Knowledge Graph Comparison and Completion Techniques for Recommender Systems : A Survey

Wenqi Liao : New York University, USA

June 30, 2023

Abstract

In this paper, we aim to explore and identify knowledge graph comparison and completion methods, as well as recommender system models, that can be leveraged to build our own educational system. We delve into two categories of techniques: subgraph matching and embedding-based methods. For each technique, we discuss their relevance and potential applications in recommender systems. Additionally, we provide a comprehensive overview of PinSage, a robust and well-established recommendation engine based on Graph Convolutional Networks (GCN). By examining these methods and models, we seek to gain insights and inspiration for developing an effective and tailored recommender system for our educational context.

1 Introduction

The objective of this paper is to conduct a comprehensive survey on knowledge graph comparison and completion techniques to facilitate the development of an educational recommender system. Knowledge graph comparison and completion play a significant role in building recommender systems, as knowledge graphs allow us to represent entities as nodes and capture their relationships as edges. This structured graph representation offers an efficient approach for similarity analysis, enabling the identification of potential similar items that a user might be interested in. Furthermore, by consolidating users' past activities and profiles into a single knowledge graph or a knowledge graph system, personalized recommendations can be generated by considering individual users' unique characteristics and preferences. It is worth noting that knowledge graph comparison is not limited to within a single graph; comparing a knowledge graph to another can lead to potential recommendations, thus promoting discovery.

To achieve this objective, this paper introduces different techniques for comparing and completing knowledge graphs, including Subgraph Isomorphism and Embedding-based methods. Subgraph Isomorphism is a technique that examines whether a smaller graph (subgraph) is isomorphic to a larger graph (target graph), enabling the identification of common patterns or subgraphs within knowledge graphs. On the other hand, Embedding-based methods focus on representing nodes or entire graphs as low-dimensional vectors (embeddings) in a continuous vector space. These embeddings capture both structural and semantic information, enabling meaningful comparisons. Within the realm of Embedding-based methods, two prominent sub-fields are discussed: translational models and deep models. Translational models are specialized for structured data with explicit relationships, while deep models are more versatile and capable of learning complex patterns from diverse data types.

In the subsequent paragraphs, each technique will be discussed in detail. This discussion will include a definition of each technique, an overview of one or two frequently used methods or algorithms associated with the technique, an exploration of how these methods or algorithms contribute to building recommender systems, and an evaluation of their suitability, particularly in the context of our educational recommender system.

2 Educational Recommender System Design

Our recommender system aims to assist students in improving their performance in exams and homework by providing personalized exercises. The system takes into account various factors such as the course content, the student's performance on quizzes, homework, and past exams. Additionally, the system incorporates a comparative analysis of different students' profiles. By comparing performance data across students, the system can predict how a particular student may perform on concepts they may have struggled with or have not encountered yet.

3 Preliminary Definition

3.1 Graph

The formal definition of a graph G is represented as $G = (V, E, L)$, where V is the set of vertices (nodes), E is the set of edges that connect the vertices, and L represents labels of nodes in graph. The edges in a graph can have additional attributes or weights to represent additional information or properties associated with the relationships.

3.2 Knowledge Graph

A knowledge graph is a graph-based data structure designed to capture and represent knowledge about the real world. It organizes information as entities and their relationships, where entities are represented as nodes (V) and relationships between entities are represented as edges(E). In a knowledge graph, data is typically represented in the form of triples, denoted as $G = \{(h, r, t)\}$, where h represents the head entity, r represents the relationship or edge, and t represents the tail entity[6].

4 Subgraph Matching

Subgraph matching plays a crucial role in building recommender systems as the relationships and structural patterns between items and users provide important information for potential recommendations that users might be interested in. It involves finding subgraphs within a larger graph that match a given pattern graph, enabling the identification of similar or related items based on their structural relationships.

4.1 Definition

4.1.1 Subgraph Isomorphism

Subgraph isomorphism determines whether a subgraph exists within a larger graph. Given a graph G with vertices and edges represented as (V, E), and a subgraph Gs with vertices and edges represented as (Vs, Es), Gs is a subgraph of G if and only if all Vs are a subset of V and all Es are a subset of E [1]. Due to the exponential search space generated by all possible subgraphs, subgraph isomorphism is an NP-complete problem [3]. This complexity makes it challenging to perform matching on large graphs. As knowledge graph comparison often involves large databases, exact matching becomes intractable, leading to the study of approximate and simulation matching techniques.

