Quaternion-Based Knowledge Graph Network for Recommendation

Zhaopeng Li[1,2], Qianqian Xu[3*]
Yangbangyan Jiang[1,2], Xiaochun Cao[1,2,6], Qingming Huang[3,4,5,6*]
[1]State Key Laboratory of Information Security, Institute of Information Engineering, CAS, Beijing, China
[2]School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China
[3]Key Laboratory of Intelligent Information Processing, Institute of Computing Technology, CAS, Beijing, China
[4]School of Computer Science and Technology, University of Chinese Academy of Sciences, Beijing, China
[5]Key Laboratory of Big Data Mining and Knowledge Management, Chinese Academy of Sciences, Beijing, China
[6]Peng Cheng Laboratory, Shenzhen, China
{lizhaopeng,jiangyangbangyan,caoxiaochun}@iie.ac.cn,xuqianqian@ict.ac.cn,qmhuang@ucas.ac.cn

ABSTRACT

Recently, to alleviate the data sparsity and cold start problem, many research efforts have been devoted to the usage of knowledge graph (KG) in recommender systems. It is common for most existing KG based models to represent users and items using real-valued embeddings. However, compared with complex or hypercomplex numbers, these real-valued vectors are of less representation capacity and no intrinsic asymmetrical properties, thus may limit the modeling of interactions between entities and relations in KG. In this paper, we propose Quaternion-based Knowledge Graph Network (QKGN) for recommendation, which represents users and items with quaternion embeddings in hypercomplex space, so that the latent inter-dependencies between entities and relations could be captured effectively. In the core of our model, a semantic matching principle based on Hamilton product is applied to learn expressive quaternion representations from the unified user-item KG. On top of this, those embeddings are attentively updated by a customized preference propagation mechanism with structure information concerned. Finally, we apply the proposed QKGN to three real-world datasets of music, movie and book, and experimental results show the validity of our method.

KEYWORDS

Recommendation, Quaternion Embedding, Knowledge Graph

ACM Reference Format:
Zhaopeng Li, Qianqian Xu, Yangbangyan Jiang, Xiaochun Cao, Qingming Huang. 2020. Quaternion-Based Knowledge Graph Network for Recommendation. In *Proceedings of the 28th ACM International Conference on Multimedia (MM'20), October 12—16, 2020, Seattle, WA, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3394171.3413992

*Corresponding author.

MM '20, October 12–16, 2020, Seattle, WA, USA

ACM ISBN 978-1-4503-7988-5/20/10…$15.00
https://doi.org/10.1145/3394171.3413992

1 INTRODUCTION

The success of recommender system (RS) brings much convenience to web applications, including social media [23], news [58], traveling [50], videos [57], and E-commerce [21], etc. Nowadays, almost every user-oriented software adopts a RS to offer users advice tailored to personal interests. For a long time, the most widely-used framework for RS is collaborative filtering (CF) [18], which only requires historical user-item interactions to model users' potential preference. Unfortunately, traditional CF-based methods could not achieve satisfactory performance when facing sparse historical user-item interactions and the cold-start problem. To address these issues, researchers usually integrate rich side information into RS, such as texts [6, 52] , images [7] and user attributes [40], as additional descriptions of users and items to compensate for the sparsity and pursue better performance. Different from these isolated attributes, Knowledge Graph (KG) enjoys a connected structure with the ability to provide more fruitful facts and link up different items [41], resulting in increasing attention in recent years.

Generally, a KG is a directed heterogeneous graph providing structured representation of facts, consisting of nodes corresponding to real-world entities and edges corresponding to relations. More specifically, the fruitful facts are expressed in triples in the form of (*head*, *relation*, *tail*), e.g., a triple (*The Truman Show*, *film.actor*, *James Eugene Carrey*) in a movie KG represents a fact that "The Truman Show" has an actor named "James Eugene Carrey". Obviously, incorporating KG provides more structural information for RS to better dig out the relational information between users and items, which helps to improve the accuracy of recommendation. In the past few years, many research efforts have been devoted to KG in RS [41, 43, 44], so as to learn effective user/item embeddings and user preference by using KG's structural information. To name a few, RippleNet [41] first proposes a preference propagation mechanism where users' interests are explicitly propagated along the paths in KG. Based on this, KGCN [43] automatically captures users' high order interests in a graph convolutional networks [17, 35] fashion. Then KGAT [44] further explicitly models the high-order relations in the unified user-item KG instead of the original KG.

Despite the great success of existing KG-aware RSs, most of them model users and items in the real number system, which actually limit their performance. Informally, the real-valued embeddings together with its dot products in Euclidean space could handle

symmetry of relations, but usually fail to model the antisymmetric relations which exist universally in KG. Even equipped with transE [4] or its variants [15, 22, 47] to model the antisymmetry, those models are prone to overfitting with an explosive number of parameters [39]. On the contrary, complex or hypercomplex numbers have more representation capability than real-valued ones and further enjoy intrinsic antisymmetrical properties. For example, the Hermitian product between two complex vectors is not symmetric. Consequently, the antisymmetric relations between two entities could be more effectively modeled. As a representative type of hypercomplex systems, the quaternion consists of a real component and three imaginary components. The Hamilton product between two quaternions is noncommutative but also very expressive since it interacts with all the real and imaginary components of both quaternions. Besides, Hamilton product could be interpreted as a spatial rotation with scaling, thus could be naturally used as rotation parameterization. Specifically, while complex numbers only exhibits 2-D rotations, quaternions express the rotation in the 3-D plane. Such larger degrees of freedom make quaternions more flexible and avoid the problem of Gimbal Locks [54]. With these advantages, the application of quaternion and its Hamilton product leads to promising performance in many areas such as speech recognition [31] and KG embedding [54]. This inspires us to incorporate quaternion embeddings into KG-based RS to achieve better modeling of user-item interactions.

In this paper, we propose Quaternion-based Knowledge Graph Network (QKGN) for recommender systems. Our aim is to effectively capture relations in historical user-item interactions and KG with the help of quaternions, so as to improve the performance of top-K recommendation. The key idea of QKGN is to design a semantic matching principle based on Hamilton product to encourage more compact interactions between entities and relations in unified user-item KG. Then the learned representations are further fed into a customized preference propagation mechanism for quaternions, followed by pairwise comparisons for triples and user-item pairs to better capture users' preference. More specifically, our main contributions are listed as follows:

- We propose a new framework QKGN, which models users and items in non-real space with customized quaternion KG embedding layer and preference propagation layer for top-K recommendation in an end-to-end manner.
- With Hamilton product to rotate the head and tail entities based on the relation in each triple, our model could better explore the information from KG than several state-of-the-art methods even without the help of preference propagation mechanism.
- The experimental results on three real-world datasets show the superiority of incorporating quaternion embeddings for KG-aware recommender systems in hypercomplex space.

2 RELATED WORK

2.1 Quaternion-Based Applications

Quaternion is a kind of hypercomplex number system, which is first invented by Hamilton [11]. Due to its rich representational capability as well as high flexibility, recently, quaternions have attracted much interest and been adopted in many areas, including robotics [20], image classification [9], KG embedding [54], automatic speech recognition [31], and general recommender system [55]. Besides, quaternion MLP [29] achieves better performance on spoken language understanding than standard real-valued MLP, and quaternion RNN [30] also outperforms traditional RNNs on the phoneme recognition task with less computation burden. Shortly, quaternions could enable neural networks to code both latent inter- and intra-dependencies [54] between multidimensional input features, and show promising performance. Notably, as far as we know, though [55] introduces quaternions representations into general recommender systems and obtains promising performance, no existing work focuses on incorporating quaternions representations into KG-aware recommender systems, which inspires us to conduct the research work in this paper.

2.2 Knowledge Graph Embedding

As a promising approach which aims to embed a KG into a continuous vector space while preserving the inherent information, knowledge graph embedding (KGE) has attracted intense research focus in recent years. The various KGE methods could be roughly divided into translational models and semantic matching models according to different score functions of entities and relations. TransE [4] is the most representative translational method, which represents both entities and relations as vectors in a same vector space, and considers the relation vector $\mathbf{r}$ as the translation from a head entity $\mathbf{h}$ to its tail entity $\mathbf{t}$, i.e., $\mathbf{h}+\mathbf{r} \approx \mathbf{t}$. Although TransE is effective in 1-to-1 relations modeling, it fails to deal with 1-to-N, N-to-1 and N-to-N relations properly. To this end, a number of models are proposed to improve TransE, e.g., TransH [47] interprets a relation as a translating operation on its relation-specific hyperplane, followed by TransR [22] and transD [15]. Besides, RESCAL [28] is a traditional semantic model based on the factorization of a three-way tensor, which matches latent semantics of entities and relations to measure plausibility. Other representative semantic models include bilinear models [19, 27, 51], neural-network-based models [8, 36], and graph convolutional network [35].

