item-entity graph to generate explainable paths.

PGPR [26] Combines MDP and Reinforcement Learning with a multi-hop path-searching agent that learns to navigate from a user to potentially “good” items conditioned on the starting user in the KG. The main drawback of the PGPR model is that it only provides transparency through the path reasoning process over the KG, and does not provide direct and clear user-item explainable recommendations. Moreover, the model only performs better when paths histories are available. As detailed in the paper, a state representation with no path history is incapable of providing adequate information for the reinforcement learning agent to learn a good policy. Again, as stated by the authors, directly applying TransE [17] as a single-hop latent matching method for recommendation outperforms the PGPR model.

EIUM [27] Develops a multi-modal fusion mechanism, wherein textual, visual, and structural features of items are extracted and incorporated into a network of multi-modal fusion constraints for user-item preference representation. A weighted pooling layer is utilized to learn and integrate the different contributions of each path between user-item pairs. The main drawback is that users’ historical records are needed for prediction and recommendation. Hence, the possibility of the cold-start recommendation problem is inevitable, especially for fresh users with no historical records.

KPRN [69] Uses RNN to extract qualified paths between user-item pairs, and adopts an LSTM network to develop sequential dependencies among entities and relations. However, extracting qualified paths using RNN is labor-intensive. Graph Neural Networks can be employed to mimic the propagation of user preference within the KG. Also, the KPRN model suffers from the cold start problems since the KG links multiple domains in combination with overlapped entities. This issue can be resolved by employing the “zero-shot learning” algorithm to resolve the cold start problems in the target domain.

RKGE [70] Uses RNN to learn the semantic representation of entities and paths for user-item preference. RKGE can automatically mine the connection patterns between entities in the KG to improve recommendation performance. Nonetheless, only one type of neural network is used by RKGE to encode the path embeddings, which cannot exhaustively extract path features, inhibiting the performance improvement of the recommendation.

DHKGE [71] DHKGE is an improvement of RKGE model [70]. DHKGE encodes path embeddings between users and items by integrating CNN and the LSTM network, and outperforms the RKGE in terms of accuracy.

In [26], Xian et al. proposed the policy-Guided Path Reasoning (PGPR) model for recommendation and interpretability through actual paths in the Freebase KG. PGPR combines Markov Decision Process and Reinforcement Learning with a multi-hop path-searching agent that learns (through a policy-guided algorithm) to navigate from a user to potentially “good” items conditioned on the starting user in the KG. In detail, given a state \( s_t = \text{tuple}(u, e_t, h_t) \), where \( u \in U \) is the starting user entity, and \( e_t \) is the agent entity of history \( h_t \) prior to step \( t \), an action space \( A_t = \{(r, e)| (e_t, r, e) \in \mathcal{G}_R, e \notin \{e_0, \ldots, e_{t-1}\} \} \) of the KG \( \mathcal{G}_R \) is first constructed. Based on \( A_t \) and a scoring function \( f((r, e)|u) \) which maps any
edge \((r, e)\) \((\forall r \in R, \forall e \in E)\) to a real valued score, a user-conditional pruned action space \(A_t(u)\) is created as \(A_t(u) = \{(r, e) | \text{rank}(f((r, e) | u)) \leq a, (e, r) \in A_t\}\), to effectively maintain potentially “good” paths conditioned on the starting user \(u\), where \(a\) is a known upper bound on the size of \(A_t\). A soft probabilistic reward \(R_T\) is then constructed to encourage the agent to explore as many “good” paths, based on the terminal state \(s_T = (u, e_T, h_T)\) as

\[
R_T = \begin{cases} 
\max \left(0, \frac{f(u, e_T)}{\max_{e \in \mathcal{E}} f(u, e)} \right), & \text{if } e_T \in \mathcal{I}, \\
0, & \text{otherwise},
\end{cases}
\]

Based on the above Markov Decision Process, optimal paths are obtained by learning a stochastic policy to maximize the expected cumulative reward for any starting user \(u\), as

\[
J(\cdot) = \mathbb{E}_\pi \left( \sum_{t=0}^{T-1} \gamma^t R_{t+1} | s_0 = (u, u, \emptyset) \right),
\]

where \(\gamma\) is the discount factor. The authors solve the optimization problem through reinforcement learning by designing a policy network \(\pi(\cdot | s, A_u)\), and a value network \(\hat{V}(s)\) that share the same feature layers. A “beam search” algorithm guided by \(T\)-hop generative probabilities \(Q_T\), and path rewards \(R_T\) is finally employed to explore candidate paths and recommended items for each user. Interpretations of recommended items are based on the highest generative probabilities.

