A Personalized Recommender System Based-on Knowledge Graph Embeddings

Ngoc Luyen Le, Marie-Hélène Abel, Philippe Gouspillou

A Personalized Recommender System based-on Knowledge Graph Embeddings

LE Ngoc Luyen[1,2], Marie-Hélène ABEL[1], Philippe GOUSPILLOU[2]

[1] Université de technologie de Compiègne, CNRS, Heudiasyc (Heuristics and Diagnosis of Complex Systems), CS 60319 - 60203 Compiègne Cedex, France
[2] Vivocaz, 8 B Rue de la Gare, 02200, Mercin-et-Vaux, France

Abstract. Knowledge graphs have proven to be effective for modeling entities and their relationships through the use of ontologies. The recent emergence in interest for using knowledge graphs as a form of information modeling has led to their increased adoption in recommender systems. By incorporating users and items into the knowledge graph, these systems can better capture the implicit connections between them and provide more accurate recommendations. In this paper, we investigate and propose the construction of a personalized recommender system via knowledge graphs embedding applied to the vehicle purchase/sale domain. The results of our experimentation demonstrate the efficacy of the proposed method in providing relevant recommendations that are consistent with individual users.

Keywords: Knowledg Graph, Node Embedding, Deep Learning, Recommender System

1 Introduction

In today's world, where there is an abundance of information, recommender systems have become increasingly important in curating and presenting relevant content to users. These systems are designed to assist individuals by providing personalized suggestions and recommendations, which makes information discovery and decision making more efficient. The use of knowledge knowledge graphs for recommender system has demonstrated the ability to produce highly accurate recommendations that are also straightforward to interpret and explain [14, 24]. The essential elements of knowledge graphs are to enhance the organization and structure of information which is able to effectively define a measure of relatedness between entities [6].

A knowledge graph by means of ontology creates a structured framework for a set of concepts or terms within a specific domain by arranging them in a hierarchical manner, and by using relation descriptors to model the connections between these concepts or terms. This provides a standardized lexicon for representing entities within that domain [11, 16]. Recently, the application of knowledge graphs has grown in popularity within the field of recommender systems and decision support systems for graph-based feature learning [20, 25]. As a part of out work, we are especially interested in constructing a personalized recommender system by utilizing knowledge graph embedding for feature learning.

Most of the work in knowledge graph embedding focuses on create continuous vector representation represented the entities and relations and define a scoring

function to measure its relatedness [21]. These vectors are learned using feature learning algorithms such as TransE [1], TransR [22], Node2Vec [4] and others which are designed to capture the structural relationships between entities in the graph. The main idea of knowledge graph embeddings is to map the symbolic entities and relations in the knowledge graph to continuous vector space, this allow us to apply mathematical operations to them, like similarity measure or clustering.

In the domain of purchasing and selling vehicles, the descriptions of the vehicles often contain a wealth of information. This information can be organized using ontologies, as demonstrated in works such as [5], [12]. To the best of our knowledge, there does not currently exist a recommender system for the vehicle domain that utilizes features from knowledge graph embeddings.

The rest of this article is organized as follows: Section 2 introduces works from the literature on which our approach is based. Section 3 presents our main contributions, starting with the problem statement and then our proposed approach for the construction of the recommender system using knowledge graph embeddings. In section 4, we test our work using a dataset from the domain of purchase/sale of used vehicles. Finally, we conclude and present the perspectives.

2 Related Works

Recommender systems are a type of application that attempts to predict a user's preferences for items and suggests the most relevant items to them through information retrieval [10]. These suggestions can aid users in various decision-making processes, such as choosing music to listen to or products to purchase. Generally, recommender systems are classified into six main categories: Collaborative Filtering, Content-based, Demographic-based, Knowledge-based, Context-aware, and Hybrid [5].

Collaborative Filtering RSs are based on the idea that users who have similar tastes in items will have similar preferences in the future. These systems analyze past interactions between users and items, such as purchase history or ratings, to identify patterns and make recommendations [18]. While content-based RSs, on the other hand, focus on the characteristics of the items themselves to make recommendations. These systems analyze the attributes of items and compare them to a user's past interactions or preferences to make suggestions [9]. With these RS types, the integration of external knowledge resource can provide more information about context or personalize on users and items which particularly useful for improve the accuracy and performance of RSs.