However, all the above methods adopt real-valued vectors for KGE, which neglect the rich representational capability of complex-valued numbers. To this end, ComplEx [39] is proposed to model antisymmetric relations with complex-valued embeddings and Hermitian dot product, shown to be effective in learning KG representations. Then, RotatE [37] proposes a rotation-based translational method which models each relation as a rotation from head to tail entity in the complex vector space. Further, QuatE [54] moves a further step by introducing the quaternion hypercomplex system to represent entities and relations, with a new scoring function assisted by relational rotation quaternions through Hamilton product, and outperforms previous work.

2.3 Knowledge Graph Aware Recommendation

Traditional CF algorithms suffer from data sparsity and cold-start problems, leading to unsatisfactory performance [3, 46]. To alleviate this issue, the usage of KG in recommender system has gained much popularity with its well-defined structure and sufficient resources. Based on the different way to utilize KG information[10], these works could be grouped into embedding-based methods

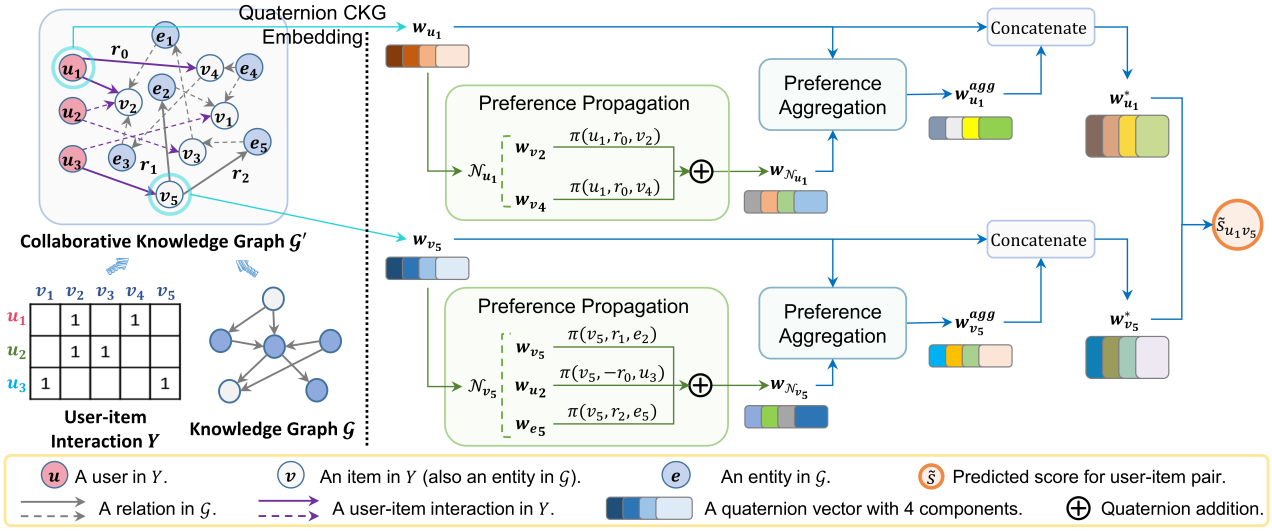


Figure 1: The framework of our Quaternion-based Knowledge Graph Network (QKGN).

[2, 5, 42, 49, 53], path-based methods [13, 14, 24, 45, 48], and unified methods [38, 41, 43, 44, 56]. Considering the limitation of space, here we only discuss a few methods, and one could refer to [10] for a detailed survey. For example, CKE [53] is a typical embedding-based method that combines items' textual embedding, visual embedding, with structural embedding to represent items and give recommendations. Besides, MCRec [13] is a representative of path-based methods that learns the representations of meta-paths to describe the user-item interactions. However, such methods require predefined effective meta-paths with domain knowledge, which limits its usage. Moreover, RippleNet [41], KGCN [43] and KGAT [44] are recent unified methods that consider semantic embeddings of entities and relations, as well as structural connectivity, to fully exploit the information in KG for better modeling of users and items. As mentioned above, most existing KG-aware recommender systems use real-valued embeddings. Thus, we aim to incorporate the richer representation capability and higher flexibility of quaternion embeddings for better recommendations.

3 METHODOLOGY

3.1 Overview

Problem Formulation. Given the user set $\mathcal{U} = \{u_1, u_2, \ldots, u_{|\mathcal{U}|}\}$ and item set $\mathcal{V} = \{v_1, v_2, \ldots, v_{|\mathcal{V}|}\}$, we could record each user-item interaction based on user's implicit feedback:

$$y_{uv} = \begin{cases} 1, & \text{if interaction } (u, v) \text{ is observed,} \\ 0, & \text{otherwise.} \end{cases} \quad (1)$$

As defined in Eq.(1), y_{uv} will be set to 1 if and only if user u have an observed interaction with item v, *e.g.*, watching a movie, listening to a music, purchasing an item, or clicking an advertisement. These interactions form the user-item interaction matrix $Y = \{y_{uv} | u \in \mathcal{U}, v \in \mathcal{V}\} \in \mathbb{R}^{|\mathcal{U}| \times |\mathcal{V}|}$.

Besides, we have a knowledge graph $\mathcal{G}$ as the side information to facilitate the recommendation. Let $\mathcal{E}$ and $\mathcal{R}$ denote the sets of entities and relations in $\mathcal{G}$, respectively. A head-relation-tail fact is represented as a triple (h, r, t), with $h, t \in \mathcal{E}$ and $r \in \mathcal{R}$. With the user-item interaction matrix Y and KG $\mathcal{G}$, we then merge them into a unified directed graph $\mathcal{G}'$, *i.e.*, the collaborative knowledge graph (CKG) [44], to exploit the usable information from KG for recommendation. Specifically, each item $v \in \mathcal{V}$ in Y will be mapped into to a corresponding entity h in KG $\mathcal{G}$ by entity linking[26] if that item could be found as an entity in KG. See the left corner of Figure 1. Accordingly, there exist $M = |\mathcal{U}| + |\mathcal{E}|$ entities and $N = |\mathcal{R}| + 1$ relations in $\mathcal{G}'$ ("+1" represents the relation of observed user-item interaction). Note that we use $-r$ to denote the inverse direction of r. For example, in triples (u_1, r_0, v_4) and $(v_4, -r_0, u_1)$, r_0 represents the relation "user.interacts" while $-r_0$ denotes "item.interacted-by".

Now with the collaborative knowledge graph $\mathcal{G}'$, our goal is to predict the top-K items which would be interested in by each user. More specifically, we aim to learn a prediction function that gives a rank-preserved score $\hat{s}(u, v)$ indicating the preference level for user u on item v.

Framework Overview. Based on existing methods, we further design our QKGN to capture the intricate interactions in real-world data with quaternion representations. As shown in Figure 1, we adopt a quaternion-based knowledge graph network, including a quaternion knowledge graph embedding layer to learn representations from fruitful facts in CKG, and a preference propagation layer to update representations for users and items concerning the structural connectivity. After that, the inner product between obtained representations could predict a score for each unseen user-item interaction, which is used for top-K recommendations.

3.2 Quaternion CKG Embedding Layer

Inspired by the promising performance of KG link prediction by QuatE [54], we design a rotation-based method to learn effective quaternion representations from KG for recommendation. Given CKG $\mathcal{G}'$, we first use quaternions to represent all entities and relations. Then we formulate the plausibility score function for triples under a semantic matching principle based on Hamilton product.

Quaternion Embeddings of CKG. Quaternion is a representative of the hypercomplex number system, which extends the traditional

complex number system to four-dimensional space. Here we first introduce its definition [11].