In [27], Huang et al. proposed an Explainable Interaction-driven User Modeling (EIUM) algorithm for movie recommendations by extracting and encoding semantic paths between user-item pairs through the MovieLens (IMDB) dataset and the Freebase KG. To learn the KG, the authors developed a multi-modal fusion mechanism, wherein textual, visual, and structural features of items are extracted and incorporated into a network of multi-modal fusion constraints for user-item preference representation. A weighted pooling layer (WPL) is utilized to learn and integrate the different contributions of each path between user-item pairs. A sequence model is then constructed to learn the semantic representation of the paths between user-item pairs. The sequence model consists of a self-attention mechanism with a position encoding module capable of capturing long-distance dependencies of the sequence with various lengths. The two learning modules, i.e., the multi-modal fusion and the sequential recommendation modules, are then combined for joint learning. In mathematical detail, for the multi-modal fusion module, the textual item\(_{textual}\) and visual item\(_{visual}\) features are extracted using fastText and AlexNet, respectively, and represented as

\[
f_c = \sigma(W_f [item_{textual}, item_{visual}] + b_f),
\]

where \([,]\) denotes concatenation, and \(W_f, b_f\) are learning parameters, with \(\sigma\) being a non-linear activation function. The structural features \(f_e\) of entities and relation is learned by applying certain structural constraints on the TransE KGE [17]. The constraints are given in the form: \(c2c : h_f + r = f_e, c2s : h_f + r = f_e, s2s : h_f + r = f_e,\) and \(s2c : h_f + r = f_e\). For the sequence module, given a user’s interaction sequence \(\{i_t, t = 1, 2, \ldots, T\}\), the authors represent the sequential interaction of entity \(e_{t_i}\) as \(B = \{\tau_t, t = 1, 2, \ldots, T\}\), and the user preference interaction as

\[
B_{emb} = \sum_{t=1}^{T} w_t \cdot e_{t_i},
\]

where \(w_t\) represents the attention score of interactions, \(\tau = (u, interact, i)\), interact denotes a predefined relation between user \(u\) and item \(i\), and the output embedding \(B_{emb}\) represents the user dynamic preference. Finally, the joint learning is obtained by minimizing the combined objective functions of the two modules as \(L = L_{rec} + \lambda L_{kg}\) where \(\lambda\) is the
regularization parameter, and \( \mathcal{L}_{\text{rec}} \) is the loss function of the recommendation module, which follows through cross-entropy function of a distance function \( D(\cdot) \), given by:

\[
\mathcal{L}_{\text{rec}} = \sum_u \sum_v \sum_{v' \in \mathcal{V}} - \log \sigma [D(u) - D(v')],
\]

with \( v \) being the ground-truth item which is higher than all other items \( v' \{v' \in \mathcal{V} = \mathcal{I}, v \notin \mathcal{V}\} \), and \( \mathcal{L}_{\text{kg}} \) is the loss function of the KG, which is formed by the combination of the loss functions of the multi-modal fusion constraints, given by

\[
\mathcal{L}_{\text{kg}} = \mathcal{L}_{\text{kg}} + \mathcal{L}_{\text{s2s}} + \mathcal{L}_{\text{c2s}} + \mathcal{L}_{\text{c2c}}
\]

\[
\mathcal{L}_{\text{kg}} = \frac{1}{4} \sum_i \|h + r - t\|, \quad i \in \{c2c, s2s, c2s, s2c\}
\]

EIUM predicts the user’s item preference with diversified semantic paths to offer path-level explanations. Thus, explanations are based on user-item interactive paths carrying different semantic information. For instance “The movie m5 is recommended since it’s the sequel to movie m3 you have watched.”

In [69], Wang et al. proposed the Knowledge-aware Path Recurrent Network (KPRN) to exploit user-item interaction for explainable movie and music recommendations by employing the MovieLens, KKBox datasets, and the IMDb KG. Specifically, qualified paths between user-item pairs are extracted over the KGs, and then a Long Short-Term Memory (LSTM) network is adopted to develop the sequential dependencies among entities and relations. After that, a pooling operation, in the form of an attention mechanism, is employed to aggregate path representations to capture prediction signals for user-item pairs. Based on the attention mechanism, path-wise explanations to recommended items are then generated by the authors. In mathematical details, given a user-item pair \((u, i)\), and a set of all possible paths \( \mathcal{P}(u, i) = \{p_1, p_2, \ldots, p_K\} \) connecting user \( u \) and item \( i \), the KPRN model takes the paths \( \mathcal{P}(u, i) \) as its input, and outputs a score \( \hat{g}_{ui} \) signifying the degree of interaction between \( u \) and \( i \), where

\[
\hat{g}_{ui} = f_\Theta(u, i|\mathcal{P}(u, i)),
\]