In the past couple of years, there have been many studies that have incorporated knowledge graphs into the construction of feature learning for recommender systems. For example, Sun et al. proposed a novel neural network to capture the semantic information of entity pairs in their work [19]. In [23], the authors presents an intriguing illustration of how knowledge graph embeddings can be integrated with other representations constructed from multimedia content to enhance the quality of recommendations. They employ translational models to develop knowledge graph embeddings, which are subsequently combined with embeddings of the items' content, such as textual and visual knowledge, to initiate a matrix factorization. In [17], the authors proposed the construction of knowledge embeddings for content-based recommender systems by learning

representations of entities in RDF graphs using external knowledge graphs from DBpedia and Wikidata. In [14], the authors introduce a novel approach called *entity2rec* for constructing knowledge graph embeddings based on the properties of entities . They define a global user-item score to compute the relatedness of properties and use a learning ranking algorithm to rank the top recommended items. The above studies have demonstrated that incorporating recommendation algorithms based on knowledge graph embeddings represents a simple and efficient approach for the inclusion of user and item knowledge entities in the recommendation process [7].

In the context of recommender systems, feature learning can be used to automatically discover and extract features from knowledge graph in order to create knowledge graph embedding. These features can be used as input to other algorithms, such as matrix factorization or neural networks. By learning a compact and informative representation of the entities and relationship in knowledge graphs, feature learning can help to capture the most important and relevant information for a recommendation task, leading to more accurate and personalized recommendations for users.

3 Our Proposition

In this section we lay out our proposition for constructing a recommender system based on knowledge graph embedding. For demonstration purposes, we introduce vehicle knowledge graph represented by means of ontologies from the e-commerce domain for the purchase/sale of vehicles.

3.1 Problem statement

In the context of an e-commerce applications, a typical recommender system consists of three main components: a set of items $I = \{i_1, i_2, .., i_n\}$, a set of users $U = \{u_1, u_2, ..., u_3\}$ and a set of interactions between users and items $D = \{d_{ui} | u \in U, i \in I\}$, where $d_{ui} = 1$ denotes that there are an interaction between user u and item i, $d_{ui} = 0$ means otherwise.

A knowledge graph is a structured representation of entities and their relationships, which can be formalized as a set $G = (E, R)$. The set E comprises of the entities and R denotes the set of relations between these entities. In the context of a recommender system, the entities E include users $u \in U$ and items $i \in I$. The relations between entities are represented as a triple (e, r, e), where $r \in R$ and $e \in E$. In general the relation can be described by a property of the entity. For example, the transmission of the vehicle item "Tesla Model S 2020" is automatic. This information can be represented in the form of a triple $(Tesla\_Model\_S, Transmission, Automatic)$. The knowledge graph can be created and structured using a domain ontology or by extracting information from large knowledge bases such as DBPedia or Wikidata.

Given a collection of interactions between users and items in a e-commerce application and the knowledge graph G, the recommendation task is to predict and rank a list of the most suitable items for a particular user from a list of candidate items through compare the score between them.

By utilizing the knowledge graph, we can integrate it into the recommendation task. In the following section, we will investigate a method to extract

information from the knowledge graph and use it to enhance the performance of the recommender system.

3.2 Knowledge graph embedding based on relation types

The wealth of information provided by the knowledge graph can be leveraged to add additional information to the learning process. Building on the work of Palumbo et al. in *entity2rec* [14, 15], we propose utilizing knowledge graph embeddings by incorporating feature learning for specific properties of entities or relation types from a knowledge graph perspective. Different methods have been developed for feature learning from knowledge graphs to create knowledge graph embeddings, such as TransE [1] or TransR [22]. These methods learn information about entities without considering their specific relation types or properties, which can make the results difficult to understand and explain. In contrast, the *entity2rec* approach [14, 15] considers the relation types and properties used to create the vector representations, allowing for observation and evaluation of the importance of different relation types in knowledge graphs for recommendation tasks.

The vector representation of two nodes should capture as much information as possible about the nodes, including their relation types with other nodes. This is because different relation types have different semantic values based on their relationship with other entities in the knowledge graph. By separating feature learning based on relation types, it is possible to assign a weight to each relation type, which can be used for overall comparison or for comparison of specific properties. For example, if comparing two vehicle entities, they may have the same vehicle type but different manufacturers. In this case, the relatedness based on the "*vehicle type*" relation type will differ when compared to the relatedness based on all relation types.