Definition 1 (Quaternion). *A quaternion $q \in \mathbb{H}$ consists of one real component and three imaginary components, defined as*

$$q = a + b\mathbf{i} + c\mathbf{j} + d\mathbf{k}, \tag{2}$$

where a, b, c, d are real numbers and $\mathbf{i}, \mathbf{j}, \mathbf{k}$ are imaginary units which satisfy the following Hamilton's rules:

$$\mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = \mathbf{ijk} = -1. \tag{3}$$

Accordingly, a quaternion vector $\boldsymbol{w} \in \mathbb{H}^k$ could be denoted as $\boldsymbol{w} = \boldsymbol{a} + \boldsymbol{b}\mathbf{i} + \boldsymbol{c}\mathbf{j} + \boldsymbol{d}\mathbf{k}$, where $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}, \boldsymbol{d} \in \mathbb{R}^k$ are real-valued vectors with dimension k.

Then, for a triple (h, r, t) in the CKG, we could represent the head entity h, tail entity t, and relation r by quaternion embeddings $\boldsymbol{w}_h = \boldsymbol{a}_h + \boldsymbol{b}_h\mathbf{i} + \boldsymbol{c}_h\mathbf{j} + \boldsymbol{d}_h\mathbf{k}$, $\boldsymbol{w}_t = \boldsymbol{a}_t + \boldsymbol{b}_t\mathbf{i} + \boldsymbol{c}_t\mathbf{j} + \boldsymbol{d}_t\mathbf{k}$ and $\boldsymbol{w}_r = \boldsymbol{a}_r + \boldsymbol{b}_r\mathbf{i} + \boldsymbol{c}_r\mathbf{j} + \boldsymbol{d}_r\mathbf{k}$, respectively.

Triple Scoring. Then we aim to obtain the plausibility score that a triple (h, r, t) exists in the CKG. In this paper, we assume the triples obey a semantic matching principle that the head h and the tail t could be matched through proper rotation based on the relation r. To this end, we turn to the products between quaternions to implement the rotation and matching operation.

From the Hamilton rules defined in Eq.(3), it could be derived that $\mathbf{ij} = \mathbf{k}$, $\mathbf{ji} = -\mathbf{k}$, $\mathbf{jk} = \mathbf{i}$, $\mathbf{kj} = -\mathbf{i}$, $\mathbf{ki} = \mathbf{j}$ and $\mathbf{ik} = -\mathbf{j}$. Obviously, such products between imaginary units naturally realize the rotations between different imaginary axes. With these rules, the following Hamilton product [11] could model spatial rotations we need.

Definition 2 (Hamilton product). *Given two quaternions $q_1 = a_1 + b_1\mathbf{i} + c_1\mathbf{j} + d_1\mathbf{k}$ and $q_2 = a_2 + b_2\mathbf{i} + c_2\mathbf{j} + d_2\mathbf{k}$, their Hamilton product is the standard multiplication of all the factors in quaternions:*

$$\begin{aligned} q_1 * q_2 = &(a_1a_2 - b_1b_2 - c_1c_2 - d_1d_2) \\ &+ (a_1b_2 + b_1a_2 + c_1d_2 - d_1c_2)\mathbf{i} \\ &+ (a_1c_2 - b_1d_2 + c_1a_2 + d_1b_2)\mathbf{j} \\ &+ (a_1d_2 + b_1c_2 - c_1b_2 + d_1a_2)\mathbf{k}. \end{aligned} \tag{4}$$

Hamilton product is not commutative, which enables it to efficiently handle antisymmetric relations in CKG. From the perspective of spatial rotations, it could be seen as a rotation in four dimensions with the scaling transformation determined by q_2:

$$q_1 * q_2 = q_1 * |q_2| \left(\frac{q_2}{|q_2|} \right), \tag{5}$$

where $|q_2| = \sqrt{a_2^2 + b_2^2 + c_2^2 + d_2^2}$ is the norm of a quaternion q_2.

Based on this, we adopt the Hamilton product to rotate the head and tail entities. Specifically, we rotate the head embedding $\boldsymbol{w}_h$ by relation embedding $\boldsymbol{w}_r$, and rotate the tail embedding $\boldsymbol{w}_t$ by the conjugation of $\boldsymbol{w}_r$, calculated as $\bar{\boldsymbol{w}}_r = \boldsymbol{a}_r - \boldsymbol{b}_r\mathbf{i} - \boldsymbol{c}_r\mathbf{j} - \boldsymbol{d}_r\mathbf{k}$. To begin with, we first normalize $\boldsymbol{w}_r$ and its conjugation:

$$\boldsymbol{w}_r^\circ(\boldsymbol{a}_r^{'}, \boldsymbol{b}_r^{'}, \boldsymbol{c}_r^{'}, \boldsymbol{d}_r^{'}) = \frac{\boldsymbol{w}_r}{|\boldsymbol{w}_r|}, \quad \bar{\boldsymbol{w}}_r^\circ(\boldsymbol{a}_{\bar{r}}^{'}, \boldsymbol{b}_{\bar{r}}^{'}, \boldsymbol{c}_{\bar{r}}^{'}, \boldsymbol{d}_{\bar{r}}^{'}) = \frac{\bar{\boldsymbol{w}}_r}{|\bar{\boldsymbol{w}}_r|}. \tag{6}$$

After that, we perform the vector Hamilton product between head embedding $\boldsymbol{w}_h$ and normalized relation embedding $\boldsymbol{w}_r^\circ$ to generate the rotated head embedding $\boldsymbol{w}_h^{'}$:

$$\begin{aligned} \boldsymbol{w}_h^{'}(\boldsymbol{a}_h^{'}, \boldsymbol{b}_h^{'}, \boldsymbol{c}_h^{'}, \boldsymbol{d}_h^{'}) &= \boldsymbol{w}_h * \boldsymbol{w}_r^\circ \\ &= (\boldsymbol{a}_h \odot \boldsymbol{a}_r^{'} - \boldsymbol{b}_h \odot \boldsymbol{b}_r^{'} - \boldsymbol{c}_h \odot \boldsymbol{c}_r^{'} - \boldsymbol{d}_h \odot \boldsymbol{d}_r^{'}) \\ &+ (\boldsymbol{a}_h \odot \boldsymbol{b}_r^{'} + \boldsymbol{b}_h \odot \boldsymbol{a}_r^{'} + \boldsymbol{c}_h \odot \boldsymbol{d}_r^{'} - \boldsymbol{d}_h \odot \boldsymbol{c}_r^{'})\mathbf{i} \\ &+ (\boldsymbol{a}_h \odot \boldsymbol{c}_r^{'} - \boldsymbol{b}_h \odot \boldsymbol{d}_r^{'} + \boldsymbol{c}_h \odot \boldsymbol{a}_r^{'} + \boldsymbol{d}_h \odot \boldsymbol{b}_r^{'})\mathbf{j} \\ &+ (\boldsymbol{a}_h \odot \boldsymbol{d}_r^{'} + \boldsymbol{b}_h \odot \boldsymbol{c}_r^{'} - \boldsymbol{c}_h \odot \boldsymbol{b}_r^{'} + \boldsymbol{d}_h \odot \boldsymbol{a}_r^{'})\mathbf{k}, \end{aligned} \tag{7}$$

where $\odot$ denotes the Hadamard product between two real vectors.

Likewise, we could obtain the rotated tail embedding:

$$\boldsymbol{w}_t^{'}(\boldsymbol{a}_t^{'}, \boldsymbol{b}_t^{'}, \boldsymbol{c}_t^{'}, \boldsymbol{d}_t^{'}) = \boldsymbol{w}_t * \bar{\boldsymbol{w}}_r^\circ. \tag{8}$$

Such operation enables more compact interactions between entities and relations in our CKG.

Then, we adopt the inner product between $\boldsymbol{w}_h^{'}$ and $\boldsymbol{w}_t^{'}$ to measure how well the rotated head and tail entities match. Precisely, the inner product of quaternion embeddings is calculated by summing up the inner products between corresponding components:

$$\begin{aligned} g(h, r, t) &= \boldsymbol{w}_h^{'} \cdot \boldsymbol{w}_t^{'} = (\boldsymbol{w}_h * \boldsymbol{w}_r^\circ) \cdot (\boldsymbol{w}_t * \bar{\boldsymbol{w}}_r^\circ) \\ &= \langle \boldsymbol{a}_h^{'}, \boldsymbol{a}_t^{'} \rangle + \langle \boldsymbol{b}_h^{'}, \boldsymbol{b}_t^{'} \rangle + \langle \boldsymbol{c}_h^{'}, \boldsymbol{c}_t^{'} \rangle + \langle \boldsymbol{d}_h^{'}, \boldsymbol{d}_t^{'} \rangle. \end{aligned} \tag{9}$$

Here $g(h, r, t)$ is the plausibility score for (h, r, t). Ideally, a higher $g(h, r, t)$ indicates that the triple is more likely to exist in the CKG, and vice versa.