and \( f_\Theta \) represents the KPRN model of parameters \( \Theta \). To generate the KPRN model, the authors employ a three-step strategy. In the first instance, the sequential embeddings of entities \( e_i \) and relation \( r_j \) are generated along paths \( p_k \) and concatenated as \([e_1, r_1, e_2, r_2, \ldots, e_l, r_l] \), where each element represents an entity or relation. An RNN in the form of LSTM is then employed to explore the sequential embeddings and generate a unified representation for encoding the holistic semantics. In the last stage, a weighted pooling operation is utilized to combine multiple paths to generate the final score. Here, the final prediction score \( \bar{g}_{ui} \) is given by

\[
\bar{g}_{ui} = \sigma(g(S)),
\]

where \( S = \{s_1, s_2, \ldots, s_K\} \) is the prediction score for \( K \) paths, and

\[
g(S) = \log \left[ \sum_{k=1}^{K} \exp \left( \frac{s_k}{\gamma} \right) \right],
\]

with \( \gamma \) being a hyperparameter for controlling exponential weights. Treating the recommendation as a binary classification problem (i.e., user-item interaction is assigned 1, otherwise 0), a negative log-likelihood loss function is employed to learn the model, with the objective function given by

\[
\mathcal{L} = - \sum_{(u, i) \in O^+} \log \bar{g}_{ui} + \sum_{(u, i) \in O^-} \log(1 - \bar{g}_{ui}),
\]

where \( O^+ = \{(u, i)|\bar{y}_{ui} = 1\}, \) and \( O^- = \{(u, i)|\bar{y}_{ui} = 0\} \) are respectively, the positive and negative interaction pairs. Path-wise explanations are given in the form
"Shakespeare in Love is recommended since you have watched Rush Hour acted by the same actor Tom Wilkinson".

In [70], Sun et al. proposed the Recurrent Knowledge Graph Embedding (RKGE) model to learn the semantic representation of entities and paths for user-item preference. In brief, RKGE uses an RNN architecture to model the semantics of paths that link the same entity pair in the KG for the user-item recommendation. Characterization of user preference towards an item is performed via a pooling operator to discriminate the saliency of different paths, and an SGD via a binary cross-entropy loss (BCELoss) is employed in the model optimization. The main advantage of RKGE is that it can automatically mine the connection patterns between entities in the KG to improve recommendation performance. Nonetheless, only one type of neural network is used by RKGE to encode the path embeddings, which cannot exhaustively extract path features, inhibiting the performance improvement of the recommendation.

To overcome the limitations of the RKGE model [70], Li et al. [71] propose the Deep Hybrid Knowledge Graph Embedding (DHKGE) model for the top-N recommendation. DHKGE encodes path embeddings between users and items used in the recommender system by integrating CNN and the LSTM network. DHKGE then utilizes an attention mechanism to distinguish the significance of multiple semantic paths between user-item pairs. The attention mechanism is mainly employed to aggregate the encoded path representations to create a hidden state vector, and a proximity score is used to compute the "closeness" between the target user and candidate items to generate the top-N recommendation. The DHKGE outperforms the RKGE in terms of accuracy.

Overall, the path-embedding methods explicitly learn the embeddings of connection patterns. The methods encode the patterns of user-item pair into latent vectors, thus making it possible to consider the effect of mutual relationships of the target user, and provide richer semantics. Most of the models under the path-embedding methods can capture the connection patterns by automatically enumerating the paths. However, for large dataset, it is impossible to mine all connection paths one-by-one if the relations are complex, without the help of pre-defined meta-structures.

3.4. The Propagation DKG-Based Learning Methods

Propagation-based methods exploit information in a KG by combining the representation of entities and relations with embeddings and higher-order connection patterns for personalized explainable recommendations. The main idea is based on embedding propagation, where the basic implementation is the graph neural networks technique. The methods fine-tune entity representations by combining the embeddings of multi-hop neighbors in the KG. This allows the user’s preferences to be predicted with the rich representations of the user and the items. We present the most recently published papers among this method, with a summary presented in Table 7.