4.1.2 Simulation Matching

Simulation matching aims to find all occurrences of a pattern graph within a target graph while preserving the exact structural and connectivity relationships. It identifies all subgraphs in the target graph that are isomorphic to the pattern graph, ensuring both correctness and completeness in finding exact matches.

4.1.3 Approximation Matching

Approximation matching relaxes the requirement of exact isomorphism and allows for some degree of variation or approximation in the matches. It aims to find subgraphs that are similar to the pattern graph but may not be identical in terms of structure or connectivity.

4.1.4 Query Graph

A query graph represents a specific pattern or structure that you want to find within a larger graph or dataset. It defines the shape, connectivity, and properties of the desired subgraph or pattern that you want to extract from the data.

4.1.5 Data Graph

A data graph, also known as the target graph, is the larger graph or dataset in which you want to search for patterns or extract information. It represents the actual graph or network data that contains the entities, relationships, and attributes of interest.

In the context of recommender systems, real-time results are often preferred as users search for items of interest. Therefore, strict accuracy is not always required. Hence, I will focus on introducing the approximate matching algorithm, specifically TurboISO. I have chosen TurboISO because, as proven in [4], it outperforms other methods such as VF2, SPath, and RI when the average degree of the graph is high. Since recommender systems often involve densely connected graphs, TurboISO is an excellent choice.

4.2 TurboISO[5]

Since subgraph matching is an NP-hard problem, traditional approaches that involve listing all possible permutations become highly inefficient and infeasible, especially for large datasets. To address this issue, efficient matching requires the discovery of a suitable matching order for the entire graph and the implementation of an effective filtering method to exclude undesirable candidates in the early stages. However, existing methods face challenges in selecting the matching order due to the exponential time complexity involved. In this context, TurboISO introduces two novel concepts: 'candidate region exploration' and 'combine and permute strategy' (COMB/PERM).

Candidate region exploration rapidly identifies candidate subgraphs containing embeddings of the query graph and computes a robust matching order for each explored region. This technique minimizes the size of each candidate region, ensuring faster matching and effectively addressing the filtering problem.

The proposed method, TurboISO, utilizes a Breadth-First Search (BFS) tree, T_q, which is generated on the query graph q from a selected root node U_s. This process guarantees that each leaf node has its own shortest path relative to the start node. The start vertex V_s of each candidate region in the data graph g is examined, and a depth-first search is performed using T_q from V_s to identify candidate data vertices for each query vertex. Each path in T_q (for each start vertex) is sorted in ascending order based on the number of candidate vertices, resulting in a matching order. For each candidate region, candidate vertices for each query vertex are found according to the obtained order.

To provide a visual understanding, a graph, figure 1, representation is presented. Once a BFS tree T_q is obtained from q, unnecessary edges, such as the one between u_2 and u_3, are removed. In CR1(u_1), query vertex u_2 has a candidate vertex v_2, while u_3 has two candidates, v_4 and v_{1005}. This results in a matching order (u_1, u_2, u_3). Subsequently, embeddings are computed using only the candidate vertices.

The COMB/PERM strategy addresses the issue of matching order by avoiding unnecessary enumerations between query and data vertices, as depicted in Figure 2. When considering only v_2, v_3, and v_4 in the data graph g, COMB/PERM generates combinations of these vertices to avoid generating all mappings that include them. If a generated combination produces a valid mapping, the strategy proceeds to permute all possible mappings for that combination. Conversely, if a

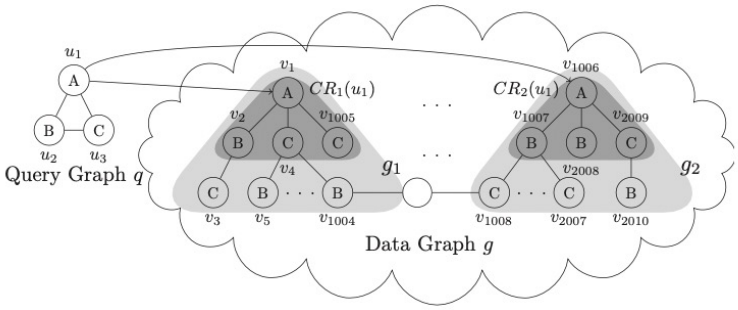

Figure 1: A query graph q and a data graph g[5].