Considering that existing triples should have higher plausibility scores than non-existing ones, we apply the following pairwise ranking loss to encourage their score difference:

$$\mathcal{L}_{\text{KG}} = \sum_{(h,r,t,t^{'}) \in \mathcal{T}_{\mathcal{G}^{'}}} -\ln \sigma\big(g(h, r, t) - g(h, r, t^{'})\big), \tag{10}$$

where $\mathcal{T}_{\mathcal{G}^{'}} = \big\{(h, r, t, t^{'}) \big| (h, r, t) \in \mathcal{G}^{'}, (h, r, t^{'}) \notin \mathcal{G}^{'}\big\}$ is the training set for quaternion CKG embedding in collaborative knowledge graph, and σ denotes the sigmoid function. Notably, to construct a negative triple $(h, r, t^{'})$, we randomly replace the tail entity in an existing triple as in [44]. The quaternion embedding layer works as a regularizer to induce each user and item to cover the rich information from KG, which is of great importance to the final recommendation (more discussion in section 4).

3.3 Preference Propagation Layer

With the quaternion embedding, we further apply a preference propagation layer for the collaborative knowledge graph, which could capture the first-order structural information for entities to help model the user preference.

Preference Propagation. For an entity h in the collaborative knowledge graph, the user preference would propagate through those triples of which h is the head entity. Following the previous work, we try to trace the propagation to find the preference reflected in h's one-hop neighbors. Let the one-hop neighbor set of h be $\mathcal{N}_h = \big\{(h, r, t) \big| (h, r, t) \in \mathcal{G}^{'}\big\}$. Since it is natural to consider that different tail t has a different importance to head h through relation r, so we apply an attention mechanism in the propagation. Similar to Eq.(9), we rotate head h and tail t by relation r, and calculate the similarity between the rotated head and tail embeddings as the

attention score $\pi(h,r,t)$ to indicate how much information should be propagated from t to h along relation r:

$$\pi(h,r,t) = \tanh(\boldsymbol{w}_t * \bar{\boldsymbol{w}}_r^{\circ}) \cdot \tanh(\boldsymbol{w}_h * \boldsymbol{w}_r^{\circ}). \tag{11}$$

This formulation encourages our model to propagate more information for those entity pairs of smaller angles in hypercomplex space. Then we normalize the score by softmax function:

$$\widetilde{\pi}(h,r,t) = \frac{\exp\big(\pi(h,r,t)\big)}{\sum_{(h,r',t')\in\mathcal{N}_h}\exp\big(\pi(h,r',t')\big)}. \tag{12}$$

With the attention score, we compute the linear combination of h's neighbor entities to capture the first-order structure information of head entity h, denoted as $\boldsymbol{w}_{\mathcal{N}_h}$:

$$\boldsymbol{w}_{\mathcal{N}_h} = \sum_{(h,r,t)\in\mathcal{N}_h} \widetilde{\pi}(h,r,t)\boldsymbol{w}_t. \tag{13}$$

Notably, we calculate $\boldsymbol{w}_{\mathcal{N}_h}$ for all entities in the collaborative knowledge graph, *i.e.*, all users, items and other entities are with corresponding propagation information $\boldsymbol{w}_{\mathcal{N}_h}$, which helps to the following information aggregation procedure.

Information Aggregation. Here we aggregate each entity's quaternion representation $\boldsymbol{w}_h$ and its neighbor propagation information $\boldsymbol{w}_{\mathcal{N}_h}$ to a k'-dimensional quaternion vector $\boldsymbol{w}_h^{agg}$ for each entity h. Especially, to fully fuse the two representations, we apply a well-designed Bi-Interaction Aggregator [44], which performs two kinds of feature interaction with LeakyReLU [25] as activation function:

$$\begin{aligned}\boldsymbol{w}_h^{agg}(\boldsymbol{a}_h^{agg}, \boldsymbol{b}_h^{agg}, \boldsymbol{c}_h^{agg}, \boldsymbol{d}_h^{agg}) &= \text{LeakyRelu}\big(\mathbf{P}_1(\boldsymbol{w}_h + \boldsymbol{w}_{\mathcal{N}_h})\big) \\ &\quad + \text{LeakyRelu}\big(\mathbf{P}_2(\boldsymbol{w}_h \odot \boldsymbol{w}_{\mathcal{N}_h})\big),\end{aligned} \tag{14}$$

where $\mathbf{P}_1, \mathbf{P}_2$ are trainable quaternion weight matrices with $\mathbf{P}_* = \boldsymbol{A}_* + \boldsymbol{B}_*\mathbf{i} + \boldsymbol{C}_*\mathbf{j} + \boldsymbol{D}_*\mathbf{k}$ and $\boldsymbol{A}_*, \boldsymbol{B}_*, \boldsymbol{C}_*, \boldsymbol{D}_* \in \mathbb{R}^{k'\times k}$.

The preference propagation layer helps to explicitly capture the local structural proximity for each user, item and knowledge entity. Considering the rich representation capability of quaternion embeddings as well as to reduce computation cost, we only take the first-order information into account in our model. In fact, we could further explore the high-order structural proximity by stacking more preference propagation layers.

3.4 Model Learning

Prediction. With the above layers, we could obtain both the original and propagation-enhanced quaternion embeddings for each user and item, i.e., $\{\boldsymbol{w}_u, \boldsymbol{w}_u^{agg}\}$ for user u and $\{\boldsymbol{w}_v, \boldsymbol{w}_v^{agg}\}$ for item v. To predict the user-item score, we first concatenate these two representations as the final representation $\boldsymbol{w}_u^*$ for u:

$$\begin{aligned}\boldsymbol{w}_u^*(\boldsymbol{a}_u^*, \boldsymbol{b}_u^*, \boldsymbol{c}_u^*, \boldsymbol{d}_u^*) &= \boldsymbol{w}_u \,\|\, \boldsymbol{w}_u^{agg} \\ &= \big[\boldsymbol{a}_u, \boldsymbol{a}_u^{agg}\big] + \big[\boldsymbol{b}_u, \boldsymbol{b}_u^{agg}\big]\,\mathbf{i} + \big[\boldsymbol{c}_u, \boldsymbol{c}_u^{agg}\big]\,\mathbf{j} + \big[\boldsymbol{d}_u, \boldsymbol{d}_u^{agg}\big]\,\mathbf{k},\end{aligned} \tag{15}$$

where $\|$ denotes the concatenate operation between two quaternion representations, i.e., concatenate the corresponding real parts and imaginary parts of two quaternions, respectively; $[\,\cdot\,,\,\cdot\,]$ denotes the concatenate operation between two real vectors. Similarly, item v's final representation $\boldsymbol{w}_v^*$ is obtained as follows:

$$\boldsymbol{w}_v^*(\boldsymbol{a}_v^*, \boldsymbol{b}_v^*, \boldsymbol{c}_v^*, \boldsymbol{d}_v^*) = \boldsymbol{w}_v \| \boldsymbol{w}_v^{agg}. \tag{16}$$

Finally, due to the rich representation capability of quaternions, we perform a simple inner product between the quaternion representations of user u and item v, to calculate a preference score for each user-item pair:

$$\hat{s}(u,v) = \boldsymbol{w}_u^* \cdot \boldsymbol{w}_v^* = \big\langle \boldsymbol{a}_u^*, \boldsymbol{a}_v^* \big\rangle + \big\langle \boldsymbol{b}_u^*, \boldsymbol{b}_v^* \big\rangle + \big\langle \boldsymbol{c}_u^*, \boldsymbol{c}_v^* \big\rangle + \big\langle \boldsymbol{d}_u^*, \boldsymbol{d}_v^* \big\rangle. \tag{17}$$

Table 1: Statistics of the datasets.