Recently, graph convolutional networks (GCN) have been deemed a state-of-the-art method for graph representation learning. In [72], Yang et al. employed GCN to develop a four-step hierarchical explainable attention GCN model called HAGERec, to explore users’ preferences from higher-order structural connections of a heterogenous KG. HAGERec simultaneously learns user and item representations via a bi-directional information propagation strategy to exploit semantic information. In detail, multi-directional connections among users $u$ and their entities $e_u$, and items $v$ and item entities $e_v$, are first formulated using a multi-hop algorithm and a “flatten operation” to compress higher-order relations and embed the entities and relations of the KG into a vector space. User and item representations are then aggregated into an embedding vector in the second step, using the following embedding formulas:

$$V_{GCN} = \text{aggregate}(N_{e_u}) = l((W_1(e_u + g(N_{e_u}))) + b_1))$$

$$g(N_{e_u}) = \sum_{e_{n_0} \in N_{e_u}, r_{K^2} \in R^2} (\text{softmax}(e_{n_0} \cdot r_{K^2} \cdot a_{e_u,e_{n_0}}))$$
\[ a_v = h^T I(W_2([e_{uv}, e_{vu}]) + b_2), \quad e_{uv, vu} = \text{softmax}(a_v) \]

and

\[ U_{GCN} = \text{aggregate}(N'_{e_u}) = \text{softmax}(V_{GCN} \cdot r_1 \cdot a_{e_u, vu}), \]

where \( N_{e_u} \) and \( N'_{e_u} \) denote the sample neighbor entities for \( e_{uv} \) and \( e_{vu} \), respectively. \( W_1, W_2, W_3, \) and \( b_1, b_2, b_3 \) are weights and biases, respectively, whereas \( h \) is the attention parameter that is learned during model training. \([e_{uv}, e_{vu}] \) is the concatenation vector operation, and \( e_{uv}, e_{vu} \) and \( r_1 \) are vectors processed by an MLP layer. The third step employs the sample neighbor entities to construct interaction signals to preserve the structure of entities and their neighbor networks to present a more complete representations for users and items. The predicted probability of a user \( u \) engaging an item \( v \) is finally constructed in the fourth step as \( \hat{y}_{uv} = p(u, v) \), and the model objective function is constructed by optimizing the BPR loss function [82] to learn the parameters of the HAGERec model as

\[ \mathcal{L} = \sum_{(u,v) \in I^+, (u,j) \in I^-} - \ln \sigma(\hat{y}_{uv} - \hat{y}_{uj}) + \lambda \| \theta \|_2^2 \]

where \((u,v)\) and \((u,j)\) are positive and negative interactions, respectively, and \( \theta \) are parameters estimated by minimizing \( \mathcal{L} \), with \( \lambda \) being the regularization term. To explain user preferences, HAGERec leverages an attention-based relation propagation algorithm to build knowledge-aware connectivity that utilizes an attention score to infer the reason behind users’ preferences.

To provide a better recommendation with user-item side information, it is important to explicitly model higher-order relations in a collaborative KG (CKG). In [73], Wang et al. proposed the Knowledge Graph Attention Network (KGAT) to explicitly model higher-order user-item connectivities in a hybrid CKG for an explainable recommendation. The KGAT framework utilizes a three-step strategy to model higher-order user-item relations. In the first stage, given a triple \((h, r, t)\), the TransR embedding is employed to learn the embeddings of each entity and relation by optimizing the translation \( e'_h + e_r \approx e'_t \) through a plausibility score \( g(h, t, r) \) given as

\[ g(h, t, r) = \|W_re_h + e_r - W_re_t\|_2^2, \]

where \( e'_h, e'_t \) are the projections of the head and tail entity embeddings \( e_h, e_t \) respectively in the relation’s \( r \) space, and \( W_r \) denotes the transformation matrix of relation \( r \). A pairwise ranking loss is then used to train the TransR embeddings. The loss function is formulated as

\[ L_{KG} = \sum_{(h,r,t') \in \mathcal{T}} - \ln \sigma(g(h, r, t') - g(h, r, t)). \]

In the second stage, using a model similar to the GCN architecture, the generated embeddings of the entities are recursively propagated and updated in the CKG based on the embeddings of neighboring entities to capture high-order connectivities. A neural attention mechanism is then employed to learn the weights associated with neighboring entities during the propagation. The degree of information propagation from \( h \) to \( t \) is formulated through a \( \text{tanh}(-) \) nonlinear activation equation given by

\[ \pi(h, r, t) = (W_re_t)^T \text{tanh}(W_re_h + e_r), \]

where \( \pi(h, r, t) \) controls the decay of each propagation on the triple \((h, r, t)\). Multiple representations are obtained for user and item nodes \( \{e_{u1}, \ldots, e_{uN} \} \) and \( \{e_{i1}, \ldots, e_{iN} \} \) after \( N \)-layer-propagations, which are then concatenated to form the final user-item representa-
tions $e_u^{*}, e_i^{*}$, respectively. Finally, the user preference for an item is predicted through the matching score $\hat{y}(u, i)$ as

$$\hat{y}(u, i) = e_u^{* T} e_i^{*}$$

Justification for item recommendation can be interpreted quantitatively by observing the attention weights.

Table 7. A summary of the Propagation DKG-based learning methods.