We adopt the widely used *node2vec* on the knowledge graph to each relation type in order to embed entities into space $\mathbb{R}^d$. First, we extract and build a subgraph for each relation type G_p. Then, we start to learning vector representation of nodes with each relation type $x_p : e \in G_p \rightarrow \mathbb{R}^d$. The model is trained by using the *node2vec* objective function [4]:

$$\max_{x_p} \sum_{e \in G_p} (-logZ_e + \sum_{n_i \in N(e)} x_p(n_i) \times x_p(e)) \tag{1}$$

where $Z_e = \sum_{v \in G_p} exp(x_p(e) \times x_p(v))$ denotes the per-node partition function. We approximate it by using negative sampling [13]. The neighborhood of an entity e, denoted as N(e), is defined using a node2vec random walk. The optimization process for this is performed using stochastic gradient ascent on the parameters that define x_p. With the vector representation, $x_p(e)$, for an entity on a relation type p, we will investigate the method of computing the relatedness between entities in the next section.

3.3 Top-n recommendations based on learning to rank

Learning to Rank (LTR) can be applied in top-n recommendation tasks to sort a list of items that a user may be interested in. The model is trained using data such as ratings, rankings, or implicit feedback, and it learns to rank items based

on this information. Once the model is trained, it can be used to predict the relevance of new items to a given user, and the top-n most relevant items can be recommended to the user. [8]. One of the main advantages of using LTR is that it allows for a more fine-grained control over the ranking process. Unlike traditional collaborative filtering methods which tend to recommend items that are similar to items, LTR methods can take into account multiple aspects of the items, such as their content and context, to generate more accurate and personalized recommendations.

Top-n item recommendation for a particular user $u \in U$ is extracted from the n-highest scores from the list of candidate items. Therefore, a ranking function $\mathfrak{r}(u, i)$ is a function that evaluates the relevance of an item to a user by assigning it a score which represents the level of relevance of the item to the user, with higher scores indicating that the item is more relevant. The ranking function's objective is to order a list of items by relevance to the user, with the items having the highest scores considered to be the most relevant. We have a set of sub-graphs extracted from a knowledge graph based on the set of relation types. The entity embedding based on relation types x_p is employed to compute the similarity scores for the ranking function as follows:

$$\mathfrak{r}(u, i) = \mathfrak{S}(x_p(u), x_p(i)) \tag{2}$$

where $\mathfrak{S}(x_p(u), x_p(i))$ denote the cosine similarity between item vector $x_p(i)$ on relation type p and user vector $x_p(u)$ on relation type p.

In our work, we employ LambdaMART [2] which learn to ranking based on the gradient boosting framework. It is a tree-based algorithm that is designed to optimize the pairwise or list-wise loss functions, which measure the difference between the predicted order of items and their true order. LambdaMART uses a combination of multiple decision trees to make the ranking predictions, and it optimizes the parameters of the trees using gradient descent.

In the next section, we will be introducing our training model that is based on feature learning utilizing relation types present in the knowledge graph. Additionally, a ranking function will be implemented to optimize the model. This approach allows us to effectively learn important features and improve the overall performance of the model.

3.4 Training model and optimisation

Given the interaction dataset from the user set U on item set I and the knowledge graph G, each user u_k is associated with a set of items from the interaction data $\vec{d_k} = \{d_{k1}, d_{k2}, ..., d_{kn}\}$, and the set of labels $\vec{y_k} = \{y_{k1}, y_{k2}, ..., y_{kn}\}$, the training set is defined $X_{ds} = \{(u_1, \vec{d_1}, \vec{d_1}), (u_2, \vec{d_2}, \vec{y_2}), ..., (u_N, \vec{d_N}, \vec{y_N})\}$, where N is the number interaction between users and items in the dataset.

With the ranking function $\mathfrak{r}(u, i)$, we can compute the relatedness score $\vec{\mathfrak{r}}(u, i)$ for pairs of user u and item i. The top-n recommendation task becomes learning a prediction function $y_k = f(\mathfrak{r}(u, i), \theta)$. with the loss function is computed as follows [14]:

$$L(\theta) = \sum_{k=1}^{N} (1 - \mathcal{A}(\pi(u_k, \vec{d_k}, \theta)) \tag{3}$$

where $\mathcal{A}(\pi(u_k, \vec{d_k}, \theta))$ denote the agreement between the permutation $\pi(u_k, \vec{d_k}, \theta$ induced from $\mathfrak{r}(u_k, \vec{d_k}, \theta$ and the ground truth relevance of items. Therefore, the

goal of the learning process is to find the set of parameters θ that minimize the loss function $L(\theta)$ on the training data X_{ds}.