		Last-FM	MovieLens-20M	Book-Crossing
User-Item Interaction	#Users	23,566	116,679	663
	#Items	48,123	16,878	11,385
	#Interactions	3,034,796	6,596,097	16,744
	Density	0.1510%	0.3349%	0.2218%
Knowledge Graph	#Entities	58,266	102,569	77,903
	#Relations	9	32	25
	#triples	464,567	499,474	151,500

Loss Function. Considering that the prediction score of a user and his/her interacted item should be higher than that of the not interacted one, we optimize a pairwise ranking loss $\mathcal{L}_Y$ between different items of certain users:

$$\mathcal{L}_{\text{Y}} = \sum_{(u,i,j)\in\mathcal{T}_Y} -\ln\sigma\big(\hat{s}(u,i) - \hat{s}(u,j)\big)\big), \tag{18}$$

where $\mathcal{T}_Y = \big\{(u,i,j)|y_{ui} = 1, y_{uj} = 0\big\}$ is the training set for user-item interactions. With such ranking loss our QKGN could predict a rank-preserved score with user preference concerned.

Besides, we adopt L_2 regularization to reduce overfitting, denoted as $\mathcal{L}_{\text{reg}} = \|\Theta\|_2^2$, where $\Theta = \{\boldsymbol{W}, \boldsymbol{P}_1, \boldsymbol{P}_2\}$ contains all the quaternion embeddings and trainable parameters in our model. Altogether, we use λ_1 and λ_2 to control the weight of quaternion CKG embedding and regularization, and jointly optimize Eq.(10) and Eq.(18) in the complete objective function of QKGN, as follows:

$$\mathcal{L}_{\text{QKGN}} = \mathcal{L}_{\text{Y}} + \lambda_1\mathcal{L}_{\text{KG}} + \lambda_2\mathcal{L}_{\text{reg}}. \tag{19}$$

Training. Inspired by the success of alternative optimization in KGAT [44], we optimize the total loss $\mathcal{L}_{\text{QKGN}}$ and the quaternion embedding loss $\mathcal{L}_{\text{KG}}$ of KG triples alternatively. In each epoch, we first jointly update all quaternion representations generated by quaternion CKG embedding and preference propagation layers according to the gradients of $\mathcal{L}_{\text{QKGN}}$. After that, the quaternion representations for all entities and relations in CKG are again updated by the gradients of $\mathcal{L}_{\text{KG}}$, so as to better utilize the rich representation learning capability of Hamilton product.

4 EXPERIMENT

To validate the effectiveness of our method, we evaluate QKGN on three real-world recommendation tasks, including music, movie and book recommendation, of which the details are shown in Table 1.

4.1 Last-FM

Dataset description. The original user-item interaction information of Last-FM dataset comes from LFM-1b[1] [34]. Here we take a subset proposed in [44], where the 10-core setting (only users and items with at least ten interactions are retained) are used to ensure the quality of the dataset. Besides, all the tracks here are mapped into entities in freebase to ensure the KG available.

In the experiments, we use the public training set and test set released by [44], where the training set contains 80% interactions and the test set includes remaining ones. We treat each observed

[1] http://www.cp.jku.at/datasets/LFM-1b/

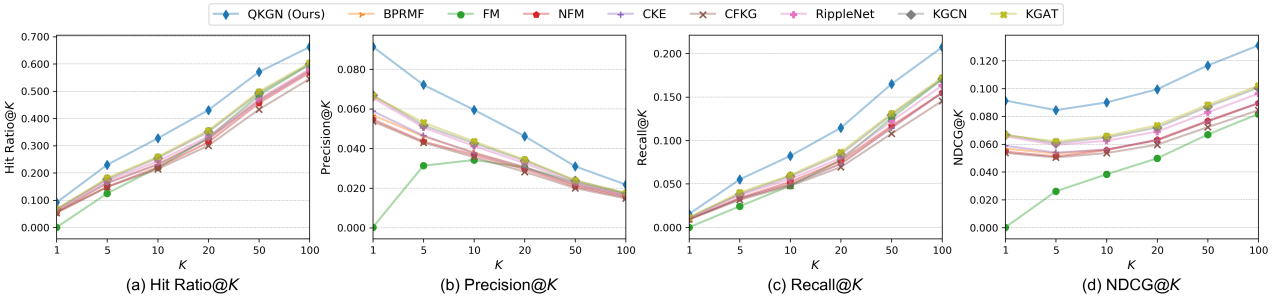


Figure 2: Hit Ratio@K, Precision@K, Recall@K, NDCG@K in top-K recommendation on Last-FM.

user-item interaction as a positive instance, and randomly sample an unobserved item for this user as a negative instance to construct the pairwise comparison.

Evaluation metrics. In the top-K recommendation scenario, we select items with the first K highest predicted scores for each user in the test set, and report the average result of four widely-used evaluation measures including Hit Ratio (HR@K), Precision (P@K), Recall (R@K) and Normalized Discounted Cumulative Gain (NDCG@K). Besides, we take the result with the highest Recall@20 as the best result for each model, and compute all the metrics on $K \in \{1, 5, 10, 20, 50, 100\}$.

Competitors. To demonstrate the effectiveness of our QKGN, we choose following eight representative models as competitors:

- **BPRMF** [32] is a traditional CF model that uses the inner product and pairwise ranking loss.
- **FM** [33] is a content-aware factorization model which adopts factorized interactions of features.
- **NFM** [12] is a strong factorization model that combines the linearity of FM with non-linearity of neural networks.
- **CKE** [53] considers structural, textual and visual information together for recommendation. Here the items are represented as KG embeddings learned by transR [22] as only KG is available.
- **CFKG** [2] integrates user-item interactions and knowledge of items into a unified graph to learn their embeddings.
- **RippleNet** [41] is a memory-network-like model that propagates users' preference alongside paths in KG.
- **KGCN** [43] is an end-to-end model that aggregates and incorporates the information in KG to calculate the representations.
- **KGAT** [44] is a state-of-the-art method that explicitly models the high-order relations from KG under GNN framework.

Implementation details. To better verify the effectiveness of quaternion knowledge graph embedding (QKGE) and preference propagation layer (PPL) for the final recommendation, we implement our proposed model together with two ablated variants:

- **w/o QKGE** only adopts a preference propagation layer to deal with the randomly initialized quaternion embeddings for all the users, items and entities in KG, without the help of QKGE.
- **w/o PPL** applies QKGE to encourage the model to learn expressive representations from collaborative knowledge graph, without a PPL to model the first-order connectivity structure.
- **QKGN** takes advantage of the jointly learning of QKGE and PPL to achieve more comprehensive modeling for the final recommendation, as described in section 3.

Table 2: Experimental results for $K = 20$ on Last-fm. For each metric, the best performance is marked with magenta and the second best is marked with cyan.

Type	Method	HR@20↑	P@20↑	R@20↑	NDCG@20↑
Baselines	BPRMF	0.32458	0.03060	0.07328	0.06275
	FM	0.32560	0.02966	0.07437	0.04796
	NFM	0.31036	0.02900	0.07372	0.06005
	CKE	0.32543	0.03056	0.07298	0.06290
	CFKG	0.29093	0.02731	0.06742	0.05767
	RippleNet	0.33353	0.03259	0.07969	0.06905
	KGCN	0.35025	0.03387	0.08418	0.07200
	KGAT	0.35589	0.03439	0.08648	0.07357
Ours	w/o QKGE	0.37754	0.03724	0.09449	0.08118
	w/o PPL	0.40720	0.04129	0.10233	0.08597
	QKGN	0.43079	0.04622	0.11451	0.09949

We run all the experiments on a Linux machine with a TITAN RTX GPU, 2 Intel Xeon CPUs (E5-2620 v4 @2.10GHz), and 128GB of RAM. We use python 3.6.7 and TensorFlow [1] to implement our QKGN model. Specifically, we set the embedding size $k = 64$ for all models, except that for RippleNet [41] the embedding size is 16 as suggested in paper due to its heavy computational overhead. A single PPL with the hidden dimension $k' = 64$ is applied in our QKGN. Also, we implement the initialization algorithm in [30] tailored for quaternion-valued networks in our model. Besides, we apply Xavier initializer and Adam [16] optimizer for all the models with the same batch size 1024. As for hyper-parameter settings, we set λ_1 as 0.1, λ_2 as $1e-5$, learning rate l as $1e-3$, and dropout rate as 0.1 for our QKGN. The grid search is applied to set the hyper-parameters of all competitors. Notably, three attentive embedding propagation layers with hidden dimensions of $\{64, 32, 16\}$ are used for KGAT [44] as suggested in their paper.