<table>
<thead>
<tr>
<th>Model</th>
<th>Key Contribution/Drawbacks</th>
</tr>
</thead>
<tbody>
<tr>
<td>HAGERec [72]</td>
<td>Simultaneously learns user and item representations via a bi-directional information propagation strategy to exploit semantic information. However, one limitation of this method is that there is the possibility of introducing irrelevant neighbors during the “neighbor sampling” or the propagation process.</td>
</tr>
<tr>
<td>KGAT [73]</td>
<td>Utilizes a three-step strategy to explicitly model higher-order user-item relations in a hybrid CKG for an explainable recommendation. It provides high accuracy recommendations by using a self-attention mechanism and a DL architecture to learn side information of recommendable items. However, the KGAT model suffers from high computational costs resulting from the massive volume of KG side information used in the learning process.</td>
</tr>
<tr>
<td>KGAT+ [74]</td>
<td>This is an improvement of the KGAT model [73]. KGAT+ addresses the computational complexity of the conventional KGAT model and maintains high accuracy and interpretability. The drawback of KGAT+ lies in the compression of the one-to-many side information. After the compression stage, the original relationships are restored through the computations of the inner product of the attention weights. Nonetheless, the restoration process may not completely reproduce the original side information in the original data before the compression. Indeed, there is the possibility that the final computed attention weight may turn out to be a positive value for a triple that should not exist in the original data and which might be used for interpretation.</td>
</tr>
<tr>
<td>RippleNet [75]</td>
<td>RippleNet propagates the users’ preferences directly over edges in the KG via an attention mechanism and then interprets the recommendations according to attention scores. Main limitations include – high computational costs, reliance on the post-analysis of the attention scores, which may not always be trustworthy, and suffers from the cold start problem.</td>
</tr>
<tr>
<td>AKUPM [76]</td>
<td>The algorithm is designed to predict the click-through rate for a user-item pair, wherein information from the user is enhanced by entities related to the user’s clicked items. Similar to RippleNet, AKUPM suffers from cold start issues.</td>
</tr>
</tbody>
</table>

In [74], Shimizu et al. employed KGAT model [73] to develop an improved model-intrinsic (KGAT+) knowledge-based explainable recommendation framework for real-world services. KGAT+ addresses the computational complexity of the conventional KGAT model and maintains high accuracy and interpretability. To achieve this, the authors proposed a five-step framework of a learning algorithm. In the first instance, the massive volume of KG side information consisting of one-to-many relationships is compressed by employing a latent class model and class membership probability. The latent class is treated as an entity, whereas the class membership probability is the relational strength between head and tail entities. In mathematical detail, for a set of targeted relations belonging to a compressed “Collaborative Knowledge Graph (CKG)” denoted as $\mathcal{R}_{\mathit{CKG}}^\prime$, a latent class of the form $\mathcal{Z} = \{ z_k : 1 \leq k \leq K \}$ is taken between the head $h$ and tail $t$ entities, respectively. Then based on these parameters, the membership probability is computed as

$$P(h,t) = \sum_{k=1}^{K} P(z_k) P(h|z_k) P(t|z_k),$$

where $z_k$ connects $h$ and $t$, with probabilities $P(h|z_k)$ and $P(t|z_k)$, respectively. The TransR KGE method is then used in the second stage to obtain the entity representation, and the parameters are updated based on the positive and negative triplets $(h,r,t)$ and $(h,r,t')$, respectively, by minimizing the pairwise ranking loss $L_{KG}$ as

$$L_{KG} = \sum_{(h,r,t) \in T} - \ln(e_{(h,r,t)} \sigma(g(h,r,t) - g(h,r,t'))).$$
where \( \sigma(\cdot) \) is a sigmoid function, and \( \epsilon(\cdot, \cdot, \cdot) \) is the probabilistic term. A normalized nonlinear activation equation bearing the probabilistic term \( \epsilon(\cdot, \cdot, \cdot) \) multiplying the one proposed in the conventional KGAT is then employed to compute the attention weights of each triplet in the third stage. After learning the embedded representations, they are concatenated to form the final user and item representations \( e^*_u \) and \( e^*_i \), respectively, and the user preference for an item is computed as \( \hat{y}(u, i) = e^*_u e^*_i \). The BPR loss is then formulated as

\[
L_{CF} = \sum_{(u,i,j) \in O} -\ln \sigma(\hat{y}(u, i) - \hat{y}(u, j))
\]

where \( (u,i) \) and \( (u,j) \) are positive and negative sample pairs, respectively. The entire model is finally optimized using the combined loss functions as

\[
\arg\min L_{KGAT} = L_{KG} + L_{CF} + \lambda\|\Theta\|^2
\]

where \( \Theta \) is a set of parameters that are estimated by minimizing \( L_{KGAT} \), and \( \lambda \) is the regularization term. In the fifth and final stage, those relations with soft probabilities compressed by the latent class model are restored back into the original state by calculating the inner product of the normalized nonlinear activation functions as

\[
\pi(h, r, t) = \sum_{k=1}^{K} \pi_{\text{norm}}(h, r, z_k) \times \pi_{\text{norm}}(z_k, r, t)
\]