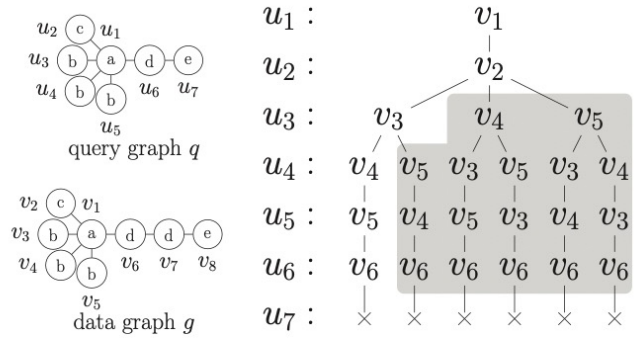

Figure 2: An example of useless enumerations[5].

combination fails to generate a valid mapping, all other permutations for that combination are discarded. For instance, if the combination (v_3, v_4, v_5) from the data vertices v_3, v_4, v_5 fails to generate a valid mapping, all other permutations of these vertices will be pruned.

Additionally, TurboISO introduces a new concept called 'neighborhood equivalence class' (NEC) to determine the query vertices for generating combinations. Query vertices in the same NEC share identical matching data vertices. For example, u_3, u_4, and u_5 belong to the same NEC, as they have the matching data vertices v_3, v_4, v_5.

4.2.1 Application in recommender System

While subgraph isomorphism is an essential component of building recommender systems, it does not play the primary role. Recommender systems focus heavily on user-item interactions, item features, user preferences, and other semantic factors. Here are some specialized applications of using TurboISO or subgraph isomorphism in general for our recommender system:

Identifying Similar Performance Patterns: Comparing performance graphs of students helps identify common strengths, weaknesses, and learning patterns, enabling targeted recommendations and interventions.

Personalized Exercise Generation: By comparing the course content graph with a student's performance graph, we can determine specific concepts to focus on and recommend related exercises based on structural similarities.

Predicting Performance on Unseen Concepts: Analyzing patterns and relationships between a student's performance graph and the course content graph allows us to predict their performance on unfamiliar concepts, enabling targeted recommendations and additional resources.

Integration with Other Recommendation Techniques: Integrating TurboISO and subgraph

isomorphism with other approaches like content-based filtering and graph neural networks provides a comprehensive understanding of student needs and more effective recommendations.

5 Embedding-based methods

The embedding-based method has emerged as a widely utilized approach in recommender systems, offering effective solutions for capturing underlying relationships and patterns between items and users. By projecting items and users into a shared latent space, this method enables accurate recommendations by effectively measuring similarity. Embeddings serve as dense representations that encode semantic information, facilitating efficient computations and personalized recommendations based on user preferences. The versatility, scalability, and adaptability of this method make it indispensable for modern recommendation systems.

Within the realm of embedding-based methods, knowledge graph (KG) embedding models can be broadly categorized into two main types: translational models and deep models. In this section, we will explore one representative embedding-based method from each category, offering a comprehensive introduction to one method and a concise overview of a recommender system model built upon it. While deep models have gained significant attention from researchers, it is also important to discuss a mature recommendation engine within this domain.

5.1 Definition

5.1.1 Knowledge Graph Embedding

Knowledge graph embedding involves the creation of dense representations that capture the semantic meaning of objects and relationships within a knowledge graph. These embeddings encode the essential characteristics of objects and relationships in a low-dimensional space, facilitating similarity computations. By measuring the similarity between two vectors, we can infer the similarity between the corresponding objects in the knowledge graph[8].

5.1.2 Translational Model

Translational models approach the modeling of relations between entities as translations in a vector space. These models encode relationships by learning the translation vector that represents the transition from one entity to another[9].

5.1.3 Deep Model

Deep models refer to neural network architectures that consist of multiple layers of neurons. By leveraging deep architectures, these models can effectively capture non-linear relationships between users and items, while also capturing high-level features and latent factors that drive user preferences.

5.1.4 Link Prediction

The link prediction problem in knowledge graphs involves predicting the likelihood of a particular relationship (or link) between two entities based on existing information. This problem can be formulated as a point-wise learning to rank problem, where the goal is to learn a scoring function denoted as $\psi : \mathcal{E} \times \mathcal{R} \times \mathcal{E} \rightarrow R$. Here, $\mathcal{E}$ represents the set of entities and $\mathcal{R}$ represents the set of relationships in the knowledge graph[12].