4 Experiments

In this section, we evaluate our approach on a real case of vehicle purchase/sale domain. The structure and organization of the vehicle's description and user profiles are carried on by using ontologies that can queried by using SPARQL language [12].

4.1 Dataset and evaluation

We extract and built our dataset from vehicle purchase/sale domain that include users are clients and items present for vehicle models on the market. We summarize the statistic of our datasets in Table 1.

User - Item Interaction	Users	393
	Items	5.537
	Iteractions	99.121
Knowledge Graph	Entities	27.561
	Relation types	6
	Triplets	822.000

Table 1. Some statistics on the dataset

In order to evaluation our work, we use a standard information retrieval metrics such as Precision at n, recall at n and Mean Average Precision (MAP) that can be used to evaluate the performance of a recommender system. Precision @n shows the proportion of recommended items that are relevant to the user among the top-n recommendations and is defined as follows:

$$P(n) = \frac{1}{|U|} \sum_{u \in U} \sum_{k=1}^{n} \frac{can(i_k, u)}{n} \tag{4}$$

where $can(i_k, u) = 1$ if the candidate item i is relevant to user u, otherwise $can(i_k, u) = 0$. While recall @k is the proportion of relevant items that are recommended among all relevant items and is calculated as follows:

$$R(n) = \frac{1}{|U|} \sum_{u \in U} \sum_{k=1}^{n} \frac{can(i_k, u)}{rel(u)} \tag{5}$$

where $rel(u)$ is the relevant items set for user u in the test set. The third metric MAP allows a measure of the average precision at n for a set of item recommendations.

$$MAP(n) = \frac{1}{|U|} \sum_{u \in U} \sum_{k=1}^{n} \frac{can(i_k, u)}{n} \times rel(i_k, u) \tag{6}$$

These metrics can be used to assess the accuracy of the recommendations made by the recommender system, with higher values indicating better performance.