The restoration permits the model to interpret the same connection as the original data instead of an ambiguous interpretation of the connection with the latent class. In brief, each relation is assigned an attention weight indicating the significance of deciding on a user-item recommendation. Thus, by observing these attention weights, the justification for item recommendation can be interpreted quantitatively.

In [75], Wang et al. proposed RippleNet, which is an end-to-end framework for a knowledge-aware recommendation. In brief, RippleNet propagates the users’ preferences directly over edges in the KG via an attention mechanism and then interprets the recommendations according to attention scores. However, RippleNet suffers from high computational costs. The model also relies on the post-analysis of the attention scores, which may not always be trustworthy. In addition, RippleNet treats the user’s historical interests as a “seed set,” which is then iteratively extended along KG links to discover the user’s hierarchical potential interests based on the candidate item. The possibility of the cold start problem can persist, especially for fresh-start users who have no historical interests. Moreover, the model does not fully explore the semantics of the relations between entities, leading to possible information loss during message passing. For the latter drawback and the computational cost problem, HAGERec [72] resolves these issues pretty well.

In contrast to the RippleNet model, AKUPM [76] employs the TransR embedding method to assign entities in the KG with different representations under various relations. AKUPM is designed to predict the click-through rate for a user-item pair, wherein information from the user is enhanced by entities related to the user’s clicked items. Specifically, the entities to be integrated into the model are initialized as the user’s clicked items, then propagated from near to far along relations in the KG to inject rich entities. In doing so, the model arrives at rich entities that can infer the user’s potential interests. In addition, the self-attention mechanism proposed in [87] is utilized by AKUPM to assign weights for entities during the propagation process, which provides attention-based interpretability for the user’s potential interests. However, similar to the RippleNet, AKUPM does not necessarily resolve the cold start issues as it still requires the user’s historical data for the model to function.

The main aim of the propagation method is to aggregate both the embeddings of entities and that of the relations and the higher-order connection patterns for personalized
recommendations. However, these models are usually computationally costly for large graphs, and models can lead to non-convergence.

4. Summary of the Systematic Review

The anatomy of the various explainable DKG-based RS techniques identified in our survey have been summarized in Figure 4, and the reviewed publications have been classified in Table 3. The percentage of total number of publications under each category has been given in Table 8. From the findings, 22.2% applied the TSELM method for explainable recommendations. Only 11.1% of the surveyed papers applied the JSELM technique for an explainable recommendation, indicating that the JSELM technique is less explored. The PEELM and PEM, each recorded 33.3% of the total number of publications. Figure 4 highlights the publications in each method, identified during the review. We can observed growing popularity in PEELM and PEM methods in recent publications.

![Figure 4. Evolution of DKG-based methods for Explainable RS.](image)

<table>
<thead>
<tr>
<th>Method</th>
<th>Reference(s)</th>
<th>Percentage (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TSELM</td>
<td>[61,63–65]</td>
<td>22.2%</td>
</tr>
<tr>
<td>JSELM</td>
<td>[58,66]</td>
<td>11.1%</td>
</tr>
<tr>
<td>PEELM</td>
<td>[27,67–71]</td>
<td>33.3%</td>
</tr>
<tr>
<td>PEM</td>
<td>[72–77]</td>
<td>33.3%</td>
</tr>
</tbody>
</table>

As observed in Table 3, the majority (72.2%) of the reviewed publications employ attention mechanism (IANM) for interpretability, while those that employ written human-understandable (WHUE) explanations constitute only 27.8%. The most used knowledge graph and knowledge graph embeddings are Freebase KG and the TransR KGE, respectively. The datasets employed in the reviewed papers can be categorized into 5 main groups: books, movies, music, news, and products. The most used datasets come from movies, music, and books, as observed in Table 3. The DL methods used mainly constitute neural networks algorithms, which comprise of RNN and its variants, GCN, MLP, and CNN. According to the reviewed publications, GCN and RNN showed promising results in explainable recommendations. Other statistical measures such as the BPR, BCEloss, and SoftMax were also employed to make final recommendations.