For the translational model category, we will explore TransE, which is one of the most popular and well-known translational methods in the field of knowledge graph embedding. TransE also serves as a benchmark for evaluating other translational models.

Additionally, we will briefly introduce a recommender system model called ECFKG, which employs TransE as the knowledge graph embedding (KGE) method. Within the realm of deep models, ConvE stands out for its remarkable capability to capture the structural characteristics and semantic relationships present in a graph. This makes ConvE an invaluable tool in recommender

systems, as it not only enhances the precision of the models but also offers higher levels of explainability. In this context, we will also discuss Ekar, a recommender system model that utilizes ConvE to refine reward shaping, thereby further improving its recommendation capabilities. Moreover, we will introduce PinSage, a mature recommendation engine based on Graph Convolutional Networks (GCN), which effectively handles the complex structures and relationships found in large-scale knowledge graphs.

5.2 TransE[10]

TransE, or Translating Embeddings for Modeling Multi-relational Data, is a significant approach in building recommender systems that deal with multi-relational data. Traditional methods designed for single-relational data are inefficient and impractical due to the diverse range of relationships and entity types involved in the context of recommender systems. Therefore, a more generic and scalable method is required.

Alternative models in the field, such as non-parametric Bayesian extensions and tensor factorization or collective matrix factorization models, offer excellent expressivity. However, their complexity often compromises interpretability and leads to potential issues of overfitting and undercutting. To address these challenges, it is crucial to shift the focus towards simpler methods that strike a better balance between accuracy and scalability.

TransE adopts a translation-based parameterization motivated by two key factors. Firstly, hierarchical relationships are prevalent in knowledge bases, and translations naturally represent these relationships. Secondly, the author was inspired by an article on learning word embeddings, which demonstrated that one-on-one relationships between entities of different types, such as the relationship between countries and their capitals, can also be effectively represented through translations.

In TransE, knowledge base embeddings are generated using an energy-based model. Given a triplet (h, l, t), TransE treats the relationship l as a translation from the head entity h to the tail entity t in a metric space. If $h + l$ is close to t, then t should be one of the closest neighbors to $h + l$. The approach utilizes a reduced set of parameters by learning a low-dimensional vector for each entity and relationship.

The energy of a triplet in the TransE framework is defined as the dissimilarity measure, denoted as $d(h + l, t)$, which can be either the L_1 or L_2 norm. The optimization process involves stochastic gradient descent with minibatches, and it includes the additional constraint that the L_2 norm of entity embeddings is 1. This constraint is crucial as it prevents the training process from trivially minimizing the loss by continuously increasing the norms.

TransE has been proven to outperform other state-of-art models on link prediction. Its translation-based parameterization is well-suited for a variety of relationships, making it a highly effective choice in building recommender systems.

5.3 ECFKG[11]

In the development of the ECFKG model, the authors recognized the lack of explicit relationship modeling in existing approaches for incorporating different information about users and items. This limitation hindered the explainability of the recommender system. To address this issue, the authors integrated the collaborative-filtering method typically used for unstructured data. However, to preserve the structured nature of the data and maintain internal relationships, they leveraged knowledge-based embeddings inspired by TransE. This approach allowed heterogeneous information to be projected into a unified low-dimensional embedding space, marking the first use of knowledge-based embeddings in explainable recommendation systems. By applying a softmax function, the authors were able to emphasize the distinctions between similar and dissimilar entities. The use of the softmax function transformed the problem into an optimization format, solvable with iterative optimization algorithms, thereby ensuring the explainability of the system. In addition, the authors extended the softmax function to calculate the probability of an entity being a possible tail entity, given the head entity and the relationship. This probability calculation enabled the

selection of the item with the highest possibility as the recommended item. Moreover, the model constructed an explainable path that links the head entity, the relationship, and the tail entity.

5.3.1 Application in Recommender System

The application of the ECFKG model in recommender systems [11] offers valuable insights that can be applied to our educational system. By considering relations such as "student getting wrong on an addition problem" and "addition belongs to algebra," we can employ a similar approach. Once we have learned knowledge graph embeddings over the graph and obtained embeddings for each student, the model can identify the most similar problem related to addition in the algebra domain. This enables us to recommend relevant practice materials to students based on their individual needs and performance.

Given that we have only addressed a single embedding method and model within the category of translational models, it is important to consider the potential benefits of employing a translational model recommender system in our specific context.