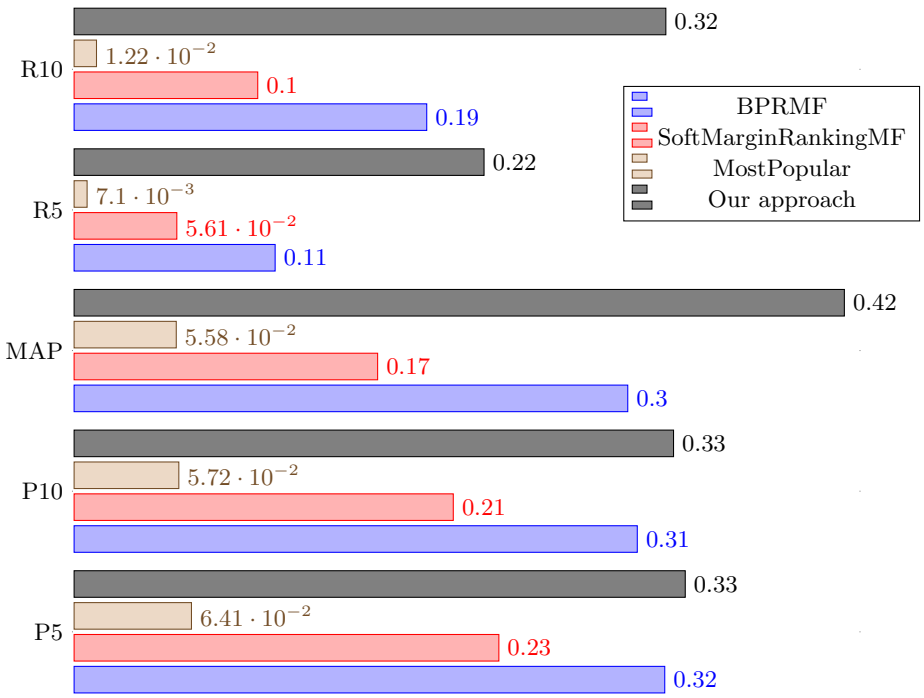

Fig. 1. Comparison with other approaches on the dataset

4.2 Results

To conduct a thorough experiment on the dataset, we have pre-configured certain hyper-parameters on the model to optimize its performance. One of the key hyper-parameters that we have set is the dimension of embeddings, which we have set to 200. This value was chosen based on prior research and experimentation, and it has been found to be effective in capturing the underlying structure of the data. Another important hyper-parameter that we have set is the number of random walks per node in the graph. This value has been set to 100 and is used to explore the connectivity of the nodes in the graph. In addition to these specific hyper-parameter settings, we have also retained the default values in the LambdaMart learning to rank and node2vec models.

The experiments focus on extracting and comparing knowledge graph embeddings of different features. Specifically, the feature sets are defined as a set of six features {*Vehicle style*, *Fuel style*, *Mileage*, *Number of seats*, *Transmission type*, *Vehicle price*}. These features have been chosen as they are considered to be the

most relevant and informative attributes of a item vehicle, and they are commonly used in the automotive domain to describe and compare different models. The goal of the experiments is to evaluate these features by supplementing them with information from knowledge graphs using relation type embeddings for each item vehicle.

From the experimental results depicted in figure 1 and table 2, several key insights can be gleaned. Our proposed approach demonstrates superior performance in terms of top-n recommendations across various evaluation metrics when compared to other approaches [3] such as MostPopular, SoftMarginRankingMF, and BPRMF on the vehicle dataset. This serves as evidence that incorporating features from a knowledge graph can provide additional information that improves the prediction of top-n items for a given user. Additionally, it highlights the potential of using knowledge graph-based features to enhance recommendation systems.

Relation Type Feature	P5	P10	MAP	R5	R10
Vehicle Style feature	0.3491	0.3501	**0.4509**	0.2179	0.3268
Fuel Type Feature	0.3348	0.3412	0.4295	0.2113	0.3163
Mileage Feature	0.2534	0.2811	0.3890	0.1889	0.2968
Number of seats Feature	0.3160	0.3017	0.4190	0.2051	0.2897
Transmission Type Feature	**0.3791**	**0.3567**	0.4311	**0.2228**	**0.3296**
Vehicle Price Feature	0.3557	0.3374	0.4059	0.2132	0.3057

Table 2. Result of on different feature evaluation from the dataset

In this work, we aimed to investigate the role of different features extracted from a knowledge graph through relation type embeddings. The results presented in table 2 demonstrate that there are variations in the achieved results when utilizing different features. By separating features by relation type, we can gain insight into the factors that influence item recommendations for users. As a result, we used these embeddings to evaluate the importance of individual features in relation to the overall results obtained. Additionally, it allows us to identify which relation types are more informative for recommendation and which are less important, and use it to propose an interpretation to users, which can make the recommendation system more transparent, and can also help to improve the user's understanding and trust of the system.

5 Conclusion and Perspectives

In this paper, we investigate the construction of an approach for a personalized recommender system based on building knowledge graph embedding, which allows for the extraction and addition of more information into the learning-to-rank process. From our work, we illustrate how we can separate building sub-graphs for each relation type in the knowledge graph and use it to build

embeddings. The results obtained from the experiment show that our approach is of interest as it provides better results compared to other approaches on our dataset. In our future work, we intend to research the exploitation of ways to explain the recommendation results to a given user by analyzing features learned from knowledge graphs.

Acknowledgment

This work was funded by the French Research Agency (ANR) and by the company Vivocaz under the project France Relance - preservation of R&D employment (ANR-21-PRRD-0072-01).