Based on our systematic review process and the classification criteria, the reviewed papers between 2018 and January 2022 shows that there has been a decrement in the total number of papers that employ the DKG-based method for an explainable recommendation. As revealed in Figure 4, for TSELM, only four papers were published between the years
2018 and January 2022. For JSELM, only two papers were recorded, and six papers each were recorded for PEELM and PEM, respectively. It can be observed under the “Number of publications” that there has been a sharp decline of explainable DKG published papers since 2018. In the next section, we have provided the major challenges that confront this field of research, and possible solutions to resolve these challenges.

5. Open Challenges and Future Directions

To achieve a reliable explainable recommendation, the DKG-based algorithms must effectively capture the entity representations for better user preference for a candidate item. This section highlights the major open challenges of the DKG-based methods presented in this paper, and suggest solutions on how to overcome these challenges. We first highlight open challenges with regard to KG and KGEs for explainable recommendations in Section 5.1, and then proceed to present the challenges of the DL methods in the various DKG-based methods in Section 5.2. In each of the highlighted challenges, we provide possible solutions in Section 5.3, thus defining a clear roadmap for future research.

5.1. Challenges Related to KGs and KGEs

- The first open challenge has to do with the trade-off between computational complexity as a result of using a large volume of side information and recommendation accuracy. Even though using such a large volume of side information achieves a high recommendation accuracy and ensures reliable interpretability, training the model to learn all the available side information without deep deliberation increases computational time. Thus, it is appropriate to examine the side information carefully.
- Moreover, although embedding-based reasoning is scalable and effective as the interpretation is conducted via computation within embeddings, it has difficulty learning reliable representations for sparse entities. This is due to the fact that good embeddings heavily rely on data richness.
- In addition, the majority of the embedding models are trained based on negative sampling. In other words, the models focus on maximizing similarities of connected entities in the KG and minimize the similarities within the sampled disconnected entities. Typical examples include ComplEx [50], RESCAL [48], TransE [17], and so forth. Negative sampling helps in reducing the time complexity during model learning by taking into account only the subset of negative instances. However, the method may deliver unstable model performance during model training due to the variability of the “sampled negative” instances in the different runs of the training process. Negative sampling also produces fluctuations which greatly weakens the prediction accuracy of the learned embeddings.
- Indeed, one way of improving recommendation performance for KGEs is through KGE regularization techniques. These methods generate extra loss functions to capture the KG structure to regularize the recommendation model [88,89]. Despite performance improvements, it can be argued that KGE regularization lacks the reasoning capacity. For higher order relations, these models only encode them implicitly instead of plugging them directly into the optimization model. However, to capture long-range connectivities and to provide interpretable results for the higher-order modeling, an explicit model has to be used to capture long-range relations.
- For most of the KGE methods, the embeddings usually have to be trained in advance with respect to the recommendation algorithm, thus, leading to a lack of end-to-end training manner.

5.2. Challenges Related to DL Methods

- For path embedding methods that regard the direct paths between user-item pairs as a sequence, it is often advisable to employ RNN to encode the paths. This is attributed to the capability of RNN to efficiently model sequences of various lengths and effectively capture the semantics of entities and the entire paths between entity
pairs. Recurrent feedback mechanisms can memorize the effects of past data samples in the latent states, thus enabling RNN and its variants, such as LSTM and GRU, to model temporal information, making it an effective tool to encode the user’s behavior sequence. However, the RNN-based method is ineffective in modeling long-dependencies and incapable of operating parallelly because of its unique architecture.

- Markov Decision Processes (MDP) has also been applied in many sequential recommendation models. However, it is essential to state that MDP such as L-order Markov Chain cannot be directly used in sequence-aware recommendation in most cases. This is because data sparsity rapidly leads to poor approximations of the transition matrices and faces difficulty in the choice of order.

- Although DKG-based RSs with GCN architectures attained good performance, their training process is time-consuming. Thus, they are not efficient in modeling rapidly-changing user interests, as they may not be adequate to comprehend real-time user interests. The proposed model of Song et al. [90] which involves a dynamic-attention network, can be employed to capture rapidly changing user’s interests.

5.3. Future Research Directions

In the following, we have summarized the possible research directions that can pave the way for novel ideas in this vibrant field of research.

- One of the prominent methods for KG-based explainable recommendation is the path-embedding method. In this method, explicit embeddings of various patterns of user-item or item-item connections in the KG are explored to provide additional guidance for the recommendation. Although embedding-based reasoning is scalable and effective as the interpretation is conducted via computation within embeddings, it has the difficulty of learning reliable representation for sparse entities. As stated earlier, this is due to the fact that good embeddings heavily rely on data richness. Thus, the explainability of the recommended item is seriously affected with huge margin of error since the model’s accuracy reduces in the case of sparse data. In the future, we plan to overcome the above-cited issue by introducing rules in the learning process. Rules help to improve both the quality of sparse entity embeddings and the results of their link predictions. Also, iteratively combining rule-based learning and KGE-based learning can improve predictions and recommendations. In all, rule-based reasoning provides a way of learning deductive, interpretable, and inferential rules. We intend to explore and modify the iterative rule-based learning, or the IterE model proposed by Zhang et al. [91]. The IterE model employs a special pruning strategy to learn rules from embeddings iteratively, and further utilize old and new triples inferred by rules to learn the embeddings.