Results. The results of all models for $K = 20$ are presented in Table 2, from which we observe that:

a) Our QKGN achieves a clear margin in performance gain over all baselines on all used metrics. Specifically, although QKGN only applies a single preference propagation layer to capture the first-order connectivity, owing to the expressiveness of quaternion representations, it still outperforms the best competitor KGAT by 7.490%, 1.183%, 2.803%, 2.592%, on four metrics, respectively.

b) Now we analyze the performance of QKGN and its ablated variants. First, these models consistently outperform all the baselines due to the richer representational capability of quaternions. Besides, removing QKGE or PPL both degrades the performance

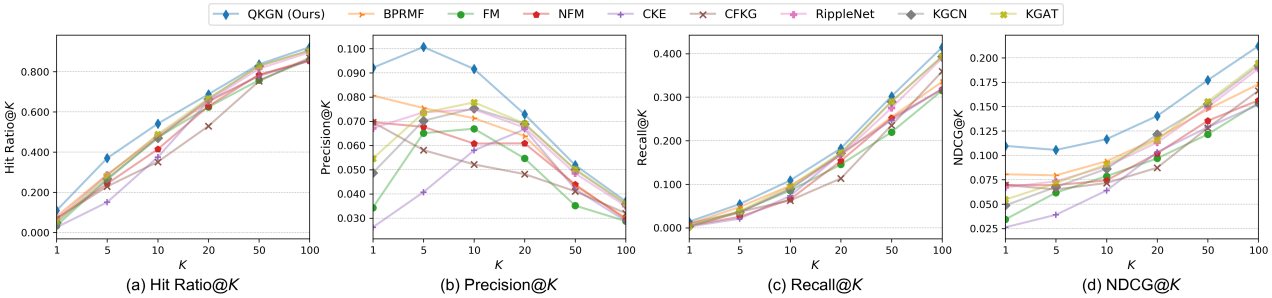


Figure 3: Hit Ratio@K, Precision@K, Recall@K, NDCG@K in top-K recommendation on MovieLens-20M.

of our model, which shows the effectiveness of the regularization of QKGE as well as the PPL. Moreover, compared with removing QKGE, the model which only removes PPL performs better, which again verifies that QKGE itself could lead to quite expressive representations.

c) Among all the competitors, RippleNet, KGCN and KGAT show strong performance compared with other baselines, which shows the importance of capturing proximity information in KG. Besides, CKE and CFKG perform comparably poorly than general recommendation model BPR and factorization models FM and NFM, indicating that they fail to make full use of KG as they simply aggregate the embedding learned by transR or transE to represent users and items.

Besides, according to Figure 2, QKGN also performs better than all baselines for all the different Ks, indicating that our model could achieve promising performance in top-K recommendation.

4.2 MovieLens-20M

Dataset description. MovieLens-20M is a subset of the full MovieLens dataset[2], including more than 20 million explicit ratings (ranging from 0.5 to 5) of more than 100 thousand users on about 27 thousand movies. We first transform these rating into implicit feedback as in [43], and use the same 10-core setting to ensure data quality. Then the same way as in Last-FM is used to split the training set and test set. We also apply the KG released by [43].

Implementation details. All the settings are the same as on Last-FM, except that for QKGN λ_1, λ_2 and l are set as 1.5, 1e − 3 and 5e − 4. Besides, only the first 10,000 users are used in experiments considering the size of KG and the limit of our computing devices.

Results. As shown in Table 3, our proposed QKGN also outperforms all the competitors on all metrics. In particular, QKGN improves the best competitor KGAT's performance by 2.050%, 0.381%, 0.988%, 2.450% on four metrics respectively. Therefore, the effectiveness of our algorithm is again validated. Besides, QKGN consistently obtains better performance than its variants, indicating that QKGE and PPL are essential components for the top-K recommendation. Moreover, observing the phenomenon that FM, NFM and CFKG could hardly perform better than traditional BPRMF, we consider that these methods fail to explore the rich side information from such a large KG (including more than 100 thousand entities), leading to unsatisfactory performance. Similarly, Figure 3 shows that our QKGN also achieves the best performance.

[2]http://grouplens.org/datasets/movielens/

Table 3: Experimental results for $K = 20$ on MovieLens-20M.

Type	Method	HR@20↑	P@20↑	R@20↑	NDCG@20↑
Baselines	BPRMF	0.64980	0.06387	0.16682	0.11747
	FM	0.62450	0.05464	0.14532	0.09685
	NFM	0.62850	0.06087	0.15369	0.10198
	CKE	0.65470	0.06666	0.17028	0.10322
	CFKG	0.52940	0.04816	0.11335	0.08732
	RippleNet	0.65490	0.06737	0.16732	0.11312
	KGCN	0.66358	0.06882	0.17107	0.12147
	KGAT	0.66720	0.06906	0.17265	0.11587
Ours	w/o QKGE	0.66040	0.06669	0.17471	0.12273
	w/o PPL	0.67920	0.07133	0.18072	0.13068
	QKGN	0.68770	0.07287	0.18253	0.14037

4.3 Book-Crossing

Dataset description. The original Book-Crossing dataset[3] [59] contains more than one million explicit ratings (ranging from 0 to 10) of more than 200 thousand books in the Book-Crossing community. Considering its sparsity, we treat all the rated books as positive samples to construct implicit feedback, and clean this dataset by the same 10-core setting. The same way as in MovieLens-20M is used to construct the training set and test set, and the KG constructed by [41] are choose as side information.

Implementation details. To show the rich representation learning capability of Hamilton products, here we set embedding size k as 16 and PPL's hidden dimension as 16 for our QKGN, while still leaving k as 64 for all the competitors. Note that since Last-FM and MovieLens-20M are relatively complex, we choose a larger dimension for quaternion embeddings than that of Book-Crossing. However, blindly quadrupling the dimension for baselines on these datasets will lead to overfitting, so we follow KGAT's paper to set their dimension as 64. We set λ_1 as 1e − 1, λ_2 as 1e − 4, and learning rate l as 5e − 4. The other settings are the same as on Last-FM.

Results. The results for $K = 20$ are recorded in Table 4. It shows that QKGN still consistently outperforms all the baselines over all the metrics. Specifically, QKGN outperforms the best competitor KGAT by 6.787%, 0.694%, 2.697%, 1.528% on four metrics, respectively. Considering that for our methods the embedding size is 16 while for competitors are 64, and our w/o PPL as well as QKGN performs better than KGAT, the superior expressiveness of quaternions and Hamilton product could be proved again. Besides, the performance gap between w/o PPL and w/o QKGE supports this

[3]http://www2.informatik.uni-freiburg.de/ cziegler/BX/

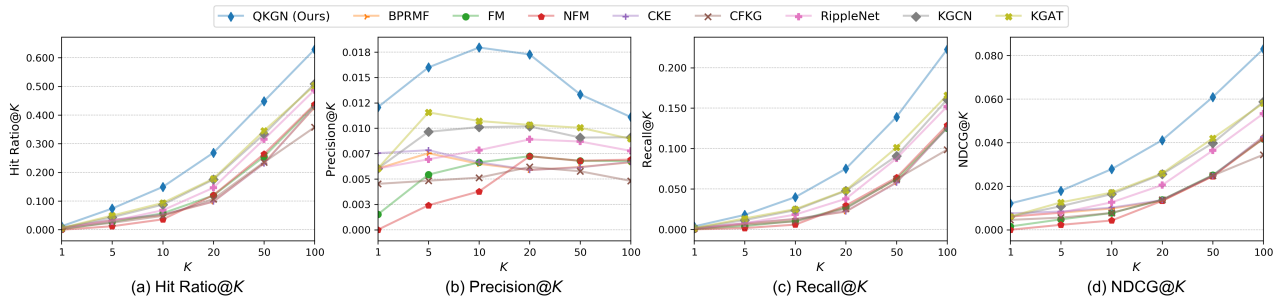


Figure 4: Hit Ratio@K, Precision@K, Recall@K, NDCG@K in top-K recommendation on Book-Crossing.

as well. Moreover, experimental results on different Ks in Figure 4 also indicate QKGN outperforms all the baselines.