References

1. Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. *Advances in neural information processing systems*, 26, 2013.
2. Christopher JC Burges. From ranknet to lambdarank to lambdamart: An overview. *Learning*, 11(23-581):81, 2010.
3. Zeno Gantner, Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. MyMediaLite: A free recommender system library. In *5th ACM International Conference on Recommender Systems (RecSys 2011)*, 2011.
4. Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864, 2016.
5. Ngoc Luyen Le, Marie-Hélène Abel, and Philippe Gouspillou. Towards an ontology-based recommender system for the vehicle sales area. In *Progresses in Artificial Intelligence & Robotics: Algorithms & Applications*, pages 126–136, Cham, 2022. Springer International Publishing.
6. Luyen Le Ngoc, Marie-Hélène Abel, and Philippe Gouspillou. Apport des ontologies pour le calcul de la similarité sémantique au sein d'un système de recommandation. In *Ingénierie des Connaissances (Evènement affilié à PFIA'22 Plate-Forme Intelligence Artificielle)*, Saint-Étienne, France, June 2022.
7. Chan Liu, Lun Li, Xiaolu Yao, and Lin Tang. A survey of recommendation algorithms based on knowledge graph embedding. In *2019 IEEE International Conference on Computer Science and Educational Informatization (CSEI)*, pages 168–171. IEEE, 2019.
8. Tie-Yan Liu et al. Learning to rank for information retrieval. *Foundations and Trends® in Information Retrieval*, 3(3):225–331, 2009.
9. Pasquale Lops, Marco de Gemmis, and Giovanni Semeraro. Content-based recommender systems: State of the art and trends. *Recommender systems handbook*, pages 73–105, 2011.
10. Jie Lu, Dianshuang Wu, Mingsong Mao, Wei Wang, and Guangquan Zhang. Recommender system application developments: a survey. *Decision Support Systems*, 74:12–32, 2015.
11. LE Ngoc Luyen, Anne Tireau, Aravind Venkatesan, Pascal Neveu, and Pierre Larmande. Development of a knowledge system for big data: Case study to plant phenotyping data. In *Proceedings of the 6th International Conference on Web Intelligence, Mining and Semantics*, WIMS '16, 2016.
12. Philippe Gouspillou Luyen Le Ngoc, Marie-Hélène Abel. Improving semantic similarity measure within a recommender system based-on rdf graphs. In *Proceedings of the 6th International Conference on Information Technology & Systems*, 2023.

13. Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 26, 2013.
14. Enrico Palumbo, Diego Monti, Giuseppe Rizzo, Raphaël Troncy, and Elena Baralis. entity2rec: Property-specific knowledge graph embeddings for item recommendation. *Expert Systems with Applications*, 151:113235, 2020.
15. Enrico Palumbo, Giuseppe Rizzo, and Raphaël Troncy. Entity2rec: Learning user-item relatedness from knowledge graphs for top-n item recommendation. In *Proceedings of the eleventh ACM conference on recommender systems*, pages 32–36, 2017.
16. M Andrea Rodriguez and Max J. Egenhofer. Determining semantic similarity among entity classes from different ontologies. *IEEE transactions on knowledge and data engineering*, 15(2):442–456, 2003.
17. Jessica Rosati, Petar Ristoski, Tommaso Di Noia, Renato de Leone, and Heiko Paulheim. Rdf graph embeddings for content-based recommender systems. In *CEUR workshop proceedings*, volume 1673, pages 23–30. RWTH, 2016.
18. J Ben Schafer, Dan Frankowski, Jon Herlocker, and Shilad Sen. Collaborative filtering recommender systems. In *The adaptive web*, pages 291–324. Springer, 2007.
19. Zhu Sun, Jie Yang, Jie Zhang, Alessandro Bozzon, Long-Kai Huang, and Chi Xu. Recurrent knowledge graph embedding for effective recommendation. In *Proceedings of the 12th ACM conference on recommender systems*, pages 297–305, 2018.
20. Kai Wang, Tiantian Zhang, Tianqiao Xue, Yu Lu, and Sang-Gyun Na. E-commerce personalized recommendation analysis by deeply-learned clustering. *Journal of Visual Communication and Image Representation*, 71:102735, 2020.
21. Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. Knowledge graph embedding: A survey of approaches and applications. *IEEE Transactions on Knowledge and Data Engineering*, 29(12):2724–2743, 2017.
22. Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of the AAAI conference on artificial intelligence*, volume 28, 2014.
23. Fuzheng Zhang, Nicholas Jing Yuan, Defu Lian, Xing Xie, and Wei-Ying Ma. Collaborative knowledge base embedding for recommender systems. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 353–362, 2016.
24. Yongfeng Zhang, Xu Chen, et al. Explainable recommendation: A survey and new perspectives. *Foundations and Trends® in Information Retrieval*, 14(1):1–101, 2020.
25. Qiannan Zhu, Xiaofei Zhou, Jia Wu, Jianlong Tan, and Li Guo. A knowledge-aware attentional reasoning network for recommendation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 6999–7006, 2020.