Given the existence of numerous distinct entities, including professors, students, quizzes, homework assignments, exams, and problems, as well as a variety of relationships such as a student's enrollment in a professor's course or a student's incorrect answer on problem 001, it becomes crucial to transform all these relationships into a structured vector space through knowledge graph embedding or a translational model. By representing all pertinent information in vector form, the analysis and recommendation processes become more transparent. Additionally, this structured approach enables the exploration of different optimization functions or classifications, further enhancing the overall effectiveness of the system.

5.4 ConvE[12]

ConvE is a powerful model that leverages 2D convolutions applied to embeddings in order to predict missing links within knowledge graphs.

The motivation behind the ConvE model stems from the inherent incompleteness often found in knowledge graphs, where important relationships between entities are missing. Link prediction becomes a valuable tool in addressing this issue by enabling the prediction of these missing relationships. However, traditional link prediction methods face limitations when dealing with large knowledge graphs containing millions of nodes, as they primarily rely on shallow models. Increasing the expressiveness of shallow models typically requires enlarging the embedding size, which results in a significant increase in memory requirements. This becomes infeasible when dealing with large knowledge graphs.

To overcome these challenges, ConvE introduces the concept of multiple layers in the model. Unlike shallow models, increasing the number of layers in a model enables an increase in the number of features independently from the embeddings themselves. By incorporating 2D convolutions, ConvE effectively captures a richer set of feature interactions compared to 1D models. However, the use of multiple layers introduces the risk of overfitting due to the fully connected nature of the embeddings.

To address these issues, the authors of ConvE utilize the convolution operator, a highly efficient and parameter-effective tool with optimized GPU implementations. The workflow of ConvE involves reshaping and concatenating the embeddings of the subject (head) entities and relationships, which serves as input for the 2D convolutional layer. This produces a feature map tensor, which is then vectorized and projected into a k-dimensional space through linear transformation. The resulting vector is matched with the embeddings of the object (tail) entities.

The ConvE model is trained using the logistic sigmoid function to minimize binary cross-entropy loss. To regularize the model, dropout is applied to the embeddings, the feature maps after the convolution operation, and the hidden units after the fully connected layer.

In [12], it is evident that ConvE stands out as the simplest multi-layer convolutional architecture for link predictions. With a single convolution layer, a projection layer, and an inner product layer, ConvE offers a higher level of expressiveness while utilizing fewer parameters compared to other models.

5.5 Ekar[13]

The motivation behind Ekar stems from the limitations of existing recommender system models, which often lack the ability to provide meaningful explanations. It would be highly desirable to generate a path from the target user to the item they might be interested in. For example, if user A watched film B and actor C performed in film B, and actor C also performed in film D, the system could recommend film D to user A, with a clear and persuasive explanation. However, traditional path generation methods, such as breadth-first search, often fall short in terms of generating semantically meaningful paths efficiently. To address this challenge, the authors of Ekar shifted their focus to reinforcement learning.

By incorporating the principles of reinforcement learning, Ekar constructs paths based on sequential decision-making, resulting in interpretable recommendations. The user sequentially navigates through a user-item-entity graph, making decisions at each step. However, a key issue with reinforcement learning models lies in their sparse rewards. This means that the training agent receives no reward until it reaches the target item. In a recommender system with a vast search space, it becomes highly likely that no useful recommendations will be generated, as the agent may continually interact with items it has already encountered—items that the user has already purchased or viewed.

To overcome this challenge, the authors of Ekar propose incorporating the Convolutional Knowledge Graph Embedding (ConvE) method. They utilize the ConvE model for reward shaping by leveraging the score function of ConvE to compute the reward. In ConvE, the score function is employed to determine the likelihood or compatibility of a given triple (head entity, relationship, tail entity) in the knowledge graph. The score function in ConvE involves several operations, including convolutional neural network (CNN) operations and matrix multiplications, as mentioned previously.

By combining the knowledge graph embedding method with reinforcement learning, Ekar achieves both effective recommendations and robust explainability. Unlike many other state-of-the-art KG-based recommendation methods, Ekar offers enhanced interpretability without compromising its overall effectiveness.

5.5.1 Application in Recommender System

When considering the application of Ekar in the context of an educational recommender system, it offers valuable insights and advantages. Building upon the example path discussed earlier, let's consider the following path: a student makes a mistake in an addition problem, addition is a part of algebra, and multiplication is also a part of algebra. Based on this path, the system can recommend that the student practices multiplication. Paths like this provide the system with high explanatory power, offering clear insights into the recommended choices.