4.4 Study of QKGN

To get deep insights into QKGN, we study the effect of parameters, together with the theoretical and running time of our model. More experimental results are placed in supplementary materials.

Parameter sensitivity. We investigate the influence of training weight parameters λ_1 for $\mathcal{L}_{\text{KG}}$, λ_2 for $\mathcal{L}_{\text{reg}}$ on Last-FM, and the dimension k of quaternion embeddings in QKGN on MovieLens-20M. In Figure 5(a), λ_1 varies from 0.1 to 2.0 and λ_2 from 1e − 5 to 1e − 3 with other parameters fixed. It could be observed that the performance is not quite sensitive to the change of λ_1 and λ_2, as all results are higher than 0.10. Specifically, a medium or even smaller λ_2 is better for regularization. Besides, as shown in Figure 5(b) where k changes from 8 to 128, the performance is improved at first as embeddings with larger dimension are more expressive, but after $k = 64$, it drops due to overfitting.

Time complexity analysis. For CKG embedding, the semantic matching principle has computational complexity $O(|\mathcal{G}^{'}|k)$. For preference propagation, assuming the average number of neighbors for each entity to be n, the attention propagation and aggregation have complexity $O(n|\mathcal{G}^{'}|k)$ and $O(|\mathcal{G}^{'}|kk^{'})$, respectively. For prediction, the time complexity is $O(|\mathcal{G}^{'}|k)$, where $|Y|$ is the number of user-item pairs. Thus the overall training time complexity is $O((n+1)|\mathcal{G}^{'}|k + |\mathcal{G}^{'}|kk^{'} + |Y|(k+k^{'}))$.

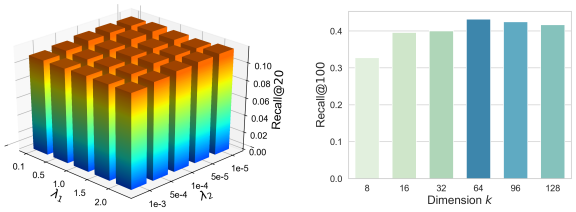
Comparison of running time. Considering the requirement of real-time recommendation, here we show the runtime of inference on Last-FM. Empirically, BPRMF, FM, NFM, CKE, CFKG, RippleNet, KGCN, KGAT and our QKGN cost around 92.6s, 2200.9s, 2263.0s, 101.2s, 843.7s, 1672.5s, 4503.7s, 132.0s and 141.3s on our Linux machine for all testing instances, respectively. As we can see, QKGN achieves medium runtime compared with baselines, which is comparable to KGAT. Consequently, it is promising to apply QKGN to real-world softwares.

5 CONCLUSION

In this paper, we explore the usage of quaternion embeddings with KG to model users and items in hypercomplex space for top-K recommendation. We develop a new framework QKGN, which applies Hamilton product to model triples in CKG with spatial rotation, and adopt a preference propagation mechanism with pairwise comparisons to update embeddings based on structural information.

Table 4: Experimental results for $K = 20$ on Book-Crossing.

Type	Method	HR@20↑	P@20↑	R@20↑	NDCG@20↑
Baselines	BPRMF	0.09804	0.00588	0.02201	0.01326
	FM	0.11916	0.00724	0.02693	0.01375
	NFM	0.12066	0.00724	0.02950	0.01328
	CKE	0.09804	0.00588	0.02218	0.01359
	CFKG	0.10709	0.00513	0.02581	0.01377
	RippleNet	0.14630	0.00890	0.03782	0.02049
	KGCN	0.17496	0.01018	0.04772	0.02550
	KGAT	0.17798	0.01033	0.04831	0.02586
Ours	w/o QKGE	0.14329	0.00694	0.03817	0.01811
	w/o PPL	0.24585	0.01546	0.07083	0.03633
	QKGN	0.26848	0.01727	0.07505	0.04114



(a) Different value of training weight parameters λ_1 and λ_2 on Last-FM

(b) Different value of embedding dimension k on MovieLens-20M

Figure 5: Visualization of parameter sensitivity.

Extensive experiments on three real-world datasets demonstrate the effectiveness of our model.

There are some avenues for future work to explore. Designing (1) different preference propagation mechanisms for users and items separately, and (2) other types of information aggregators except for Bi-Interaction Aggregator, may further improve the performance.

ACKNOWLEDGMENTS

This work was supported in part by the National Key R&D Program of China under Grant 2018AAA0102003, in part by National Natural Science Foundation of China: 61620106009, 61861166002, U1936208, U1636214, 61931008, 61836002, 61672514 and 61976202, in part by Key Research Program of Frontier Sciences, CAS: QYZDJ-SSW-SYS013, in part by Beijing Education Committee Cooperation Beijing Natural Science Foundation (No.KZ201910005007), in part by the Strategic Priority Research Program of Chinese Academy of Sciences, Grant No.XDB28000000, in part by Beijing Natural Science Foundation (4182079), and in part by Youth Innovation Promotion Association CAS.

REFERENCES

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. 2016. Tensorflow: A system for large-scale machine learning. In *Symposium on Operating Systems Design and Implementation*. 265–283.

[2] Qingyao Ai, Vahid Azizi, Xu Chen, and Yongfeng Zhang. 2018. Learning Heterogeneous Knowledge Base Embeddings for Explainable Recommendation. *Algorithms* 11, 9 (2018), 137.

[3] Shilong Bao, Qianqian Xu, Ke Ma, Zhiyong Yang, Xiaochun Cao, and Qingming Huang. 2019. Collaborative Preference Embedding against Sparse Labels. In *ACM MM*. 2079–2087.

[4] Antoine Bordes, Nicolas Usunier, Alberto García-Durán, Jason Weston, and Oksana Yakhnenko. 2013. Translating Embeddings for Modeling Multi-relational Data. In *NIPS*. 2787–2795.

[5] Yixin Cao, Xiang Wang, Xiangnan He, Zikun Hu, and Tat-Seng Chua. 2019. Unifying Knowledge Graph Learning and Recommendation: Towards a Better Understanding of User Preferences. In *WWW*. 151–161.

[6] Rose Catherine and William W. Cohen. 2017. TransNets: Learning to Transform for Recommendation. In *RecSys*. 288–296.

[7] Jingyuan Chen, Hanwang Zhang, Xiangnan He, Liqiang Nie, Wei Liu, and Tat-Seng Chua. 2017. Attentive Collaborative Filtering: Multimedia Recommendation with Item- and Component-Level Attention. In *SIGIR*. 335–344.

[8] Xin Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. 2014. Knowledge vault: a web-scale approach to probabilistic knowledge fusion. In *KDD*. 601–610.

[9] Chase J Gaudet and Anthony S Maida. 2018. Deep quaternion networks. In *IJCNN*. 1–8.

[10] Qingyu Guo, Fuzhen Zhuang, Chuan Qin, Hengshu Zhu, Xing Xie, Hui Xiong, and Qing He. 2020. A Survey on Knowledge Graph-Based Recommender Systems. *CoRR* abs/2003.00911 (2020). arXiv:2003.00911 https://arxiv.org/abs/2003.00911

[11] William Rowan Hamilton. 1848. XI. On quaternions; or on a new system of imaginaries in algebra. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 33, 219 (1848), 58–60.

[12] Xiangnan He and Tat-Seng Chua. 2017. Neural Factorization Machines for Sparse Predictive Analytics. In *SIGIR*. 355–364.

[13] Binbin Hu, Chuan Shi, Wayne Xin Zhao, and Philip S. Yu. 2018. Leveraging Meta-path based Context for Top- N Recommendation with A Neural Co-Attention Model. In *KDD*. 1531–1540.

[14] Xiaowen Huang, Quan Fang, Shengsheng Qian, Jitao Sang, Yan Li, and Changsheng Xu. 2019. Explainable Interaction-driven User Modeling over Knowledge Graph for Sequential Recommendation. In *ACM MM*. 548–556.

[15] Guoliang Ji, Shizhu He, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Knowledge Graph Embedding via Dynamic Mapping Matrix. In *ACL*. 687–696.

[16] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *ICLR*.

[17] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR*.

[18] Yehuda Koren, Robert M. Bell, and Chris Volinsky. 2009. Matrix Factorization Techniques for Recommender Systems. *IEEE Computer* 42, 8 (2009), 30–37.