- As stated in Section 5.2, although RNN is capable of efficiently modeling sequences of various lengths and capturing the semantics of entities and the entire paths between entity pairs, the method is ineffective in modeling long-dependencies and incapable of operating parallelly because of its unique architecture. Thus, item prediction and explainability of the recommended items are hugely affected. Future research works can focus on integrating the IterE rule-based system and “batch RNNs” method. The design for the “batch RNNs”, can be summarized as follows: design the batch RNN to consist of several RNNs, such that every RNN learns the semantic representation of an individual path, while the IterE pruning algorithm [91] is employed to remove irrelevant entity-item associations. One can argue that multiple RNNs can lead to overfitting as each network will have different learning parameters. This can be avoided by designing the networks in a manner that they all share the same learning parameters to avoid overfitting the model.

- From the open challenges cited in Section 5.1, it has been noted that a large volume of side information is a major cause of high computational cost. To overcome the drawback of using a large volume of side information, an efficient route search model
based on the reinforcement learning method can be performed to extract only the rich side information.
• To overcome the issues associated with negative sampling highlighted in Section 5.1, the following technique can be adopted. For those KGEs whose loss functions can be formulated or converted to a square loss, one can adopt the square loss optimization procedure proposed in [92] for Non-sampling KGE (NS-KGE) learning. This method avoids all the negative sampling in the KG by considering all the negative instances in the KG for model learning.

6. Conclusions

Generally, explainable DKG-based RS are algorithms suggesting suitable items to the users, and explaining how those items were suggested. These algorithms consider various factors related to the users and the items. This study conducted a systematic review of publications from 2018 to January 2022 on DKG-based RS to discover state-of-the-art methodologies of explainable RS. Following a search of four academic repositories: MDPI, ACM Digital Library, Science Direct, and Google Scholar, 288 reviewed publications were collected, and a preliminary review was performed to remove duplicates, of which 103 publications remained. After conducting the primary filtering based on the abstract, title, and “exclusion” and “inclusion” criteria, 38 papers passed the eligibility stage. Out of the 38 papers, 18 were selected for the final review. The 18 collected papers were categorized into groups based on how the KG/KGEs or the path-embeddings are learned and merged with the DL methods. A deep dive into the mathematical formulations and algorithms was conducted. The contributions and drawbacks of the algorithms used in those surveyed papers were presented, thus, making it easier for researchers to improve on those algorithms. The survey revealed that there is growing popularity in PEELM and PEM algorithms between 2018 and January 2022, especially those algorithms that employ RNN and GCN for explainability. However, there was an overall decline in explainable DKG-based RS publications. This decline in publications is mainly due to the various challenges associated with DKG-based methods for explainable RS. Thus, the major challenges confronting this new field of research were provided. We concluded with possible solutions for each open challenge, thus defining a clear roadmap for future research.

In summary, this survey paper has presented an analytical framework based on a fine-grained hierarchical taxonomy to systematically classify recently published papers in the field of explainable DKG-based RSs. According to the above, the strengths and weaknesses of each DKG method in the context of explainable recommendations have been discussed, and the relevant challenges to be pursued to promote the design of the next wave of explainable RSs have also been analyzed. This is a new research field; thus, we hope this review paper can assist readers in better understanding of works in this vibrant area of research.

Author Contributions: R.F.D.: Conceptualization, Methodology, Writing original draft, Investigation, Writing review & editing. C.Z.: Supervision, Resources, Funding acquisition. I.T.: Validation, Investigation, Writing review & editing. J.K.A. and B.D.: Writing review & editing. All authors read and approved the final manuscript.

Funding: This paper is supported by the Key Research and Development Plan (Social Development) projects BE2016630 and BE2017628 of Jiangsu province, the Scientific Research Project Z201603 of Wuxi Health and Family Planning Commission.

Conflicts of Interest: The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References


71. Yang, Z.; Dong, S. HAGERec: Hierarchical attention graph convolutional network incorporating knowledge graph for explainable recommendation. Knowl. Based Syst. 2020, 204, 106194. [CrossRef]


73. Shimizu, R.; Matsutani, M.; Goto, M. An explainable recommendation framework based on an improved knowledge graph attention network with massive volumes of side information. Knowl. Based Syst. 2021, 239, 107970. [CrossRef]