In the educational context, the reinforcement learning aspect of Ekar holds significant potential. Student learning is a continuous process, where concepts are built upon one another, such as how machine learning builds upon linear algebra as a prerequisite. With the sequential decision-making approach, we can develop a personalized recommendation system for each student based on their past experiences and courses, taking into account their previous performance and using it to analyze their current performance. This approach allows for a continuous, rather than discrete, understanding of the student's progress. Leveraging the system's excellent explainability, we could even generate automated learning reports within the system.

Considering that we have only addressed a single embedding method and model within the category of deep models, it is crucial to recognize the potential benefits of employing a deep model recommender system in our specific educational context.

With a neural network architecture, the model is capable of uncovering complex patterns and relationships within the knowledge graph that may be obscure and difficult for us to realize on our own. As a result, the recommendations provided by the system will be more refined without sacrificing explainability. Additionally, leveraging the high-explainability feature of deep models, the system can help educational institutions and professors gain a better understanding of how students perceive their classes in a general sense. This understanding can be instrumental in improving the quality of teaching and enhancing the overall learning experience.

5.6 PinSage[14]

PinSage is a recommender system model that distinguishes itself from other models through its scalability and improved embedding quality.

Scalability is a major concern in traditional Graph Convolutional Network (GCN) based models, as most algorithms operate on the entire graph Laplacian, which becomes impractical at production scale. PinSage addresses this challenge with several key features.

Firstly, PinSage performs local convolutions instead of full-graph convolutions. This means the model only focuses on the neighborhood of the node of interest. By transforming the representations of the node's neighbors using a dense neural network and employing an aggregator or pooling function, PinSage generates a vector representation of the node's neighborhood. This neighborhood vector is then combined with the node's own vector and fed into another dense neural network layer to produce a new vector representation for the node.

Secondly, PinSage utilizes a producer-consumer pattern to optimize GPU utilization. With multiple GPUs available on a single machine, each GPU takes a portion of the minibatch (a subset of training data) and performs computations. Since the node number can be quite large, the batch size is set to be large as well, ranging from 512 to 4096. However, the adjacent list and feature matrix for each node are stored on the CPU. To avoid inefficient communication between the GPU and CPU during the GPU's convolutional process, the CPU feeds the required adjacent lists and feature matrix to the GPU at the beginning of each minibatch iteration.

Thirdly, PinSage employs a MapReduce approach to reduce repeated computations. Given that a node can be a neighbor of many other nodes, computing embeddings for different nodes can result in countless repetitions. PinSage uses MapReduce to project all pins (nodes) into a low-dimensional latent space and then joins the low-dimensional pin representations, utilizing pooling to reduce the computation needed and obtain the board (graph) embedding. This approach ensures that the latent vector for each node is computed only once.

PinSage also enhances embedding quality, leading to significant performance improvements in recommendation, through several key features.

Firstly, PinSage uses random walks to determine the importance of neighbors for a given node. Instead of using the entire neighborhood for computation, PinSage defines the importance based on the visit count of each node during a random walk starting from a specific node. Nodes with higher visit counts are considered neighbors, and this importance is taken into account when aggregating the vector representations of the neighborhood.

Secondly, PinSage employs stacking convolutions to provide more information about local graph structures. By stacking one convolution on top of another, PinSage can capture richer information about the relationships within the graph.

Lastly, during training, PinSage gradually introduces harder and harder negative examples. These hard negative examples are items that are related to the query item but not considered positive items. By increasing the level of hardness, PinSage forces the model to distinguish items at a deeper level, enhancing its ability to make accurate recommendations.

PinSage brings the GCN model to a production scale, overcoming scalability challenges, and provides valuable insights for researchers seeking to leverage GCN more effectively in the field of recommendation systems.

5.6.1 Application in Recommender System

Given PinSage's maturity as a recommendation engine that effectively considers both semantic and graph structure, our next step would involve applying the PinSage engine to our specific educational system and leveraging our educational schema.

6 Conclusion

In this paper, we have explored two major fields that are applicable in knowledge graph comparison and building recommender systems: subgraph matching and embedding-based methods. Subgraph matching techniques excel in analyzing and comparing knowledge graph structures, making them valuable when graph structure plays a crucial role in the system. We have introduced TurboISO, a subgraph matching technique suitable for highly dense connected graphs, which can provide insights into conducting structure comparison between graphs, making it suitable for recommender systems as well.