[19] Timothée Lacroix, Nicolas Usunier, and Guillaume Obozinski. 2018. Canonical Tensor Decomposition for Knowledge Base Completion. In *ICML*. 2869–2878.

[20] Luis Lechuga-Gutiérrez, Jesus Medrano-Hermosillo, and Eduardo Bayro-Corrochano. 2018. Quaternion Spiking Neural Networks Control for Robotics. In *LA-CCI*. 1–6.

[21] Tzu-Heng Lin, Chen Gao, and Yong Li. 2019. CROSS: Cross-platform Recommendation for Social E-Commerce. In *SIGIR*. 515–524.

[22] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning Entity and Relation Embeddings for Knowledge Graph Completion. In *AAAI*. 2181–2187.

[23] Chun-Yi Liu, Chuan Zhou, Jia Wu, Yue Hu, and Li Guo. 2018. Social Recommendation with an Essential Preference Space. In *AAAI*. 346–353.

[24] Weizhi Ma, Min Zhang, Yue Cao, Woojeong Jin, Chenyang Wang, Yiqun Liu, Shaoping Ma, and Xiang Ren. 2019. Jointly Learning Explainable Rules for Recommendation with Knowledge Graph. In *WWW*. 1210–1221.

[25] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. 2013. Rectifier nonlinearities improve neural network acoustic models. In *ICML*. 3.

[26] David N. Milne and Ian H. Witten. 2008. Learning to link with wikipedia. In *CIKM*. 509–518.

[27] Maximilian Nickel, Lorenzo Rosasco, and Tomaso A. Poggio. 2016. Holographic Embeddings of Knowledge Graphs. In *AAAI*. 1955–1961.

[28] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2011. A Three-Way Model for Collective Learning on Multi-Relational Data. In *ICML*. 809–816.

[29] Titouan Parcollet, Mohamed Morchid, Pierre-Michel Bousquet, Richard Dufour, Georges Linarès, and Renato De Mori. 2016. Quaternion Neural Networks for Spoken Language Understanding. In *SLT*. 362–368.

[30] Titouan Parcollet, Mirco Ravanelli, Mohamed Morchid, Georges Linarès, Chiheb Trabelsi, Renato De Mori, and Yoshua Bengio. 2019. Quaternion Recurrent Neural Networks. In *ICLR*.

[31] Titouan Parcollet, Ying Zhang, Mohamed Morchid, Chiheb Trabelsi, Georges Linarès, Renato de Mori, and Yoshua Bengio. 2018. Quaternion Convolutional Neural Networks for End-to-End Automatic Speech Recognition. In *ISCA*. 22–26.

[32] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian Personalized Ranking from Implicit Feedback. In *UAI*. 452–461.

[33] Steffen Rendle, Zeno Gantner, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2011. Fast context-aware recommendations with factorization machines. In *SIGIR*. 635–644.

[34] Markus Schedl. 2016. The LFM-1b Dataset for Music Retrieval and Recommendation. In *ICMR*. 103–110.

[35] Michael Sejr Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. 2018. Modeling Relational Data with Graph Convolutional Networks. In *ESWC*. 593–607.

[36] Richard Socher, Danqi Chen, Christopher D. Manning, and Andrew Y. Ng. 2013. Reasoning With Neural Tensor Networks for Knowledge Base Completion. In *NIPS*. 926–934.

[37] Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. 2019. RotatE: Knowledge Graph Embedding by Relational Rotation in Complex Space. In *ICLR*.

[38] Xiaoli Tang, Tengyun Wang, Haizhi Yang, and Hengjie Song. 2019. AKUPM: Attention-Enhanced Knowledge-Aware User Preference Model for Recommendation. In *KDD*. 1891–1899.

[39] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex Embeddings for Simple Link Prediction. In *ICML*. 2071–2080.

[40] Hongwei Wang, Fuzheng Zhang, Min Hou, Xing Xie, Minyi Guo, and Qi Liu. 2018. SHINE: Signed Heterogeneous Information Network Embedding for Sentiment Link Prediction. In *WSDM*. 592–600.

[41] Hongwei Wang, Fuzheng Zhang, Jialin Wang, Miao Zhao, Wenjie Li, Xing Xie, and Minyi Guo. 2018. RippleNet: Propagating User Preferences on the Knowledge Graph for Recommender Systems. In *CIKM*. 417–426.

[42] Hongwei Wang, Fuzheng Zhang, Miao Zhao, Wenjie Li, Xing Xie, and Minyi Guo. 2019. Multi-Task Feature Learning for Knowledge Graph Enhanced Recommendation. In *WWW*. 2000–2010.

[43] Hongwei Wang, Miao Zhao, Xing Xie, Wenjie Li, and Minyi Guo. 2019. Knowledge Graph Convolutional Networks for Recommender Systems. In *WWW*. 3307–3313.

[44] Xiang Wang, Xiangnan He, Yixin Cao, Meng Liu, and Tat-Seng Chua. 2019. KGAT: Knowledge Graph Attention Network for Recommendation. In *KDD*. 950–958.

[45] Xiang Wang, Dingxian Wang, Canran Xu, Xiangnan He, Yixin Cao, and Tat-Seng Chua. 2019. Explainable Reasoning over Knowledge Graphs for Recommendation. In *AAAI*. 5329–5336.

[46] Zitai Wang, Qianqian Xu, Ke Ma, Yangbangyan Jiang, Xiaochun Cao, and Qingming Huang. 2019. Adversarial Preference Learning with Pairwise Comparisons. In *ACM MM*. 656–664.

[47] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge Graph Embedding by Translating on Hyperplanes. In *AAAI*. 1112–1119.

[48] Yikun Xian, Zuohui Fu, S. Muthukrishnan, Gerard de Melo, and Yongfeng Zhang. 2019. Reinforcement Knowledge Graph Reasoning for Explainable Recommendation. In *SIGIR*. 285–294.

[49] Xin Xin, Xiangnan He, Yongfeng Zhang, Yongdong Zhang, and Joemon M. Jose. 2019. Relational Collaborative Filtering: Modeling Multiple Item Relations for Recommendation. In *SIGIR*. 125–134.

[50] Zhenxing Xu, Ling Chen, Yimeng Dai, and Gencai Chen. 2017. A Dynamic Topic Model and Matrix Factorization-Based Travel Recommendation Method Exploiting Ubiquitous Data. *IEEE TMM* 19, 8 (2017), 1933–1945.

[51] Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2015. Embedding Entities and Relations for Learning and Inference in Knowledge Bases. In *ICLR*.

[52] Zheng-Jun Zha, Jiawei Liu, Di Chen, and Feng Wu. 2020. Adversarial Attribute-Text Embedding for Person Search With Natural Language Query. *IEEE TMM* 22, 7 (2020), 1836–1846.

[53] Fuzheng Zhang, Nicholas Jing Yuan, Defu Lian, Xing Xie, and Wei-Ying Ma. 2016. Collaborative Knowledge Base Embedding for Recommender Systems. In *KDD*. 353–362.

[54] Shuai Zhang, Yi Tay, Lina Yao, and Qi Liu. 2019. Quaternion Knowledge Graph Embeddings. In *NIPS*. 2731–2741.

[55] Shuai Zhang, Lina Yao, Lucas Vinh Tran, Aston Zhang, and Yi Tay. 2019. Quaternion Collaborative Filtering for Recommendation. In *IJCAI*. 4313–4319.

[56] Jun Zhao, Zhou Zhou, Ziyu Guan, Wei Zhao, Wei Ning, Guang Qiu, and Xiaofei He. 2019. IntentGC: A Scalable Graph Convolution Framework Fusing Heterogeneous Information for Recommendation. In *KDD*. 2347–2357.

[57] Yipeng Zhou, Jiqiang Wu, Terence H. Chan, Siu-Wai Ho, Dah-Ming Chiu, and Di Wu. 2018. Interpreting Video Recommendation Mechanisms by Mining View Count Traces. *IEEE TMM* 20, 8 (2018), 2153–2165.

[58] Qiannan Zhu, Xiaofei Zhou, Zeliang Song, Jianlong Tan, and Li Guo. 2019. DAN: Deep Attention Neural Network for News Recommendation. In *AAAI*. 5973–5980.

[59] Cai-Nicolas Ziegler, Sean M. McNee, Joseph A. Konstan, and Georg Lausen. 2005. Improving recommendation lists through topic diversification. In *WWW*. 22–32.