Moving on to embedding-based methods, we have discussed two sub-categories: translational models and deep models. Translational models are focused on the semantic aspect of the graph and we introduced TransE, a commonly-used Knowledge Graph Embedding (KBE) method capable of handling multi-relational data. We then introduced ECFKG, a recommender system model built upon TransE, which aids in making the relationships in the knowledge graph explicit.

Deep models, on the other hand, capture both semantic and structural information in knowledge graphs, making them a popular choice in recommender systems. We have introduced ConvE, a well-known KBE method, which is useful for predicting missing relationships between entities in the knowledge graph. Additionally, we discussed Ekar, a model based on ConvE that combines KBE and reinforcement learning to construct explainable paths for recommending items.

Furthermore, we introduced PinSage, the first application of Graph Convolutional Networks (GCN) in a production-scale recommender system. PinSage effectively handles both semantic and structural information in knowledge graphs.

Considering the applicability of these techniques in our educational recommender system, we can leverage subgraph matching techniques when our educational knowledge graph exhibits complex and essential structural similarities that other recommender models may fail to capture. Translational models can be highly useful for analyzing semantic similarity between students and educational entities, extracting explicit relationships that enhance the explainability of the overall system. Deep models, combining both semantic and structural aspects, should be a primary consideration. With the sequential features of some deep models, we can build our educational recommender system following specific timelines, such as school years or recommended course schedules, while maintaining the system's explainability.

Based on these considerations, we suggest starting the system development with the PinSage model and subsequently incorporating any structural or semantic improvements from other techniques. This iterative approach will allow us to refine and enhance the system over time.

References

[1] Somkunwar M R, Vaze V M. Subgraph Isomorphism Algorithms for Matching Graphs: A Survey[J]. IJETT, 2017, 1(1).

[2] Cordella L P, Foggia P, Sansone C, et al. A (sub) graph isomorphism algorithm for matching large graphs[J]. IEEE transactions on pattern analysis and machine intelligence, 2004, 26(10): 1367-1372.

[3] Wang, X., Zhang, Q., Guo, D. et al. A survey of continuous subgraph matching for dynamic graphs. Knowl Inf Syst 65, 2023, 945–989.

[4] Ma T, Yu S, Cao J, et al. A comparative study of subgraph matching isomorphic methods in social networks[J]. IEEE Access, 2018, 6: 66621-66631.

[5] Han W S, Lee J, Lee J H. Turboiso: towards ultrafast and robust subgraph isomorphism search in large graph databases[C]//Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data. 2013: 337-348.

[6] Liu J, Duan L. A survey on knowledge graph-based recommender systems[C]//2021 IEEE 5th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC). IEEE, 2021, 5: 2450-2453.

[7] Tschida C. Knowledge Graphs and Knowledge Graph Embeddings[J]. 2020.

[8] Sun Z, Zhang Q, Hu W, et al. A benchmarking study of embedding-based entity alignment for knowledge graphs[J]. arXiv preprint arXiv:2003.07743, 2020.

[9] Palumbo E, Rizzo G, Troncy R, et al. Translational models for item recommendation[C]//The Semantic Web: ESWC 2018 Satellite Events: ESWC 2018 Satellite Events, Heraklion, Crete, Greece, June 3-7, 2018, Revised Selected Papers 15. Springer International Publishing, 2018: 478-490.

[10] Bordes A, Usunier N, Garcia-Duran A, et al. Translating embeddings for modeling multi-relational data[J]. Advances in neural information processing systems, 2013, 26.

[11] Ai Q, Azizi V, Chen X, et al. Learning heterogeneous knowledge base embeddings for explainable recommendation[J]. Algorithms, 2018, 11(9): 137.

[12] Dettmers T, Minervini P, Stenetorp P, et al. Convolutional 2d knowledge graph embeddings[C]//Proceedings of the AAAI conference on artificial intelligence. 2018, 32(1).

[13] Song W, Duan Z, Yang Z, et al. Explainable knowledge graph-based recommendation via deep reinforcement learning[J]. arXiv preprint arXiv:1906.09506, 2019, 13.

[14] Ying R, He R, Chen K, et al. Graph convolutional neural networks for web-scale recommender systems[C]//Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery data mining. 2018: 974-983.